

physics analysis as a differentiable program

Nathan Simpson, Lukas Heinrich



LUND
UNIVERSITY



neos

Analysis with a *neural network observable*

The difference?

Uses the expected CLs as the loss function

Why haven't we done this before?




Loss functions need to
be differentiable.

Backpropagation \Rightarrow
Chain rule.

$$\frac{\partial \text{loss}}{\partial \varphi} = \frac{\partial \text{loss}}{\partial \text{likelihood}} \times \frac{\partial \text{likelihood}}{\partial \text{model parameters}} \times \frac{\partial \text{model parameters}}{\partial \text{cut values}} \times \dots$$

Every step needs to be
differentiable.

Not possible!

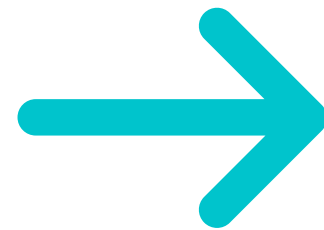
- Histograms 
- Model construction
(HistFactory) 
- Profile likelihood fit 

Until now...

- Histograms

Kernel density estimates!

- Model construction
(HistFactory)



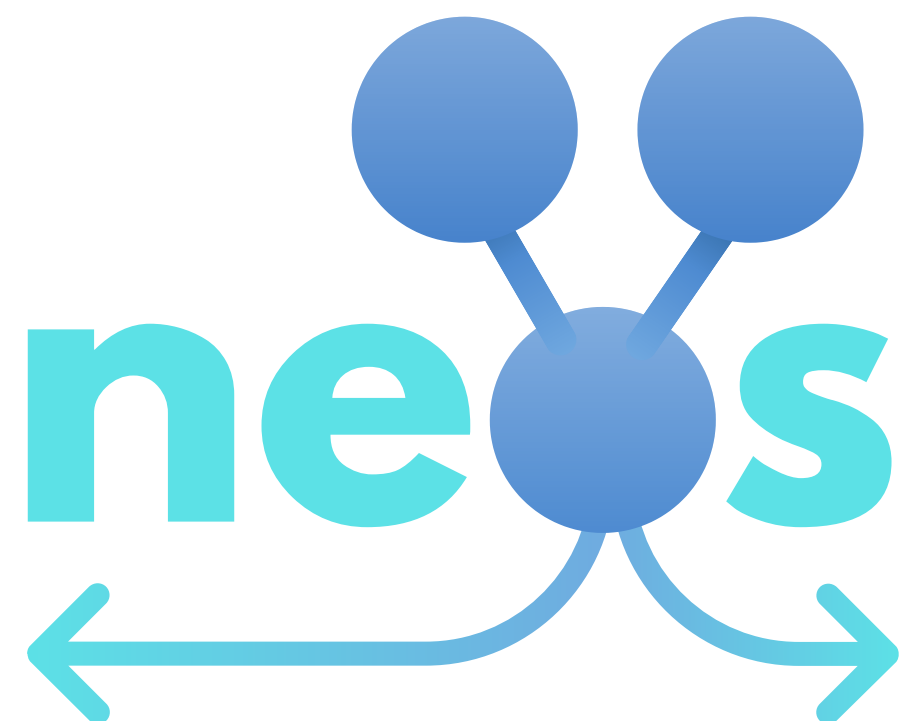
made
differentiable
using

Automatic differentiation
software! (jax + pyhf)

- Profile likelihood fit

Fixed point differentiation!

Mix it all together:



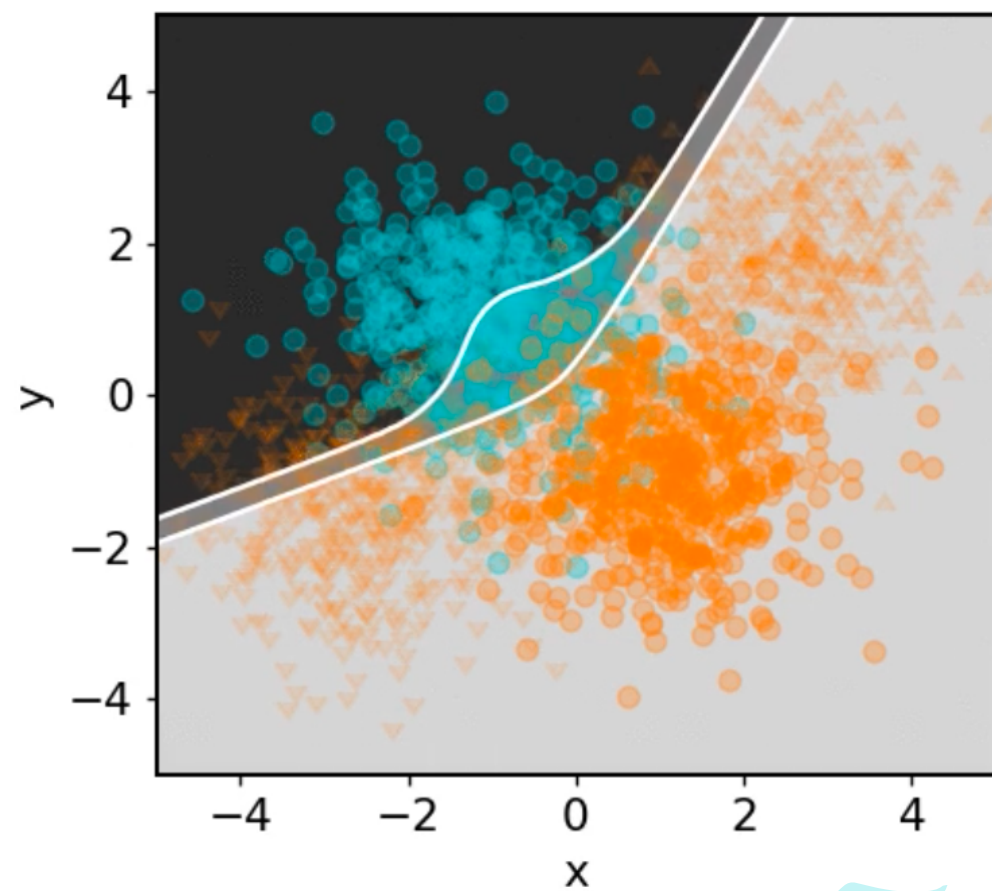
nice

**~~neural~~ end-to-end
optimised statistics**

github.com/gradhep/neos

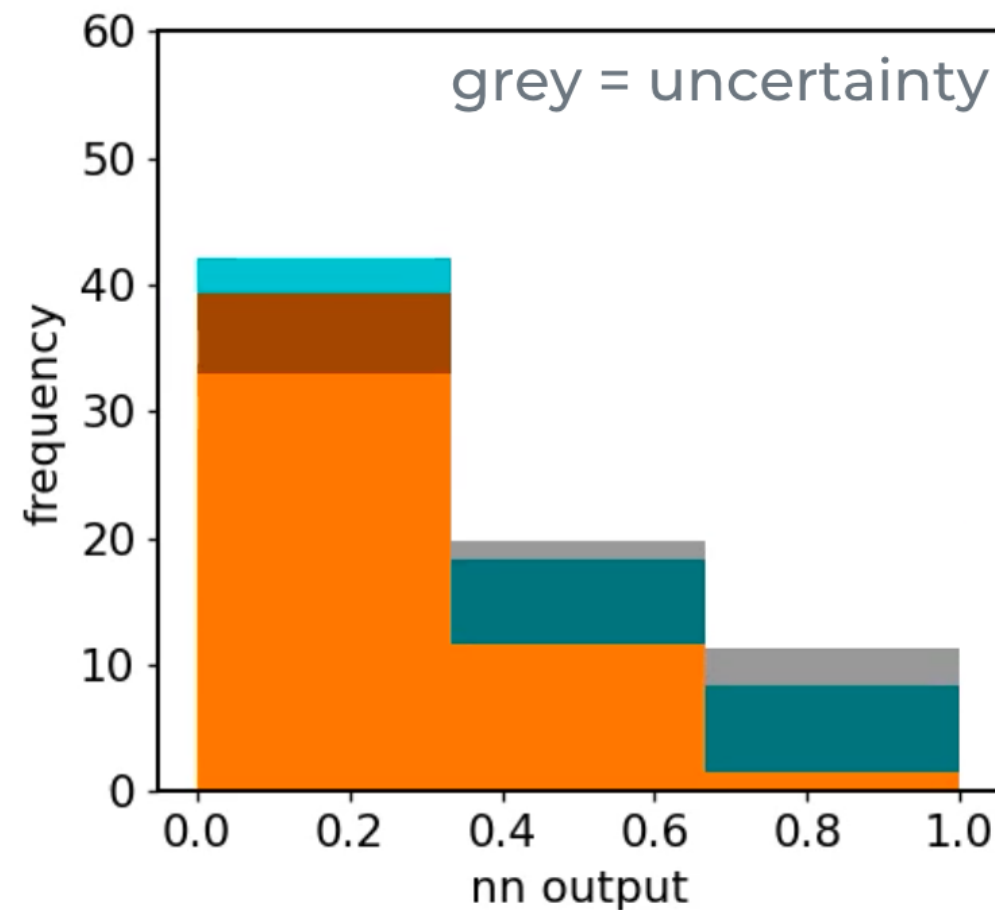
1)

predict one number per event



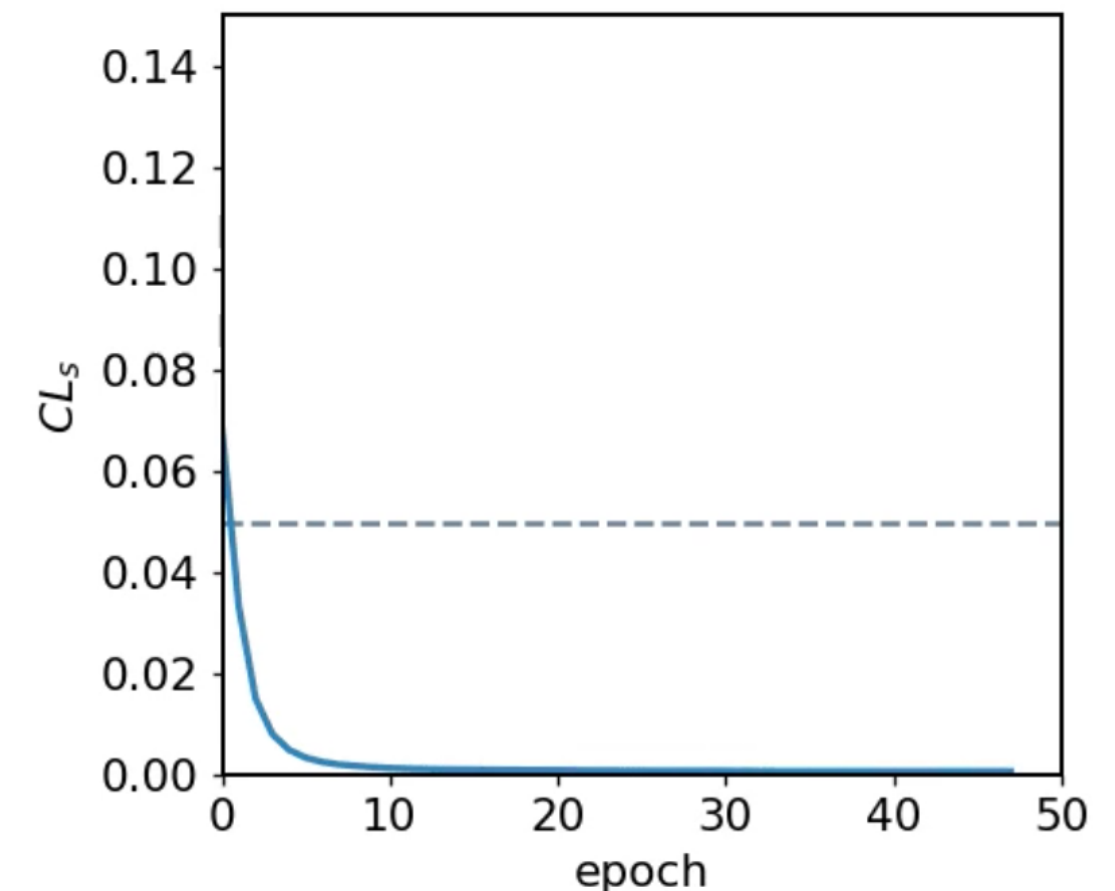
2)

histogram that number over a batch of events



3)

make statistical model + calculate expected CLs



orange = background
either side of bkg = up/down variations
cyan = signal

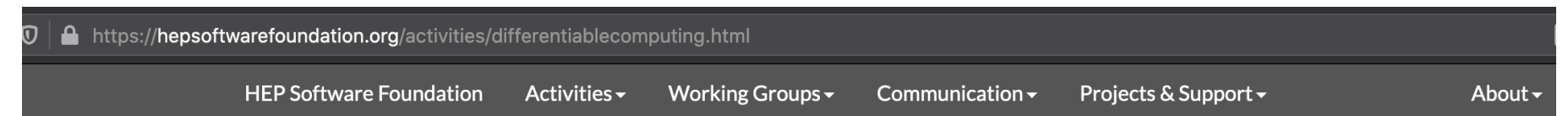
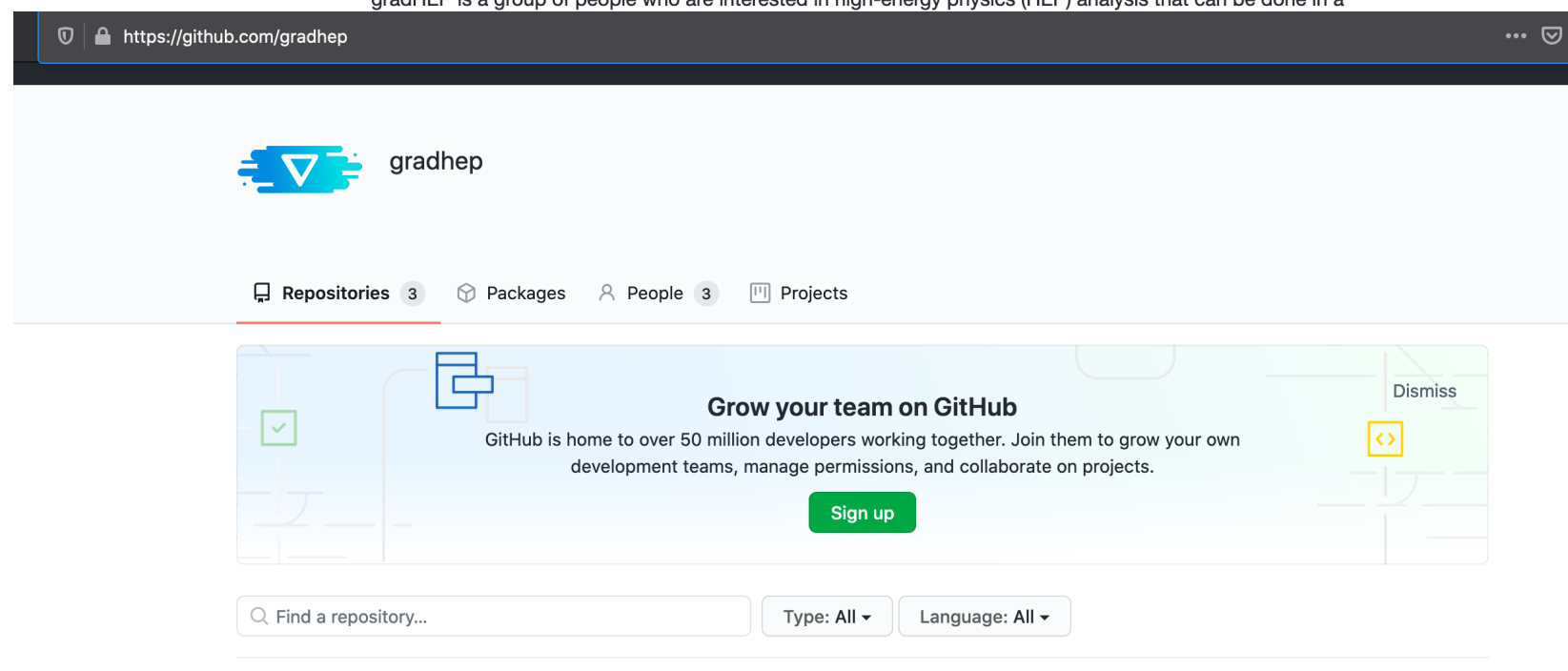
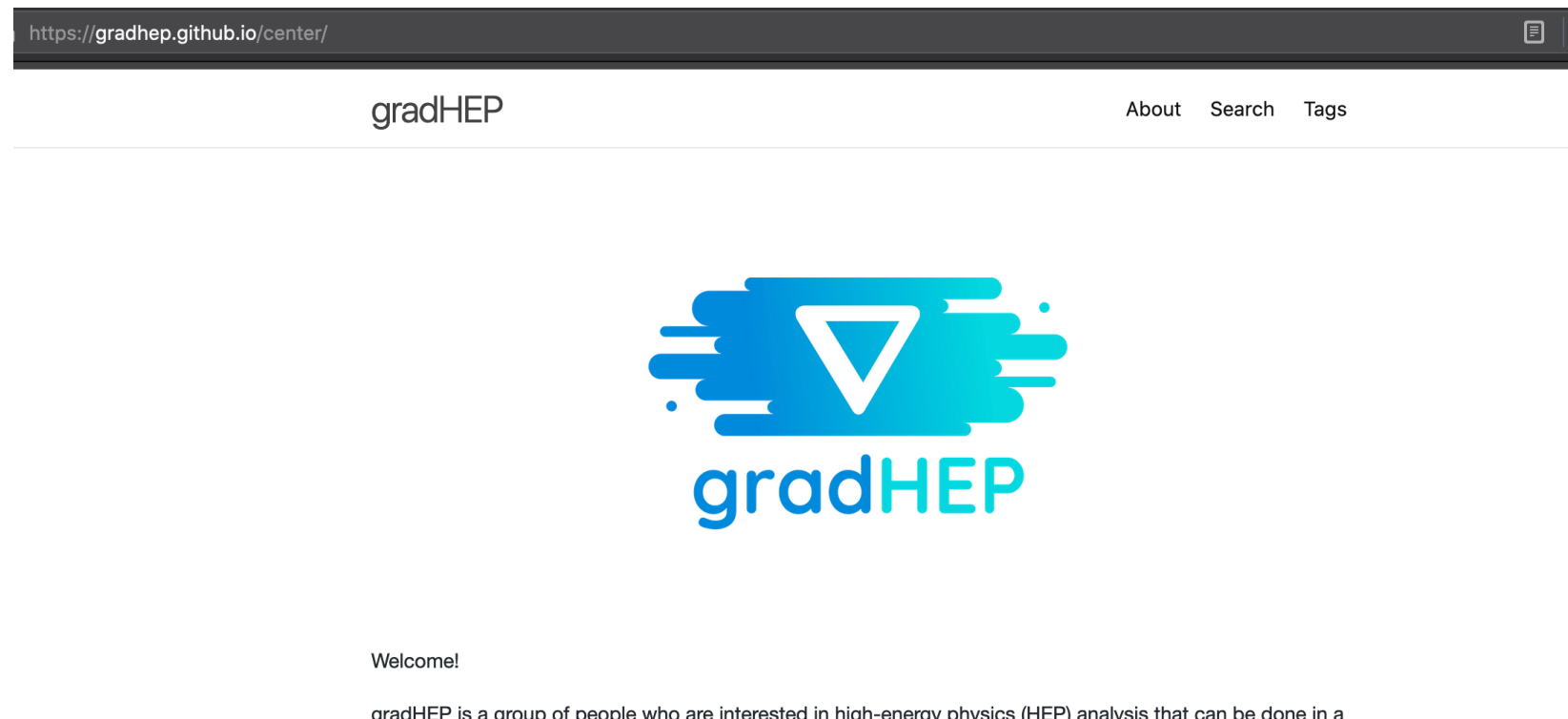
4)

update weights



The future?

A common group has formed around differentiable computing: **gradHEP** -- this is just the beginning :)



Differentiable Computing

Why make things differentiable?

When we write a program to do some physics, that program will likely have some free parameters. These could be as simple as the position of a cut, or as complicated as the parameters of a neural network. In either case, we want these parameters to be optimized such that we get the best possible result, whatever that may be. In the case of a HEP analysis, for instance, this could correspond to having the highest sensitivity to new phenomena.

So, how exactly do we get there?

It turns out that we can do exactly what we do when we train a neural network – update the parameters using gradient descent. This is where the differentiable part comes in: we can only optimize with respect to an objective that has a tractable gradient, since that's how we tell our

Differentiable Programming in High-Energy Physics

Atilım Güneş Baydin (Oxford), Kyle Cranmer (NYU), Matthew Feickert (UIUC),
Lukas Heinrich (CERN), Alexander Held (NYU), Mark Neubauer (UIUC),
Nathan Simpson (Lund), Nick Smith (FermiLab), Giordon Stark (UCSC),
Savannah Thais (Princeton), Gordon Watts (U. Washington)

August 29, 2020

Abstract

A key component to the success of deep learning is the use of gradient-based optimization. Deep learning practitioners compose a variety of modules together to build a complex computational pipeline that may depend on millions or billions of parameters. Differentiating such functions is enabled through a computational technique known as automatic differentiation. The success of deep learning has led to

thanks for listening :)