Instrumenting and Studying Adam and Other Optimization Algorithms in Pytorch

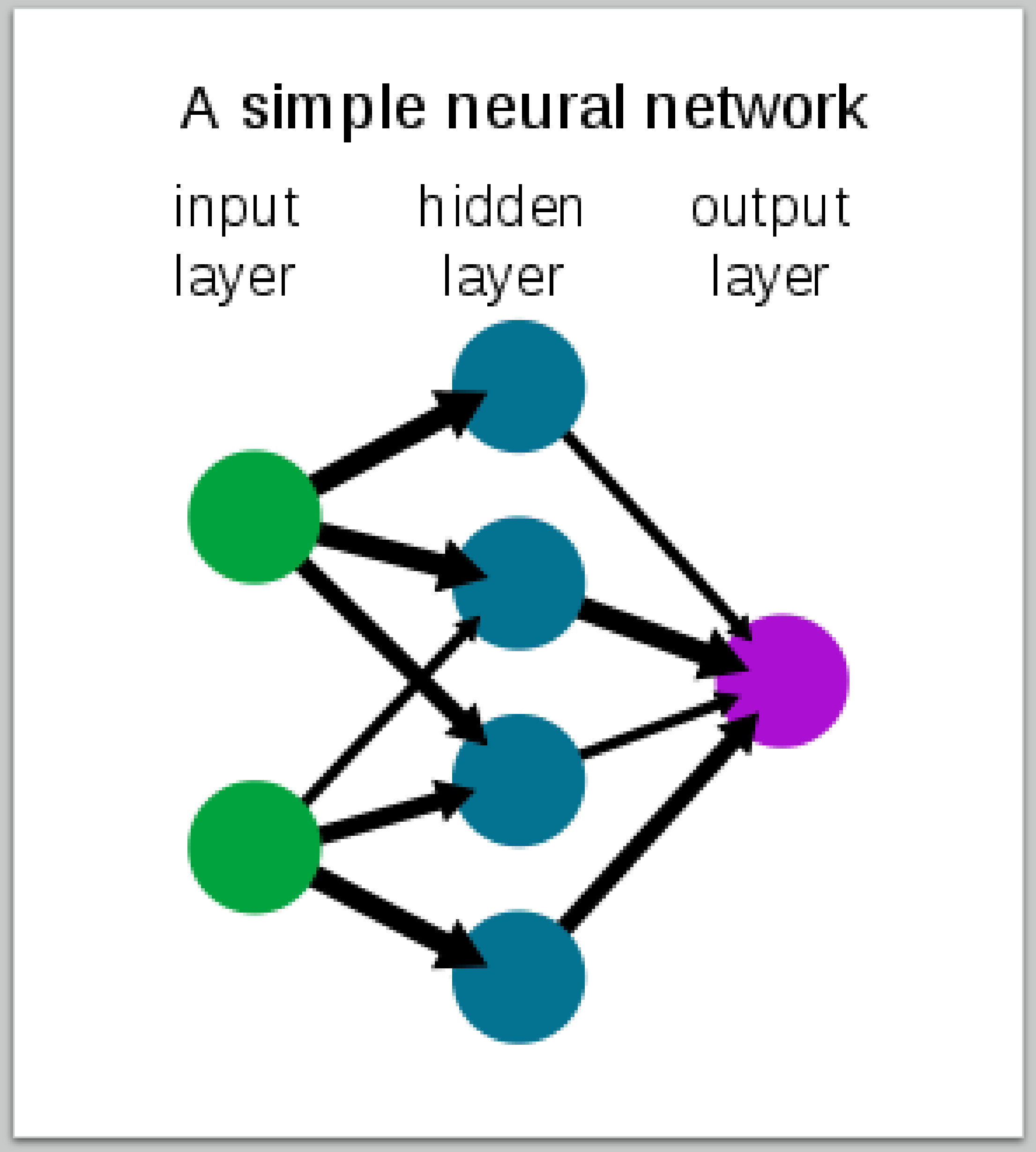
Sullivan Bailey-Darland

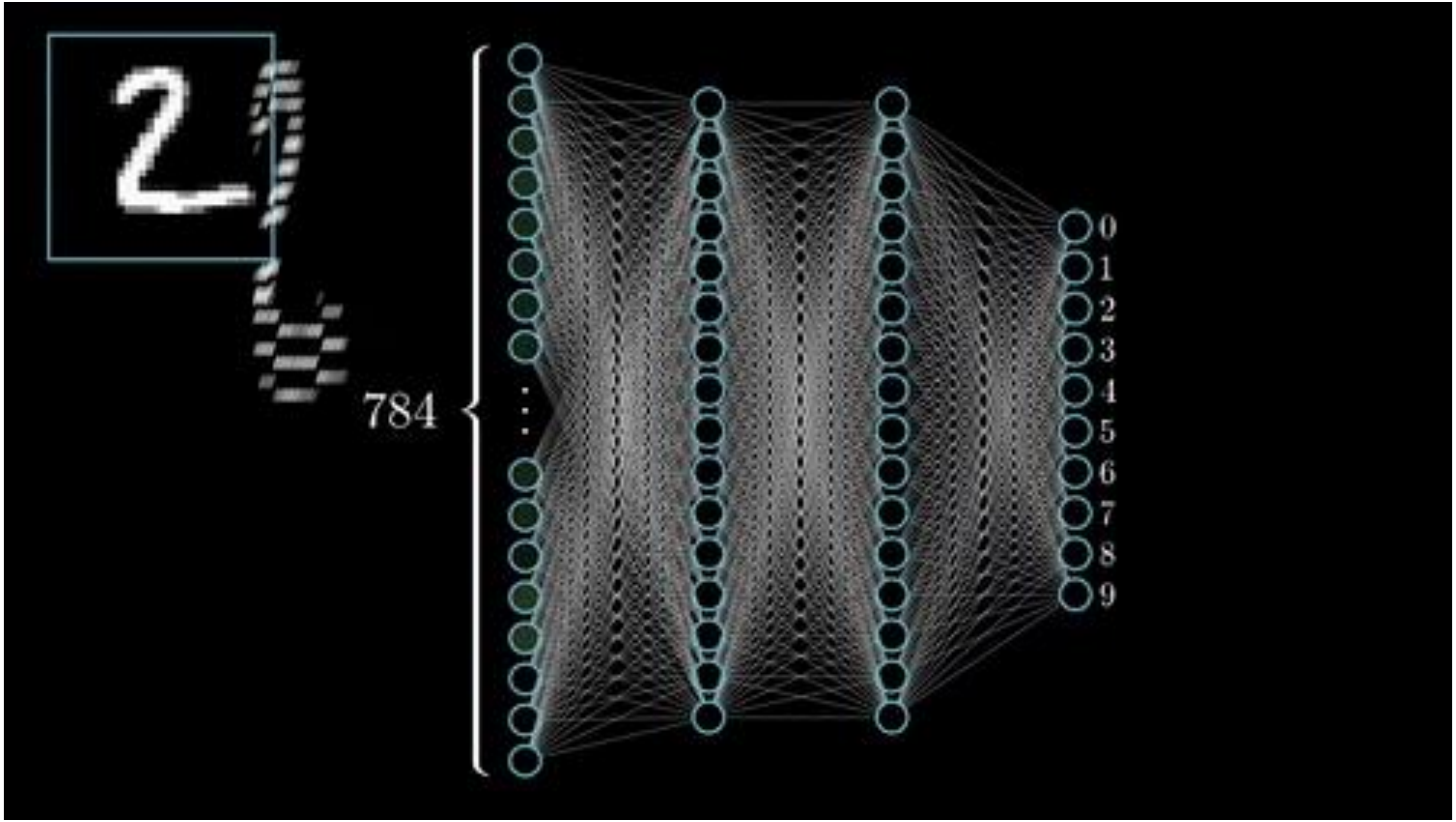
Oregon State University

DIANA Undergraduate Fellow 2021

What is machine learning?

- Computer algorithms that can learn.
- Used in a wide variety of problems
- Neural networks are a type of ML.





784
0
1
2
3
4
5
6
7
8
9

What is optimization? (for neural networks)

- Training a model is iterative
- Needs loss/cost function
- Uses gradient descent
- Minibatch and epoch

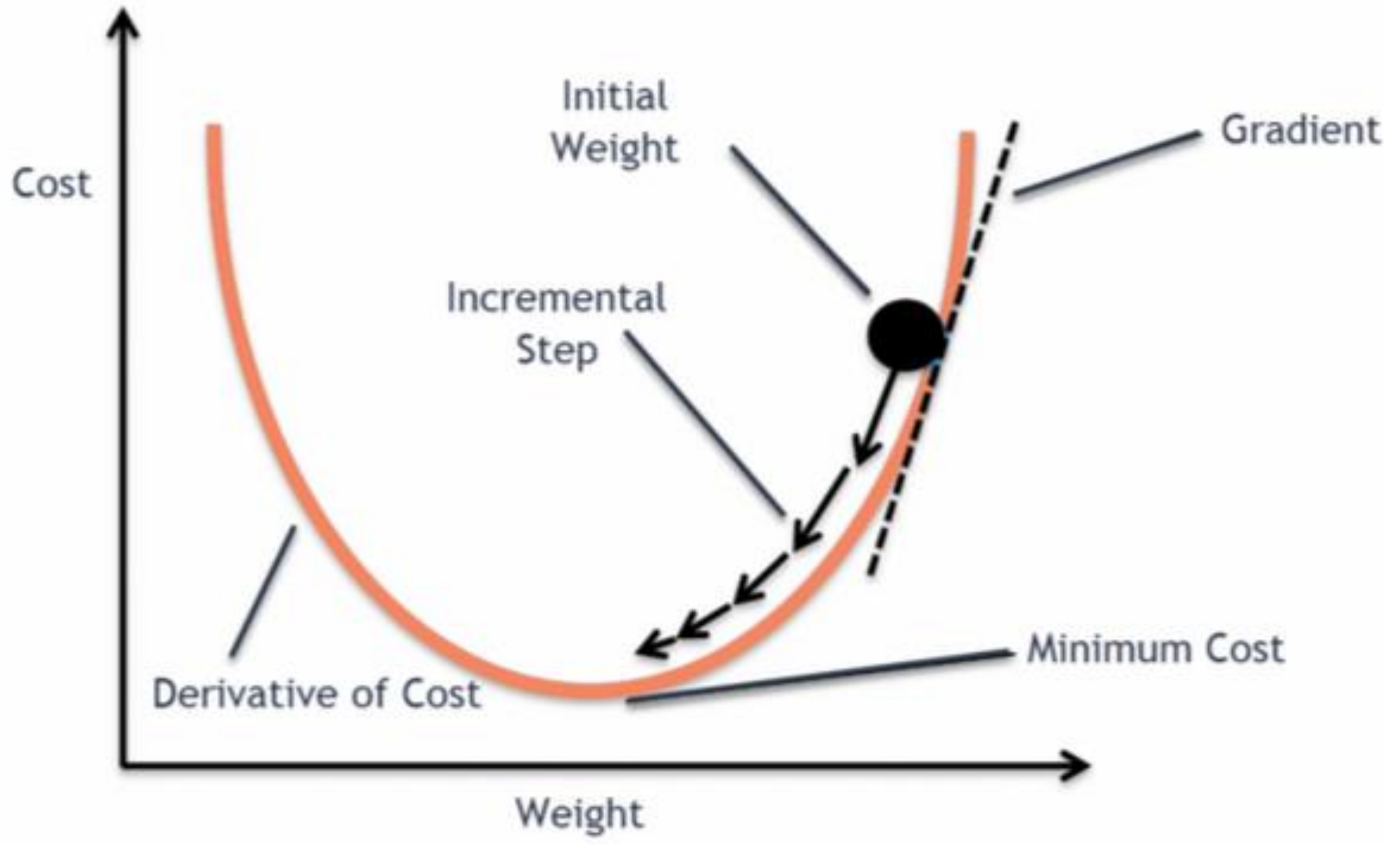
Updated weights

Learning rate

$$w_{i+1} = w_i - \eta \frac{\partial C}{\partial w}(w_i)$$

Old weights

Minibatch gradient



What is Adam? ("<u>Ada</u>ptive <u>m</u>oment estimation")

Commonly used optimization algorithm developed in 2014[1]

Instead of using just the gradient, calculates momentum variables

Uses these to take a smarter path and train faster and better

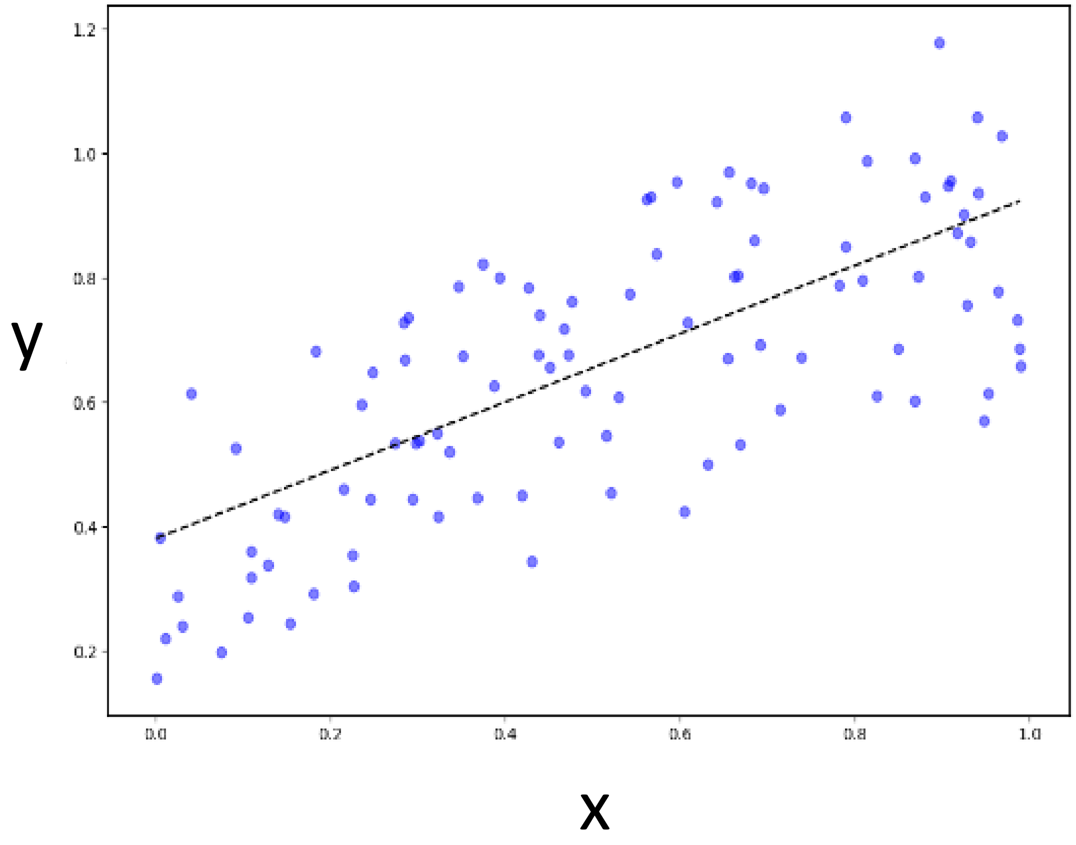
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
  $v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

1. Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv [cs.LG]*. Opgehaal van http://arxiv.org/abs/1412.6980

Visualizing a simple example: **linear regression**

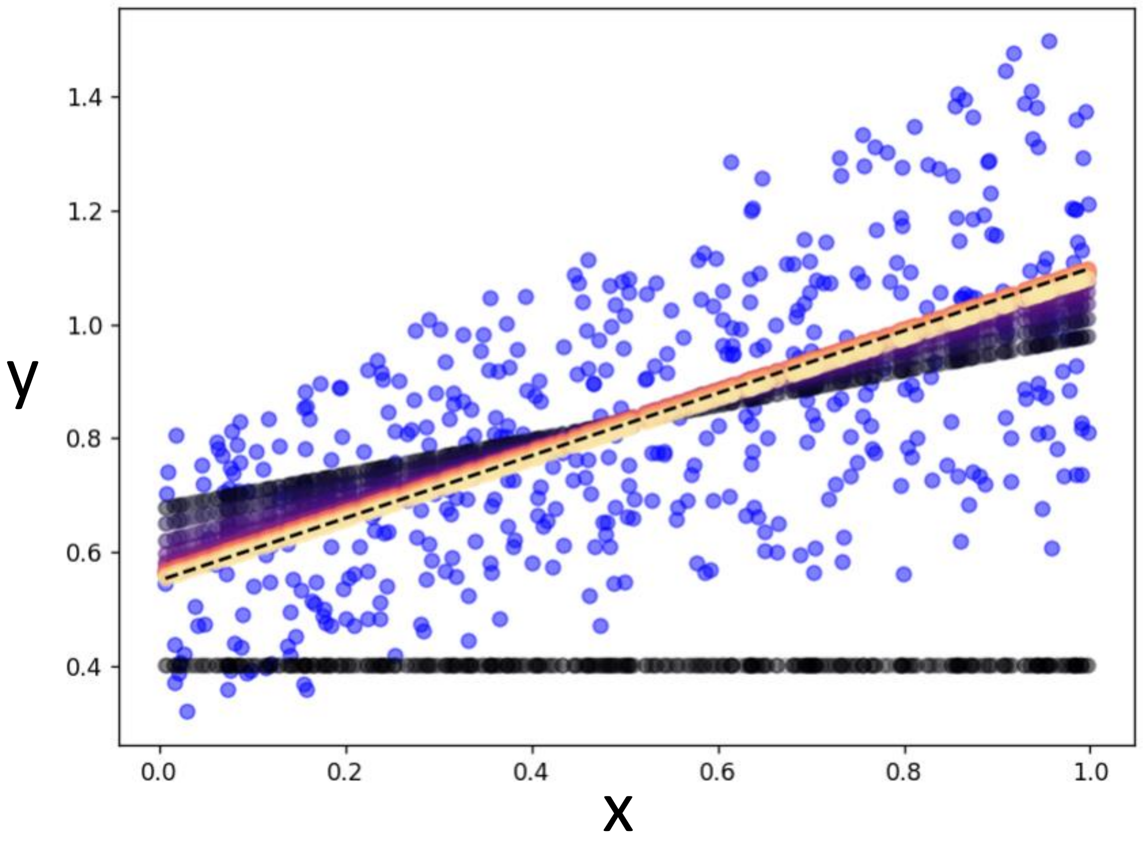
- Minimizes $\sum(y_{pred} - y_{data})^2$
- Only 2 parameters (y = **m**x + **b**), can be plotted.
- Easy to visualize model prediction

Dataset the model was trained on

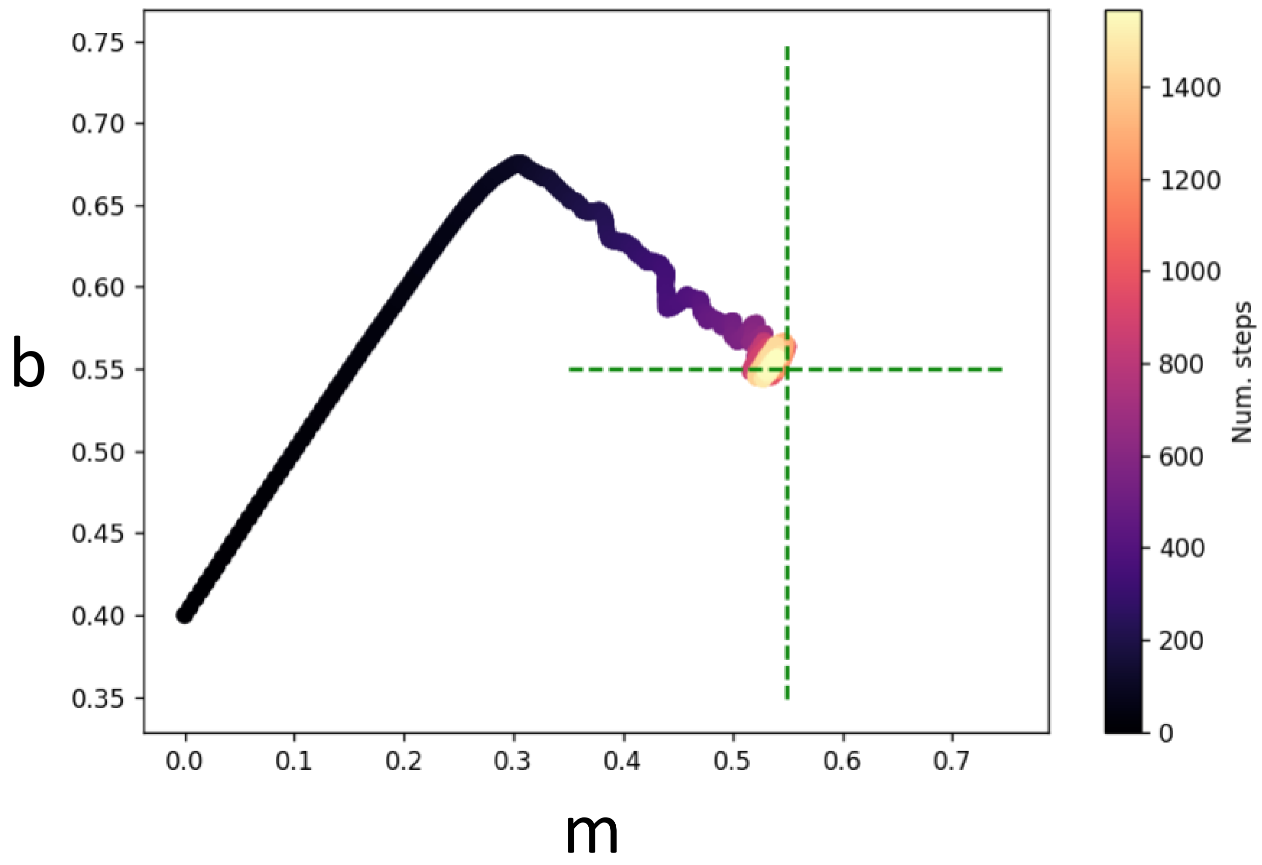


$y = mx + b$

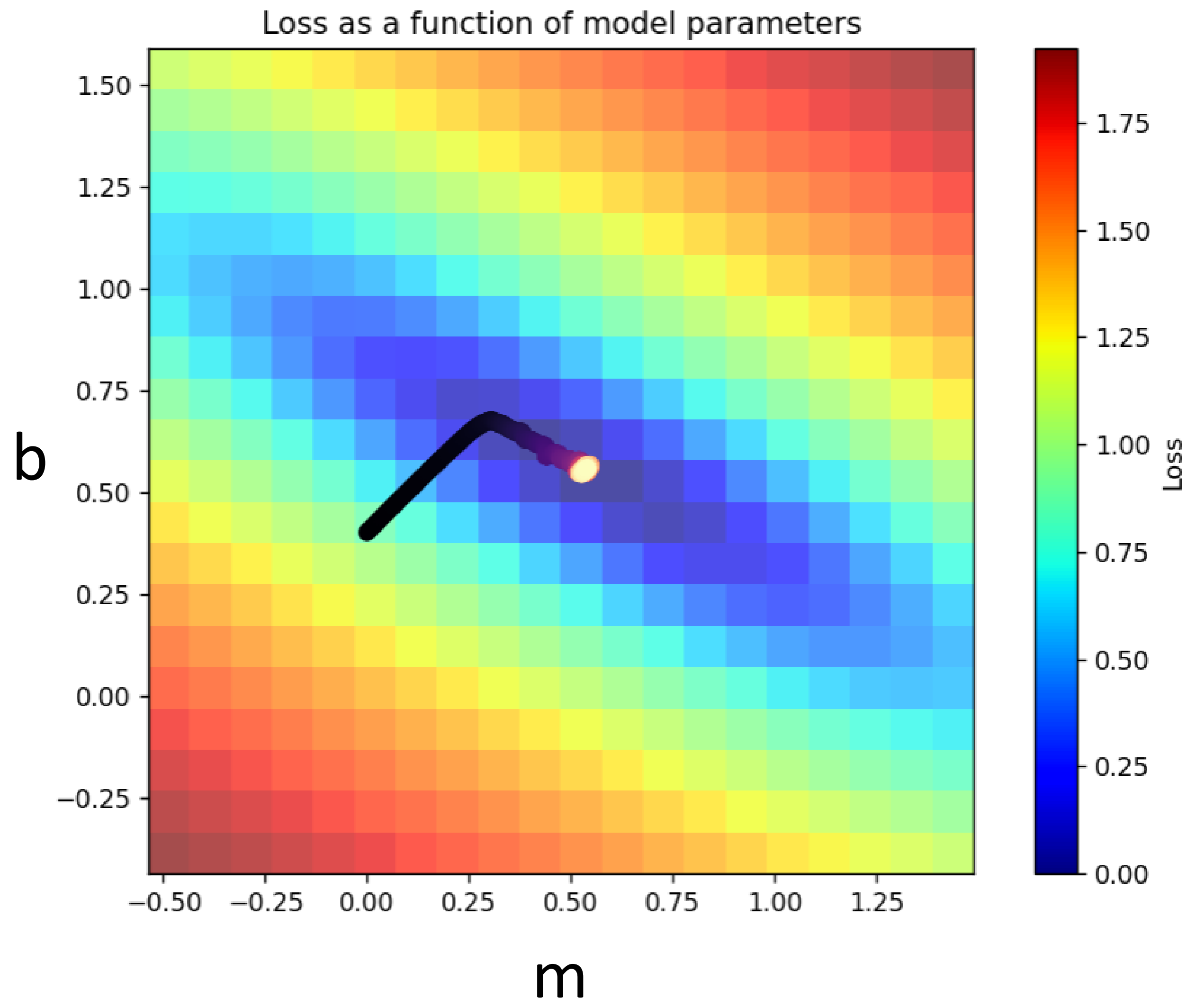
Model predictions over training

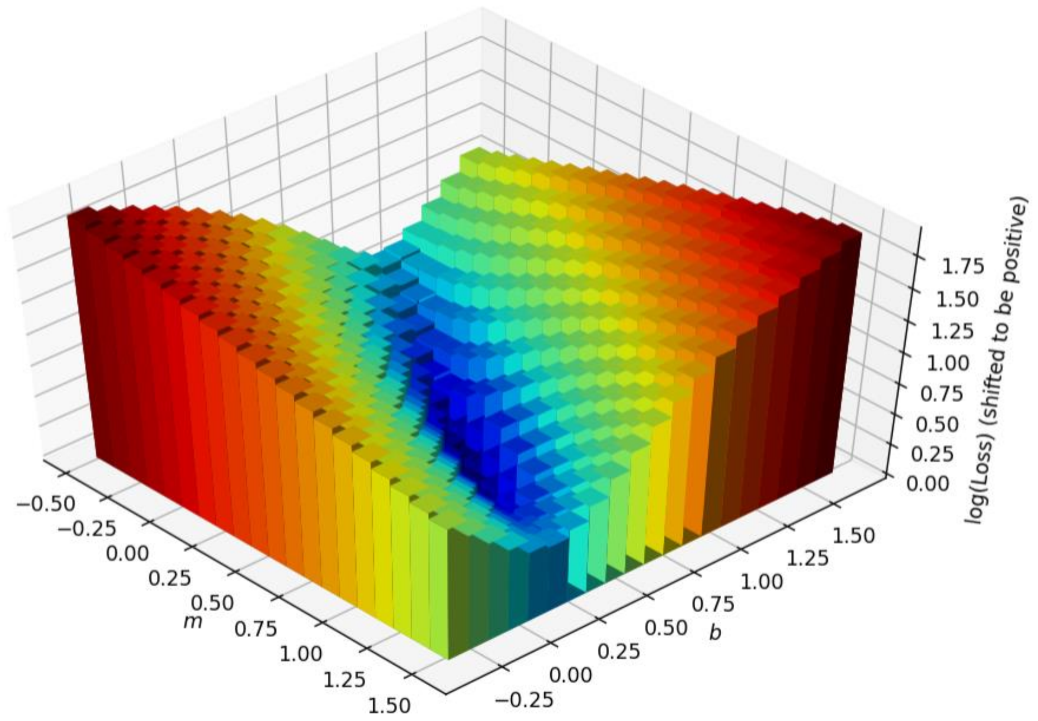


Model parameters over training



Loss as a function of the parameters





Building tools for analyzing larger models

Too many parameters

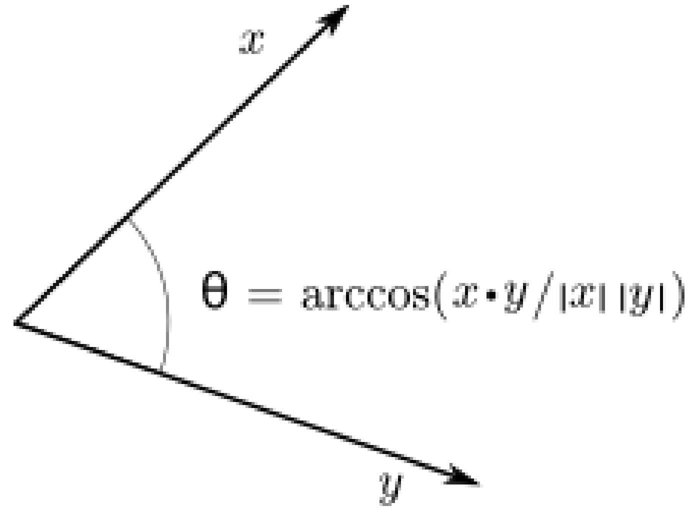
Too many steps

Over each epoch, look at the amount and "direction" of the change in parameters.

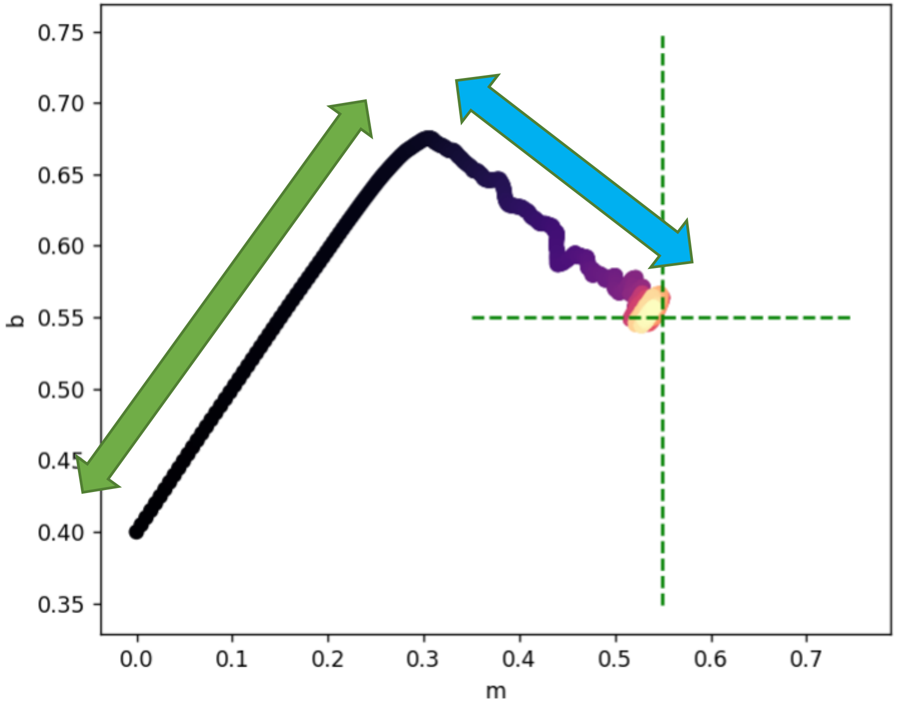
Step size/ amount of change

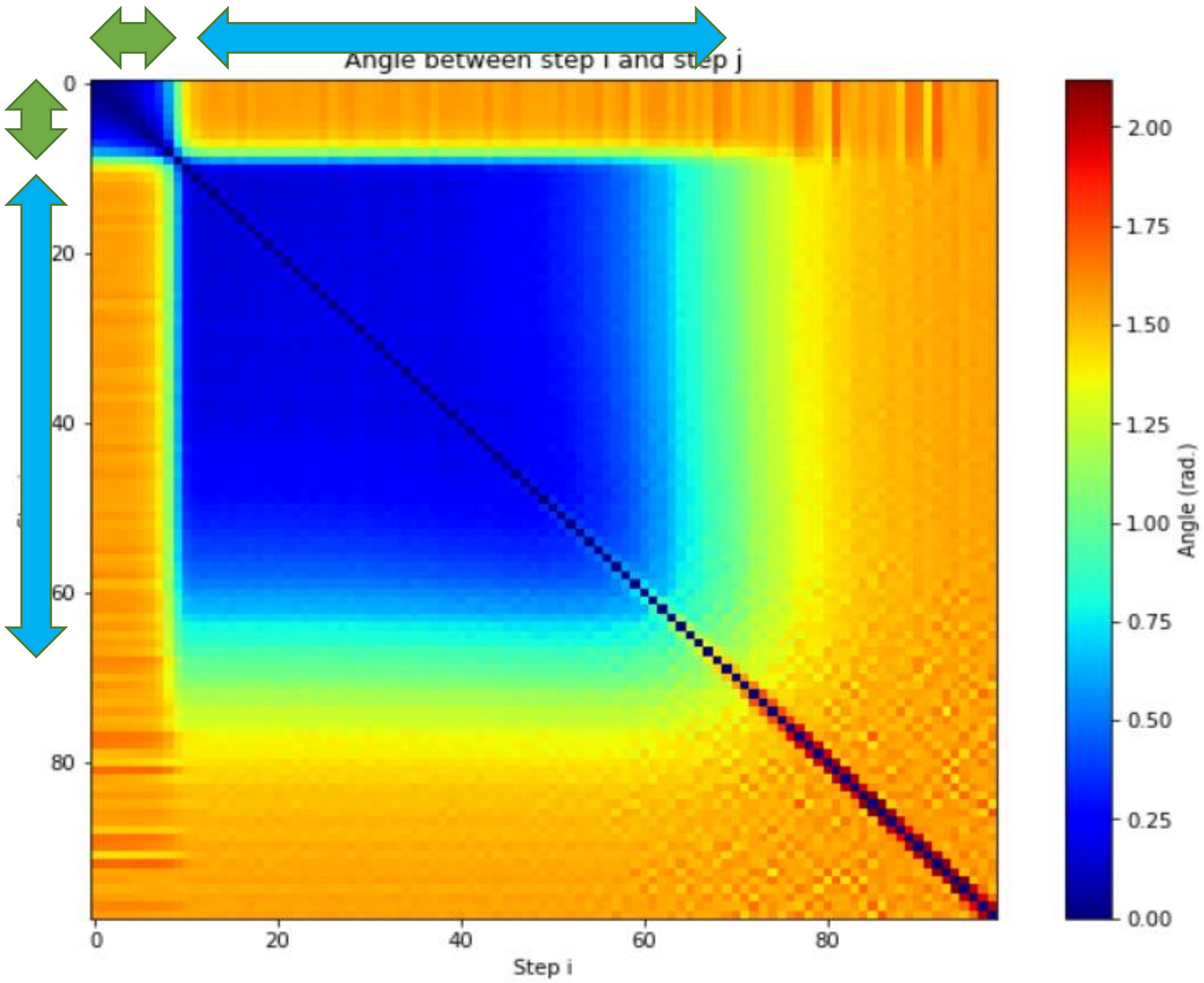
$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{N} |x_i|^2\right)^{1/2} = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}$$

Angle between steps/ direction of change

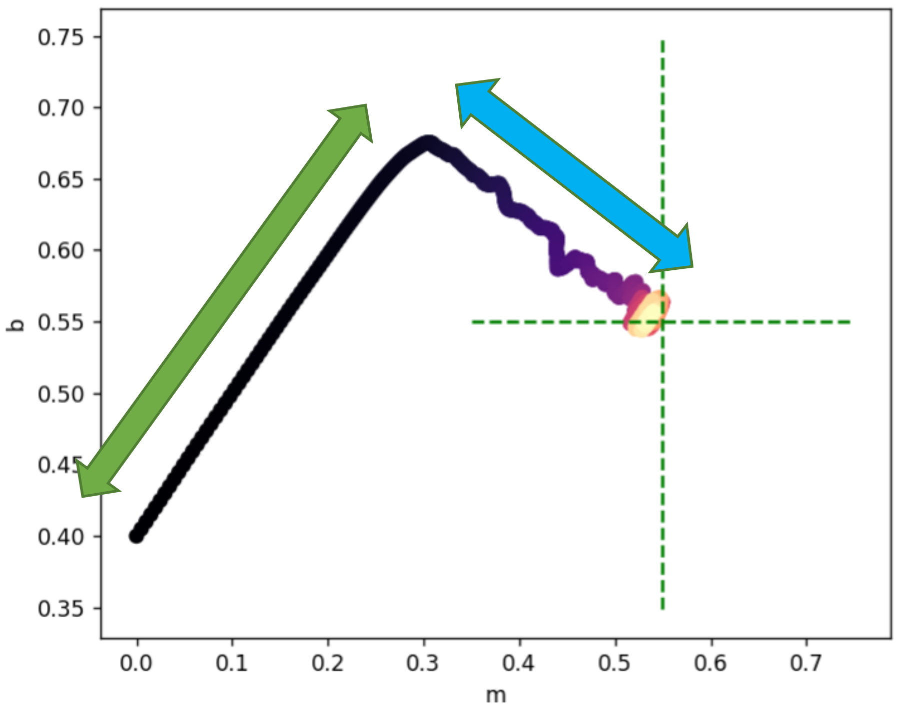


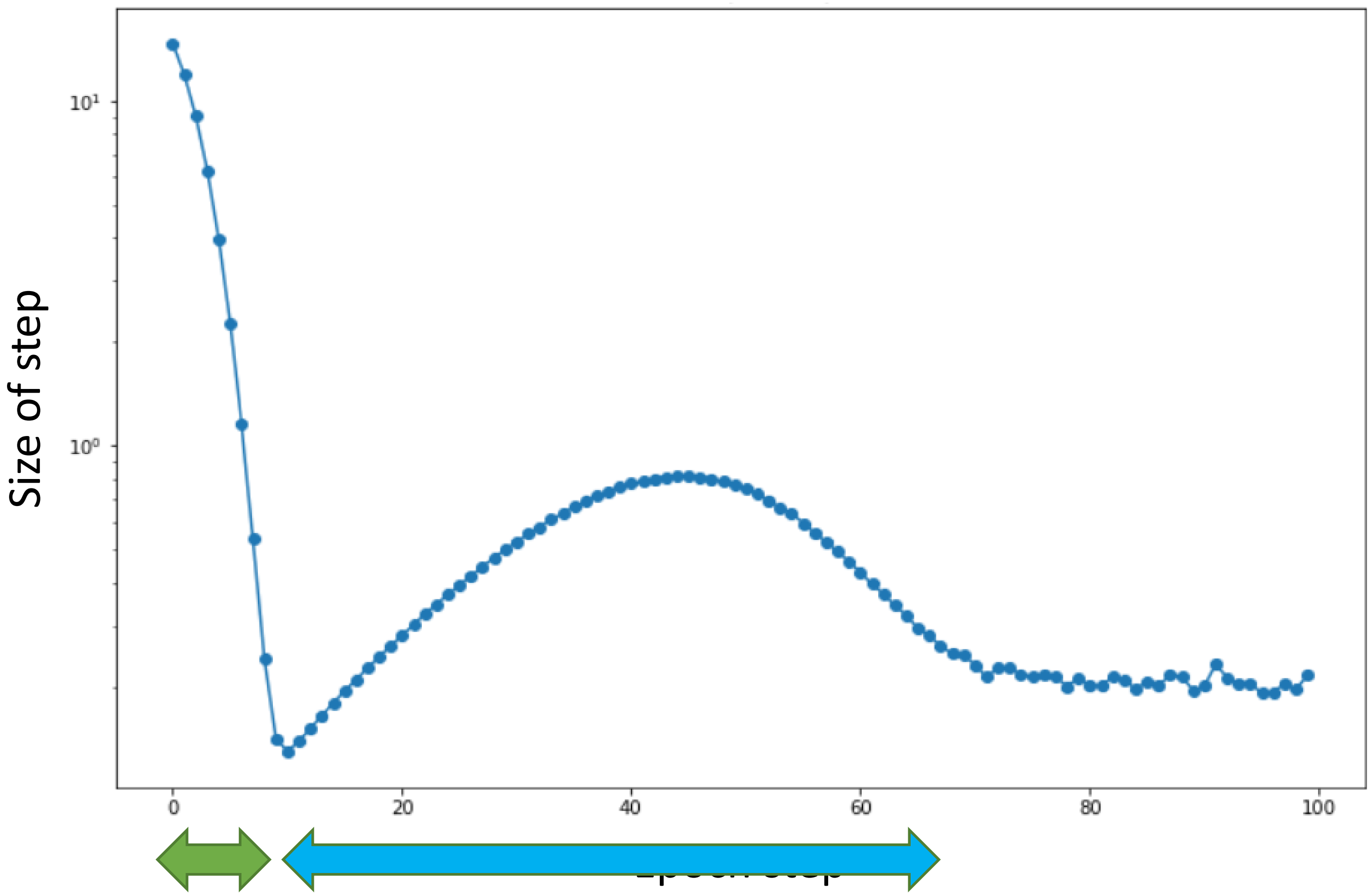
For linear regression



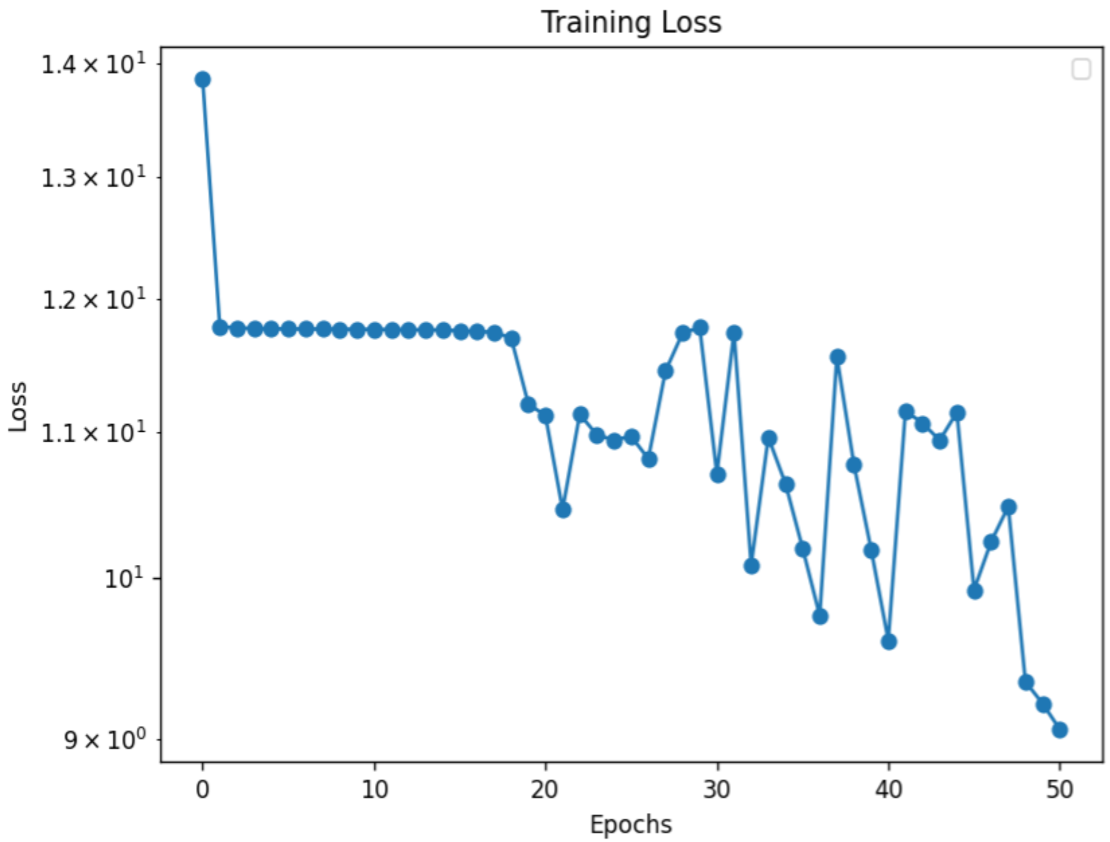


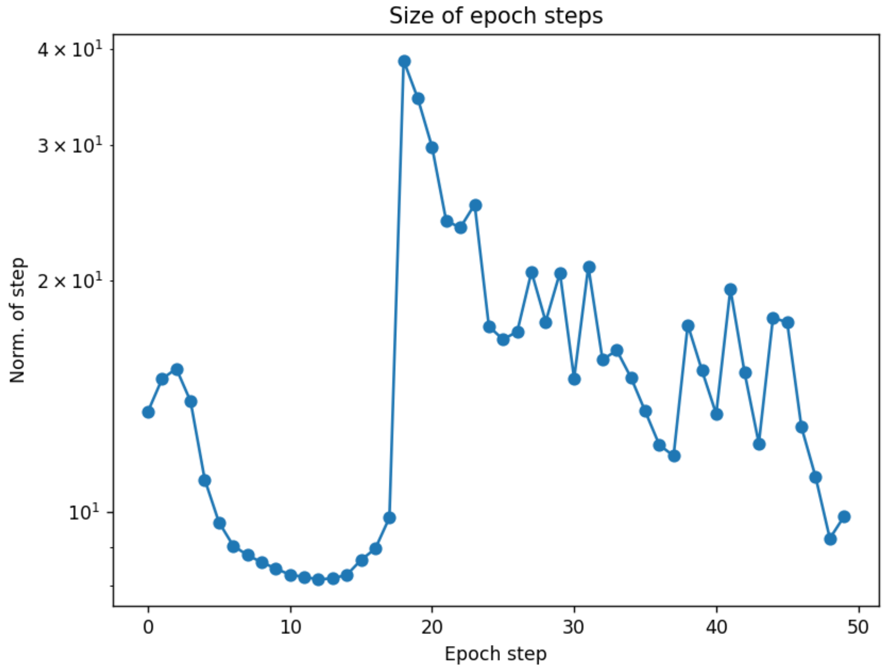
For linear regression

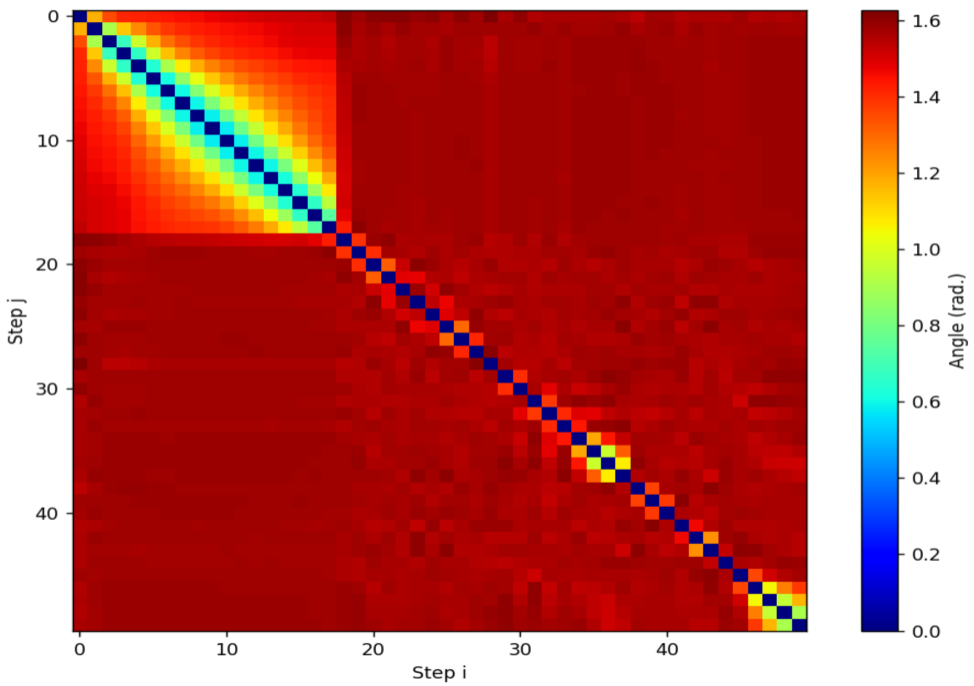




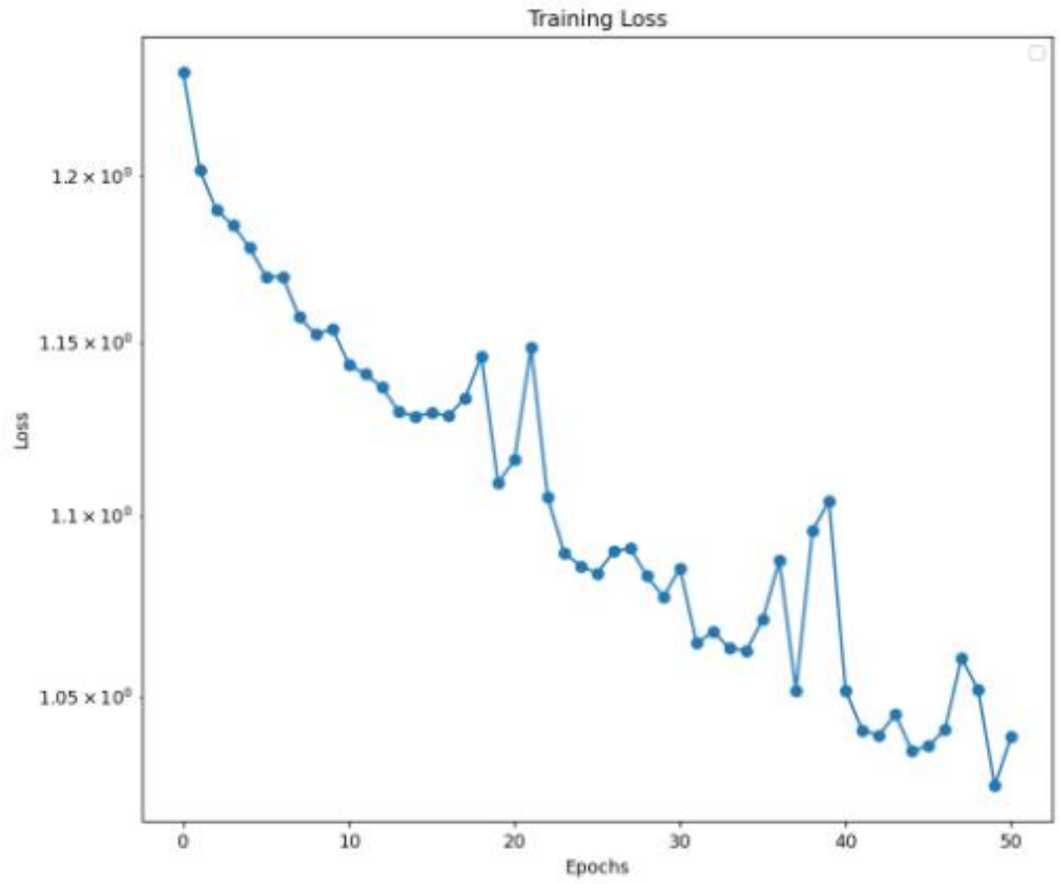
Real application: DDplus model

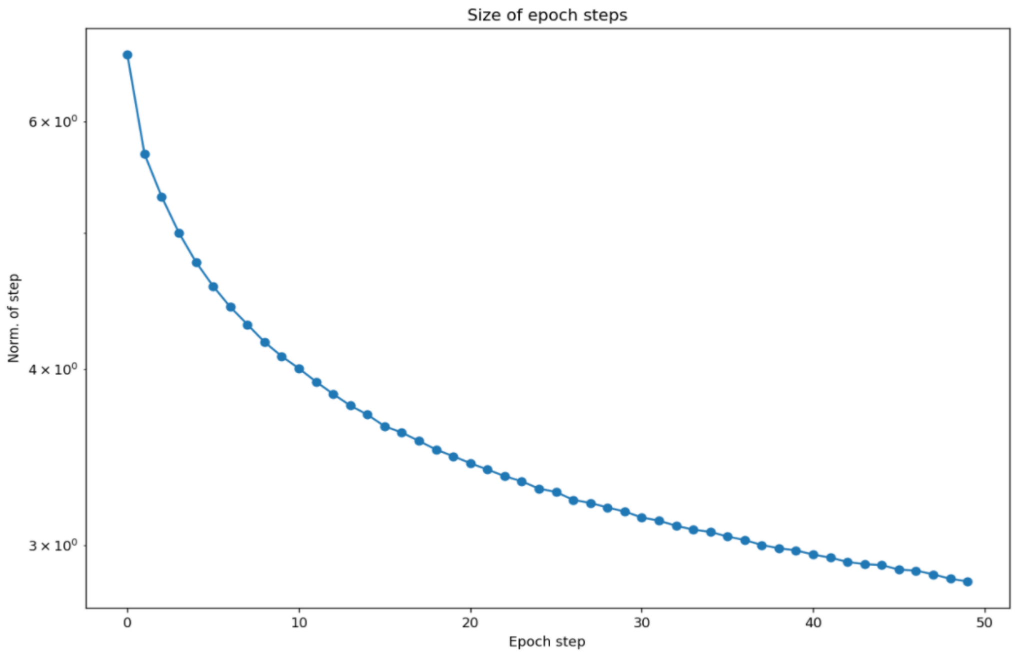


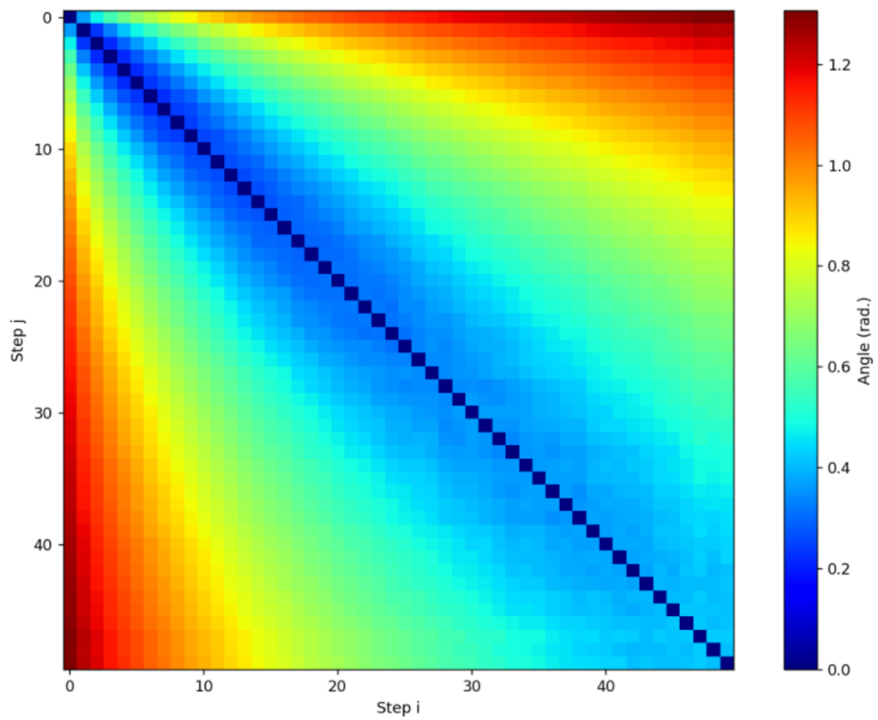




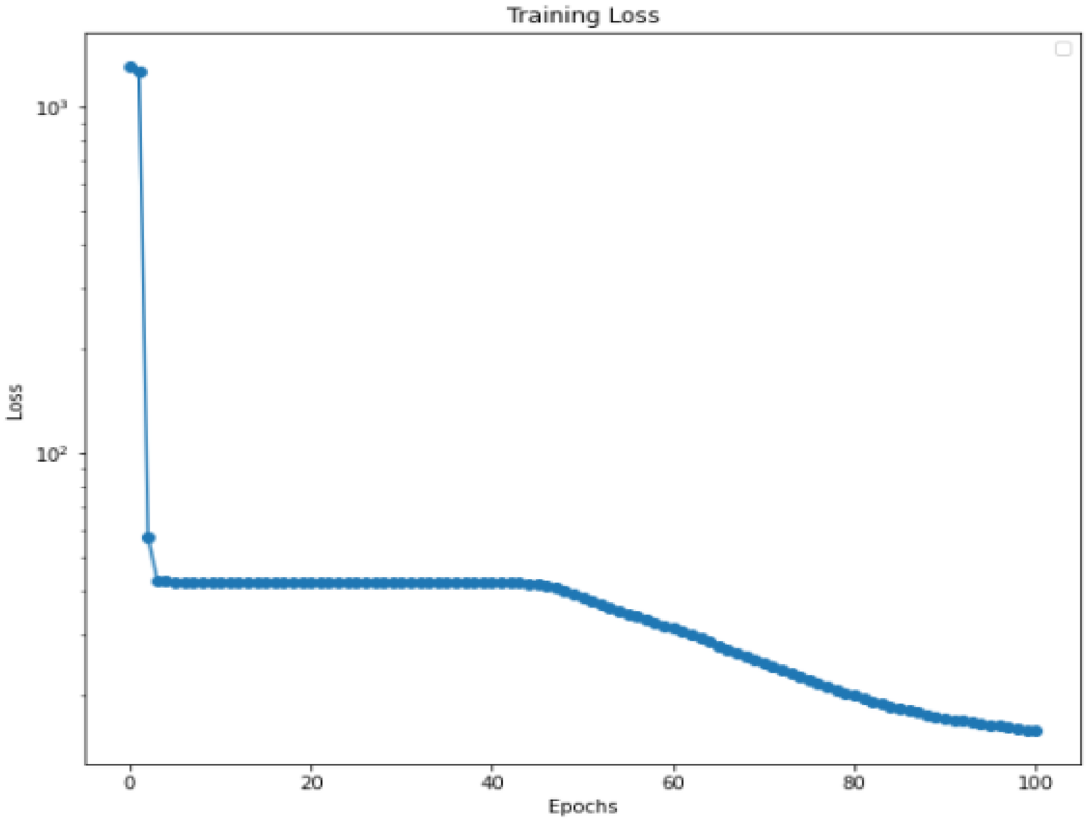
Real application: trained DDplus model

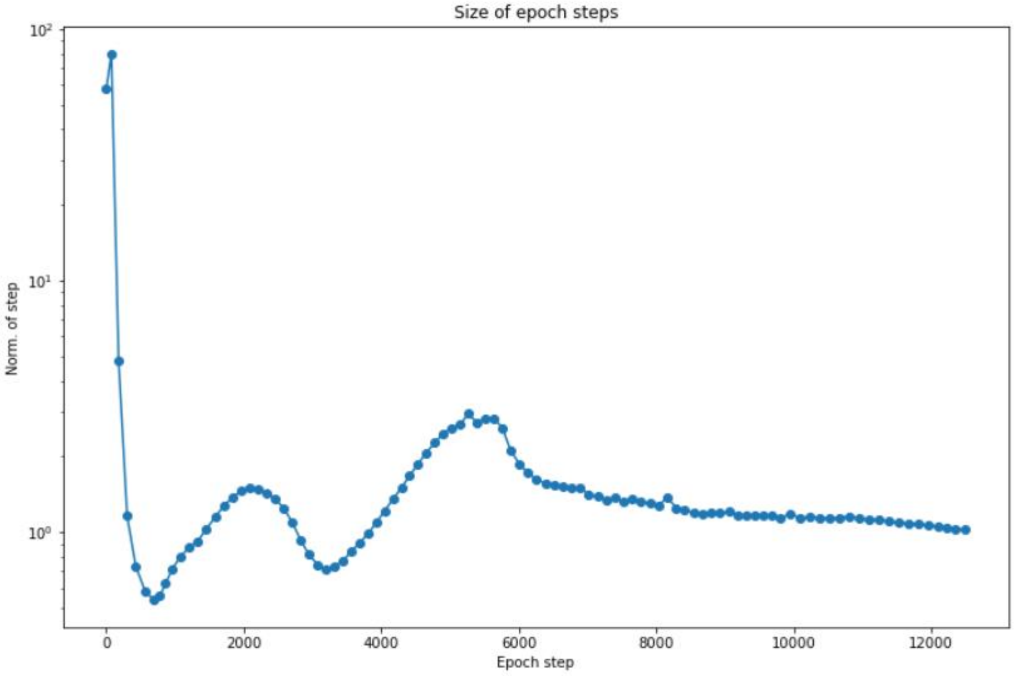


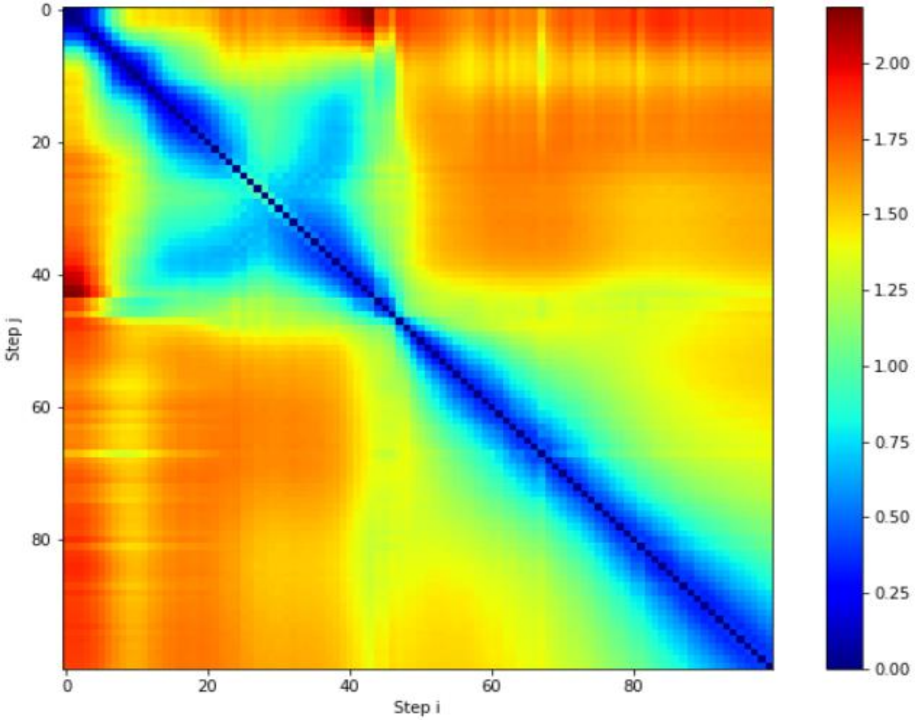




Real application: AllCNN model







Speeding up optimization?

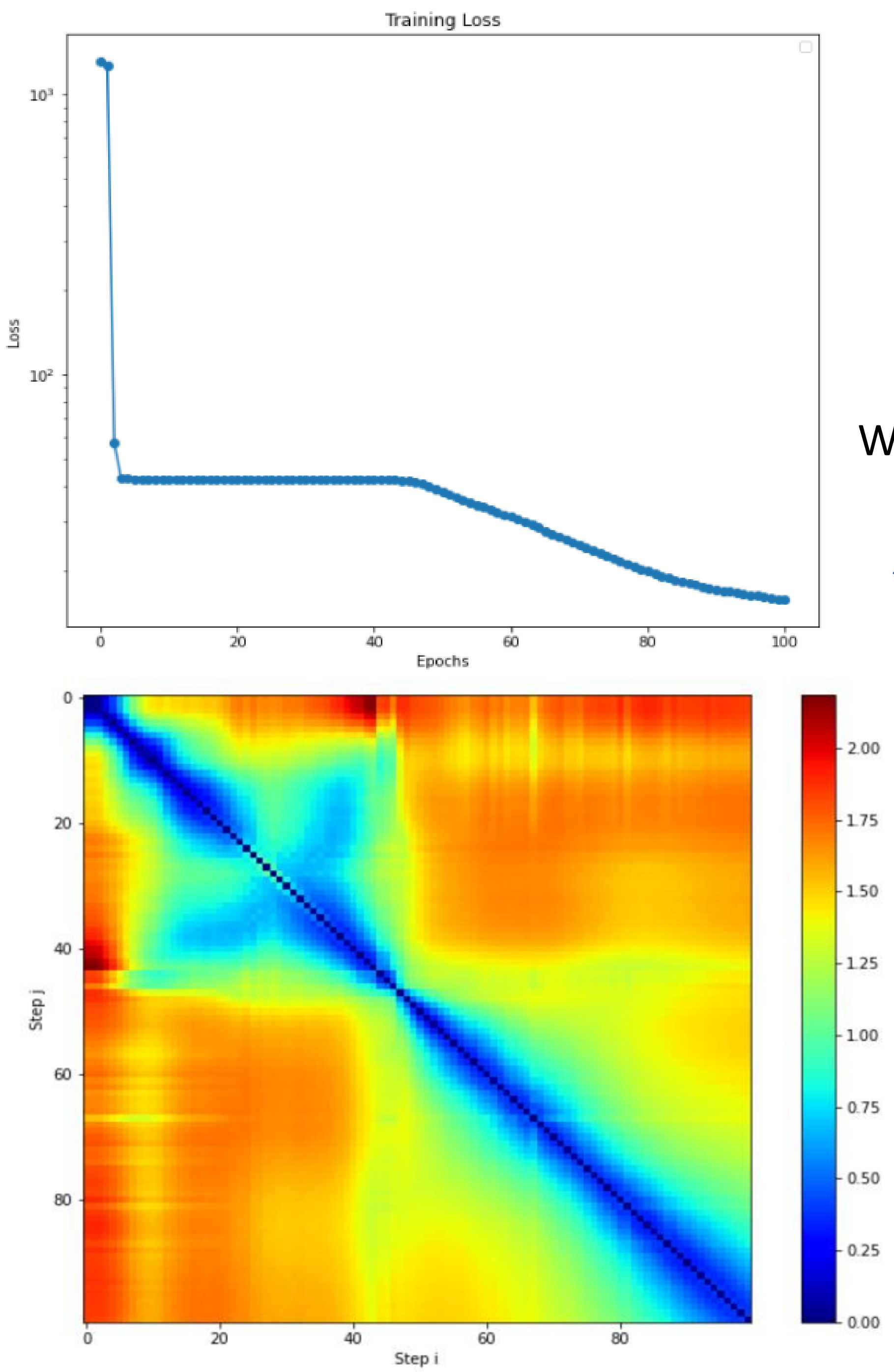
If consecutive steps are in the same direction, can you take larger steps?

Adaptive learning rate
Increase the learning rate if epochs are in the same direction

EVE algorithm
Evaluate loss in epoch direction, and if the loss continues decreasing take a larger step

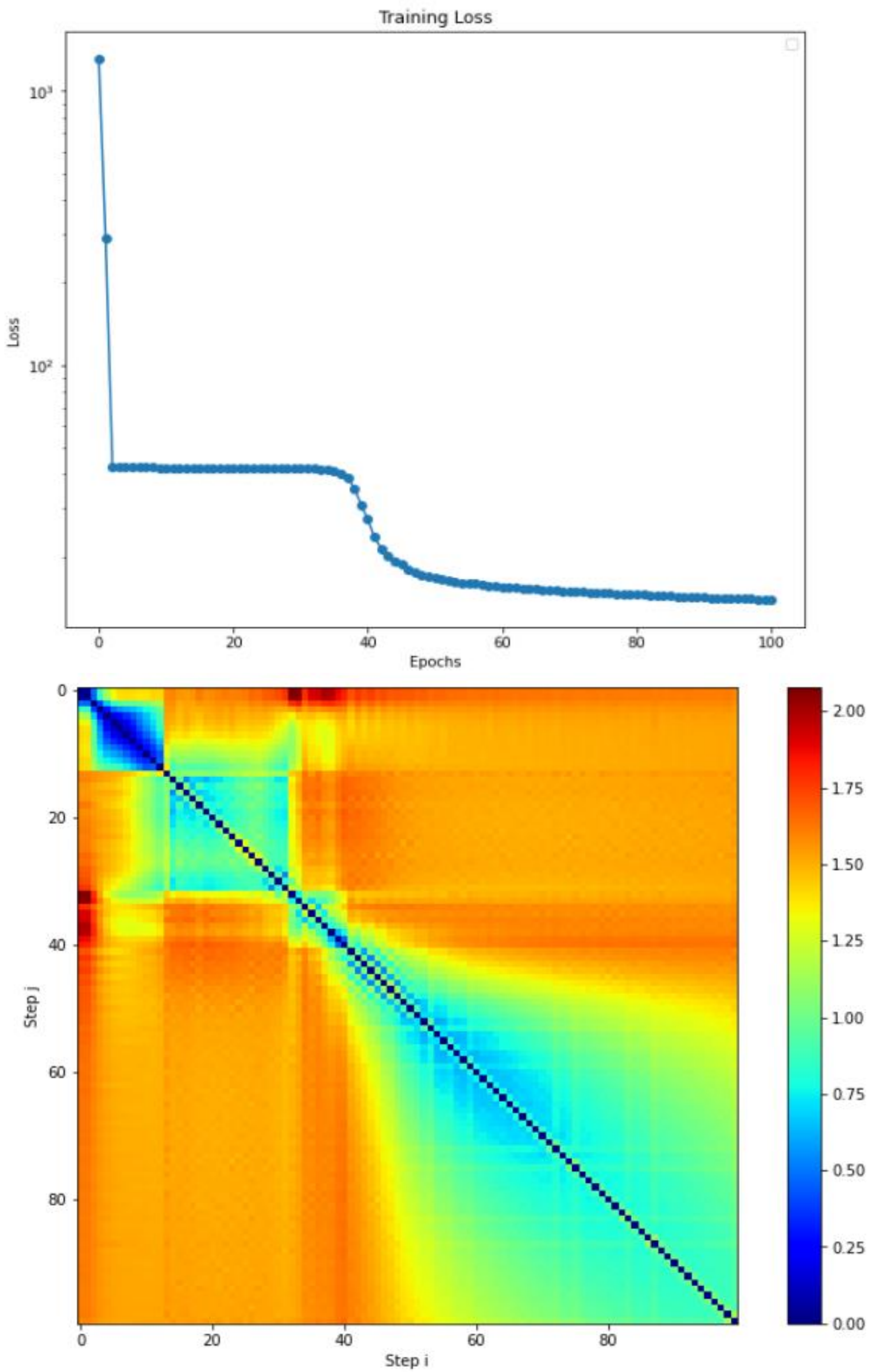
If the loss repeatedly jumps up, should you take smaller steps?

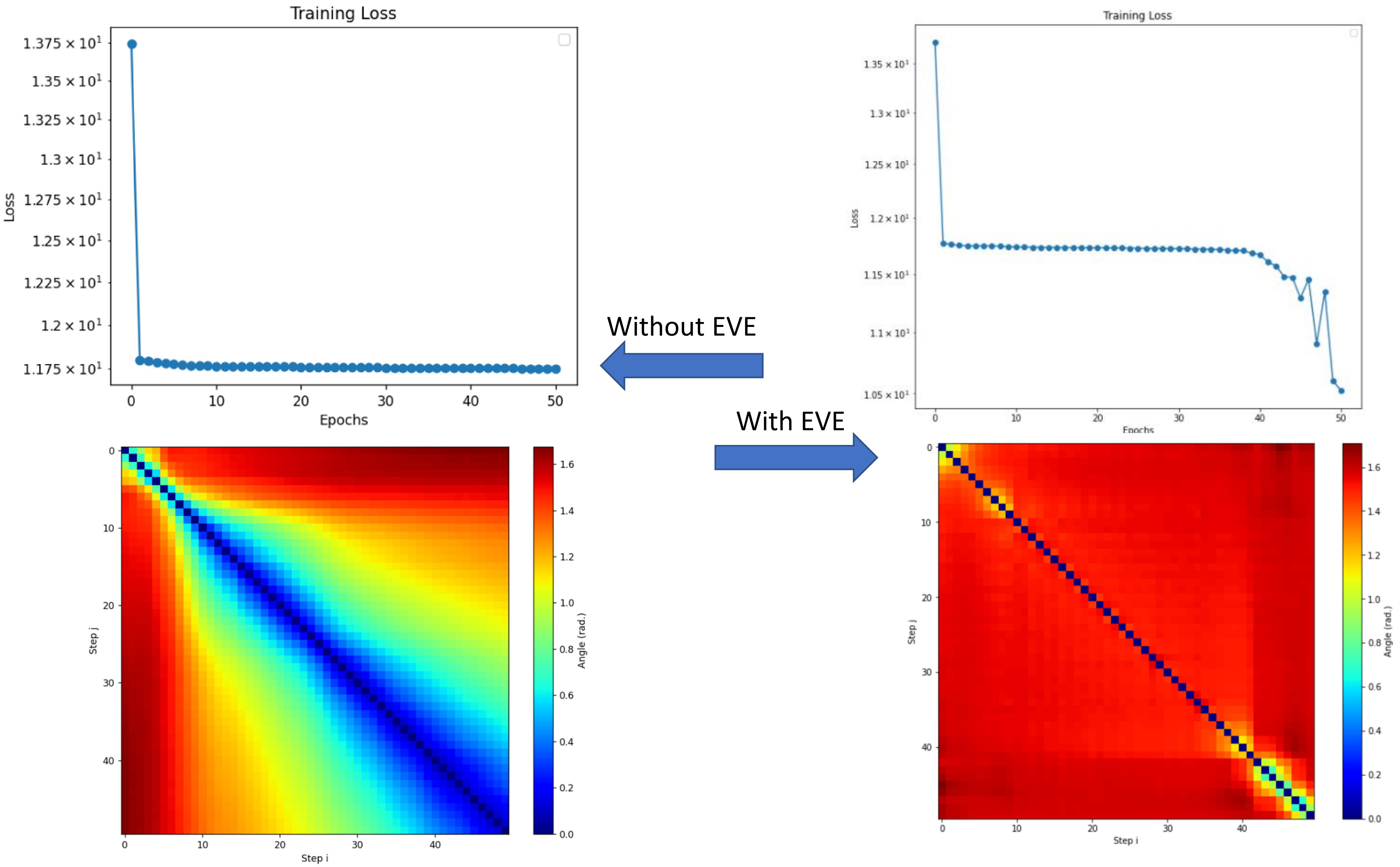
Careful learning rate
If the loss jumps up, decrease the learning rate



Without adaptive learning rate

With adaptive learning rate




Training Loss
Loss
Epochs
Step j
Step i
Angle (rad.)
Without EVE
With EVE
Training Loss
Loss
Epochs
Step j
Step i
Angle (rad.)

Current conclusions

- Training seems relatively unique for each model
- Visualizations can help compare training between models
- Early evidence that these methods can speed up some of the training

Image credits

- Neural network: https://upload.wikimedia.org/wikipedia/commons/thumb/9/99/Neural_network_example.svg/800px-Neural_network_example.svg.png
- Gradient descent: https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f
- Vector norm: https://towardsdatascience.com/why-norms-matters-machine-learning-3f08120af429
- Inner product angle: https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Inner-product-angle.png/1174px-Inner-product-angle.png