Notes on the Forward-Forward Algorithm

Akhil Sadam[1], **Dr.Tan Bui Thanh**[2]

[1]Department of Aerospace Engineering and Engineering Mechanics, UT Austin
[2]The Oden Institute for Computational Sciences, UT Austin

E-mail: akhil.sadam@utexas.edu, tanbui@oden.utexas.edu
February 2023

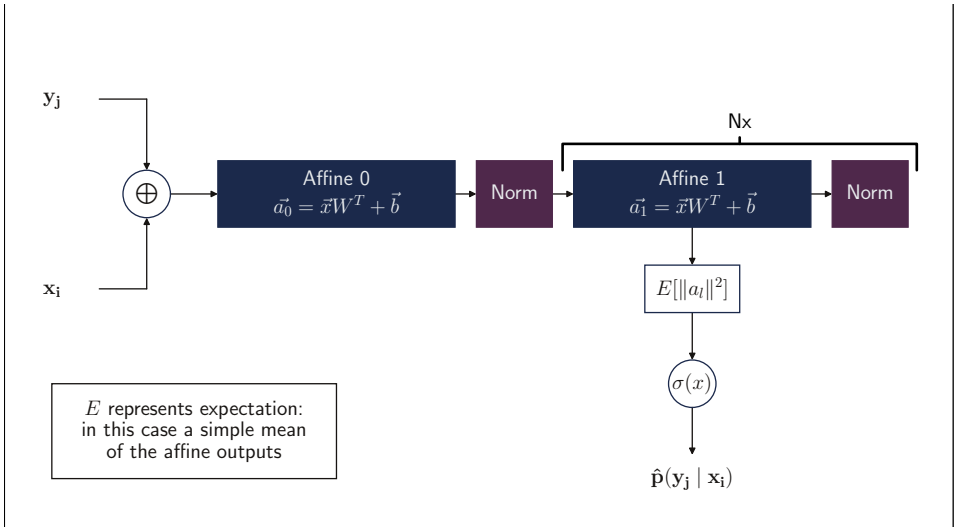
1. Motivation

This document primarily derives missing formulas from Dr. Hinton's paper [**?**], providing a more rigorous framework for further investigation. Note that the Forward-Forward Algorithm (FFA) performs only classification.

2. Base Forward-Forward Architecture (Supervised)

Assume a supervised problem with m labels and t test samples as below.

$$\mathbf{X_{test}} = \{x_i \mid i \in 1, ..., t\}, \ \mathbf{Y_{labels}} = \{y_j \mid j \in 1, ..., m\} \tag{2.1}$$



(a) Supervised FFA Architecture

In prediction mode, each test sample x_i is combined with every label $y_j, \forall j \in 1, ..., m$. The net [Fig. 1a] outputs m predictions of $\hat{p}(y_j \mid x_i)$. For a single-label classifier, the label with the highest probability is chosen. To classify all samples the FFA requires $m * t$ runs. Normalization is required after each layer to ensure that prior layer $\|a_{l-1}\|^2$ does not affect later layers. Training is a greedy layer-by-layer approach.

3. FFA Training

a_{l-1} will represent a_{l-1}^{norm} in this section, as only one layer is considered.

3.1. Loss Derivation

Consider a single layer of the FFA. Note the following:

Each layer is trained to predict $\hat{p}(y_j \mid x_i)$ from a_{l-1}. Training data consists of two datasets, assuming n known points (x_i, y_i).

$$\mathbf{D}_+ := \{(x_{+i}, y_i) \mid i \in 1, ..., n\} := \{(x_i, y_i) \mid i \in 1, ..., n\} \tag{3.1}$$

$$\mathbf{D}_- := \{(x_{-i}, y_i) \mid i \in 1, ..., n\} := \{(x_i*, y_i) \mid i \in 1, ..., n\} \tag{3.2}$$

The positive dataset is simply the correctly labeled points. The negative dataset is defined as incorrectly-labeled or corrupted data; the FFA paper [**?**] does not identify a best choice. Notice also that the negative dataset need not be the same size as the positive dataset, just a scalar multiple. The later implementation uses x_i* as a random sample (taking exactly one negative sample per datapoint).

$$x_i* \in \{x_j \mid j \in 1, ..., n, j \neq i\} \tag{3.3}$$

Probabilities for the positive dataset should be maximized, and minimized for the negative dataset, hence the name. A threshold θ is used as a margin, like in SVM. Derivation of the training procedure for a batch (n,m elements from +,- datasets) is as follows. Here the first layer is considered; any later layer would only replace x with the corresponding a_{l-1}^{norm}.

Definition 3.1 (Single Layer Distribution)
From definition of affine layer,

$$a_{+,-} = x_{+,-}W^T + b \tag{3.4}$$

Define goodness.

$$g_{+,-} := ||a_{+,-}||^2 \tag{3.5}$$

Assume a cumulative distribution function for probabilities.
Thresholding is used for better stability/model confidence.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

$$\hat{p}(y_j \mid x_{+i,-i}) = \sigma(a_{+,-}) \tag{3.7}$$

Want predicted distribution to match the labelled distribution after training.

$$\hat{p}(g_+ > \theta), \quad \hat{p}(g_- < \theta) \quad \overset{desired}{=\!=\!=} 1 \tag{3.8}$$

$$\Leftrightarrow \quad \hat{p}(g_+ - \theta > 0), \quad \hat{p}(\theta - g_- > 0) \quad \overset{desired}{=\!=\!=} 1 \tag{3.9}$$

$$\Rightarrow \quad \sigma(g_+ - \theta), \quad \sigma(\theta - g_-) \quad \overset{desired}{=\!=\!=} 1 \tag{3.10}$$

Converting to a predicted distribution Q(over the two inputs x_+ and x_-):

$$Q(x) = \frac{1}{n+m}[\sigma(g_+ - \theta) \oplus \sigma(\theta - g_-)] \overset{desired\ convergence\ to}{\Longrightarrow} P(x) = \frac{1}{n+m} \tag{3.11}$$

KL-Divergence (Kullback-Liebler) can now be used to measure the difference between the predicted and true distributions.

Definition 3.2 (KL Minimization Loss - Loss 0)

Minimize divergence to true distribution P of correct labels.

KL Divergence is defined as:

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right) = -\sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{Q(x)}{P(x)}\right) \tag{3.12}$$

Substituting:

$$L_0 := D_{KL}(P \parallel Q) = -\mathbb{E}[\ln[\sigma(g_+ - \theta) \oplus \sigma(\theta - g_-)]] \tag{3.13}$$

$$\tag{3.14}$$

Definition 3.3 (KL Maximization Loss - Loss 1)

Alternatively, the KL divergence on the opposite distribution can be maximized.

Let Q_2 be the distribution predicting probabilities of the incorrect labels, and now maximize the divergence to the true distribution P.

$$Q_2(x) = \frac{1}{n+m}[\sigma(\theta - g_+) \oplus \sigma(g_- - \theta)] \tag{3.15}$$

Similar in 3.2, we have:

$$L_1 := -D_{KL}(P \parallel Q_2) = \mathbb{E}[\ln[\sigma(\theta - g_+) \oplus \sigma(g_- - \theta)]] \tag{3.16}$$

The analysis here will ignore bias terms for simplicity. δ will denote the learning rate. Recall $g_{+,-} = ||a_{+,-}||^2$. g is the goodness, and a is the activation. The $+, -$ suffixes will be dropped if both are included in the term.

Definition 3.4 (Weight Update for Loss 0)

$$\Delta W = \delta \frac{dL_0}{dW} = \delta \frac{d\,\mathbb{E}[\ln[\sigma(g_+ - \theta) \oplus \sigma(\theta - g_-)]]}{dW} \tag{3.17}$$

$$= \delta \frac{\partial\,\mathbb{E}[\ln(Q)]}{\partial g} \nabla_W g = 2\delta \frac{\partial\,\mathbb{E}[\ln(Q)]}{\partial g} a x^T \tag{3.18}$$

Note:

$$\frac{\partial\,\mathbb{E}[\ln(Q)]}{\partial g} = \mathbb{E}[\frac{1}{\sigma(\cdots)}\frac{\partial\sigma(\cdots)}{\partial g}] = \mathbb{E}[[1-\sigma(\cdots)]\,\mathrm{sign}_Q(g)] \tag{3.19}$$

So,

$$\Delta W = 2\frac{\delta}{n+m}\sum[(1-\sigma(g_+-\theta))\oplus(\sigma(\theta-g_-)-1)]ax^T \tag{3.20}$$

Definition 3.5 (Weight Update for Loss 1)

$$\Delta W = -\delta\frac{dL_1}{dW} = -\delta\frac{d\,\mathbb{E}[\ln[\sigma(\theta-g_+)\oplus\sigma(g_-\theta)]]}{dW} \tag{3.21}$$

$$= -\delta\frac{\partial\,\mathbb{E}[\ln(Q_2)]}{\partial g}\nabla_W g = -2\delta\frac{\partial\,\mathbb{E}[\ln(Q_2)]}{\partial g}ax^T \tag{3.22}$$

Similarly,

$$\Delta W = -2\frac{\delta}{n+m}\sum[(\sigma(\theta-g_+)-1)\oplus(1-\sigma(g_--\theta))]ax^T \tag{3.23}$$

$$= 2\frac{\delta}{n+m}\sum[(1-\sigma(\theta-g_+))\oplus(\sigma(g_--\theta)-1)]ax^T \tag{3.24}$$

3.2. Loss Functions

The FFA paper [**?**] only mentions the minimization loss L_0. The FFA equations 1,3 are detailed in our notation for comparison to 3.4.

$$g_+ = ||a_+||^2 = \sum_{j=1}^{n} a_j^2 \tag{3.25}$$

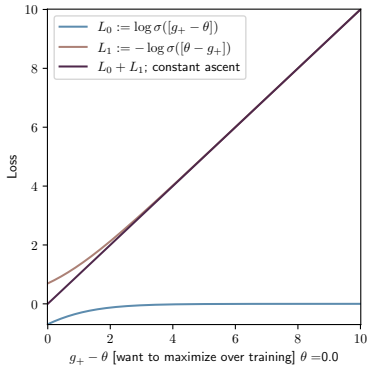
$$p(+) = \sigma(g_+-\theta) \tag{3.26}$$

$$\Delta w_j = 2\delta\frac{\partial\,\mathbb{E}[\ln(p)]}{\partial g_j}a_j x \tag{3.27}$$

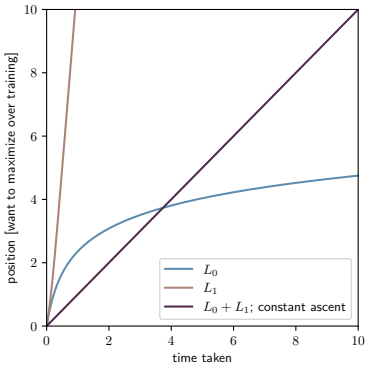
L_0 is clearly a loss that learns slowly near to the supposed minima, while L_1 is a loss that learns slowly at the start. Notice $L_0 + L_1 = x$, so a switch between the two during learning ($L_0 \to L_1$) should be best. Plots for 3 thresholds (0.0, 1.0, 2.0) are shown. While (b) and (c) require integration against a somewhat inconsistent setpoint, (a) allows straightforward comparison. The primary takeway: L_0 loss is slow but maintains a bounded loss, while L_1 loss is fast but easily explodes.

4. Results

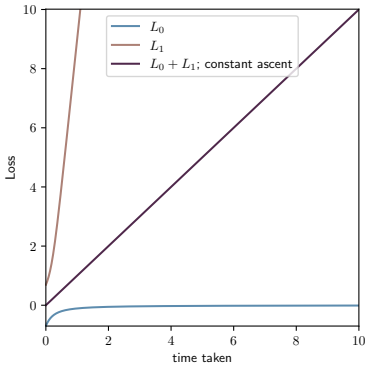
The MNIST (numbers) dataset [**?**] was used to test the FFA losses. A 784x800x10 network was used in all cases with consistent batching rates. The learning rate was



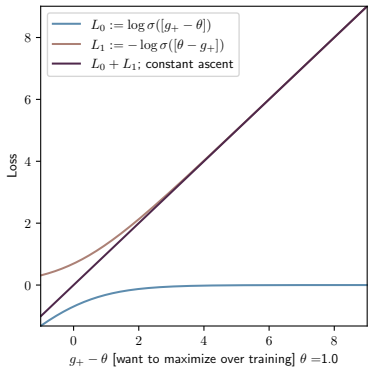
(a) FFA Loss



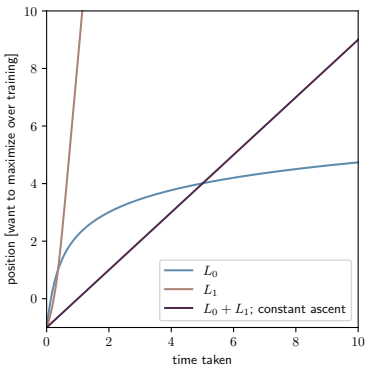
(b) FFA Time to Position



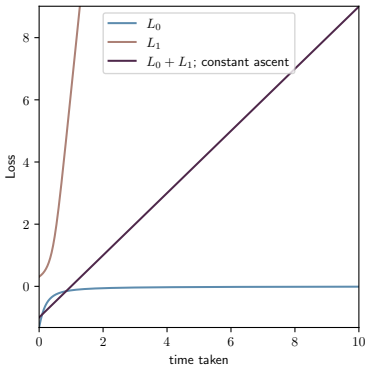
(c) FFA Loss over Time



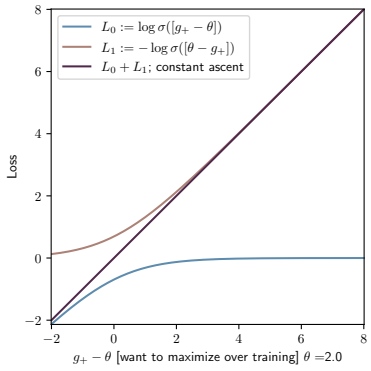
(a) FFA Loss



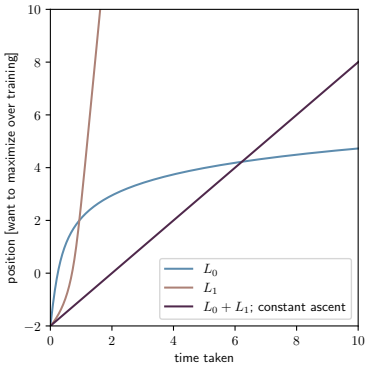
(b) FFA Time to Position



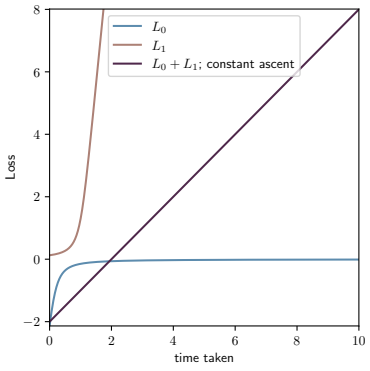
(c) FFA Loss over Time



(a) FFA Loss



(b) FFA Time to Position



(c) FFA Loss over Time

increased along with the batch size throughout training. FFA0 uses the FFA L_0 loss and FFA1 uses the FFA L_1 loss. FFA nets marked as adj., use the adjusted loss functions, where the $\ln[\cdots]$ is replaced with $\ln[1 + \cdots]$,to prevent the loss from exploding. The FFA was trained similar to a DNN (each layer at a time instead of each layer in parallel)

to eliminate differences.

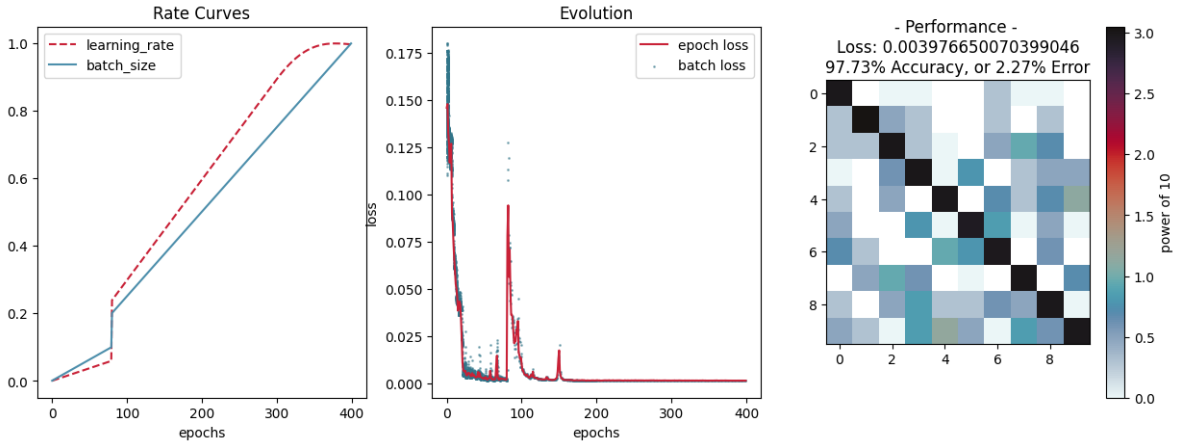
$$L_0 = -\mathbb{E}[\ln[1 + \sigma(g_+ - \theta) \oplus \sigma(\theta - g_-)]] \tag{4.1}$$

$$L_1 = \mathbb{E}[\ln[1 + \sigma(\theta - g_+) \oplus \sigma(g_- - \theta)]] \tag{4.2}$$

Net	Opt	Epochs	Rate	Start	Final	Time[m:s]	Accuracy
DNN_base	ASGD	400	1e-3	50	60000	0:24	97.73%
FFA0	ASGD	400	1e-3	50	60000	1:22	NAN
FFA1	ASGD	400	1e-3	50	60000	1:14	NAN
adj_FFA0_0	ASGD	400	1e-3	50	3200	4:23	50.67%
adj_FFA0_1	ASGD	400	1e-3	50	60000	1:23	49.80%
adj_FFA1_0	ASGD	400	1e-3	50	3200	4:23	19.24%
adj_FFA1_1	ASGD	400	5e-4	50	3200	4:23	24.84%
adj_FFA1_2	ASGD	400	1e-3	50	60000	1:21	21.38%
adj_FFA0_2	Adam	400	1e-3	50	60000	1:13	NAN
adj_FFA1_3	Adam	400	1e-3	50	60000	1:18	NAN

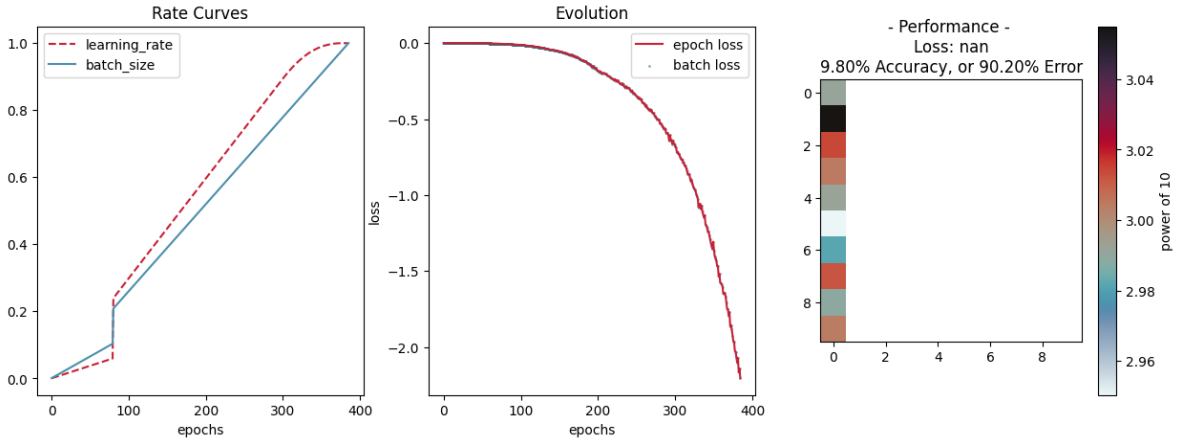
Start and Final are the batch sizes used at the start and end of training. Clearly, the FFA L_0 loss performs better, but nowhere near as well as standard backpropagation.

5. Appendix

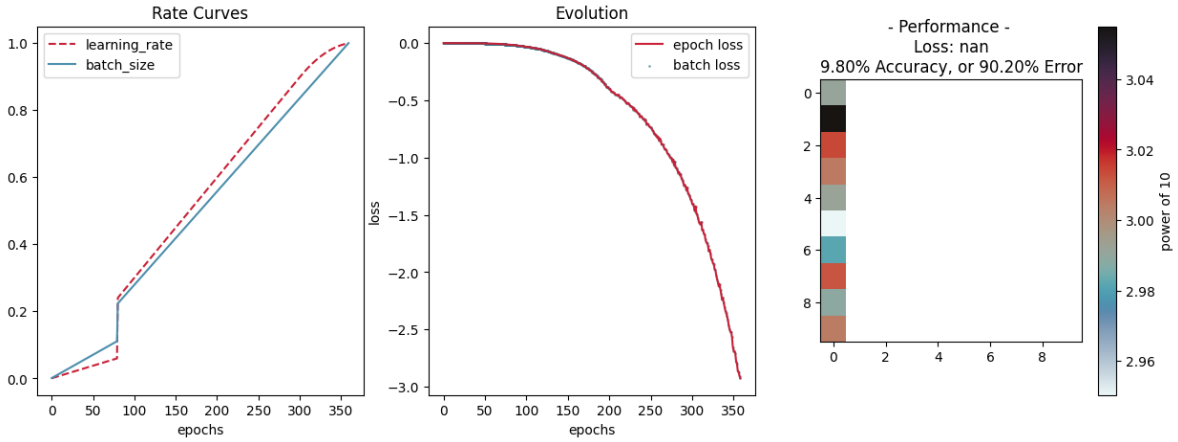


(a) DNN Baseline

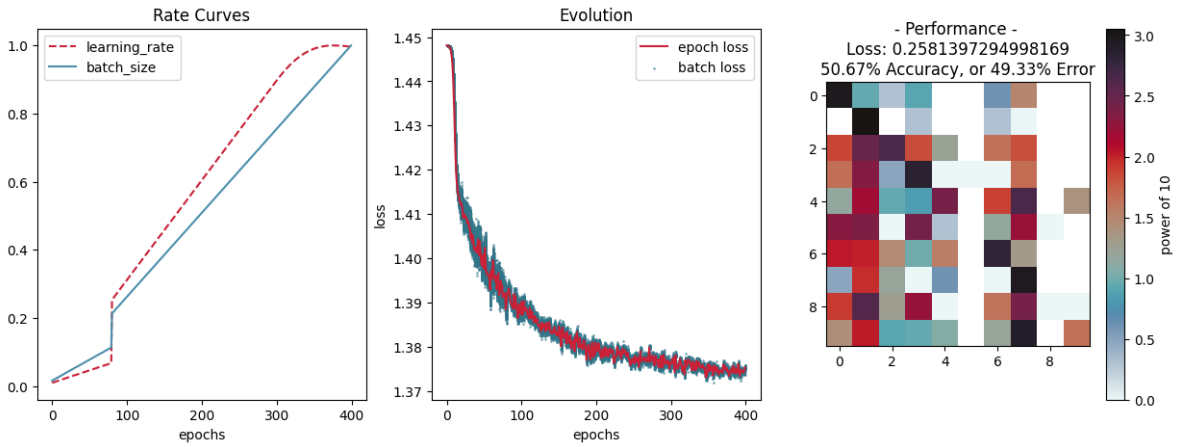
6. References



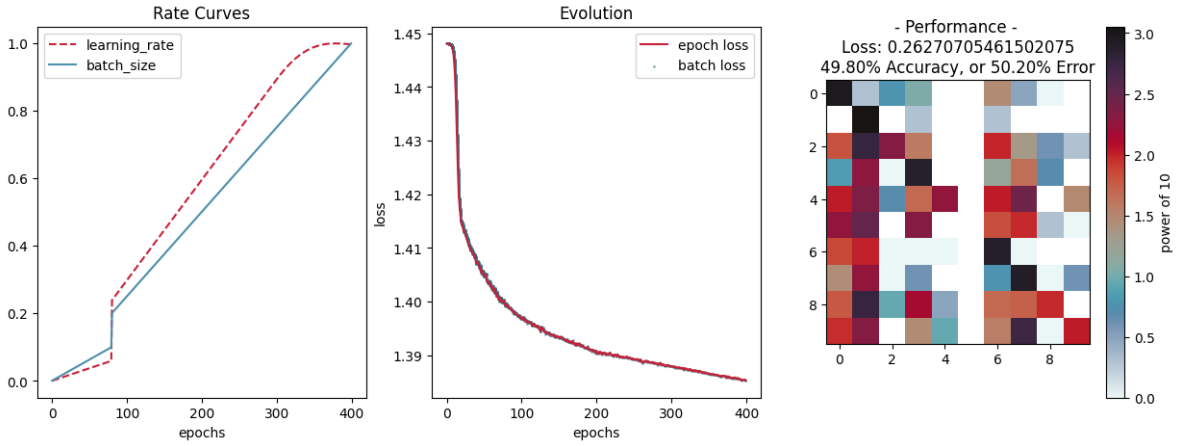
(a) FFA0



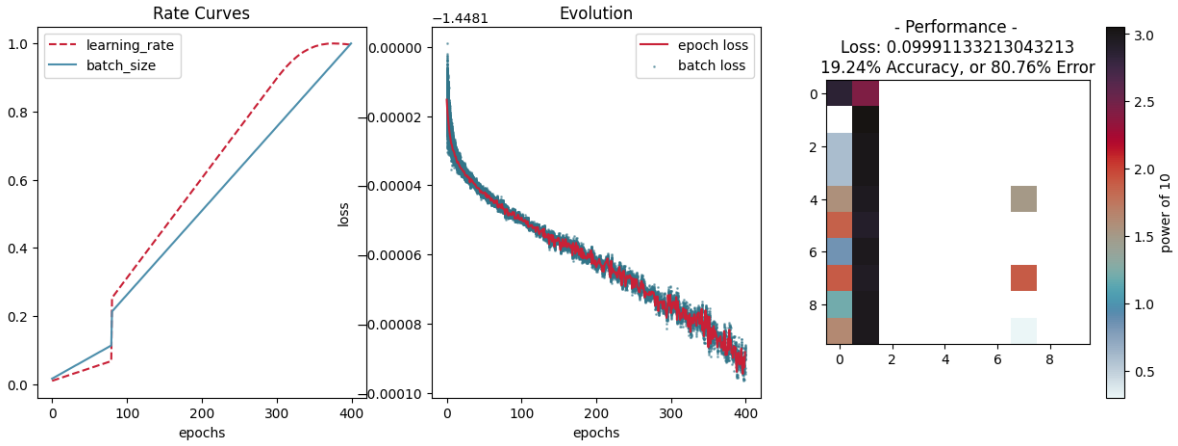
(a) FFA1



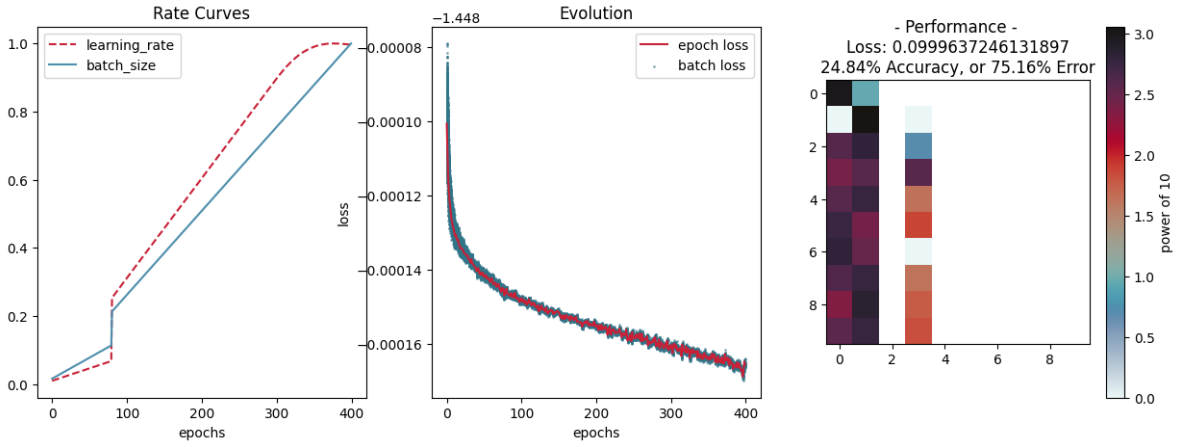
(a) adj_FFA0_0



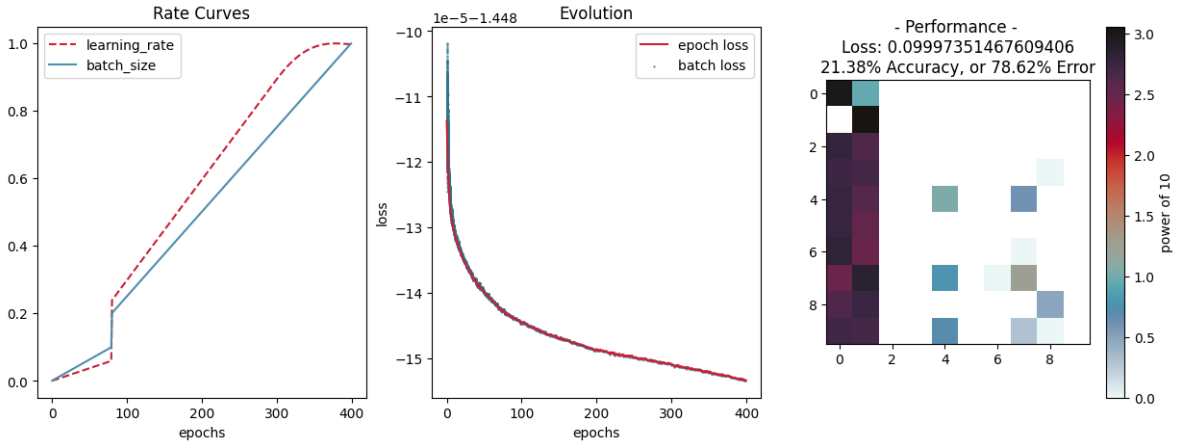
(a) adj_FFA0_1



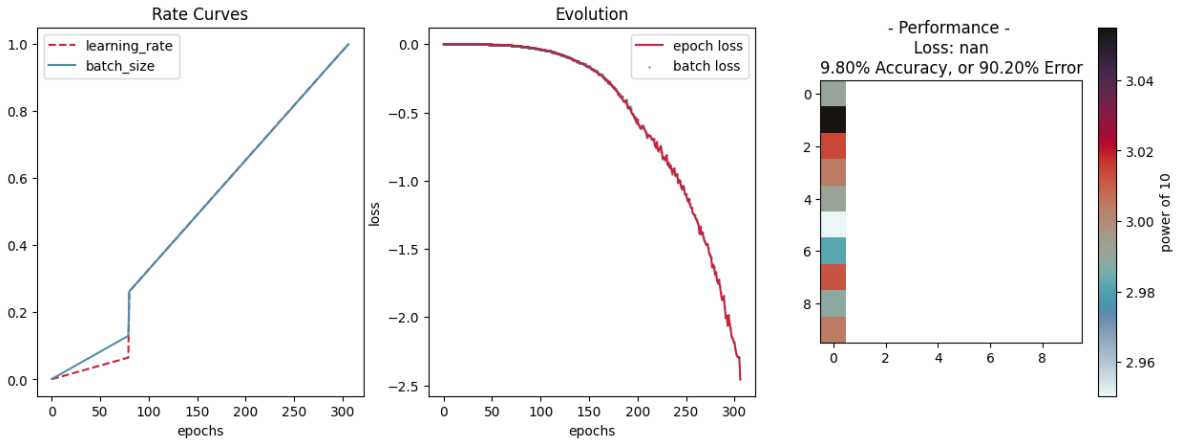
(a) adj_FFA1_0



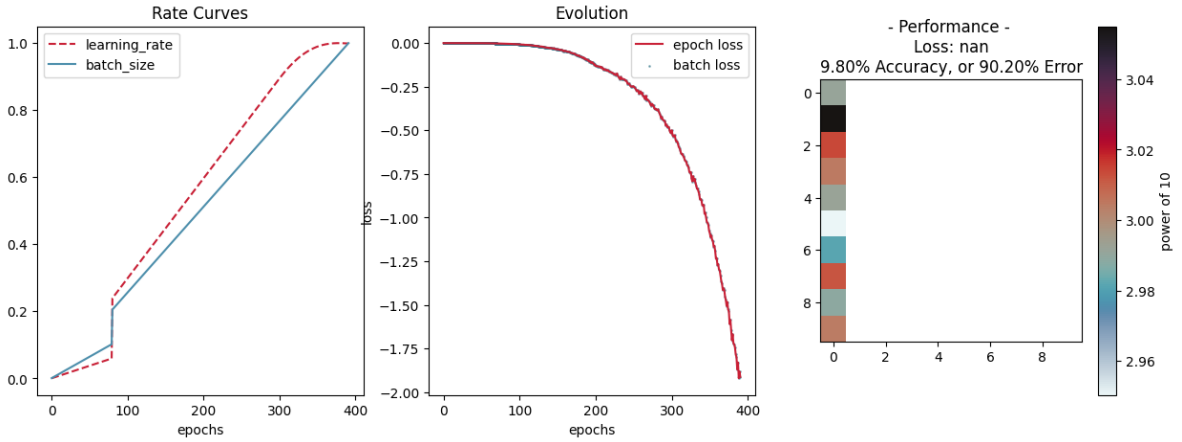
(a) adj_FFA1_1



(a) adj_FFA1_2



(a) adj_FFA0_2



(a) adj_FFA1_3