Please take the first ~5 minutes of class to fill out the survey, if you haven't done it:

https://tinyurl.com/phys805survey

Then, find a partner and go through the following with them:

- Install miniconda locally (or wherever you plan to do coursework)
- Create a new conda environment with Python ≥ 3.9
  - `pip install jupyterlab`
- Go to this URL and clone the repository: `https://github.com/mariel-pettee/phys_805_fall_2025`
- Start Jupyter Lab
- Navigate to `in_class_work` and go through Notebook 0.
  - Don't just read in silence! Have both partners look at one screen and discuss as you go.

Development workflows for physicists working with ML

Managing your software environment

OS / container

docker

environments

(mini)conda, (micro)mamba, venv

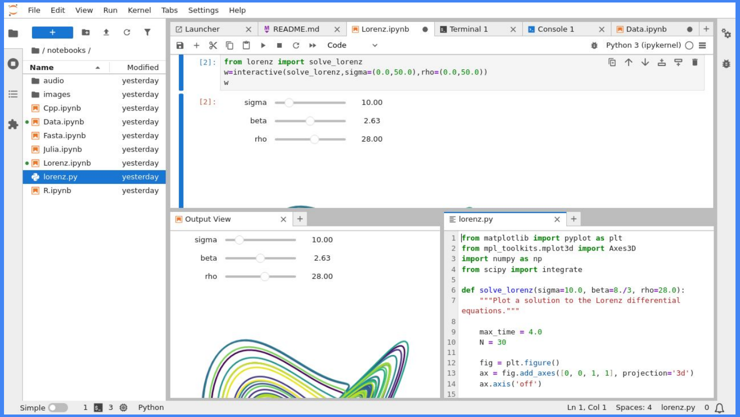
packages

pip, conda, uv

See also: poetry, pixi, ...

Developing code

Sandbox for developing code



More polished & modular codebase

```
├── LICENSE            <- Open-source license if one is chosen
├── Makefile           <- Makefile with convenience commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default mkdocs project; see www.mkdocs.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── pyproject.toml     <- Project configuration file with package metadata for
│                         {{ cookiecutter.module_name }} and configuration for tools like blac
│
```

https://github.com/drivendataorg/cookiecutter-data-science

O'REILLY

jupytercon

AUG 21-24, 2018
NEW YORK, NY

jupytercon.com
#JupyterCon

FAIR development

PAPER • OPEN ACCESS

FAIR AI models in high energy physics

Javier Duarte, Haoyang Li, Avik Roy, Ruike Zhu, E A Huerta, Daniel Diaz, Philip Harris, Raghav Kansal, Daniel S Katz, Ishaan H Kavoori ▾ Show full author list

Published 29 December 2023 •

Machine Learning: Science and Technology, Volume 4, Number 4

Citation Javier Duarte *et al* 2023 *Mach. Learn.: Sci. Technol.* **4** 045062

DOI 10.1088/2632-2153/ad12e3

Article PDF

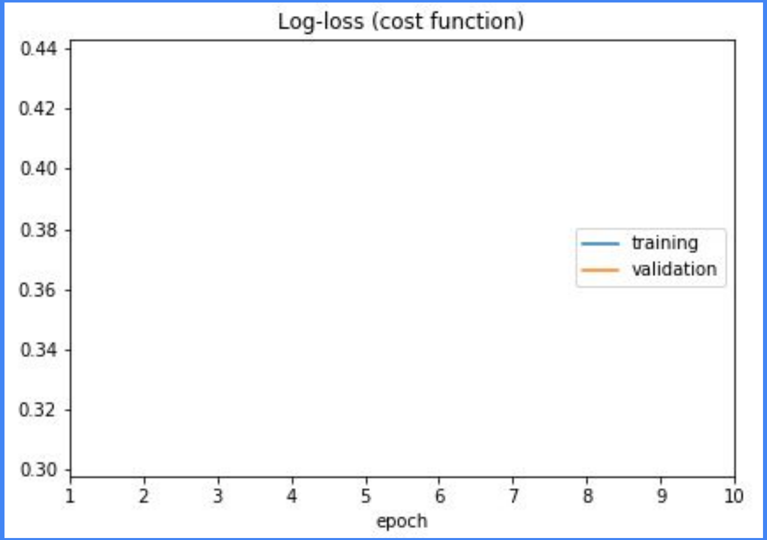
Abstract

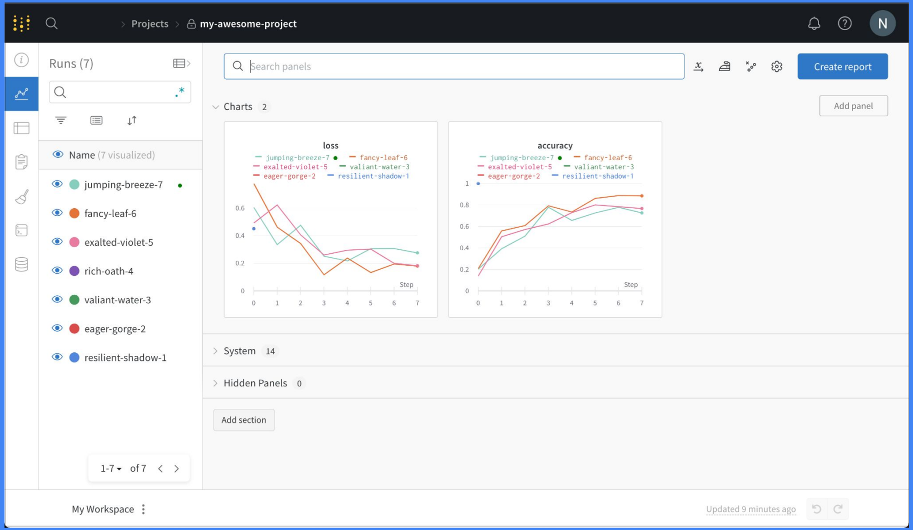
The findable, accessible, interoperable, and reusable (FAIR) data principles provide a framework for examining, evaluating, and improving how data is shared to facilitate scientific discovery. Generalizing these principles to research software and other digital products is an active area of research. Machine learning models—algorithms that have been trained on data without being explicitly programmed—and more generally, artificial intelligence (AI) models, are an important target for this because of the ever-increasing pace with which AI is transforming scientific domains, such

Tracking your training runs

LiveLossPlot

wandb





Which deep learning libraries should you use?

PyTorch

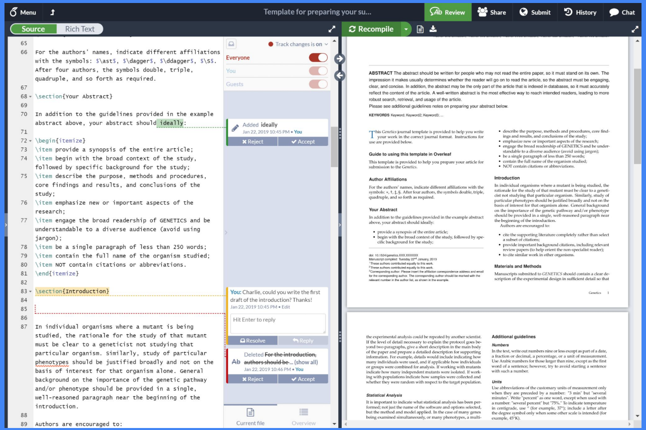
TensorFlow

Keras

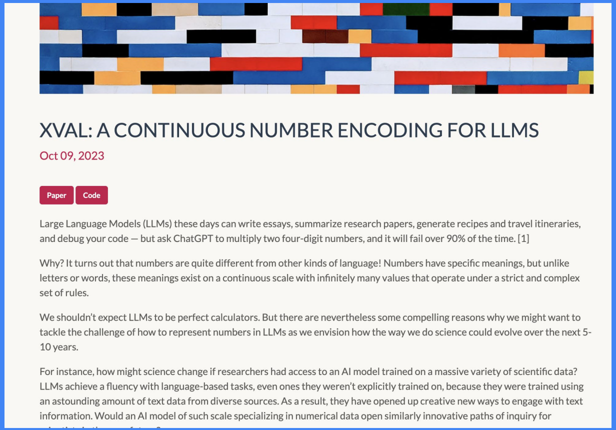
PyTorch Lightning

Writing up your results

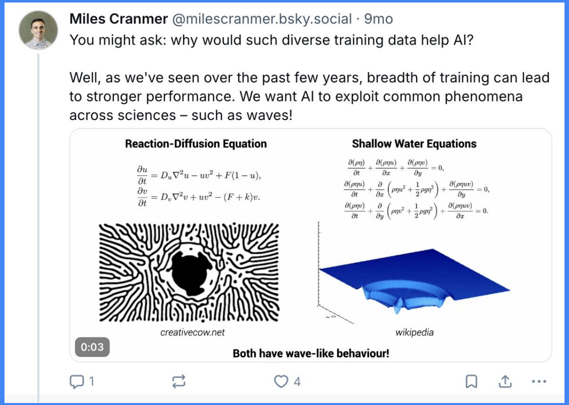
Overleaf / LaTeX



Blog posts



Tweets



What are other critical aspects of your workflow?

Documenting in Git & Overleaf

Working with your partner again:

- Open Jupyter Lab and create a new Jupyter notebook
  - Make a plot of 3 different decaying sine waves and save it as a PDF

- Make a new Git repository for your work in this course & push the notebook to it
  - Then, add some folders & a short README, and push again
  - Make a new branch with a different notebook and push that, too

- Download the NeurIPS 2025 LaTeX template and use it in Overleaf
  - Make up a title and abstract, and add your plot to the paper as a figure

What does it mean to train a neural network?

Why are neural networks special?

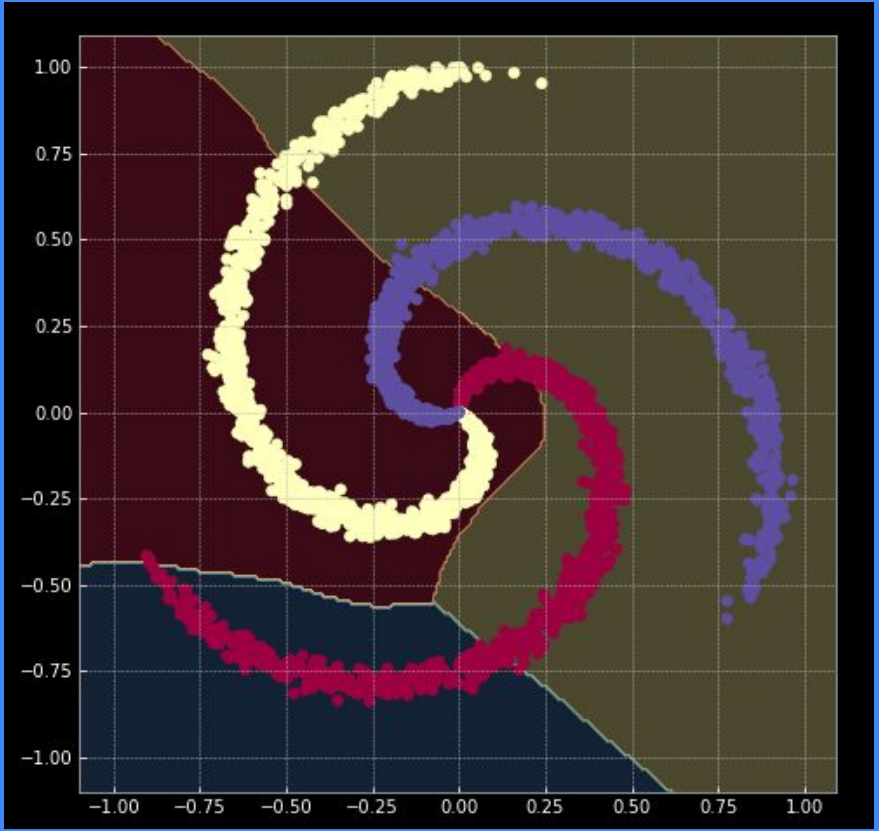
Universal approximation theorem, ~1990s:

- Neural networks can approximate *any* bounded, continuous function to within arbitrary precision... with only one hidden layer!
- *Caveat:* No guarantees that the required network is small, achievable in finite time, or able to generalize well...

Why are they so powerful?

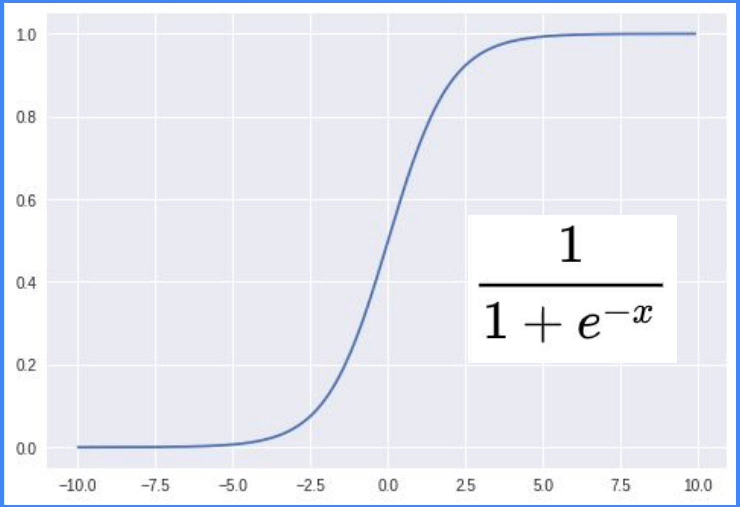
- Can capture complex & nonlinear relationships in the input data
- Automated determination of the optimal features to predict output
- Flexible input structures (sequences, images, labeled, unlabeled, etc.)

Classification with ML = semi-autonomously finding optimal **nonlinear** separations between classes

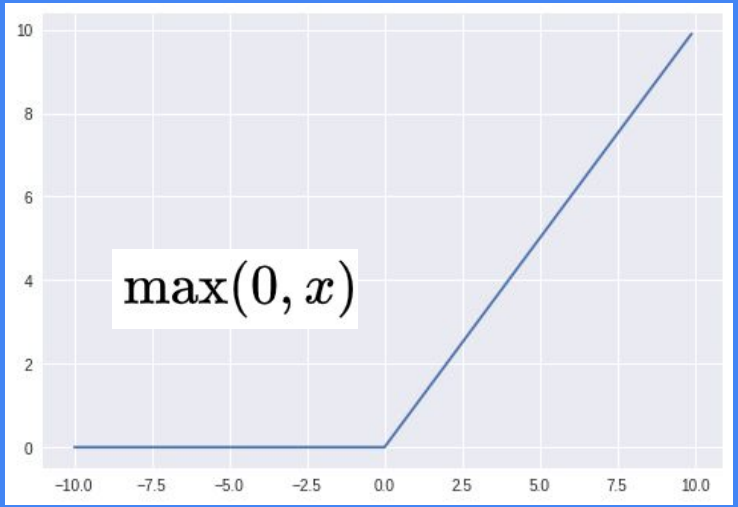


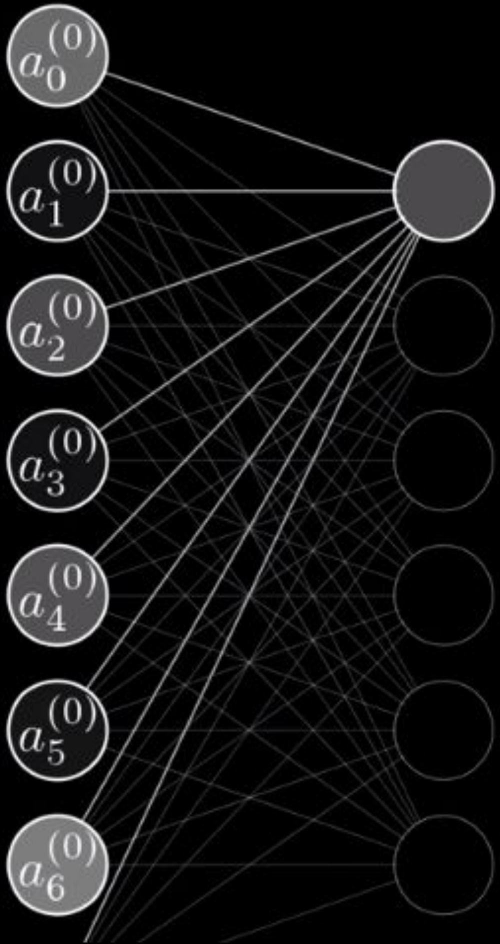
Activation functions: the key to nonlinearity

Sigmoid



ReLU (Rectified Linear Unit)

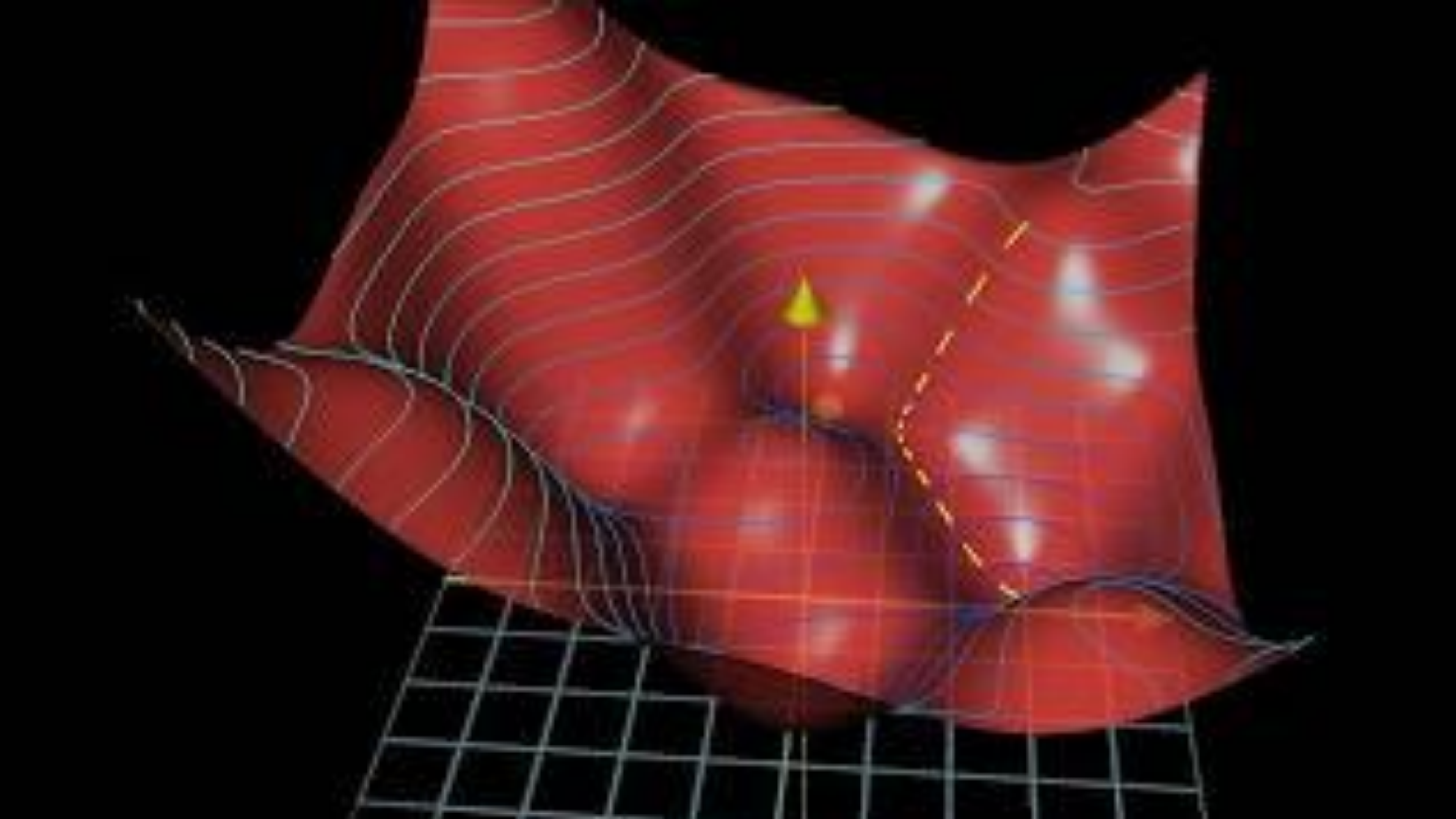




Sigmoid ⟵ Activation function

$$a_0^{(1)} = \sigma\left(w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + \underset{\text{Bias}}{b_0} \right)$$

$$\underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix}}_{\text{Weights}} \underbrace{\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}}_{\text{Previous layer's activations}} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}}_{\text{Biases}}$$



Learning via the loss function

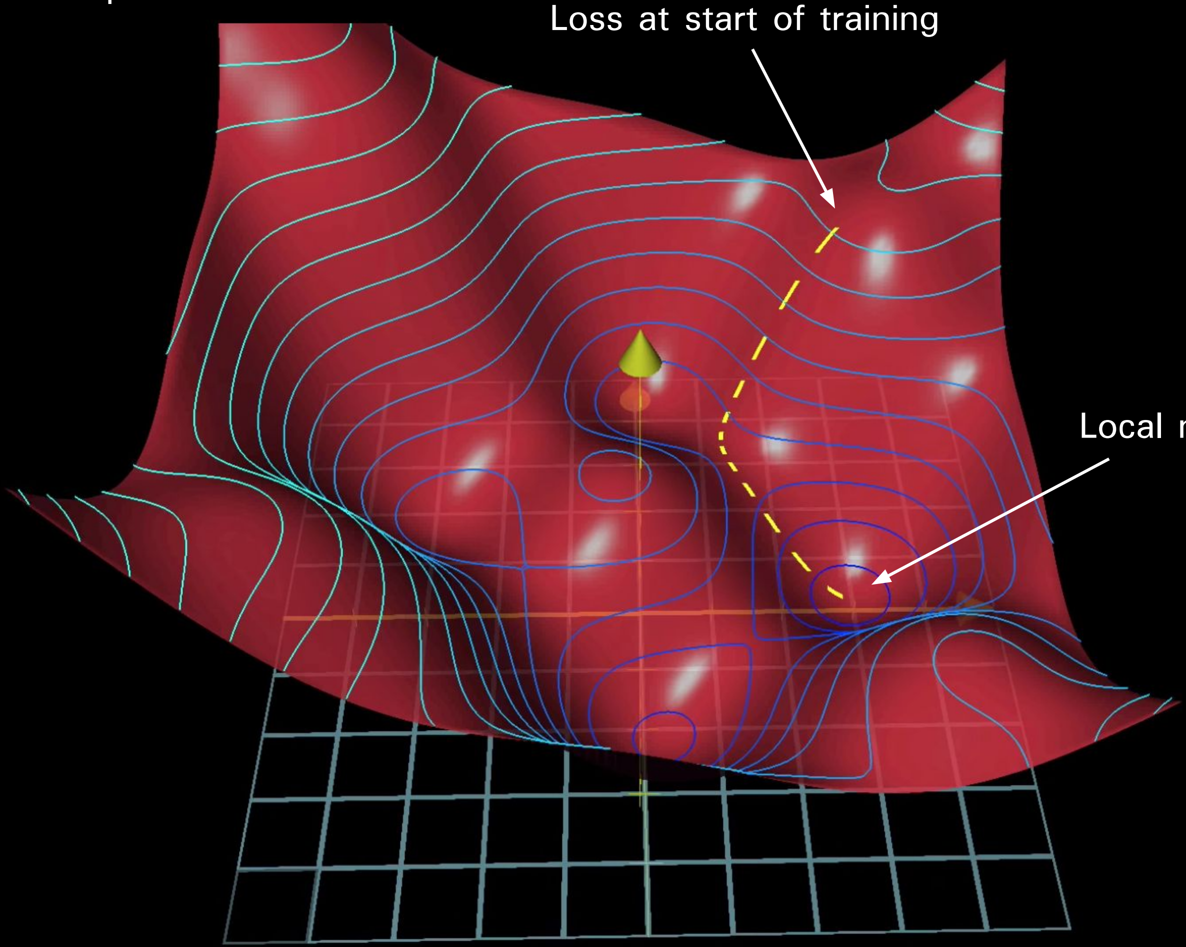
"Loss" = a measure of error in your model, or "regret" for decisions made

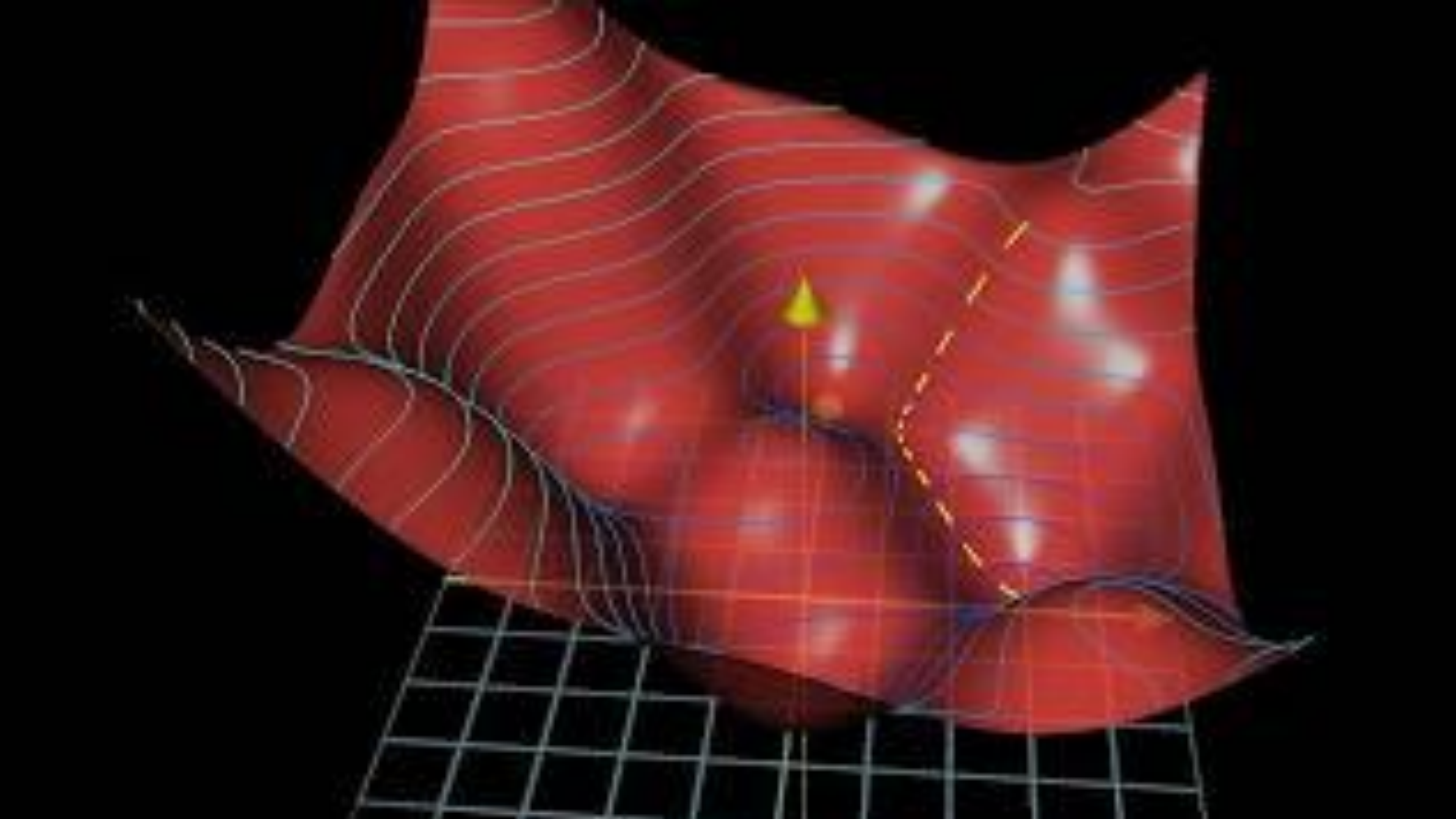
$$\lambda(x) = C(t - x)^2$$

Basic idea: **minimize the loss!**

- Loss helps quantify the difference between the model's prediction and the true value for every sample in the training dataset
- Take the negative gradient of the loss with respect to the weights/biases to get a vector
- Apply this vector (or, typically, an average vector after a few iterations) to your weights & biases to adjust them
- Make a new set of predictions, rinse, & repeat
- Eventually, hopefully approach a local or global minimum

Loss Landscape





Learning via the loss function

"Loss" = a measure of error in your model, or "regret" for decisions made $\lambda(x) = C(t - x)^2$

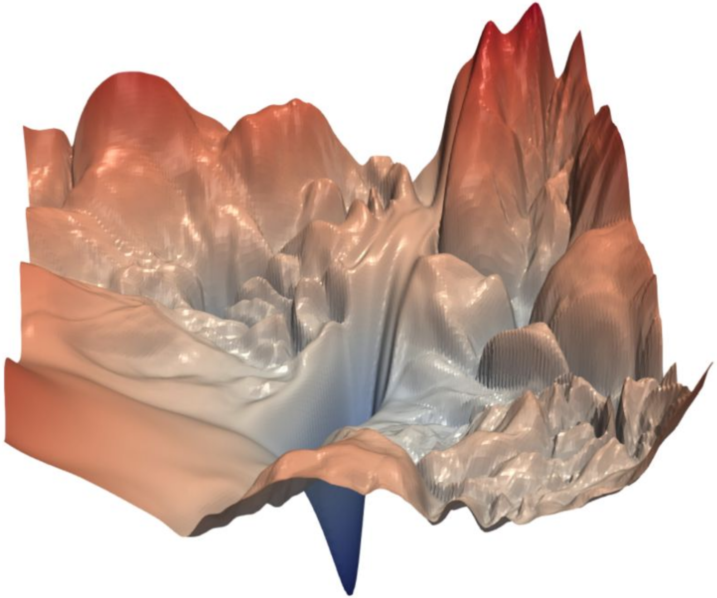
Basic idea: **minimize the loss!**

- Loss helps quantify the difference between the model's prediction and the true value for every sample in the training dataset
- Take the negative gradient of the loss with respect to the weights/biases to get a vector
- Apply this vector (or, typically, an average vector after a few iterations) to your weights & biases to adjust them
- Make a new set of predictions, rinse, & repeat
- Eventually, hopefully approach a local or global minimum

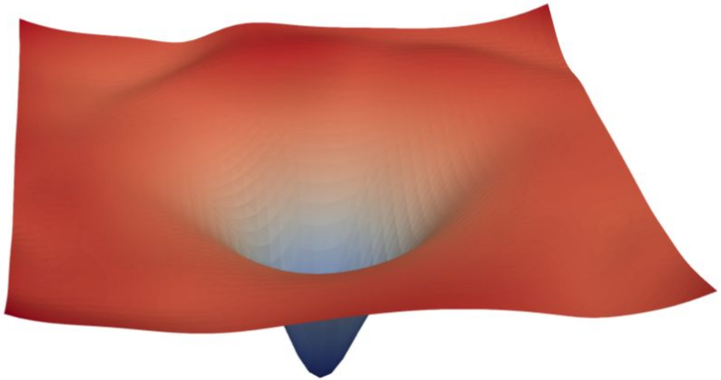
https://udlbook.github.io/udlfigures/

Visualizing the Loss Landscape of Neural Nets

Hao Li[1], **Zheng Xu**[1], **Gavin Taylor**[2], **Christoph Studer**[3], **Tom Goldstein**[1]
[1]University of Maryland, College Park [2]United States Naval Academy [3]Cornell University
{haoli,xuzh,tomg}@cs.umd.edu, taylor@usna.edu, studer@cornell.edu



(a) without skip connections



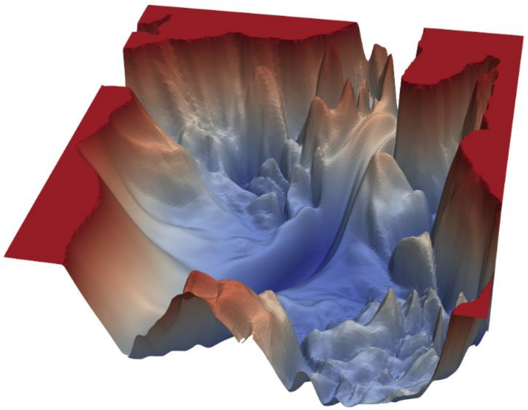
(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

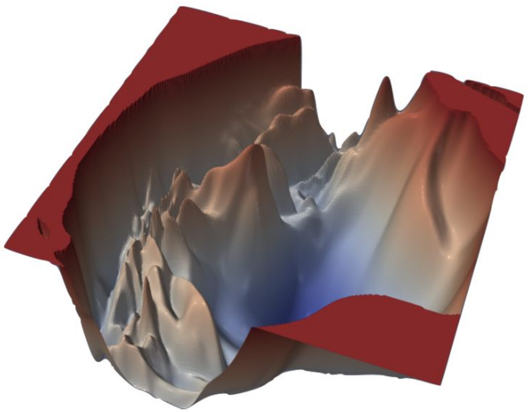
32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

https://www.cs.umd.edu/~tomg/projects/landscapes/

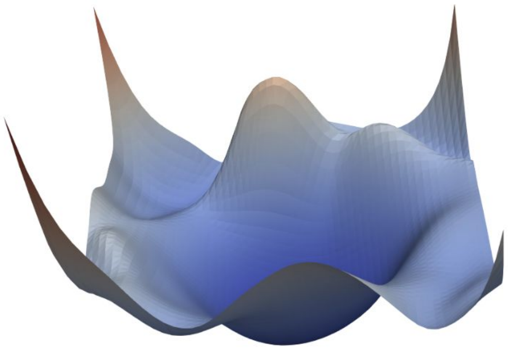
VGG-56



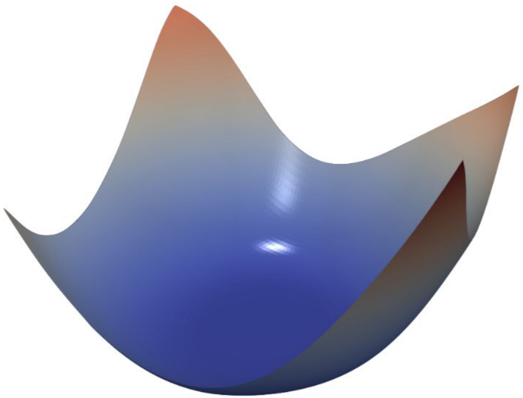
VGG-110



Renset-56



Densenet-121



For next class:

- Pick a paper from NeurIPS ML4PS workshop 2024
- Write about it in a 1-slide summary

Coming up:

- Problem Set 1 will be posted this Friday and due 1 week later
- Then you'll have 2 weeks to work on Project 1
- And so on...