Forward-Forward algorithm

Learning without backpropagation

Maciej Żelaszczyk

April 19, 2023

PhD Student in Computer Science
Division of Artificial Intelligence and Computational Methods
Faculty of Mathematics and Information Science

m.zelaszczyk@mini.pw.edu.pl

Warsaw University of Technology

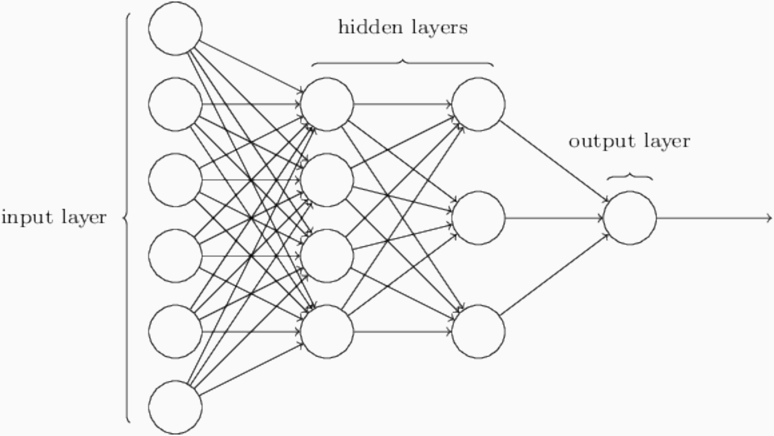
Dominant learning paradigm

The learning procedure for current deep learning models:

- Specific architecture.
- Forward pass.
- Loss (supervised, self-supervised, unsupervised).
- Backward pass via backpropagation (i.e. chain rule).
- Use obtained gradients in a weight-update procedure (e.g. Adam).

Backpropagation

- Forward pass: information flows from input to output layer.
- Backward pass: gradient flows from cost function back.



Source: Nielsen, M. A., *Neural Networks and Deep Learning*

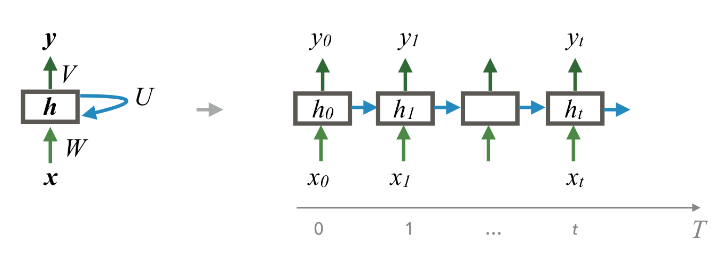
Are neural networks biologically plausible?

Different views:

- Forward pass may be plausible [Rosenblatt, 1958].
- Hebbian learning supported by evidence [Martin et al., 2000], [Malenka and Bear, 2004], [Dan and Poo, 2004].
- Backpropagation is implausible.
- Or is it? [Guerguiev et al., 2017], [Richards et al., 2019], [Lillicrap et al., 2020]
- No overall consensus.

Backpropagation through time

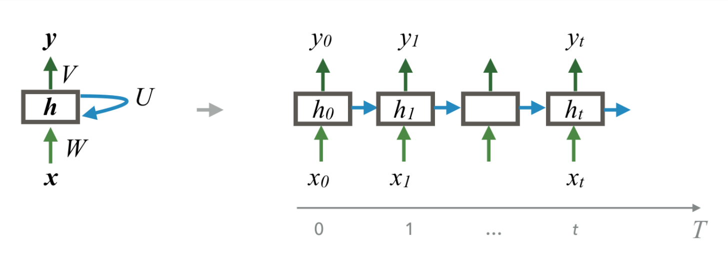
- Forward pass: whole sequence.
- Backward pass: gradient back through whole sequence.
- Computationally expensive.



Source: Gakhov, A., *Recurrent Neural Networks. Part 1: Theory*

Truncated backpropagation through time

- Forward pass: part of sequence.
- Backward pass: gradient back through part of sequence.
- Update weights.
- Move to next part of sequence.
- Every k_1 steps backpropagate through k_2 steps.
- $k_1 = k_2$ different than $k_1 = 1, k_2 > 1$.



Source: Gakhov, A., *Recurrent Neural Networks. Part 1: Theory*

Particularly implausible case

What it would mean to backpropagate through time:

- Run through several time steps forward.
- Weights remain unchanged during this phase.
- Pause the system to perform backpropagation through a couple of steps.
- Gradients are flowing *back in time*.
- Update the weights.
- Run forward again.

In reality, the brain needs to perform learning on the fly.

Additional hurdle

Backpropagation assumes perfect knowledge of the computational model of the forward pass:

- The system has to be differentiable end-to-end.
- What if we need to use non-differentiable operations?
- Max pooling in CNNs:

$$\frac{d}{dx}\max\{a, b\} = \begin{cases} a & \text{if } a > b \\ b & \text{if } a < b \\ \text{does not exist} & \text{if } a = b \end{cases} \tag{1}$$

- In practice, this particular case is not a problem.

Additional hurdle

- Less amenable cases exist, e.g. binarization:

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{2}$$

$$\frac{d}{dx} f(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \text{does not exist} & \text{if } x = 0 \end{cases} \tag{3}$$

- This can still be handled via the *Straight-Through Estimator* [Bengio et al., 2013].
- Relevant for quantized nets [Yin et al., 2019].
- What if we wanted to use $\sin \frac{1}{x}$ in the architecture?
- How about a non-differentiable black-box? E.g. a specific algorithm?
- RL can do this but with low sample efficiency.

Forward-Forward algorithm

- Instead of a forward and backward pass, have two forward passes.
- The passes are operationally equivalent.
- They operate on different data.
- Their objectives are related to *goodness* and mirrored.
- Goodness is calculated locally for each hidden layer.
- The first pass processes real data and its objective is to increase the goodness.
- The second pass processes fake data and its objective is to decrease the goodness.

Forward-Forward algorithm

A simple choice of a goodness function:

$$g(L) = \sum_{j \in L} y_j^2 \tag{4}$$

- L is the set of neuron indices for a given layer.
- y_j is the activation of the j-th neuron in L, e.g. after ReLU.
- y_j is measured before **layer normalization**.
- The objective for real data is to maximize goodness.
- The objective for fake data is to minimize goodness.
- Both the maximization and minimization are done relative to a threshold.

Forward-Forward algorithm

The procedure can be reformulated as a classification objective.

- For a given input we want to classify it as real or fake.

Predicted probability that the input is real:

$$p(\text{real}) = \sigma\left(\sum_{j \in L} y_j^2 - \theta\right) \tag{5}$$

- θ is the threshold value.
- σ is the sigmoid function.
- Can apply standard loss functions, e.g. binary cross-entropy.
- Still need gradients for weight update, but no need for backpropagation.

More than one hidden layer

The procedure as described so far is flawed in one significant way:

- Suppose that for the first layer the training procedure has converged.
- The activations of the first layer are high for real data and low for fake data.
- These activations are fed into the second layer.
- The second layer can then use the magnitude of the activations from the first layer.
- Only the first layer has learned meaningful features.
- There is no incentive for the network to learn other features.

Layer normalization

The problem can be mitigated:

- Length of the vector of activations measures their magnitude.
- Normalize the hidden vector.
- Goodness information from previous layer is removed.
- Only the direction of the previous hidden vector is retained.
- Information on *relative* activations from previous layer is preserved.

Fake data

Real data comes from a particular dataset. Fake data can be obtained in two ways:

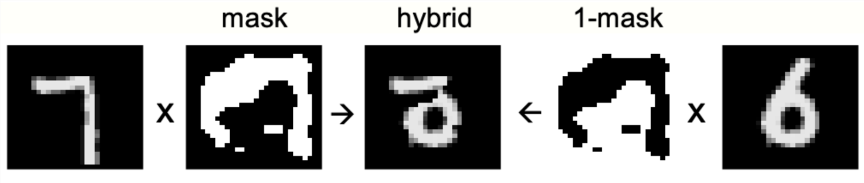
1. Design a specific procedure to produce such data.
2. Let the system generate fake data.

Hand-crafted fake data

For the MNIST dataset, we can consider the following procedure:

1. Random binary mask, same size as input images.
2. Repeatedly apply the filter $[0.25, 0.5, 0.25]$ horizontally and vertically.
3. Threshold the blurred image at 0.5.
4. This is the final mask.
5. Sample two images from the dataset.
6. Apply mask to first one, reverse of the mask to second one.
7. Add two masked images.
8. This is the final fake image.

Hand-crafted fake data



Source: [Hinton, 2022].

Unsupervised training

Train without labels:

- Fully-connected, $4 \times 2,000$, ReLU, 100 epochs.

Train simple classifier with labels:

- Last three hidden layers $\rightarrow$ softmax over labels.

Results on test set:

- FF: 1.37%
- Backpropagation: $\sim 1.4\%$

The unsupervised version can learn representations useful to a simple classifier.

Supervised training

How do we actually use labels for FF?

- One way is to include labels as input.
- Real data: image with correct label.
- Fake data: image with incorrect label.

For MNIST:

- Use images with padding.
- Replace 10 pixels of padding with a representation of the label.
- One-hot encoding is one possible representation.
- Fully-connected, $4 \times 2,000$, ReLU, 60 epochs.

Supervised training

One training/inference regime:

- Last three hidden layers $\rightarrow$ softmax over labels.
- Softmax trained jointly with net.
- Neutral label of all 0.1s as input for inference.

A different training/inference regime:

- Train FF without additions.
- For inference, accumulate goodness of last three hidden layers.
- Run net for image with all the possible label inputs.
- Choose label with highest accumulated goodness.

Results on test set:

- FF: 1.36%
- Backpropagation: ~ 20 epochs to reach similar performance.

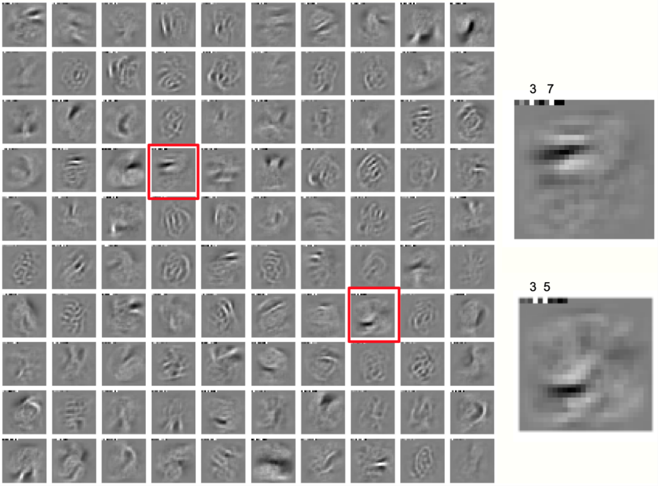
Supervised training

Improved training/inference regimes:

- Use predictions to choose hard negatives.
- For version with softmax, use neutral label in a forward pass to choose the incorrect label with highest score as a hard negative.
- For version with accumulated goodness, use incorrect label with highest accumulated goodness as a hard negative.
- Hard negatives can be used in a sampling procedure.
- Using hard negatives cuts training time by a factor of 3.

Supervised training

For jittered MNIST:



Source: [Hinton, 2022].

Information flow

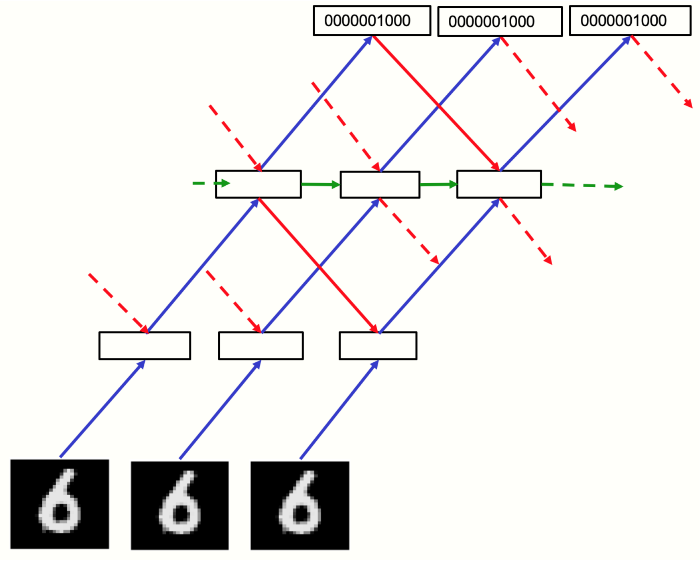
Greedy learning of individual local layers:

- Information flows forward but not back.
- May be a weakness in comparison with backpropagation.
- Use a multi-layer RNN with bottom-up and top-down signal flow - GLOM [Hinton, 2021].

Architecture:

- Treat image as static video.
- Input: image. Output: class representation.
- Intermediate layers.
- Activity vector at each layer is determined by normalized activity vectors at: (1) the layer above, (2) the layer below, (3) the same layer, at the previous time step.

Information flow



Source: [Hinton, 2021].

Information flow

- Both real and fake data are run through the net in time.
- Hidden layers updated synchronously with damping:

$$\mathbf{h}_t = \alpha \mathbf{h}_t^* + (1 - \alpha)\mathbf{h}_{t-1}^{\text{norm}} \tag{6}$$

where $\mathbf{h}_t^*$ is the currently computed state and $\mathbf{h}_{t-1}^{\text{norm}}$ is the normalized state at the previous time step.

Information flow

MNIST:

- $2 - 3$ intermediate layers, $2,000$ neurons each.
- For each image, hidden layers initialized by single bottom-up pass.
- 8 synchronous steps with damping; $\alpha = 0.7$.
- Single pass to produce probabilities for all classes and use them to sample hard negatives.
- Inference:
  - Run net for 8 steps with each of the labels.
  - Average goodness over steps 3 to 5.
  - Pick label with highest goodness.

Results: 1.31% test error after 60 epochs.

Moderate scaling

CIFAR-10:

- Each image has $3 \times 32 \times 32 = 3,072$ dimensions.
- A fully-connected net trained with backpropagation fails badly.
- Compare FF with a network based on local receptive fields.
- This net is not a CNN - no weight sharing.

Moderate scaling

Net:

- $2-3$ hidden layers, $3,072$ ReLUs each.
- Each hidden layer: $3 \times 32 \times 32$ with 3 hidden units at each location.
- Each hidden unit has a 11×11 receptive field for 363 bottom-up inputs from the layer below.

FF:

- Use GLOM-based architecture.
- For bottom-up inputs same as comparision net.
- In the last hidden layer each hidden unit has 10 top-down inputs.
- Other hidden layers have 363 top-down inputs from a 11×11 receptive field from the layer above.

Moderate scaling

Training procedure:

- Both methods use weight decay.
- FF can be either trained to maximize or minimize the activities for real data.

Two inference methods for FF:

- One-pass softmax.
- For each image-label pair let the network run for 10 iterations and accumulate goodness over iterations 4 to 6.

Moderate scaling

Results:

- Performance of FF on test set is worse than for backpropagation but not by much.
- The gap in test error does not increase with the number of layers.
- Backpropagation reduces the training error much more quickly.

Moderate scaling

learning procedure	testing procedure	number of hidden layers	training % error rate	test % error rate
BP		2	0	37
FF min ssq	compute goodness for every label	2	20	41
FF min ssq	one-pass softmax	2	31	45
FF max ssq	compute goodness for every label	2	25	44
FF max ssq	one-pass softmax	2	33	46

Source: [Hinton, 2022].

Moderate scaling

BP		3	2	39
FF min ssq	compute goodness for every label	3	24	41
FF min ssq	one-pass softmax	3	32	44
FF max ssq	compute goodness for every label	3	21	44
FF max ssq	one-pass softmax	3	31	46

Source: [Hinton, 2022].

A link to GANs

One of the forms of FF can be understood as an incarnation of a GAN:

- The core FF net with a measure of goodness corresponds to a discriminator.
- The softmax label predictor along with the procedure used to generate hard negatives corresponds to a generator.
- This simplified setup does not include an adversarial objective.
- No backpropagation at all.
- Can the lack of competition between networks stabilize training?

Open questions

This is early-stage research. Muliple open problems:

- Can the analogy to GANs be used with a more powerful generator, e.g. for images?
- What about other goodness functions? E.g. minimizing the sum of unsquared activities on positive and maximizing it on negative data.
- Activation functions different than ReLU.
- Can we have local goodness functions for different regions of an image?
- Can the sequential processing be cast in a Transformer setup?
- Would it be beneficial to use approaches where there are disparate goodness measures, some maximized and some minimized?

Bengio, Y., Léonard, N., and Courville, A. C. (2013).
Estimating or propagating gradients through stochastic neurons for conditional computation.

Dan, Y. and Poo, M.-M. (2004).
Spike timing-dependent plasticity of neural circuits.
Neuron, 44(1):23–30.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017).
Towards deep learning with segregated dendrites.
eLife, 6.

Hinton, G. (2021).
How to represent part-whole hierarchies in a neural network.

Hinton, G. (2022).

The forward-forward algorithm: Some preliminary investigations.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020).
Backpropagation and the brain.
Nature reviews. Neuroscience, 21(6):1761–1770.

Malenka, R. C. and Bear, M. F. (2004).
Ltp and ltd: An embarrassment of riches.
Neuron, 44(1):5–21.

Martin, S. J., Grimwood, P. D., and Morris, R. G. M. (2000).
Synaptic plasticity and memory: An evaluation of the hypothesis.
Annual Review of Neuroscience, 23(1):649–711.

Richards, B., Lillicrap, T., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R., de Berker, A., Ganguli, S., Gillon, C., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G., Miller, K., Naud, R., Pack, C., Poirazi, P., Roelfsema, P., Sacramento, J., Saxe, A., Scellier, B., Schapiro, A., Senn, W., Wayne, G., Yamins, D., Zenke, F., Zylberberg, J., Therien, D., and Kording, K. (2019).
A deep learning framework for neuroscience.
Nature Neuroscience, 22(11):1761–1770.

Rosenblatt, F. (1958).
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, 65(6):386–408.

Yin, P., Lyu, J., Zhang, S., Osher, S. J., Qi, Y., and Xin, J. (2019).

Understanding straight-through estimator in training activation quantized neural nets.
In *International Conference on Learning Representations*.