Throughput Machine Learning in HTC

Ian Ross

Data Engineer, Center for High Throughput Computing

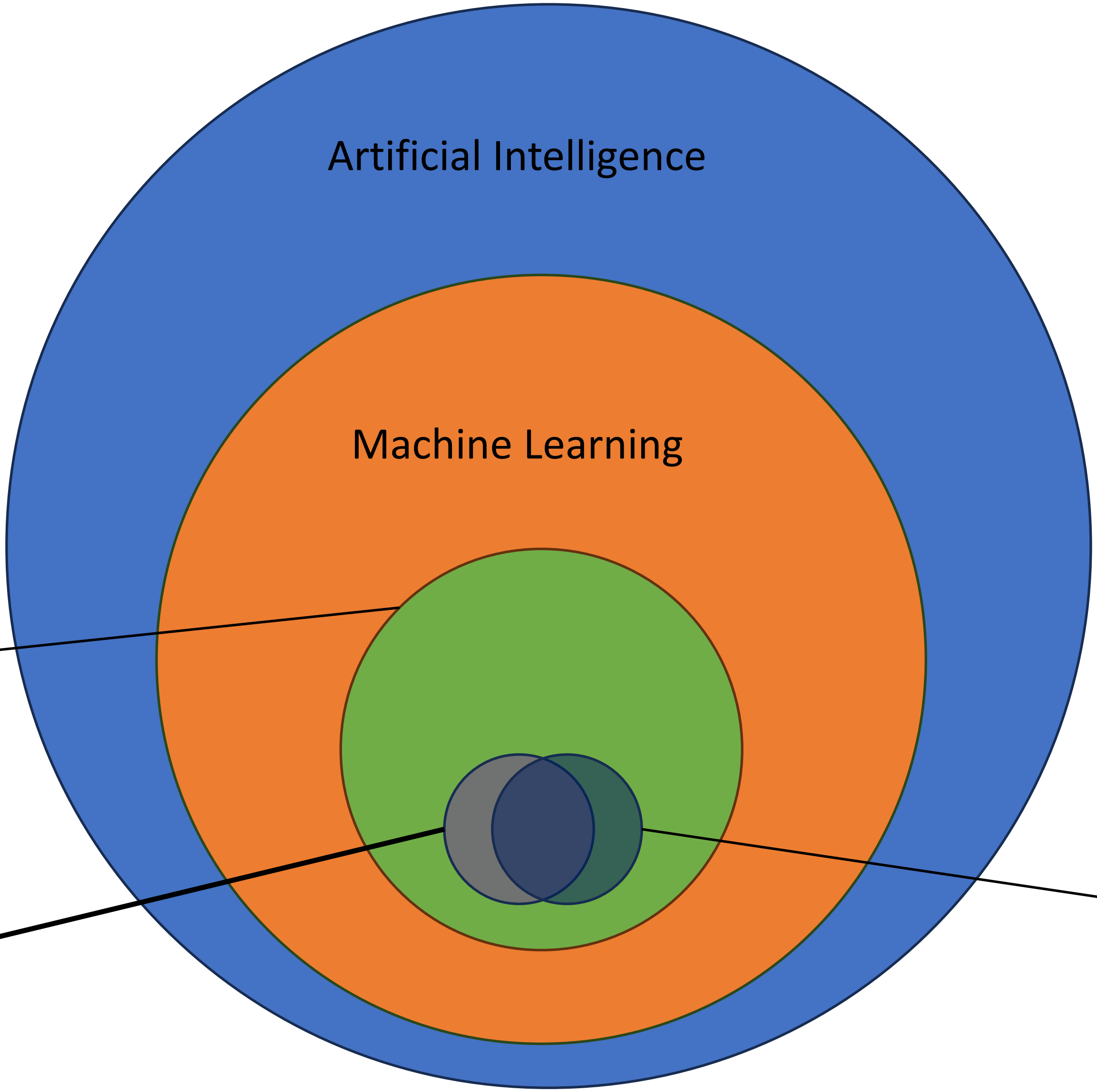
CHTC Center For High Throughput Computing

Outline

- Artificial Intelligence and Machine Learning – a too-brief overview
- Throughput Machine Learning
- Example use cases
- ML workflows and usage in CHTC
- Ongoing work
- Future plans

AI/ML – a too-brief overview

- Artificial intelligence – Methods and software to enable machines to *observe, identify, and react to stimuli to achieve a defined goal*
- Machine learning – Algorithms and practices to enable machines to recognize patterns in data and generalize to new data to achieve tasks *without instruction*
    - Subset of AI



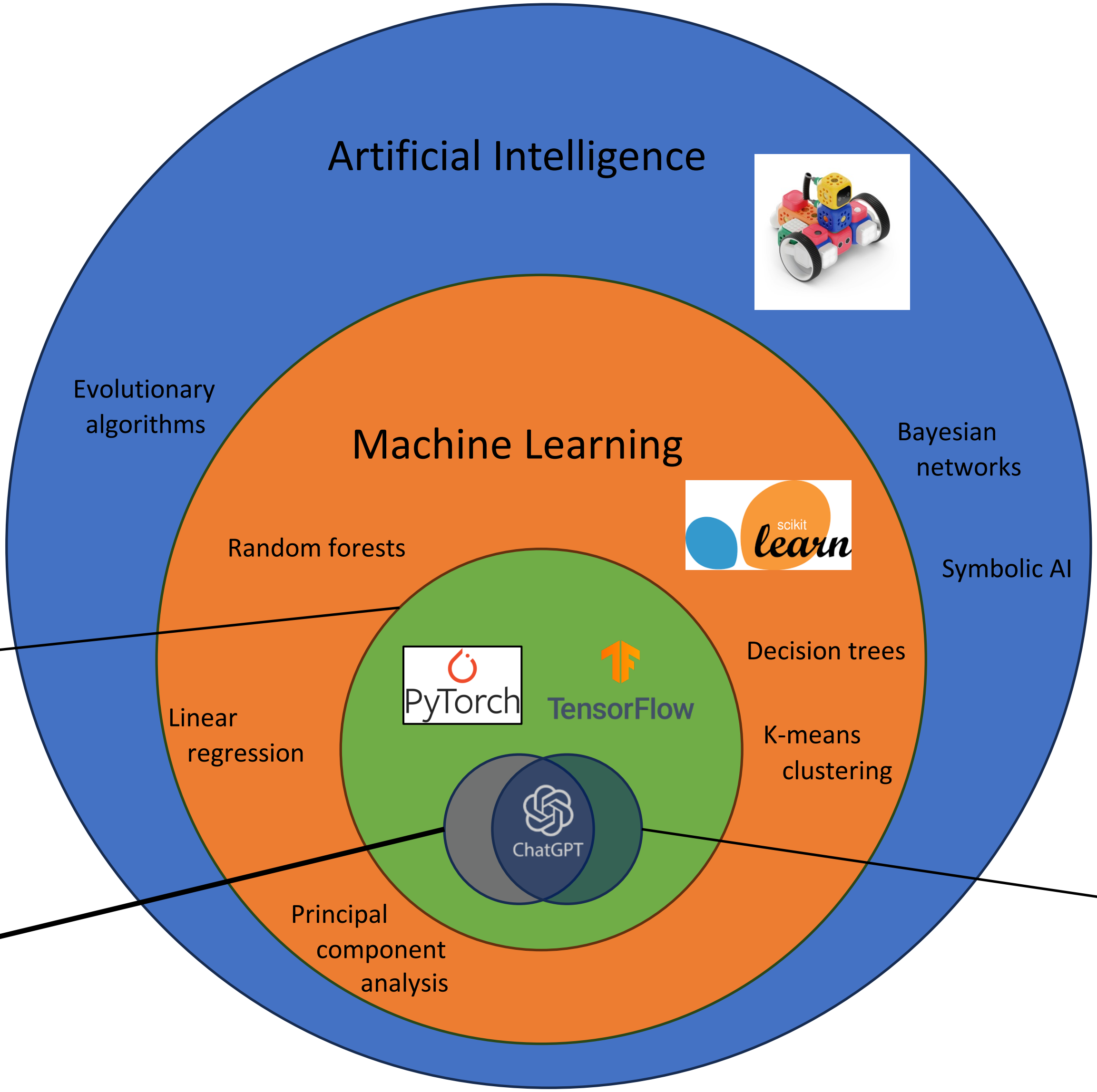
Deep learning

Using **artificial neural networks** to capture connections and strengths (weights) between inputs and outputs

Generative AI

Models trained to *generate* data similar to that used to train

Large language models



Deep learning

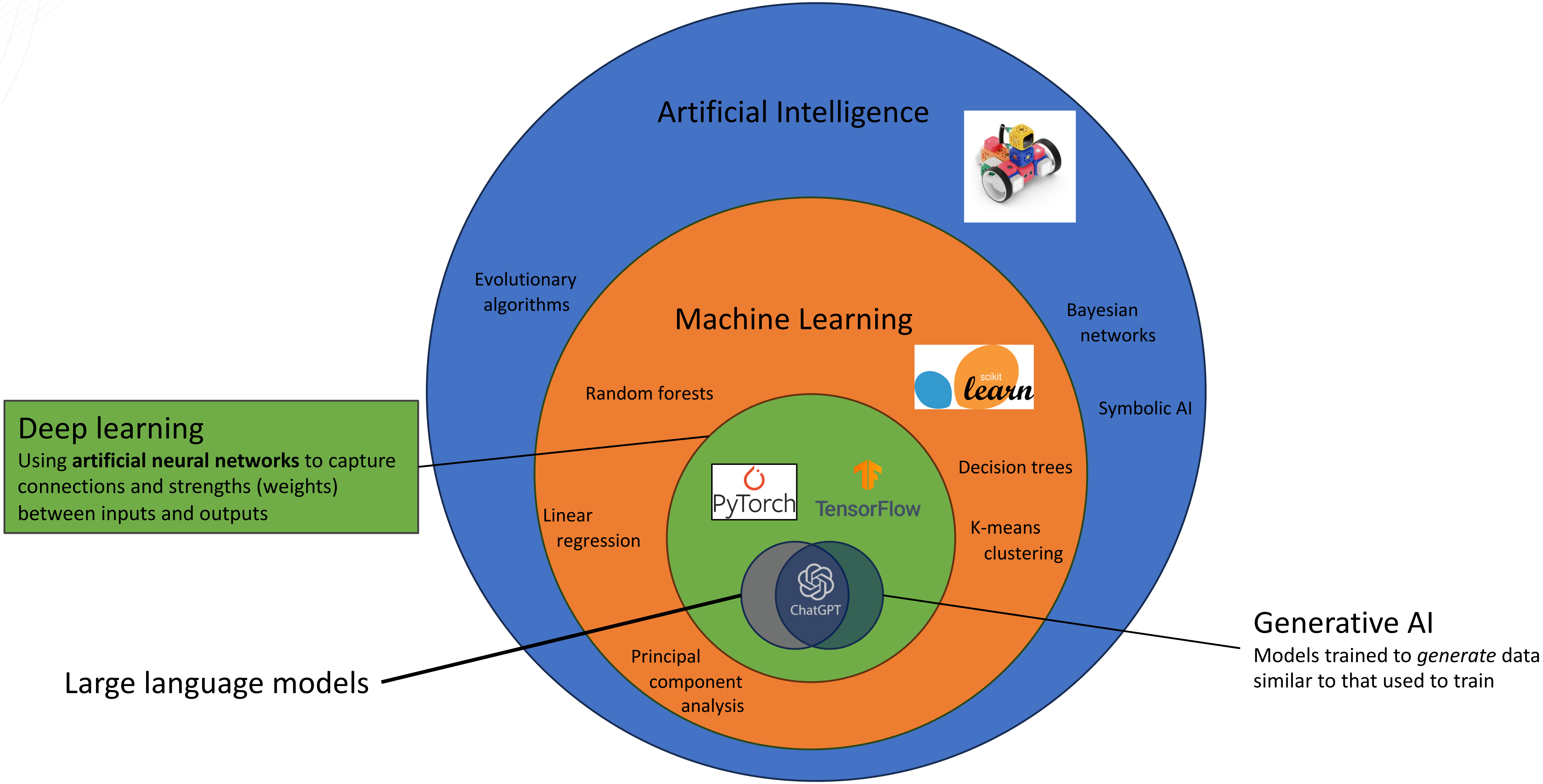
Using **artificial neural networks** to capture connections and strengths (weights) between inputs and outputs

Generative AI

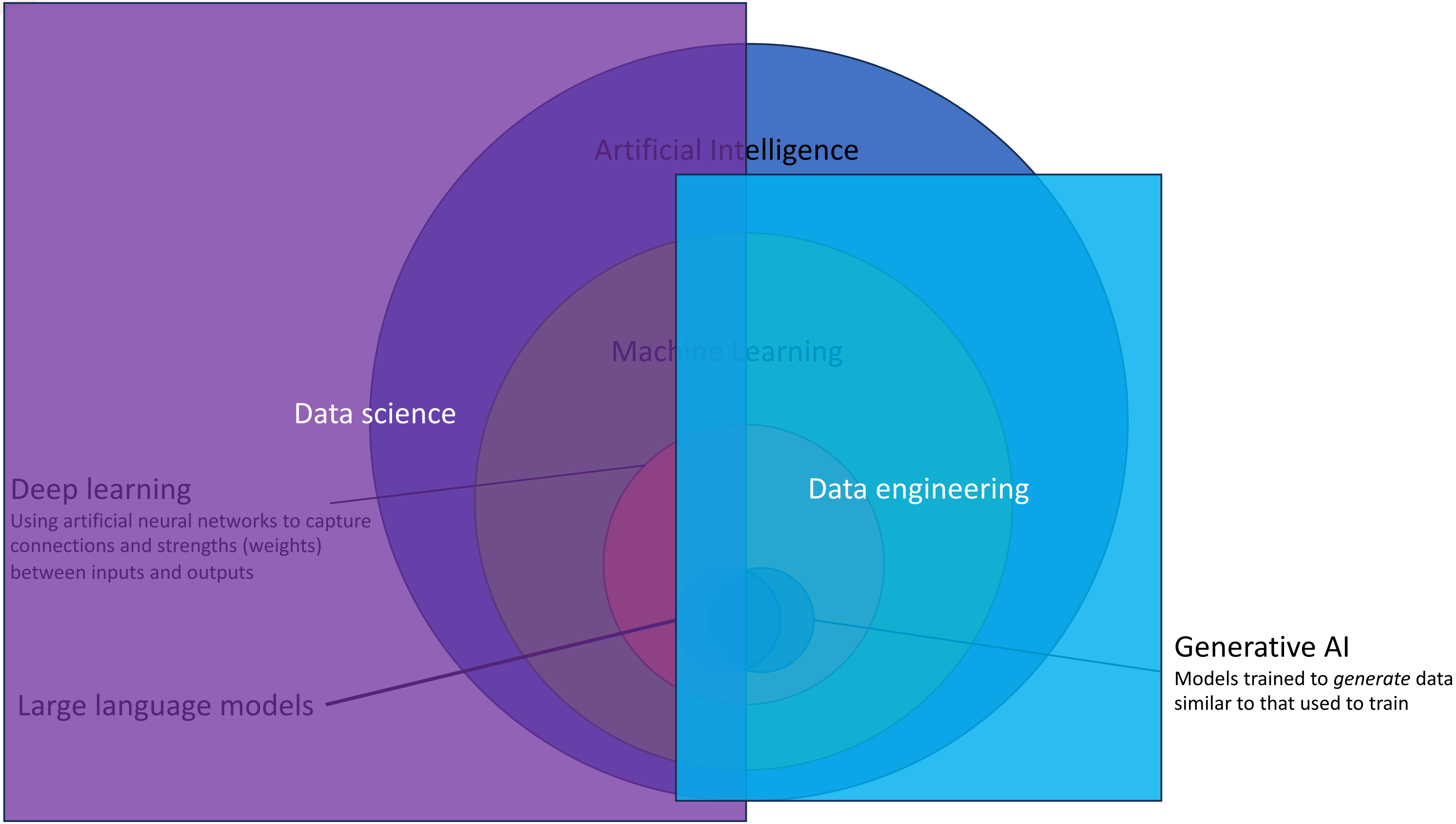
Models trained to *generate* data similar to that used to train

Large language models

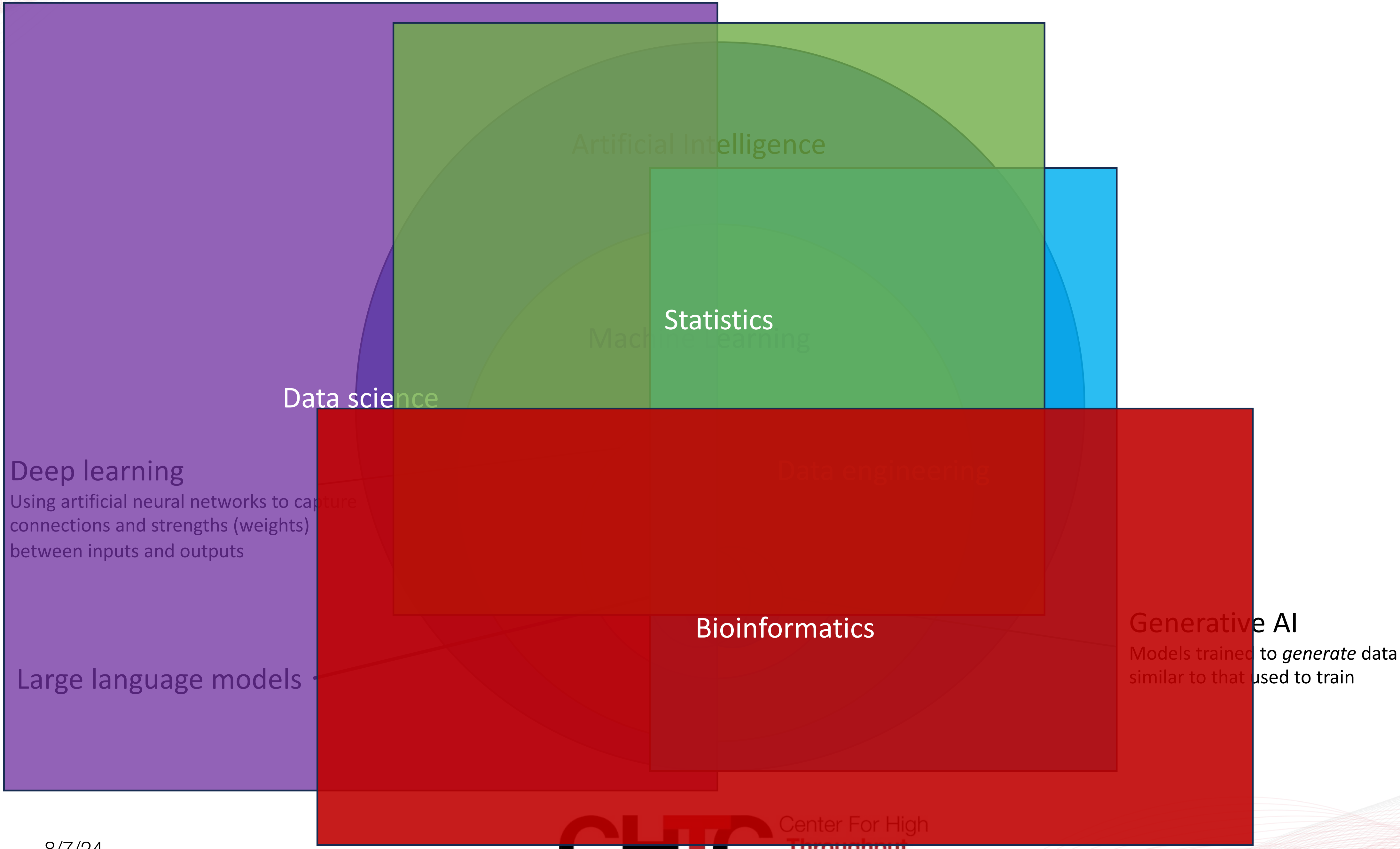
Toy robot photo by Robo Wunderkind on Unsplash



Toy robot photo by Robo Wunderkind on Unsplash



Artificial Intelligence
Machine Learning
Data science
Data engineering
Deep learning
Using artificial neural networks to capture connections and strengths (weights) between inputs and outputs
Large language models
Generative AI
Models trained to *generate* data similar to that used to train

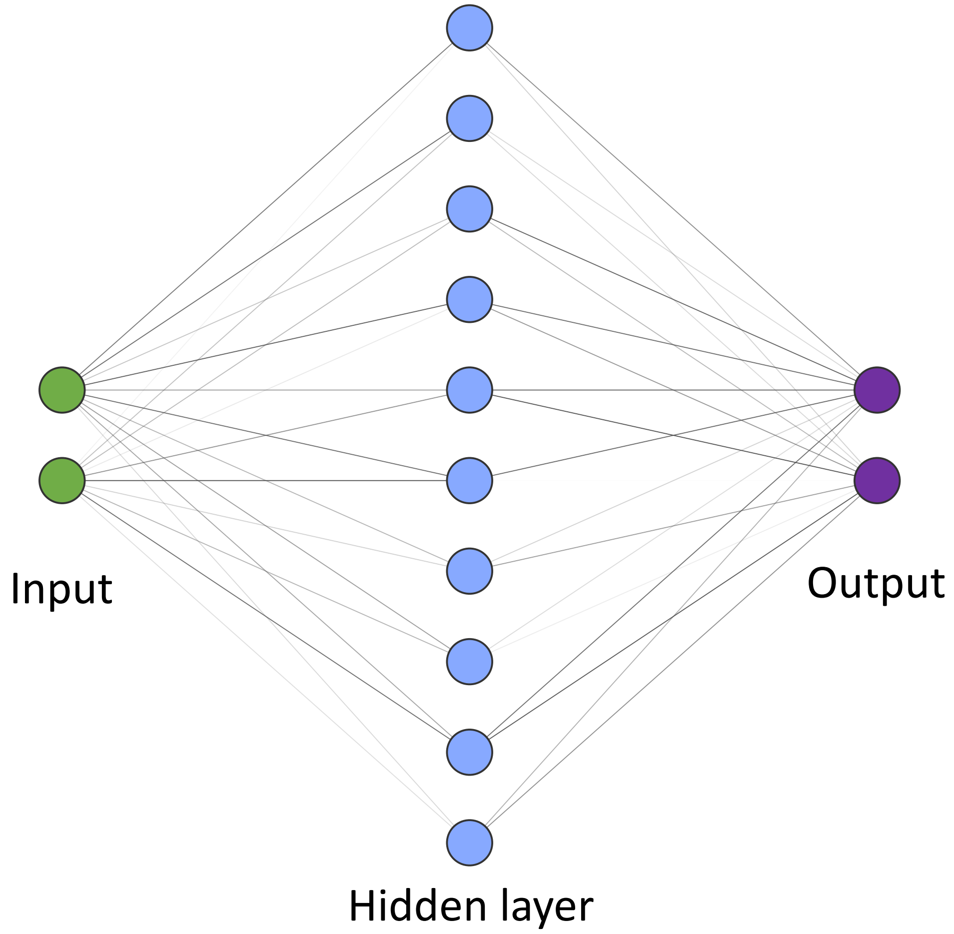

Artificial Intelligence
Statistics
Machine Learning
Data science
Data engineering
Deep learning
Using artificial neural networks to capture connections and strengths (weights) between inputs and outputs
Large language models
Bioinformatics
Generative AI
Models trained to *generate* data similar to that used to train

We care about these things primarily as tools and techniques that enable new and novel SCIENCE

…but there are foundations to understand (and complications to tackle!)

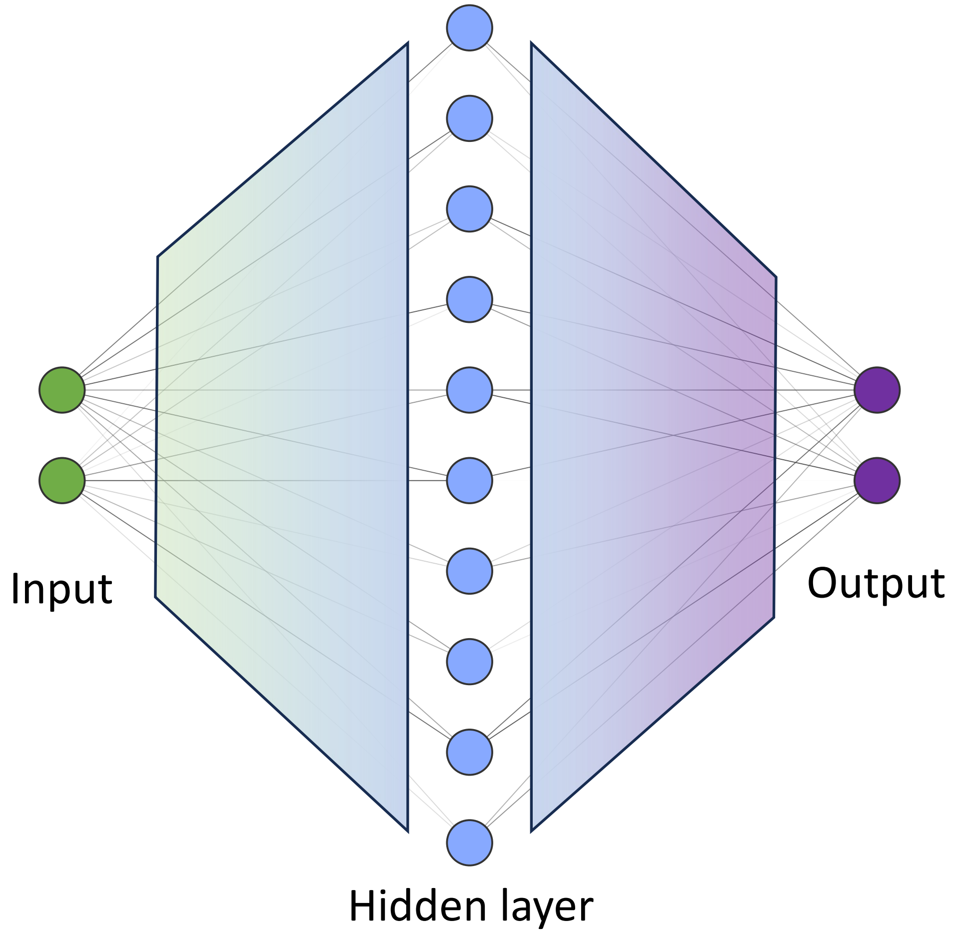
Neural networks, simplified

- Historically analogus to neurons in a brain, where electrical signals spark pathways to carry signals
- Artificial neurons are arranged into *layers*, (input, output, or hidden)
- The *activation* (value) of a neuron depends on the values of the *previous layer* and *predefined weights*



Neural networks, simplified

- These *weights* are tuned during *training*, in order to maximize success for the defined task
- These architectures can get *very* complicated, and this is where a lot of recent innovation is happening
  - Attention mechanisms (transformers), (R)-CNNs, LSTM, …

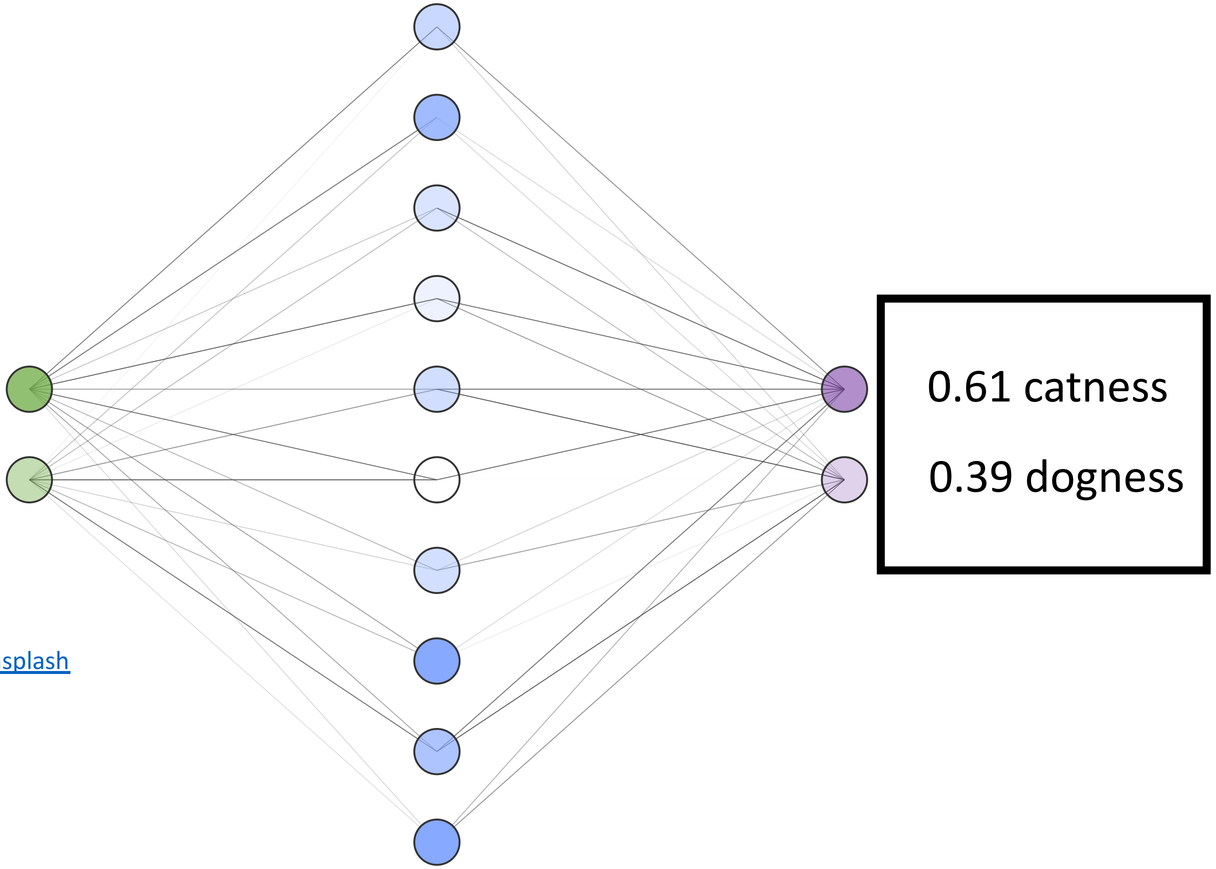


Dog/cat classifier “example”

- Imagine we had thousands of pictures of our pets (and our friends’ pets and our friends’ friends’ pets and…)
- We might initialize a set of weights and then start the training process…

Training, oversimplified

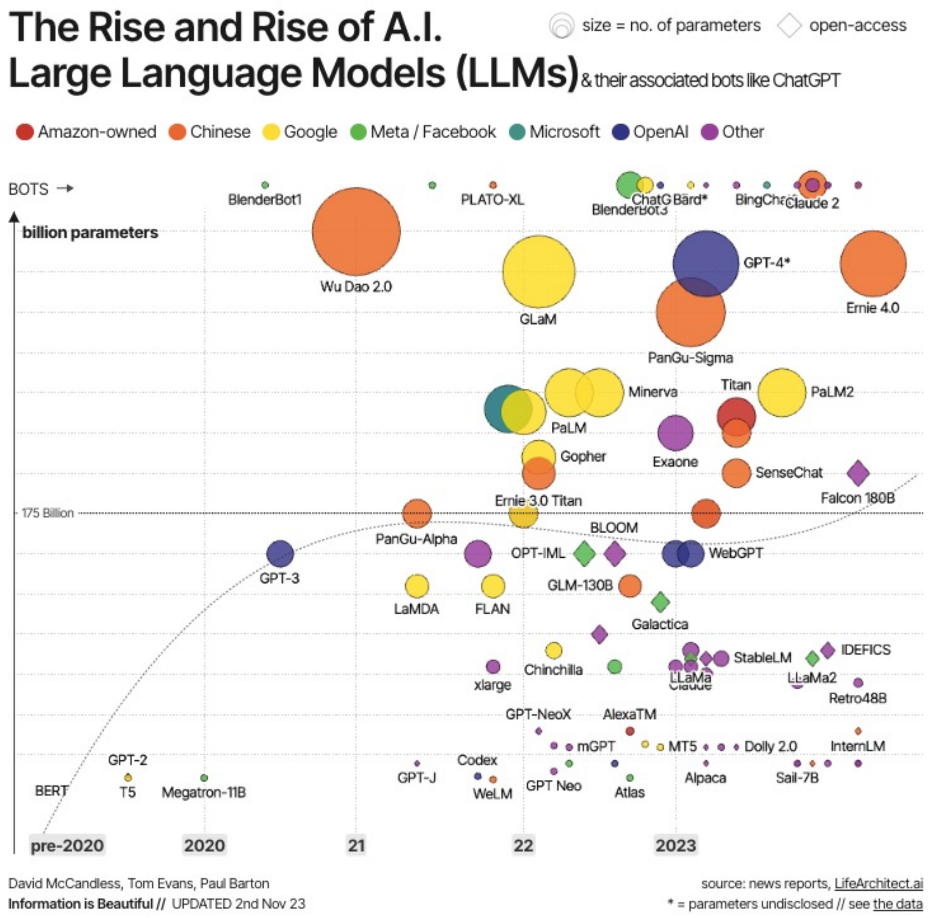
Dog photo by Matt Bango on Unsplash



Compare these values to the truth across the entire training dataset, calculate how the predictions change as we shift the weight parameters slightly, move in “correct” direction, repeat. And repeat. And repeat…

What makes training *so hard*?

- Size
  - Llama3.1 8B – 1.46 *million* GPU hours
  - Llama3.1 405B – 30.85 *million* GPU hours
- Algorithms
  - Regularization, optimization, back-propagation
- Iterative process
  - Convergence, hyperparameter tuning

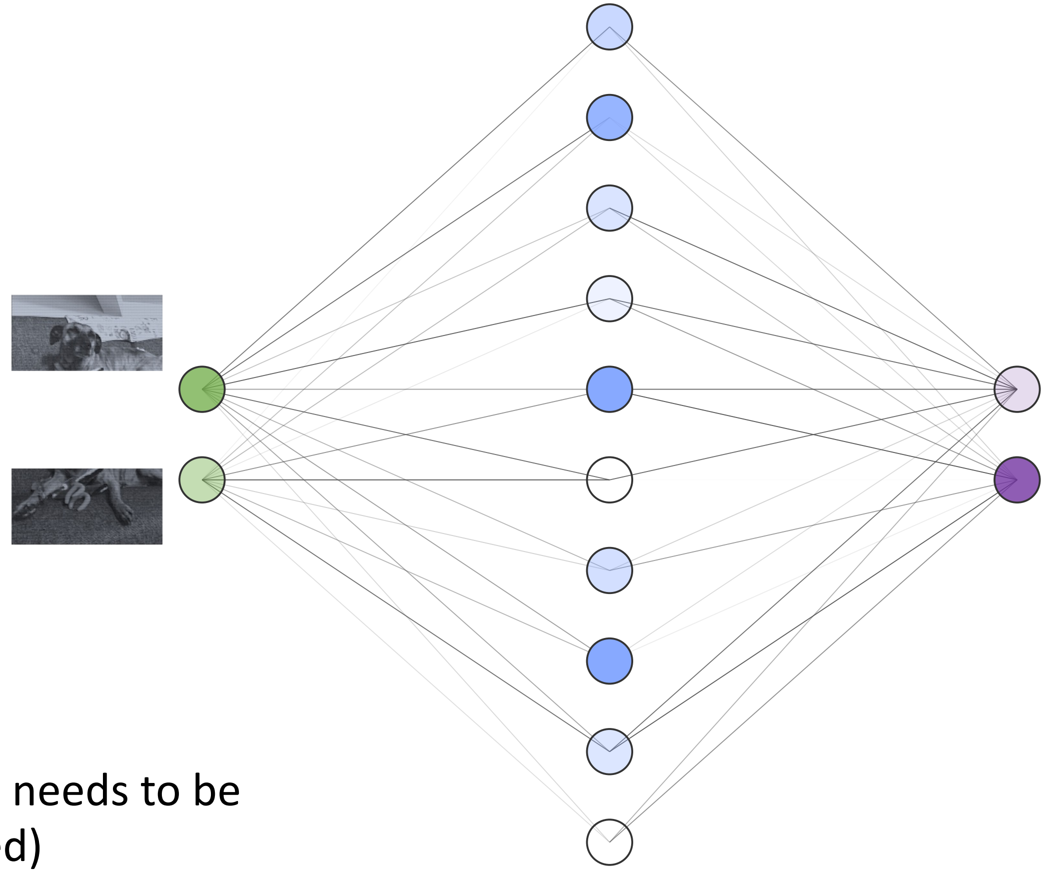


Back to dogcats..

- We're now happy with our model. Now it's time to deploy it!
- We'll pass in some photos from CHTC's social channel (*not from the training set*) and see what our model *infers* from these pictures

Best friends photo by Alec Favale on Unsplash

Inference tests

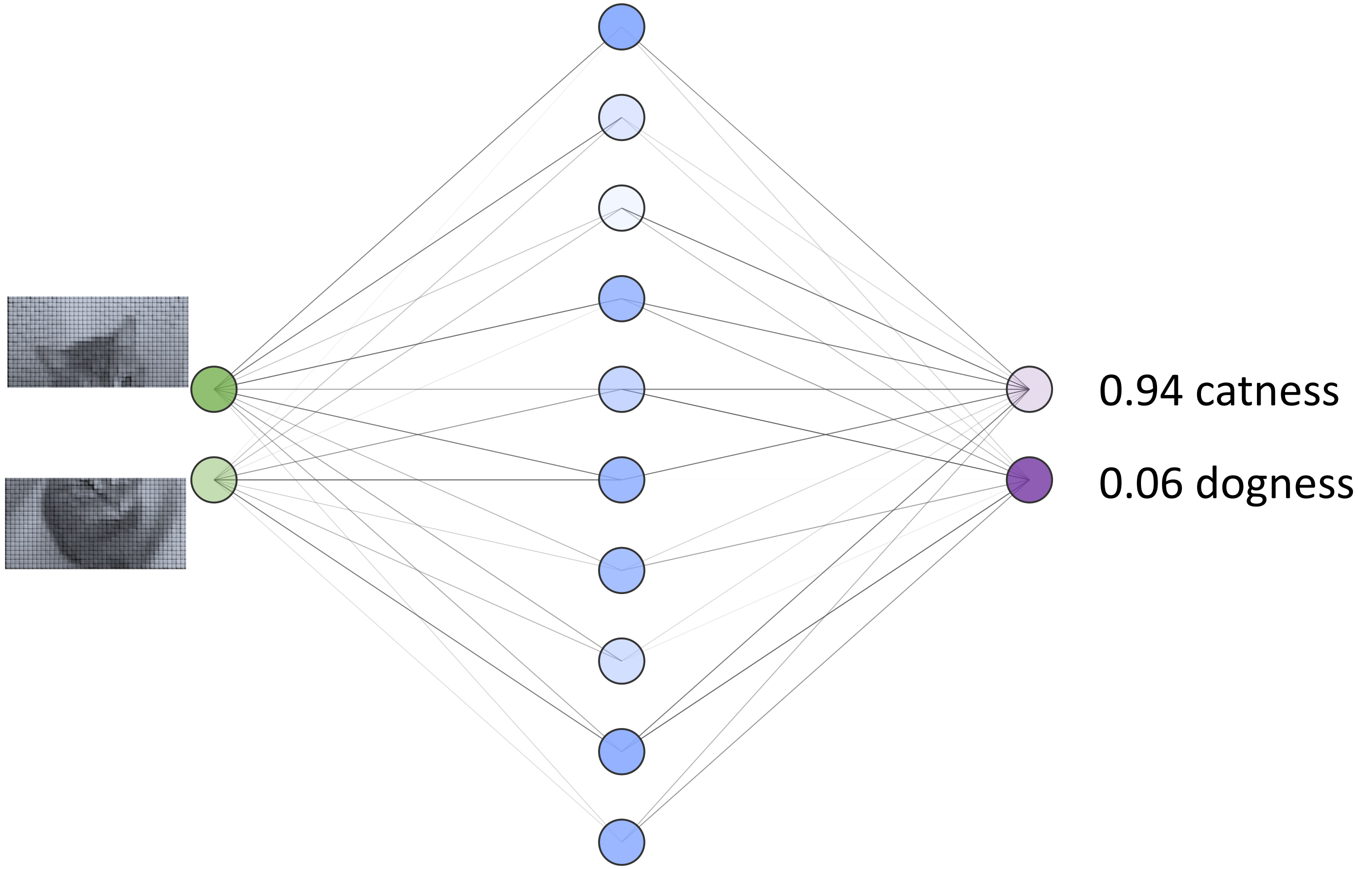


0.01 catness

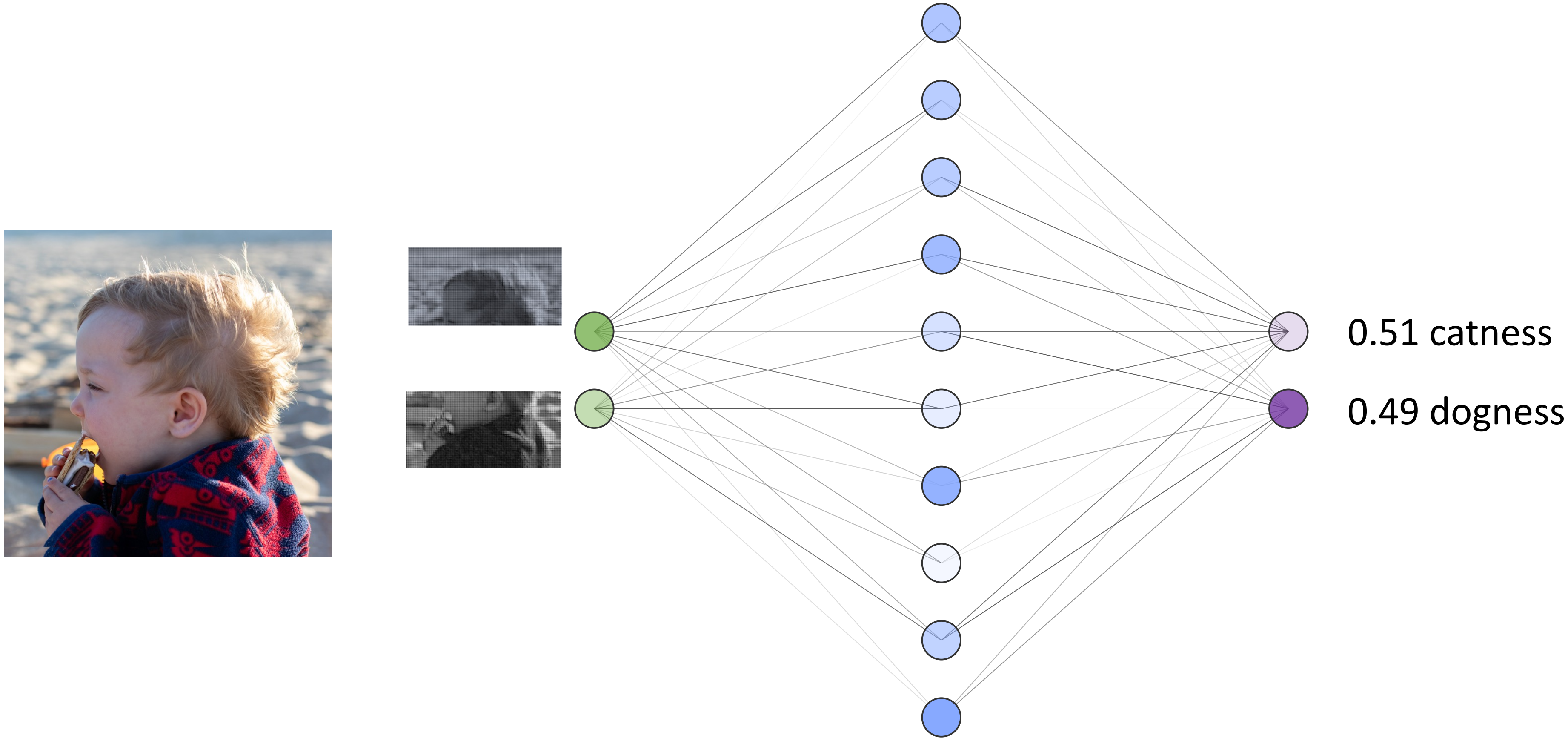
0.99 dogness

Input data (in training too!) needs to be preprocessed (e.g. tokenized)

Inference tests



Inference tests



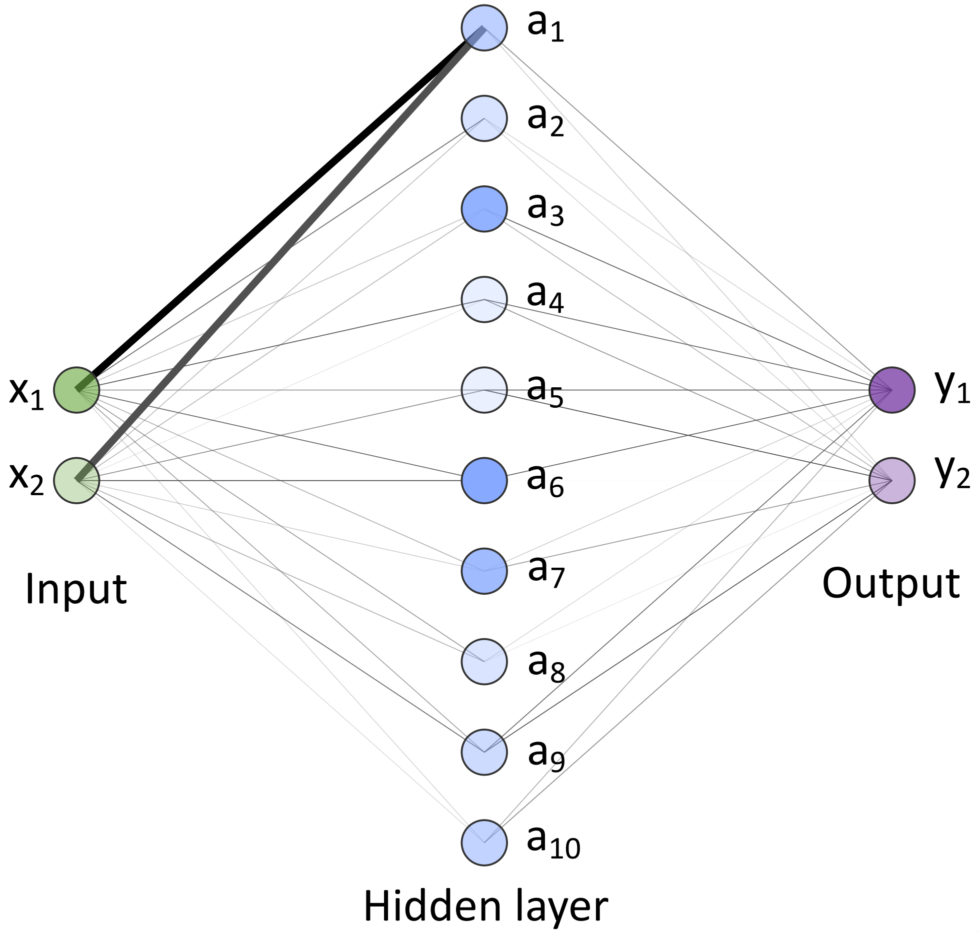
Limits of our model

- These weights are trained on our pets dataset. Weights (and therefore the hidden layer) of this model are designed to capture 💫✨*pet essence*✨💫.
- Our model *does not know how to identify anything else.* A picture of my son is *outside of the training distribution* and we should not expect it to perform in this case.
- Let's take a closer look at what this inference requires computationally..

A bit of math…

- Nodes are a linear combination of weighted nodes from the previous layer:

$$a_1 = W_{11} * x_1 + W_{21} * x_2$$



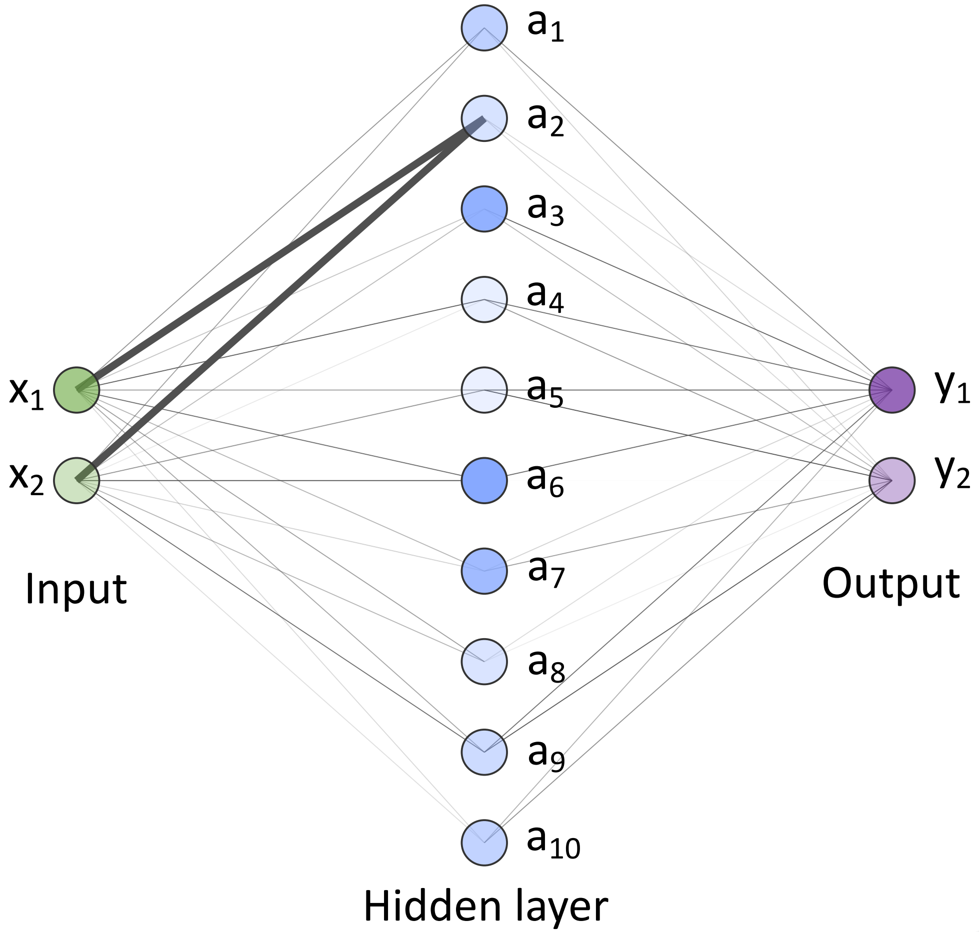
$a_1 = W_{11} * x_1 + W_{21}*x_2$

$a_2 = W_{12} * x_1 + W_{22}*x_2$

…

$a_n = W_{1n} * x_n + W_{2n}*x_n$

…and similar for the y_n



So… why GPUs?

GPUs are successful because they do one thing really well:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

Inference and training often boil down to ***parallelized matrix multiplication***, which GPUs excel at. *Problem solved, right?*

AI and ML – remaining challenges

- GPUs bring their own layer of complexity
  - Dropouts, user education, administration, cost, availability
- Data and computing needs
  - Size and scale aren't unique to ML, but ML work tends to be on the heavy side
- Training can be a long process
  - Checkpointing required, especially in the OSPool

CHTC Center For High Throughput Computing

AI and ML – remaining challenges

- GPUs bring their own layer of complexity

These are not **new** challenges! Defining resource needs, scheduling, data movement, workflow orchestration, learning new technologies… Sound familiar?

  - Checkpointing required, especially in the OSPool

Throughput machine learning in HTC

- Throughput machine learning applies our tried and tested distributed computing technologies to ML applications
  - Data movement
  - Checkpointing
  - Workflow management (DAGMan)
- Resources (GPUs) are a bit more “valuable” and policy is king
  - Prioritization, scheduling, pre-emption

Throughput machine learning – Use case 1

I have 18 million scientific articles, and I want to search for, extract, and synthesize visual artifacts across them!

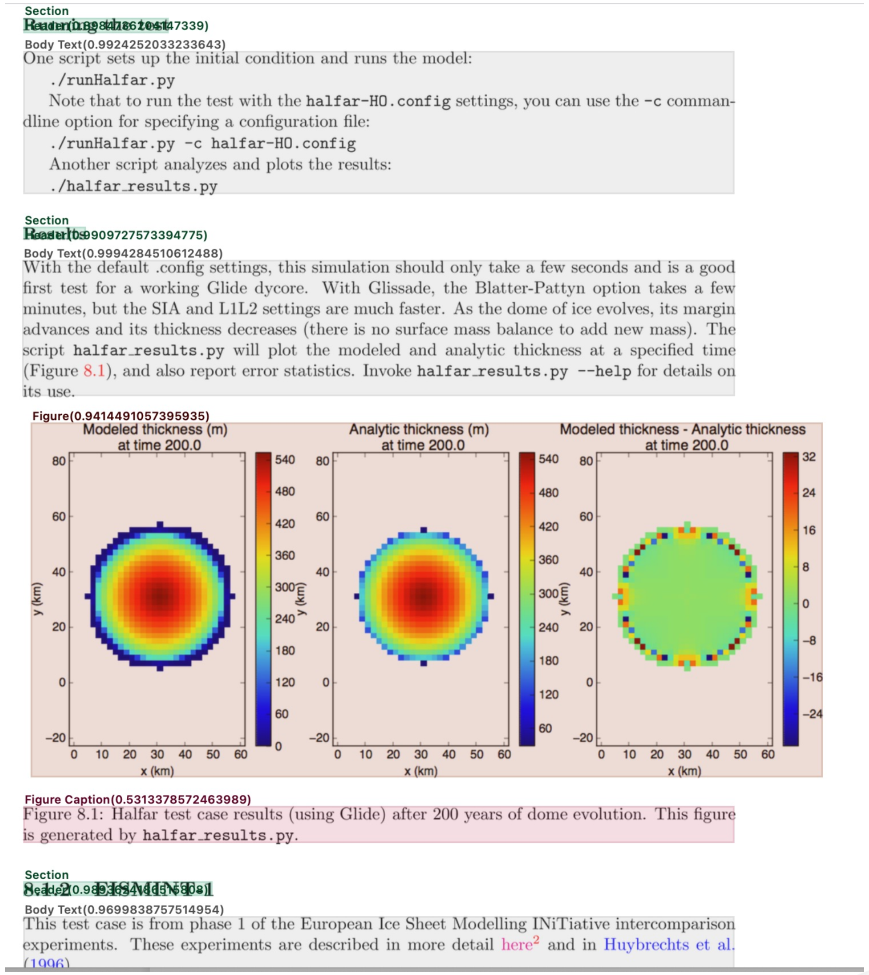
Is this a throughput workflow?

What is the smallest, self-contained computational task that is part of your work? How big is the list of these tasks? What is needed to run one task?

Scaling out with HTC

The atomic task is running our COSMOS visual pipeline on an individual PDF. Each PDF is on the order of 1 MB, produces 4 MB of output, and takes 5 seconds on a GPU or 5 minutes on a CPU. The model is ~8GB, and I have a docker container ready to go. However, these PDFs are bound by publisher agreements and can't leave UW campus.

Great! This sounds ideal for CHTC, with standard input file transfer mechanisms. Have fun!



Section

Body Text(0.9924252033233643)

One script sets up the initial condition and runs the model:

```
./runHalfar.py
```

Note that to run the test with the halfar-HO.config settings, you can use the -c commandline option for specifying a configuration file:

```
./runHalfar.py -c halfar-HO.config
```

Another script analyzes and plots the results:

```
./halfar_results.py
```

Section

Body Text(0.9994284510612488)

With the default .config settings, this simulation should only take a few seconds and is a good first test for a working Glide dycore. With Glissade, the Blatter-Pattyn option takes a few minutes, but the SIA and L1L2 settings are much faster. As the dome of ice evolves, its margin advances and its thickness decreases (there is no surface mass balance to add new mass). The script halfar_results.py will plot the modeled and analytic thickness at a specified time (Figure 8.1), and also report error statistics. Invoke halfar_results.py --help for details on its use.

Figure(0.9414491057395935)

Figure Caption(0.5313378572463989)

Figure 8.1: Halfar test case results (using Glide) after 200 years of dome evolution. This figure is generated by halfar_results.py.

Section

Body Text(0.9699838757514954)

This test case is from phase 1 of the European Ice Sheet Modelling INiTiative intercomparison experiments. These experiments are described in more detail here[2] and in Huybrechts et al. (1996)

CHTC Center For High Throughput Computing

Throughput machine learning – Use case 2

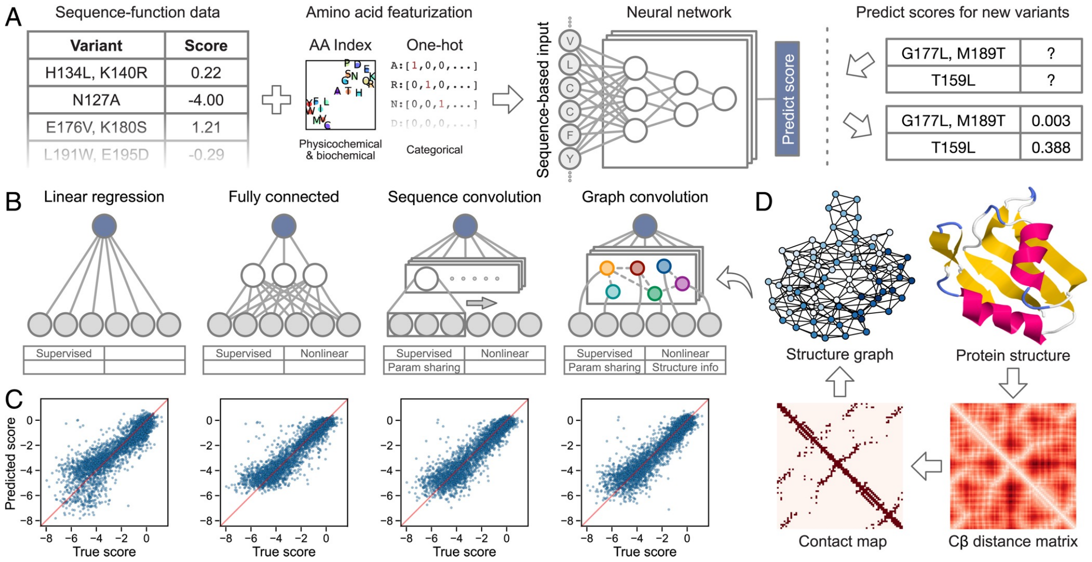
I want to train many models, empirically measure their predictive power, and use those models to drive scientific exploration.

Is this a throughput workflow?

What is the smallest, self-contained computational task that is part of your work? How big is the list of these tasks? What is needed to run one task?

Scaling out with HTC

The atomic task is training a single model from our dataset, which is 10GB. We want to test many different architectures, but anticipate GPU runtimes on the order of days for each model. We want to test as many model architectures as possible. CPU, memory, and disk requirements are minimal.

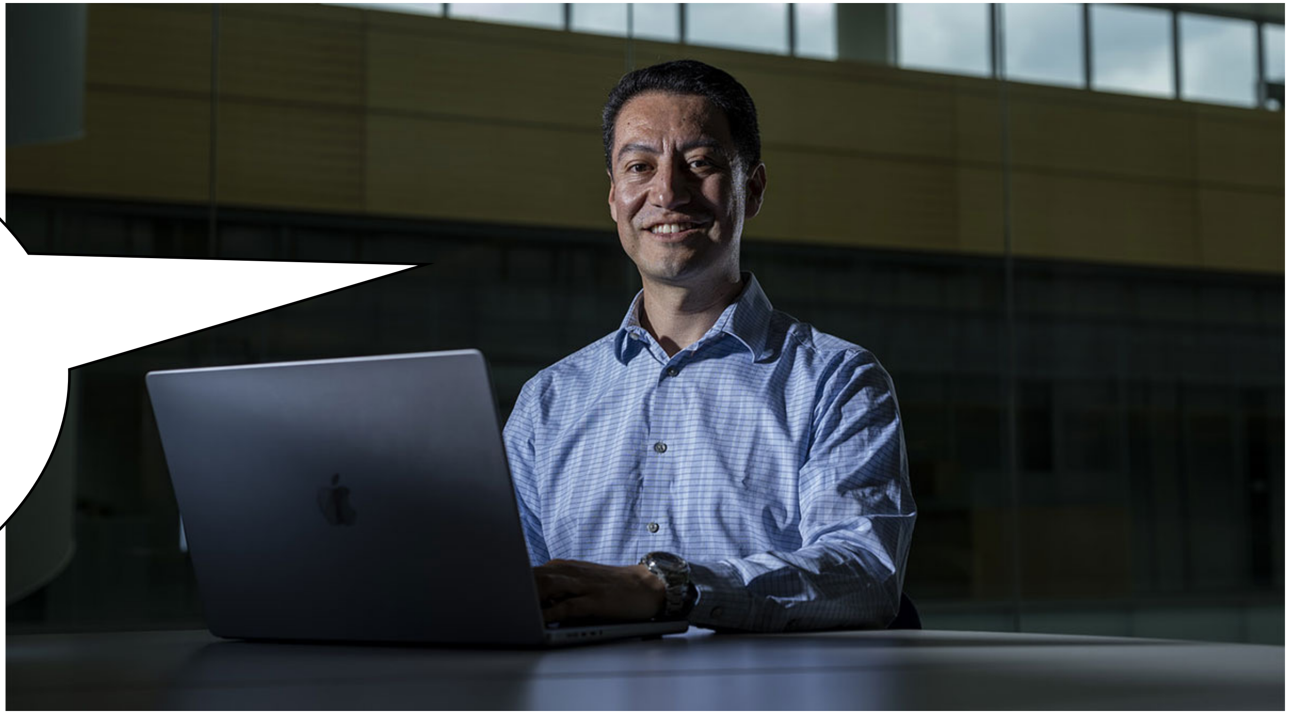


https://www.pnas.org/doi/full/10.1073/pnas.2104878118

Welcome to the OSPool! Let's learn about OSDF and job checkpointing!

Throughput machine learning – Use case 3

I want to create a foundation model for bioimaging and want to scale training across multiple nodes!



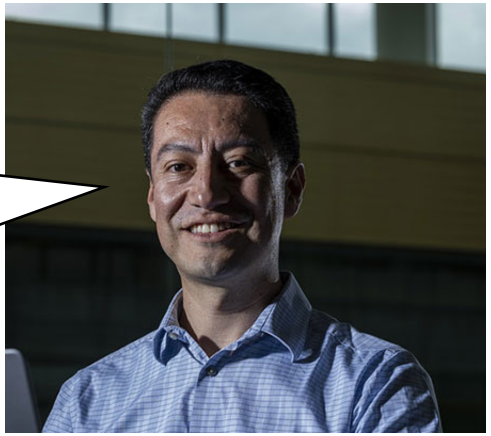
Is this a throughput workflow?

What is the smallest, self-contained computational task that is part of your work? How big is the list of these tasks? What is needed to run one task?

CHTC Center For High Throughput Computing

Scaling out with HTC?

Our dataset is 2TB. The model architecture we want to use is too big to fit on one GPU, and the memory needs are on the order of 128GB.



This isn’t a great fit for our usual computing philosophy, but let’s talk more and work together to see what we can do!

Where to go from here?

- Think about your workflow!
- A world of education opportunities
  - Pytorch, HuggingFace, <your software of choice> documentation
  - YouTube for explanations
  - arXiv for preprints
- Explore available pre-make images: Docker Hub, NGC catalog
- OSPool documentation
- CHTC documentation

GPU submit keywords

- **`request_gpus = 1`** the same way you would CPUs, memory, or disk
- **`Require_gpus = (Capability > 7.5)`** to require a certain [CUDA Compute Capability]

- New in HTCondor 23.x:
  - **`gpus_minimum_capability = 7.5`**
  - **`gpus_minimum_memory = 40000`**
  - To do this in earlier versions:

```
require_gpus = (Capability >= 7.5) && (GlobalMemoryMb >= 40000)
```

CHTC Center For High Throughput Computing

Questions?

- Talk to us! Don't let computing be a barrier to your research!