Supplementary Material

In this section, we provide a detailed description of how to implement Friendly Training (FT) of (Marullo et al. 2021), that is based on an approach that is directly inherited from Adversarial Training (AT) (Goodfellow, Shlens, and Szegedy 2015; Madry et al. 2018) and, more specifically, from the so-called Friendly Adversarial Training (FAT) (Zhang et al. 2020). Several of the following details are redundant for readers that are already aware of AT, but the description also specifically formalizes how to implement the developmental plan required by FT. What follows is an excerpt from (Marullo et al. 2021), with minor notation changes to make it coherent with the paper associated to this supplementary material.

Each example x_i is altered to $\tilde{x}_i$ by adding a specific perturbation δ_i,

$$\tilde{x}_i = x_i + \delta_i, \tag{6}$$

where $\delta_i \in \mathbb{R}^d$. Given a mini-batch $\mathcal{B}$, we denote with Δ the matrix that collects the perturbations associated to the mini-batch examples. In detail, the i-th row of Δ is the perturbation δ_i associated with the i-th example in $\mathcal{B}$. For convenience in the notation, we avoid mentioning training epochs in what follows, and we describe the training procedure as the iterative processing of mini-batches of data, updating w after each of them. We re-define the aggregated loss L by providing the network with $\tilde{x}$ instead of x, and introducing the dependency on Δ,

$$L(\mathcal{B}, \Delta, w) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \ell\left(f\left(\tilde{x}_i, w\right), y_i\right), \tag{7}$$

where $\tilde{x}_i$ is defined as in Eq. (6), $(x_i, y_i) \in \mathcal{B}$ (i.e., $\mathcal{B}$ is the mini-batch at iteration γ). Jointly optimizing Eq. (7) with respect to Δ and w allows the network not only to adapt its weights and biases in order to better cope with the learning criterion, but also to alter the data in $\mathcal{B}$ by translating them in those space regions that can be more easily classified. However, the loss of Eq. (7) does not introduce any constraints on each δ_i and, differently from w, the set Δ is specifically associated with the data in $\mathcal{B}$ (i.e, each training example is associated with its own perturbation), meaning that the number of variables of the optimization problem becomes a function of the size of the training data.

For these reasons, Eq. (7) is framed in the context of a developmental plan. In detail, we consider a single matrix Δ that is shared by all the mini-batches (the total number of rows in Δ is equal to the size of a single mini-batch). At the beginning of each training iteration, we keep w fixed, we initialize Δ to zeros and then we estimate the appropriate perturbations for the current data in $\mathcal{B}$ by gradient descent over the variable Δ (second argument of Eq. (7)). We denote with τ the number of iterative steps of such inner optimization. The value of τ controls the extent of data alteration. For small values of τ the network will only marginally simplify the data, while for a larger τ data alteration will be more aggressive. After having transformed/simplified the data of the mini-batch, we forward them to the classifier and we update the values of w minimizing the loss.

Algorithm 2: Friendly Training.

Input: Training set $\mathcal{X}$, initial weights and biases w, batch size b, max FT steps γ_{max}, max simplification steps ξ_{max_simp}, $\tau^1 \geq 1$, learning rates $\alpha > 0$ and $\beta > 0$, shared matrix Δ with b rows and d columns.
Output: The final w.

1: $\xi \leftarrow 0$
2: **for** $\gamma = 1$ to γ_{max} **do**
3: Sample a set of mini-batches, $B = \{\mathcal{B}_z\}$, from $\mathcal{X}$
4: **for** each mini-batch $\mathcal{B}_z \in B$ **do**
5: $\xi \leftarrow \xi + 1$
6: Compute τ following Eq. (8)
7: Set all the entries of Δ to 0
8: **for** $i = 1$ to τ **do**
9: Compute $\Delta_{grad} = \frac{\partial L(\mathcal{B}_z, D, w)}{\partial D}\Big|_{D=\Delta}$, see Eq. (7)
10: $\Delta = \Delta - \beta \cdot \Delta_{grad}$
11: **end for**
12: Compute $w_{grad} = \frac{\partial L(\mathcal{B}_z, \Delta, v)}{\partial v}\Big|_{v=w}$, see Eq. (7)
13: $w = w - \alpha \cdot w_{grad}$
14: **end for**
15: **end for**
16: **return** w

The training procedure is detailed in Algorithm 2, and further details are provided in the following lines. Notice that while the weight update equation (line 13) can include any existing adaptive learning rate estimation procedure, in our current implementation Δ is updated by a fixed small learning rate (line 10). While Algorithm 2 formally returns the weights after having completed the last training iteration, as usual, the best configuration of the classifier can be selected by measuring the performance on a validation set, when available. Another important fact to mention is that when the prediction on examples perfectly matches target, line 9 will return zero gradient, hence no simplifications are performed. In our implementation, following (Marullo et al. 2021), we slightly relaxed this condition by zeroing the rows of Δ_{grad} (line 9) associated to those examples that are classified with a large confidence above a given threshold c, that implements a selective early-stopping condition on the inner optimization.

In order to determine the appropriate number of inner optimization steps, the initial value $\tau^1 \geq 1$ is a fixed hyperparameter, while a quadratic law progressively reduces τ in function of ξ, that is the total number of updates of w,

$$\tau = \tau^1 \cdot \max\left(1 - \frac{\xi - 1}{\xi_{max\_simp}}, 0\right)^2. \tag{8}$$

The choice of determining τ as a function of ξ and not of γ is due to the dynamics of this implementation of FT, in which the classifier must be updated right after having computed a simplified batch of data (otherwise we would need to store all the simplified batches and then pass them to the classifier).

Experiments and Parameters. In the experiments of the main paper, we denoted the original Friendly Training implementation with FT and we determined the optimal parameters by running grid search. Concerning Advanced digit and shape recognition, Sentiment analysis and the CIFAR10 experiment, we considered the following grid of values: $\beta \in \{0.01, 0.1, 1.0, 5.0, 10.0\}$, $\tau^1 \in [10, 80, 120]$, $c \in [0.90, 0.95, 0.98]$, $\xi_{max\_simp} \in \{0.25 \cdot \gamma_{max}|B|, 0.5 \cdot \gamma_{max}|B|, 0.85 \cdot \gamma_{max}|B|\}$. For each γ, we sampled a set B of mini-batches composed of non-overlapping batches that, overall, cover the whole datasets. Concerning the CIFAR10-N10 experiment, the grid was about the following ranges: $\beta \in \{0.001, 0.01, 0.1, 1.0, 5.0, 10.0\}$, $\tau^1 \in [5, 10, 40, 80, 120]$, $c \in [0.85, 0.90, 0.95, 0.98]$, $\xi_{max\_simp} \in \{0.1 \cdot \gamma_{max}|B|, 0.25 \cdot \gamma_{max}|B|, 0.5 \cdot \gamma_{max}|B|, 0.85 \cdot \gamma_{max}|B|\}$. In all the experiments, for each γ, we sampled a set B of mini-batches composed of non-overlapping batches that, overall, cover the whole dataset, while γ_{max} was set as in the case of NFT. We also considered the case in which the 1-hot target was concatenated to the input of the auxiliary network, thus conditioning its prediction to the class of the provided example (Mirza and Osindero 2014). Target conditioning can facilitate the learning of more specific transformations in the case of similar examples that belong to different classes.

We implemented models and ran experiments with PyTorch 1.7.1. In all the experiments the seeds for random initialization are 2, 3, 4 (`torch.manual_seed` in the PyTorch library). See the code archive for all the remaining details (the reader can find corresponding command lines in the `scripts` folder).

Optimal parameters We report in Table 3 and Table 4 the hyperparameters corresponding to the best models in the case of FT and NFT, respectively. Models are chosen according to the validation error.

	Parameter	mn-back	mn-rot-back	mn-rot	rectangles	convex	wines	imdb	cifar10	cifar10-n10
FC-A	$\frac{\xi_{max\_simp}}{\gamma_{max}\lvert B\rvert}$	0.25	0.5	0.5	0.25	0.25	—	—	—	—
	τ^1	10	80	10	10	120	—	—	—	—
	β	10	0.01	0.01	5	5	—	—	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	—	—
	c	0.98	0.95	0.98	0.98	0.98	—	—	—	—
FC-B	$\frac{\xi_{max\_simp}}{\gamma_{max}\lvert B\rvert}$	0.25	0.5	0.5	0.25	0.5	0.5	0.25	—	—
	τ^1	10	10	80	120	10	80	120	—	—
	β	0.01	0.01	1	10	0.01	10	10	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	—	—
	c	0.95	0.95	0.98	0.98	0.95	0.98	0.9	—	—
CNN-A	$\frac{\xi_{max\_simp}}{\gamma_{max}\lvert B\rvert}$	0.5	0.25	0.25	0.5	0.85	—	—	—	—
	τ^1	120	10	80	10	80	—	—	—	—
	β	0.1	0.1	0.01	0.1	5	—	—	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	—	—
	c	0.98	0.98	0.9	0.98	0.98	—	—	—	—
CNN-B	$\frac{\xi_{max\_simp}}{\gamma_{max}\lvert B\rvert}$	0.5	0.5	0.25	0.85	0.25	—	—	0.25	—
	τ^1	80	120	120	10	10	—	—	10	—
	β	0.01	0.01	1	1	0.1	—	—	0.01	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	0.0001	—
	c	0.9	0.95	0.98	0.9	0.9	—	—	0.98	—
ResNet	$\frac{\xi_{max\_simp}}{\gamma_{max}\lvert B\rvert}$	—	—	—	—	—	—	—	—	0.1
	τ^1	—	—	—	—	—	—	—	—	10
	β	—	—	—	—	—	—	—	—	10
	α	—	—	—	—	—	—	—	—	0.0001
	c	—	—	—	—	—	—	—	—	0.85

Table 3: FT optimal parameters. Selected hyperparameters in the case of FT, drawn from the grid search described in the main text.

	Parameter	mn-back	mn-rot-back	mn-rot	rectangles	convex	wines	imdb	cifar10	cifar10-n10
FC-A	$\frac{\gamma_{max\_simp}}{\gamma_{max}}$	0.25	0.25	0.25	0.85	0.85	—	—	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	—	—
	β	0.0001	1e-05	1e-05	1e-05	0.0005	—	—	—	—
	η_{max}	2000	2000	2000	500	2000	—	—	—	—
	n_f	64	64	64	128	128	—	—	—	—
	ν	4	4	4	4	4	—	—	—	—
FC-B	$\frac{\gamma_{max\_simp}}{\gamma_{max}}$	0.5	0.5	0.5	0.5	0.25	0.5	0.5	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	—	—
	β	0.0001	1e-05	0.0005	0.0001	1e-05	1e-05	0.0001	—	—
	η_{max}	1000	1000	500	1000	500	500	1000	—	—
	n_f	96	128	128	64	128	n.a.	n.a.	—	—
	ν	4	4	4	4	4	n.a.	n.a.	—	—
CNN-A	$\frac{\gamma_{max\_simp}}{\gamma_{max}}$	0.85	0.25	0.25	0.5	0.25	—	—	—	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	—	—
	β	1e-05	1e-05	0.0005	0.0001	1e-05	—	—	—	—
	η_{max}	500	2000	1000	500	2000	—	—	—	—
	n_f	96	96	96	96	128	—	—	—	—
	ν	4	4	4	4	4	—	—	—	—
CNN-B	$\frac{\gamma_{max\_simp}}{\gamma_{max}}$	0.25	0.25	0.25	0.25	0.25	—	—	0.85	—
	α	0.0001	0.0001	0.0001	0.0001	0.0001	—	—	0.0001	—
	β	1e-05	0.0005	0.0001	0.0001	0.0005	—	—	1e-05	—
	η_{max}	500	2000	2000	1000	1000	—	—	500	—
	n_f	64	64	128	64	128	—	—	64	—
	ν	4	4	4	4	4	—	—	2	—
ResNet	$\frac{\gamma_{max\_simp}}{\gamma_{max}}$	—	—	—	—	—	—	—	—	0.7
	α	—	—	—	—	—	—	—	—	0.1
	β	—	—	—	—	—	—	—	—	0.0005
	η_{max}	—	—	—	—	—	—	—	—	1000
	n_f	—	—	—	—	—	—	—	—	96
	ν	—	—	—	—	—	—	—	—	2

Table 4: NFT optimal parameters. Selected hyperparameters in the case of NFT, drawn from the grid search described in the main text (see Section 3).