## Università degli Studi di Napoli Federico II Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione



## Neuro BackPropagation Lab

Giuliano Aiello

2025

# Contents

]		lusion																							
	1.1	Goal																							
			re Stack																						
	1.3	Projec	t Struct	ure																					
: ]	Resi	ilient I	Backpro	pa	ga	tic	on	$\mathbf{r}$	ec	h	ni	qι	ıes	8											
2.	2.1	Modul	e Overv	iew												 									
		2.1.1	Model																						
		2.1.2	Tester													 									
		2.1.3	Trainer													 									
		2.1.4	Utils .													 									
4	2.2	Impler	nentatio	ns												 									
		2.2.1	Rprop-													 									
		2.2.2	Rprop-	⊦ .												 									

iv CONTENTS

# Chapter 1

# **Prolusion**

#### 1.1 Goal

This report provides a comprehensive overview of a Python project whose goal is to develop and compare different adaptive backpropagation techniques involved in a machine learning process, as Rprop (Resilient BackPropagation).

The project follows the "Empirical evaluation of the improved Rprop learning algorithms" article by Christian Igel and Michel Hüsken (2001).

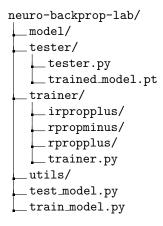
#### 1.2 Software Stack

- Python 3.9.6
- $\bullet$  PyTorch 2.6.0

The project is equipped with a requirements.txt file which allows for seamless installation of dependencies, by executing pip install -r requirements.txt.

3

### 1.3 Project Structure



- model includes the neural network model architecture.
- tester handles the testing flow of the ready-to-use trained\_model.pt.
- trainer handles the examined backpropagation techniques and the training flow of the model, saving it as trained\_model.pt.
- $\bullet$  utils offers utility functions designed to support the root project scripts.

## Chapter 2

# Resilient Backpropagation Techniques

Rprop algorithms differ from the classical back-propagation algorithms by the fact that they are independent of the magnitude of the gradient, but only depend on its sign.

#### 2.1 Module Overview

The part of the project which is shared across all Rprop techniques is presented follows.

#### 2.1.1 Model

This class represents the artificial neural network model architecture to be trained and tested.

It is based on torch.nn.Module<sup>1</sup> class. Layers are fully connected using torch.nn.Linear(.) and the network feeds forward as follows:

- 1. the first layer flattens the input MNIST image, by transforming it from a multidimensional vector to a 784-sized (since a  $28 \times 28$  sized image is manipulated) one-dimensional vector;
- the hidden layer receives the transformed vector and processes it into a 128-sized vector with a ReLU activation function to introduce non-linearity, a choice that was the result of empirical experiments;
- 3. the output layer extracts the final predictions by transforming the 128-sized vector into a 10-sized vector, which corresponds to the number of possible classes for classification.

<sup>&</sup>lt;sup>1</sup>https://pytorch.org/docs/stable/generated/torch.nn.Module.html, 2025

- 2.1.2 Tester
- 2.1.3 Trainer
- 2.1.4 Utils

## 2.2 Implementations

### 2.2.1 Rprop-

This is Rprop-.

#### 2.2.2 Rprop+

This is Rprop+.

## 2.3 Comparisons

Here I will show comparisons.

2.3. COMPARISONS 7

# Acronyms

 $\mathbf{MNIST}\,$  Modified National Institute of Standards and Technology database 5

 ${f ReLU}$  Rectified Linear Unit 5

**Rprop** Resilient BackPropagation 1, 5