University of St.Gallen

A Convexity-dependent Two-Phase Training Algorithm for Deep Neural Networks

KDIR Conference 2025 in Marbella, Spain, 22–24 October
Presentation slot: KDIR25-6A

Tomas Hrycej, Bernhard Bermeitinger, Massimo Pavone, Götz-Henrik Wiegand and Siegfried Handschuh

From insight to impact.

What if…

…the loss landscape isn't just a chaotic non-convex jungle,

but instead, has an underlying structure that we can exploit?

?

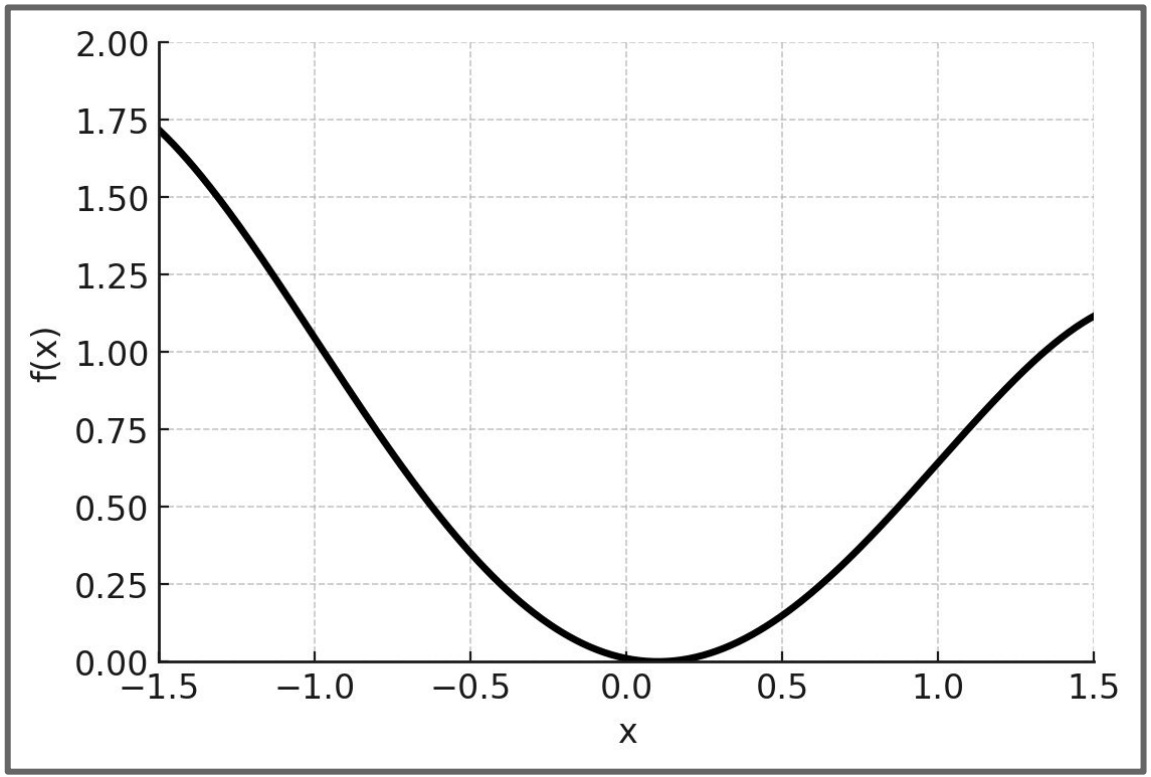
Theoretical Foundation

The convexity of the loss!

The Non-Convex Loss Landscape

- Example Function:

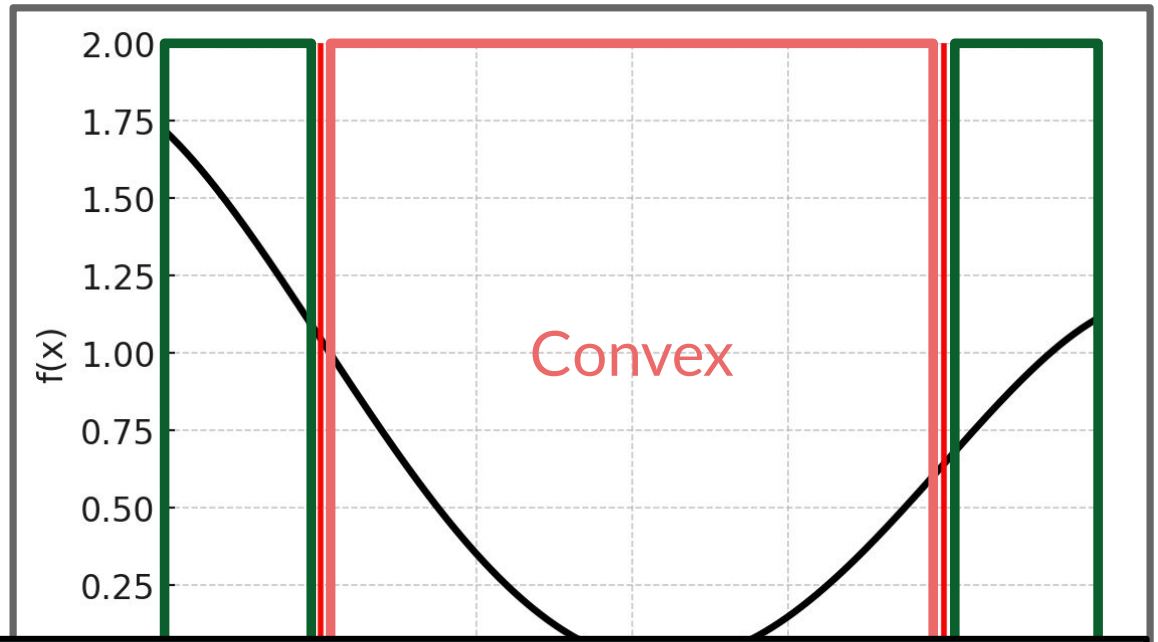
$$f(x) = \left(-\frac{1}{6}x^4\right) + (x - 0.1)^2$$



The (Non-)Convex Loss Landscape

We can divide in two regions:

- Non-Convex Region ( )
- Convex Region ( )



Mathematical Assumption: Near the minimum of a smooth non-convex function, the region is always locally convex.

Optimization Algorithms

There is more than Adam.

First- and Second-Order Optimization Algorithms

First-Order Methods
(Adam, SGD, …)

- Good for non-convex regions with complex loss landscape
- Suboptimal in convex regions
- Computational efficient per step

Second-Order Methods
(Quasi-Newton, Conjugate Gradient, …)

- Suboptimal in non-convex regions
- Optimal for convex problems (Superlinear Convergence)
- More expensive per step than Adam/SGD

Journal of Research of the National Bureau of Standards Vol. 49, No. 6, December 1952 Research Paper 2379

Methods of Conjugate Gradients for Solving Linear Systems[1]

Magnus R. Hestenes [2] and Eduard Stiefel [3]

An iterative algorithm is given for solving a system $Ax=k$ of n linear equations in n unknowns. The solution is given in n steps. It is shown that this method is a special case of a very general method which also includes Gaussian elimination. These general algorithms are essentially algorithms for finding an n dimensional ellipsoid. Connections are made with the theory of orthogonal polynomials and continued fractions.

1. Introduction

One of the major problems in machine computations is to find an effective method of solving a system of n simultaneous equations in n unknowns, particularly if n is large. There is, of course, no best method for all problems because the goodness of a method depends to some extent upon the particular system to be solved. In judging the goodness of a method for machine computations, one should bear in mind that criteria for a good machine method may be different from those for a hand method. By a hand method, we shall mean one in which a desk calculator may be used. By a machine method, we shall mean one in which sequence-controlled machines are used.

method; (b) the conjugate gradient method presented in the present monograph.

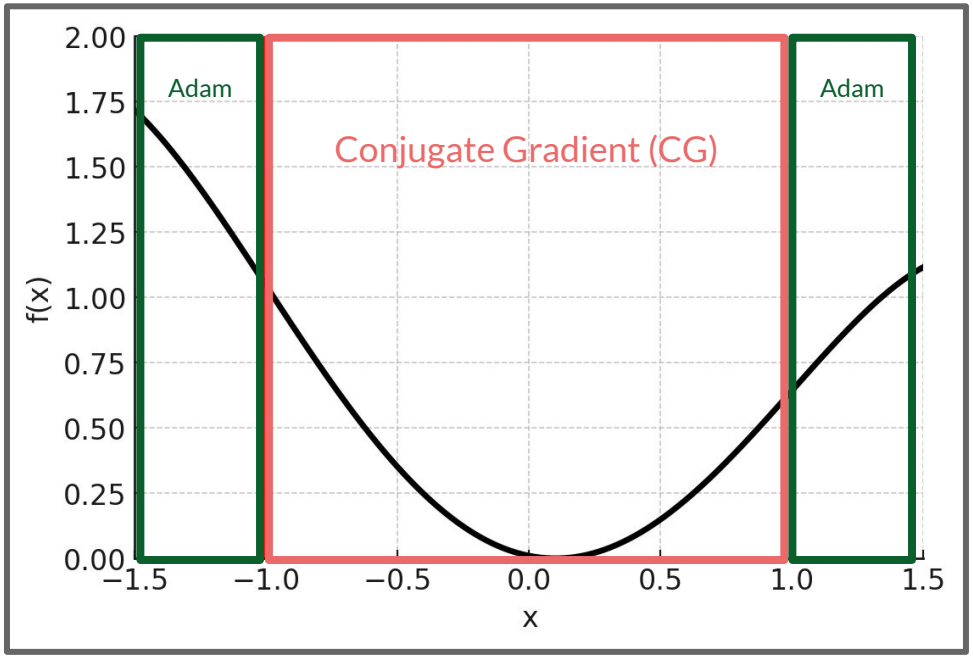
There are many variations of the elimination method, just as there are many variations of the conjugate gradient method here presented. In the present paper it will be shown that both methods are special cases of a method that we call the method of conjugate directions. This enables one to compare the two methods from a theoretical point of view.

In our opinion, the conjugate gradient method is superior to the elimination method as a machine method. Our reasons can be stated as follows:

(a) Like the Gauss elimination method, the method of conjugate gradients gives the solution in n steps if no rounding-off error occurs.

When to use what tool?

- Non-Convex
  - First-Order Optimizer
  - → Adam
- Convex
  - Second-Order Optimizer
  - → Conjugate Gradient (CG)

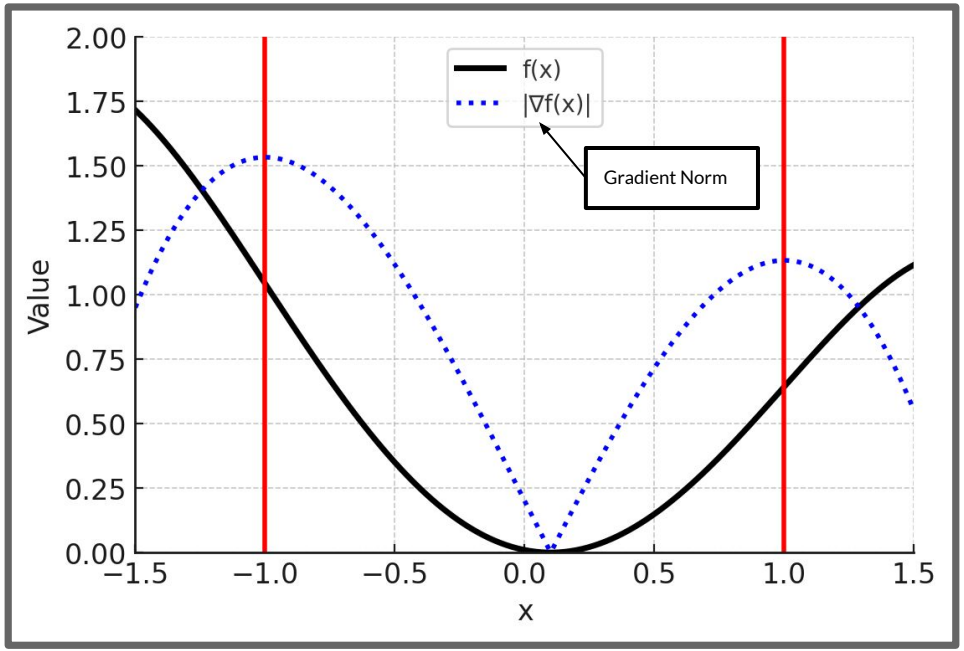


But when do we know when to switch?

Detecting the Transition

What the gradient norm tells us!

The Gradient Norm



The gradient norm behaviour reveals the geometry of the loss landscape: Convex or Non-Convex

Gradient Norm switching Idea

Non-Convex Region

Optimizer: Adam

Increasing Gradient norm

Transition Point

Gradient norm maximum

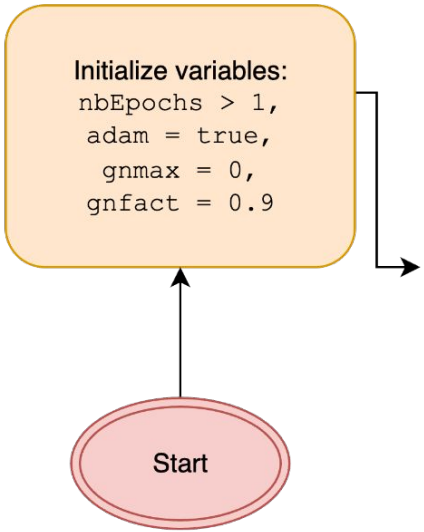
Convex Region

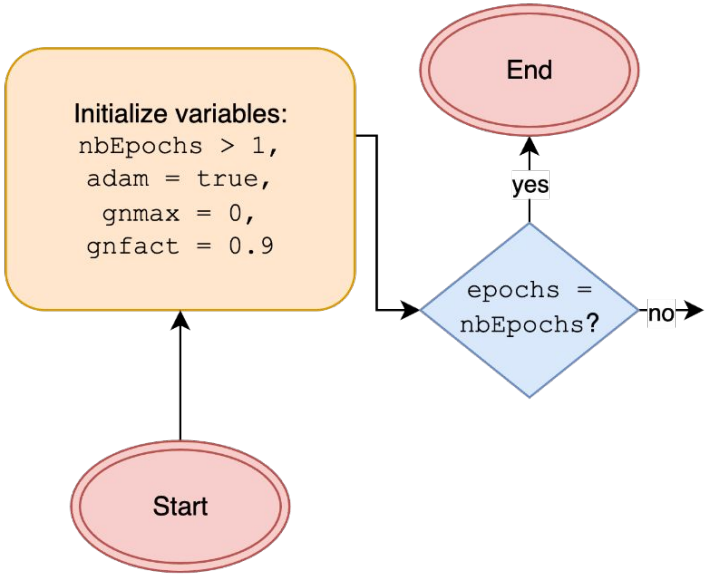
Optimizer: Conjugate Gradient

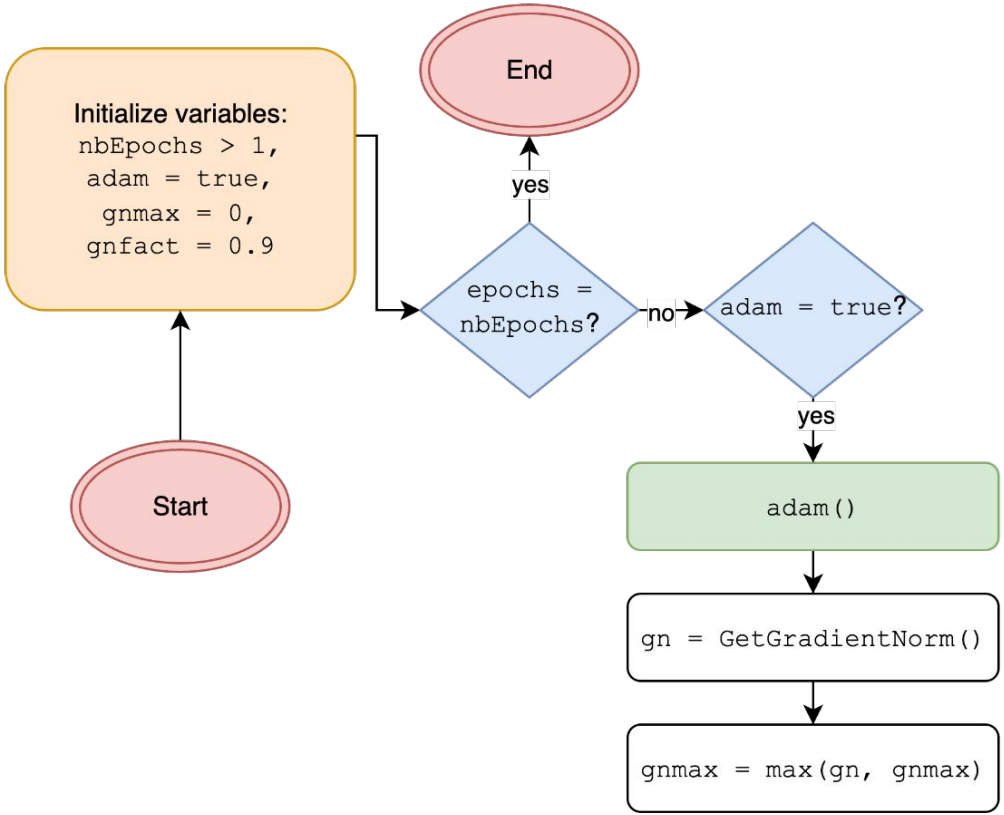
Decreasing Gradient norm

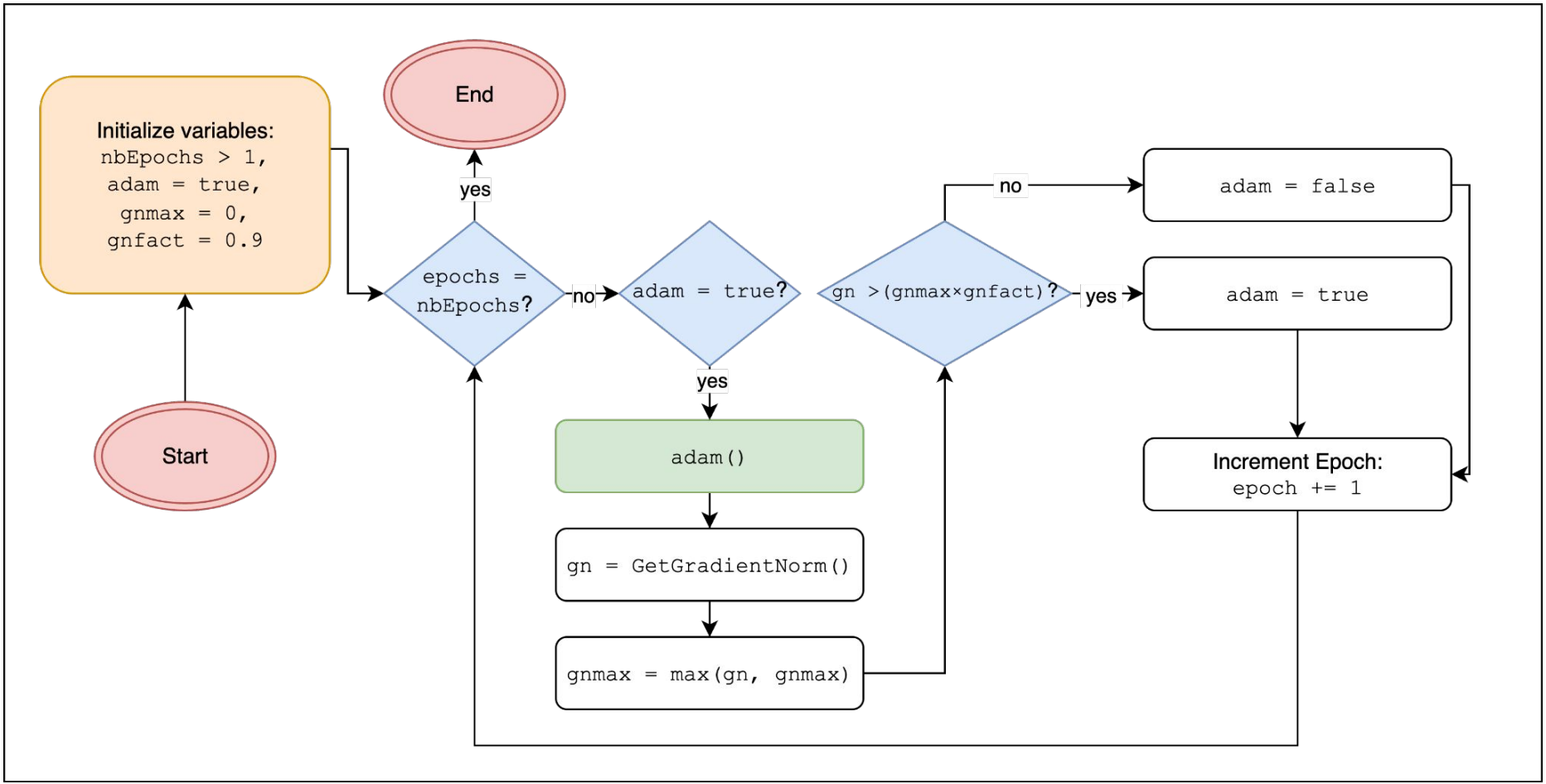
The Two-Phase Algorithm

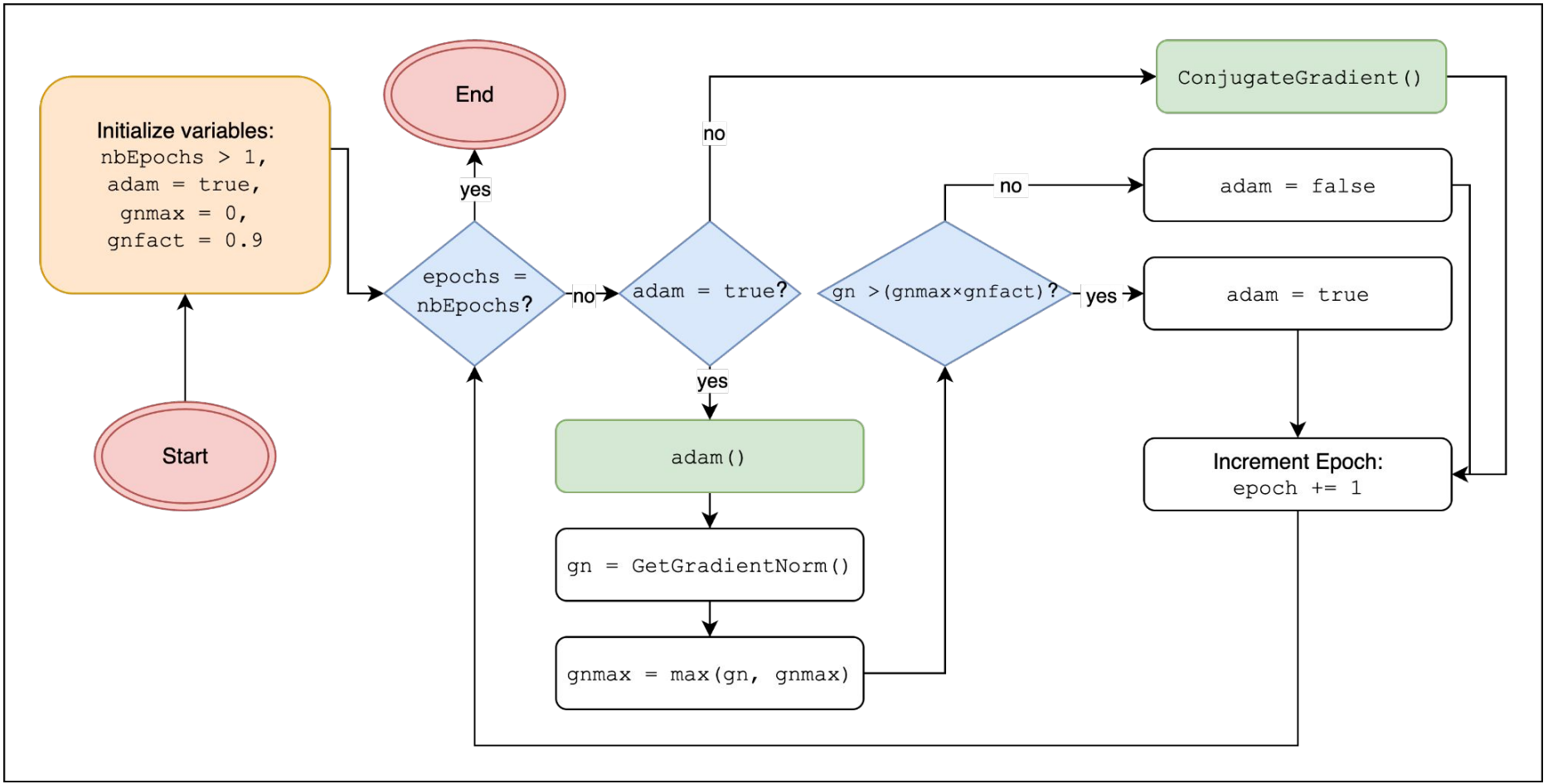
Theory meets practice!


Initialize variables:
nbEpochs > 1,
adam = true,
gnmax = 0,
gnfact = 0.9
Start


End
Initialize variables:
nbEpochs > 1,
adam = true,
gnmax = 0,
gnfact = 0.9
yes
epochs =
nbEpochs?
no
Start


End
Initialize variables:
nbEpochs > 1,
adam = true,
gnmax = 0,
gnfact = 0.9
yes
epochs =
nbEpochs?
no
adam = true?
yes
Start
adam()
gn = GetGradientNorm()
gnmax = max(gn, gnmax)


Initialize variables:
nbEpochs > 1,
adam = true,
gnmax = 0,
gnfact = 0.9
Start
End
yes
epochs = nbEpochs?
no
adam = true?
yes
adam()
gn = GetGradientNorm()
gnmax = max(gn, gnmax)
gn >(gnmax×gnfact)?
no
adam = false
yes
adam = true
Increment Epoch:
epoch += 1


Initialize variables:
nbEpochs > 1,
adam = true,
gnmax = 0,
gnfact = 0.9
Start
End
epochs = nbEpochs?
yes
no
adam = true?
no
yes
ConjugateGradient()
gn >(gnmax×gnfact)?
no
yes
adam = false
adam = true
adam()
gn = GetGradientNorm()
gnmax = max(gn, gnmax)
Increment Epoch:
epoch += 1

The Two-Phase Algorithm

- Theoretically switching at the peak

How do we know if we are over the peak?

- Practically, we define switching:
  - 90% of the maximum

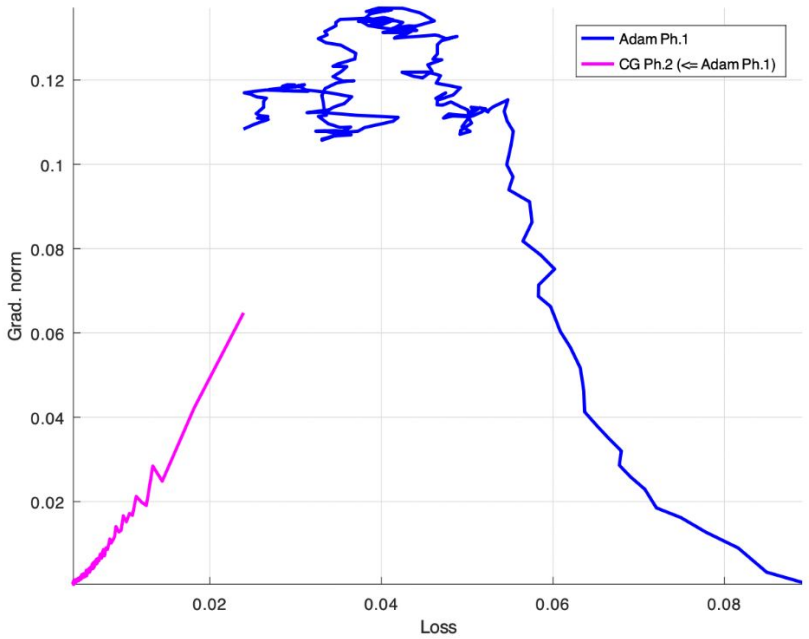
Data: nbEpochs > 1
$adam \leftarrow true$;
$gnmax \leftarrow 0$;
$gnfact \leftarrow 0.9$;
for $epoch \leftarrow 1$ **to** $nbEpochs$ **do**
  if $adam$ **then**
    ADAM();
    $gn \leftarrow$ GETGRADIENTNORM();
    $gnmax \leftarrow \mathbf{max}(gn, gnmax)$;
    $adam \leftarrow gn > (gnmax * gnfact)$;
  else
    CONJUGATEGRADIENT();
  end
end

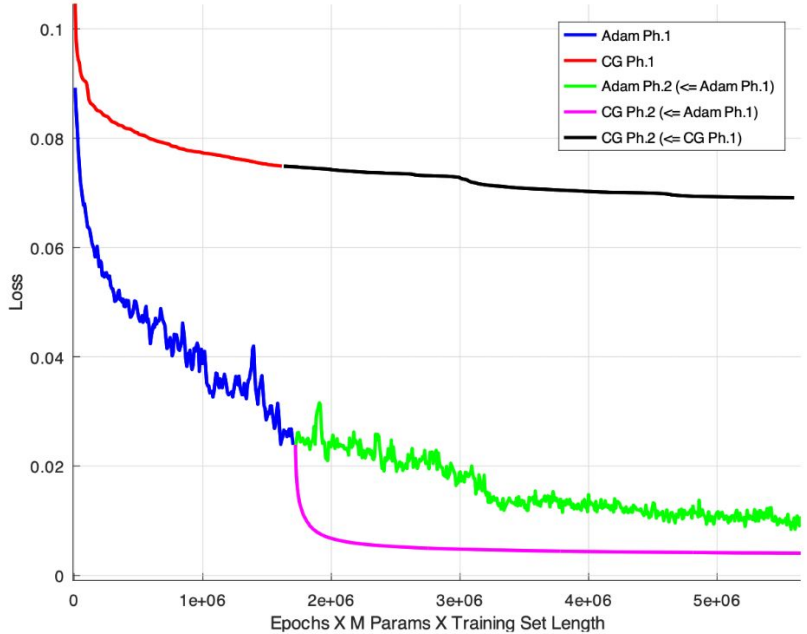
Algorithm 1: Two-Phase Algorithm to switch from Adam to CG when the gradient norm peak has reached. Model and data are left out for brevity.

Experimental Results

Does it work?

Training on Vision Transformer (VIT) - CIFAR-10





Experimental Results

Model	Dataset	Val. Loss (Adam)↓	Val. Loss (Adam+CG)↓	Val. Accuracy (Adam)↑	Val. Accuracy (Adam+CG)↑
VIT[1]	MNIST	0.0061	**0.0044** (27.87 %)	0.965	**0.974** (+0.93 %)
	CIFAR-10	0.0997	**0.0991** (0.60 %)	0.428	**0.435** (+1.64 %)
VGG-5[2]	MNIST	0.0014	**0.0011** (21.43 %)	0.993	**0.994** (+0.10 %)
	CIFAR-10	0.0531	**0.0491** (7.53 %)	0.710	**0.719** (+1.27 %)

Ø 7.2 % lower Val. Loss, Ø 1.9 % better Val. Accuracy and faster convergence!

[1] - Vision Transformer (VIT) from Dosovitsky et al. 2021(https://arxiv.org/abs/2010.11929)
[2] - Smaller variant from VGG-11 Simonyan et al. 2014 (https://arxiv.org/abs/1409.1556)

Take Home Message

1. *Optimization should follow the loss shape.*
2. *The gradient norm tells you when to switch gears.*
3. *Use geometry-aware optimization*

!

University of St.Gallen

Train Smarter — Not Harder!

A Convexity-dependent Two-Phase Training Algorithm for Deep Neural Networks

KDIR 2025: Tomas Hrycej, Bernhard Bermeitinger, Massimo Pavone, Götz-Henrik Wiegand and Prof. Siegfried Handschuh

DS-NLP Research Lab
Chair: Prof. Siegfried Handschuh

Institute for Computer Science, ICS
University of St. Gallen (HSG)

https://info.datascience-nlp.ai/@QDSResearch

Speaker:
M.Sc. Götz-Henrik Wiegand

Institute for Computer Science, ICS
University of St. Gallen (HSG)

https://info.datascience-nlp.ai/@GHWiegand