



# Safety in Generative AI

## DSC 291 2025 Fall

Yu-Xiang Wang



Check-in here

# A bit about me

**Yu-Xiang Wang 王宇翔**

Associate Professor

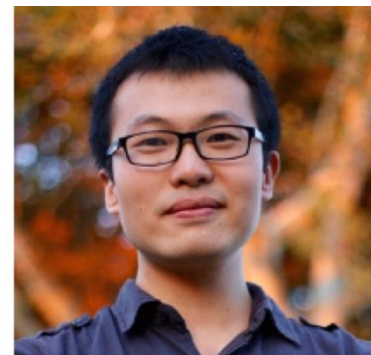
Halicioğlu Data Science Institute  
and Department of Computer Science and Engineering

UC San Diego

*Office:* HDSI 352

*E-mail:* yuxiangw AT ucsd.edu

*Yu-Xiang* is pronounced approximately as ['ju:'ʃi:ʌŋ],  
namely, *y~eu~ee - sh~ih~ah~ng* .



**Research area:** Statistical Machine Learning. Optimization, reinforcement learning, differential privacy, deep learning.

## **Short biography:**

China => Singapore

=> PhD from Carnegie Mellon University

=> Scientist at Amazon

=> Professor at UC Santa Barbara

=> Professor at UC San Diego

## **Homepage:**

<https://cseweb.ucsd.edu/~yuxiangw/>

# Meet your TAs



Erchi Wang

Office Hour: Thursday 11 am  
in HDSI First Floor



Yingyu Lin

Office Hour: TBD

# Outline today

- How does this course work?
- Discussion on AI Safety
- Deep Learning Refresher

# DSC 291 Safety in Generative AI (Fall 2025)

Quarter: Sep 30 → Dec 4, 2025 · Tue/Thu · 80 minutes each

## Course Overview

INSTRUCTOR:

Prof. Yu-Xiang Wang

TEACHING ASSISTANTS:

Yingyu Lin, Erchi Wang

TA OFFICE HOUR:

TBD

LECTURES:

Tuesday and Thursday 9:30 AM - 10:50 AM at PCYNH 121

RESOURCES

[[syllabus](#), [Piazza](#), [Gradescope](#), [Canvas](#)]

## Important Dates

- **Thu 9/25:** No class (instructor traveling)
- **Tue 11/11:** No class – Veterans Day
- **Thu 11/27:** No class – Thanksgiving
- **Tue 12/2:** No class – (NeurIPS)
- **Thu 12/4:** In-class Quiz
- **TBD (between 12/8-12/12)** Mini-Symposium - Project presentations

## Your weekly routines

- **Lectures:** In-person attendance required.
- **Weekly homework** Done in group of 4. Due each saturday.
- **Readings:** Due before the Thursday lecture each week. You need to update a reading log (with one-paragraph summary of what you've read).

## Project Timeline

- **Weeks 1–2:** team formation & idea brainstorming
- **Thu 10/30: Project Proposal Due** (2 pages)
- **Thu 11/20: Midterm Project Update** (5-minute check-in)
- **TBD (between 12/8 to 12/12): Final Presentation + Written Report Due**

## Evaluation Breakdown

- 5% Weekly reading log
- 5% Class Attendance / Participation
- 40% Homework (5% each, 8 total)
- 25% Project
- 25% Final Quiz
- **Bonus 5% In-class presentation (limited spots — sign up early)**

# High-level Schedule of the course

- Part 1: Generative AI
  - Deep Learning
  - Large Language Model
  - VAEs and Diffusion Models
  - Using GenAI as an Agents
  - Foundation models: Pretraining and Post-training
- Part 2: Safety Risks and Mitigation
  - Jailbreaking / Prompt Injection
  - Deepfake and Plagiarism
  - Data poisoning and model collapse
  - Existential Threats of AI
- Part 3: Watermarking

## Weekly Schedule

[Open all](#)[Close all](#)[Print](#)

### ▼ Pre-Week · Thu 9/25/25 — No Lecture (instructor traveling)

- **Reading / videos:** [[Hinton interview](#), [Tucker Carlson-Sam Altman](#), [Davos AI Panel \(LeCun, Russell and others\)](#) ]

### ▼ Week 1 · Foundations of GenAI (Tue 9/30, Thu 10/2)

#### Reading (due before Thursday lecture):

- **Required Reading:** [The Foundation Model](#) (Read the intro and at least one subsection from Section 2, 3, 4, or 5.)
- **Optional Reading:** Sutton's [The Bitter Lesson](#); [GPT-2 Multitask Learners](#); [GPT-3 Few-Shot Learners](#); [Sparks of AGI](#); [Emergent Abilities](#); ; [Are Emergent Abilities a Mirage?](#)

#### Tue 9/30 – Course Intro, Deep Learning Refresher

- **Mini-lecture:** course overview, deep learning basics.
- **Activities:** (1) discussion on AI safety videos, (2) vibe-coding deep learning from scratch.

#### Thu 10/2 – Foundation Models & Emerging Abilities

- **Mini-lecture:** zero-shot, few-shot / in-context learning; emergence; prompt engineering (with math + coding).
- **Activities:** (1) groups explain one emerging ability from readings; (2) vibe-code a simple in-class game; (3) groups explain one potential vulnerability.

#### Homework (due Saturday):

- **HW1:** vibe-code a ResNet image classifier (JAX/PyTorch) for MNIST.

# Lecture Attendance is required

- “Check in” when you come to the lectures
  - Scan QR code
  - Or click <https://cs-291-sign-in.erchiw.workers.dev/>
- Username: dsc291
- Password: Written on board



# Reading Assignment (weekly)

## ▼ Week 1 · Foundations of GenAI (Tue 9/30, Thu 10/2)

### Reading (due before Thursday lecture):

- **Required Reading:** *The Foundation Model* (Read the intro and at least one subsection from Section 2, 3, 4, or 5.)
- **Optional Reading:** Sutton's *The Bitter Lesson*; GPT-2 *Multitask Learners*; GPT-3 *Few-Shot Learners*; *Sparks of AGI*; *Emergent Abilities*; ; *Are Emergent Abilities a Mirage?*

- Submit a **reading log** before the Thursday lecture

“I read the introduction of “the Foundation Model” paper and Section 2.4 on Reasoning and Search. I find the use of AI for solving geometry problems fascinating.”

- Submit on “Gradescope”

# Homework are due every Saturday

## Homework (due Saturday):

- HW1: vibe-code a ResNet image classifier (JAX/PyTorch) for MNIST.

## Homework (due Saturday):

- HW2: vibe-code a one-layer Transformer (JAX/PyTorch) and train on movie-review data.

- Done in group of 4 students + AI (e.g., Gemini 2.5):
  - One submission due every Saturday midnight
- Submit on Gradescope

# Project are done in group of 4 students

## Project Timeline

- **Weeks 1–2:** team formation & idea brainstorming
- **Thu 10/30: Project Proposal Due** (2 pages)
- **Thu 11/20: Midterm Project Update** (5-minute check-in)
- **TBD (between 12/8 to 12/12): Final Presentation + Written Report Due**

- **Project Idea:**

- a deep-dive on prompt injection attacks
- Understanding the full cycle of LLMs development (estimate how much in \$ it takes for each phase)

...

# I believe in “active learning”, so I will give it a try!

- Mini-lecture
  - I cover course materials and show you what I think
- Activities:
  - Discussion
  - Hands-on (vibe) coding
- In-class student presentation: 10 min presentation slots
  - For you to present your findings from your reading or experiments.
  - **First slot next Thursday!**

Sign up sheet: <https://tinyurl.com/dsc291signup>

# Did you watch these videos?



# Activity: get into groups and chat with your classmates

- What does Geoff Hinton worry about?
- What do AI experts such as Daniella Rus, Yann LeCun, Stuart Russell agree on, and what do they not agree on?
- What are Tucker Carlson's worries?
- Which question(s) to Sam Altman made him most unease?

# Let's categorize AI Safety Threats

AI taking over

Educator Over reliance

Human abuse AI : Scams

AI failure

Societal risks/consequences

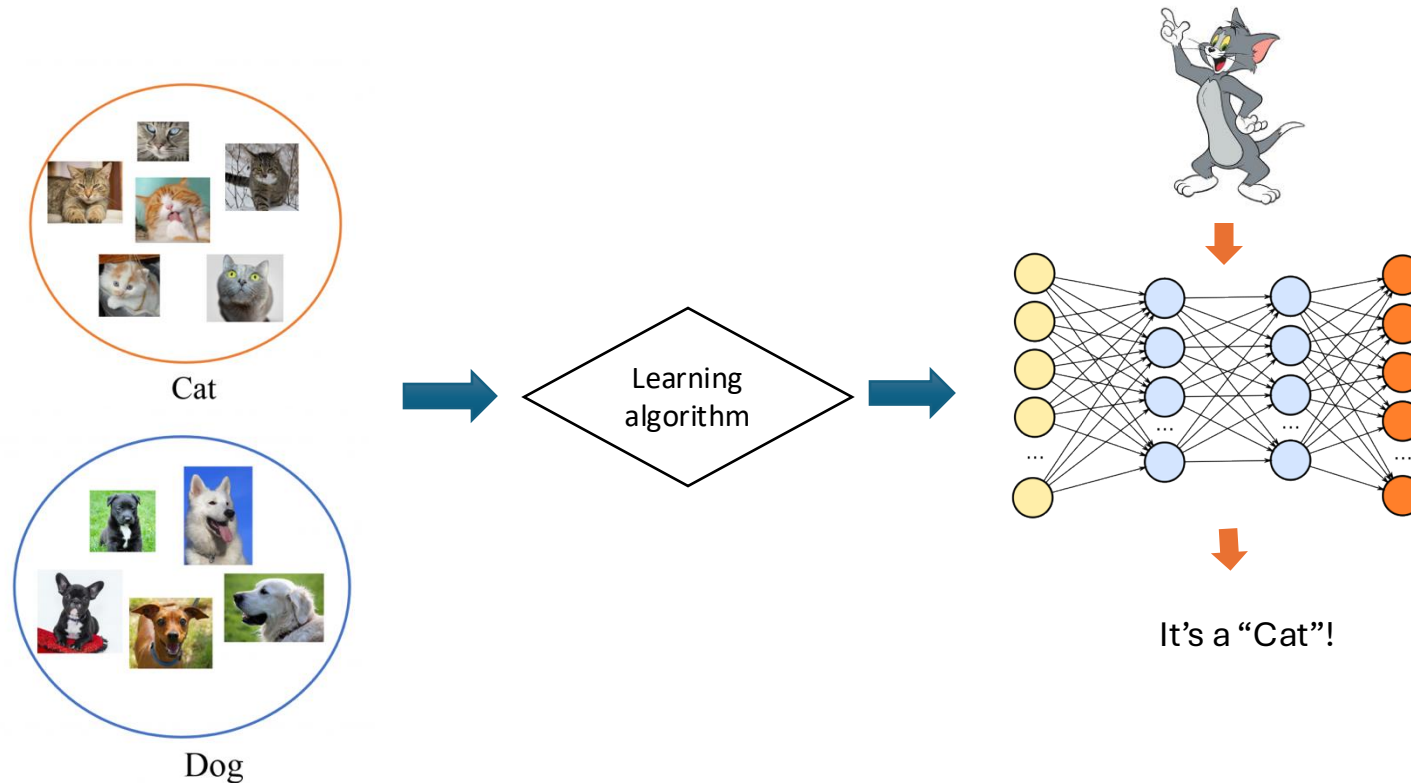
Privacy

bias, fairness in loan/mortgage

# Outline today

- How this course works?
- Discussion on AI Safety
- **Deep Learning Refresher**

# Main concepts we learned from DSC 240 (or any other Intro to ML class you took)



Features, hypothesis class, loss function, optimization

# Problem setup for machine learning problems

- Loss function

$$\ell(h, (x, y))$$

- Empirical Risk function

$$\hat{R}(h, \text{Data}) = \frac{1}{n} \sum_{i=1}^n \ell(h, (x_i, y_i))$$

- (Population) Risk function

$$R(h, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(h, (x_i, y_i))]$$

# Recap: Risk Decomposition

$$\begin{aligned} & \mathbb{E}[R(\hat{h})] - R(h_{\text{Bayes}}) \\ \leq & \underbrace{\mathbb{E}[\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})]}_{\text{Optimization Error}} + \underbrace{R(h^*) - R(h_{\text{Bayes}})}_{\text{Approximation Error}} + \underbrace{\mathbb{E}[R(\hat{h}) - \hat{R}(\hat{h})]}_{\text{Generalization Error}} \end{aligned}$$

How close am I from minimizing the empirical risk?

How much worse the best “representable” classifier is from the best classifier out there.

How different the empirical risk of my classifier is from its population risk?

# Machine learning can be viewed as a collection of techniques in minimizing the three types of errors

	Optimization error	Generalization Error	Approximation Error
Definition	$\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})$	$R(\hat{h}) - \hat{R}(\hat{h})$	$R(h^*) - R(h_{\text{Bayes}})$
Challenges	<ul style="list-style-type: none"> <li>Finding ERM for some loss functions is NP-Hard.</li> <li>Efficiency isn't enough. Need to be scalable.</li> </ul>	<ul style="list-style-type: none"> <li>We do not observe Risk!</li> <li>Don't have infinite data.</li> <li>Large generalization error <math>\Leftrightarrow</math> Overfitting</li> </ul>	<ul style="list-style-type: none"> <li>Don't know data distribution.</li> <li>No knowledge of Bayes optimal classifier.</li> <li>Large approx. error <math>\Leftrightarrow</math> Underfitting!</li> </ul>
What we have learned to address these challenges?	<p>"Just-relax" Surrogate loss, Gradient Descent, SGD</p> <p>Other more specialized solutions to optimization problems</p>	<p>Holdout, Cross-Validation</p> <p>Regularization</p> <p>Statistical learning theory</p>	<p>Better features</p> <p>More flexible decision boundaries</p> <p>Better probabilistic models</p> <p><b>Ensemble learning: Boosting, Bagging</b></p> <p><b>Feature expansion/ Kernels</b></p> <p><b>Neural Networks / Representation Learning</b></p>

# Key questions in modelling and tradeoffs

- How to come up with suitable hypothesis class?
  - We want the approximation error to be small
  - We want it to be (statistically) efficiently learnable
- How to come up with suitable loss functions?
  - We want the loss function to reflect that performance metric of interest.
  - We want the loss function to be efficiently optimizable

# Two philosophy for answering the two questions

- Deterministic / Discriminative view:
  - Loss function is a surrogate the performance metric that we care about.
    - e.g. logistic loss, hinge-loss, etc. upper bounds the 0-1 loss
  - Geometric view: Hypothesis class specifies the shapes of the decision boundary.
- Probabilistic / Generative view:
  - Loss function can be derived from Max-Likelihood Principle.
  - Hypothesis class is specified by a probabilistic model of the data-generation process.

# Maximum Likelihood Estimation

- MLE defines an optimization problem to solve for estimating the parameters given data.
  - Find the parameter that maximizes the likelihood
  - Find the **distribution within a set of distributions** that maximizes the probability (likelihood) of observing the data

## Problem 5 Maximum Likelihood Estimation

Let the data space be  $[0, 1]$ . And the data we observed be  $0.1, 0.01, 0.5$ . Let the set of probability distributions be  $\mathcal{P} = \{P_1, P_2, P_3\}$ , where  $P_1$  is a uniform distribution on  $[0, 1]$ ;  $P_2$  is a uniform distribution on  $[0, 0.2]$ ;  $P_3$  is a distribution with probability density  $p(x) = 2(1 - x)$  defined on  $[0, 1]$ .

Work out the likelihood function and the maximum likelihood estimate from  $\mathcal{P}$ .

# Examples of supervised learning problems

	Binary classification	Multi-class classification	Regression
Feature space	$\mathbb{R}^d$	$\mathbb{R}^d$	$\mathbb{R}^d$
Label space	$\{-1, 1\}$	$\{1, 2, 3, \dots, K\}$	$\mathbb{R}$
Popular Performance metric	Classification error (0-1 loss) for test data	Classification error (0-1 loss) for test data	Mean Square Error (MSE) vs ground truth
Popular surrogate loss (for training)	Logistic loss	Multiclass logistic loss aka. Cross-Entropy loss	Square loss
Probabilistic interpretation	MLE for a Bernoulli (Linear) Model	MLE for a Multinomial model	MLE for a Gaussian Observation model

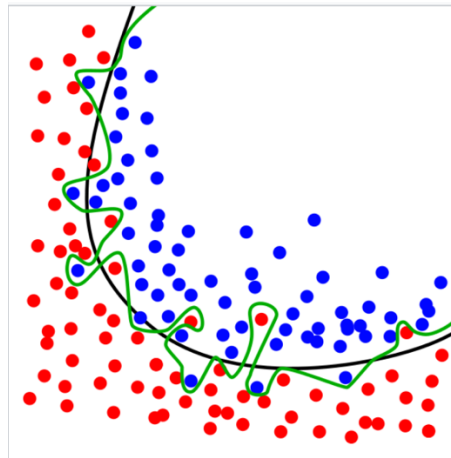
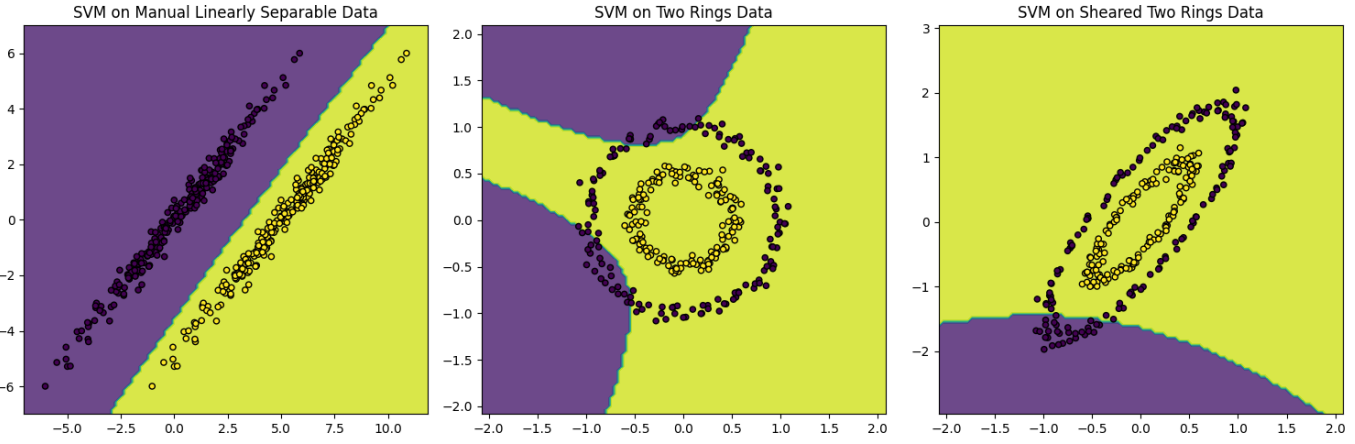
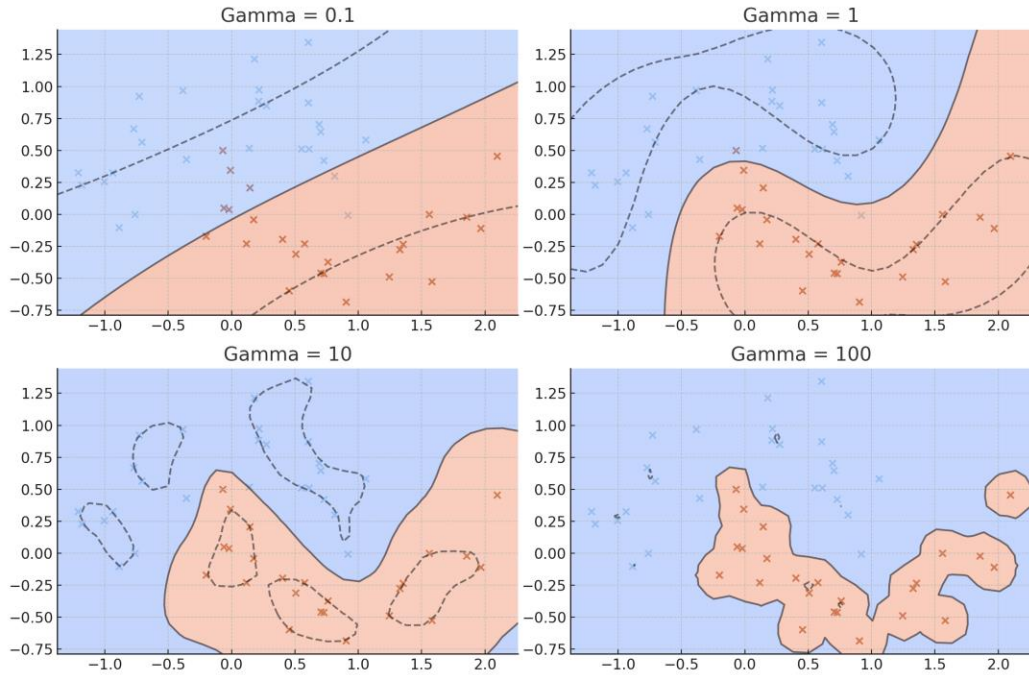
# Classifiers that we learned from 240

- Linear classifier
- Decision Tree (Decision Stumps)
- Naïve Bayes classifier
- Voting classifiers  $\leq$  Bagging, Boosting
- Feature-expanded linear classifiers  $\leq$  Kernel methods
- Neural Networks  $\leq$  Learning representation

## Important learning goals:

1. What are the parameters
2. Fix a parameter, how does the classifier make predictions
3. Sketching the decision boundary in a 2d plane

# They produce different decision boundaries



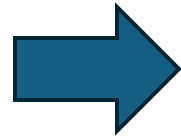
$H_{\text{final}}$

$$= \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} \right)$$

$$= \begin{array}{|c|c|c|} \hline \text{blue} & \text{blue} & \text{red} \\ \hline \text{blue} & \text{red} & \text{red} \\ \hline \end{array}$$

Overfitting vs underfitting?

# How did we go from “problem solving agent” to “Skynet and terminators”



‘Godfather of AI’ shortens odds of the technology wiping out humanity over next 30 years

Geoffrey Hinton says there is 10% to 20% chance AI will lead to human extinction in three decades, as change moves fast

- ‘We need dramatic changes’: is societal collapse inevitable?



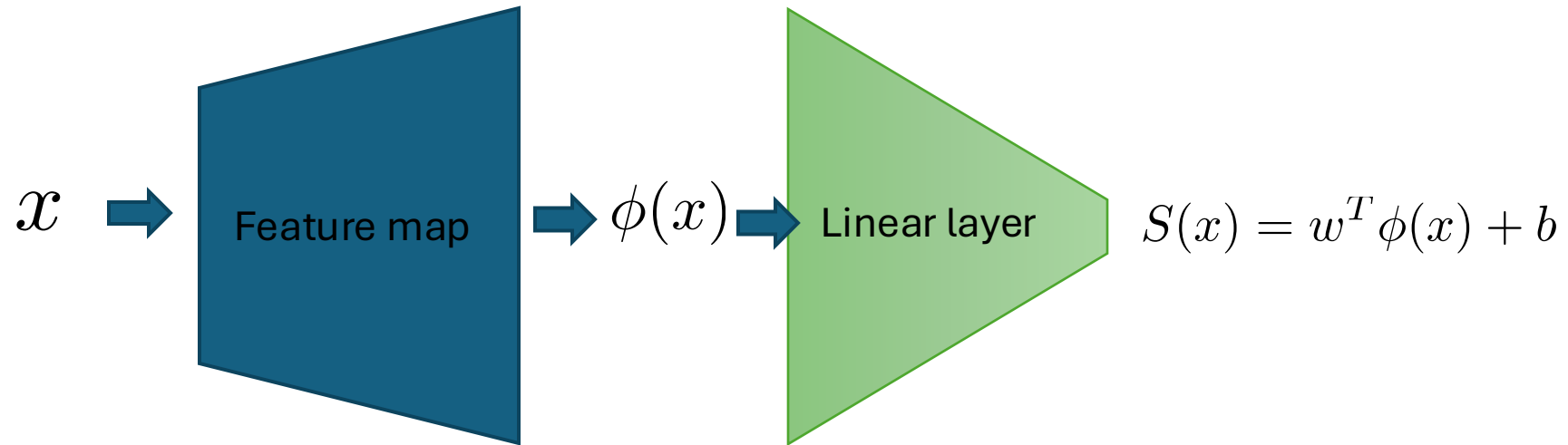
📷 Geoffrey Hinton said humans will be like toddlers compared with the intelligence of highly power AI systems. Photograph: Pontus Lundahl/TT News Agency/AFP/Getty Images



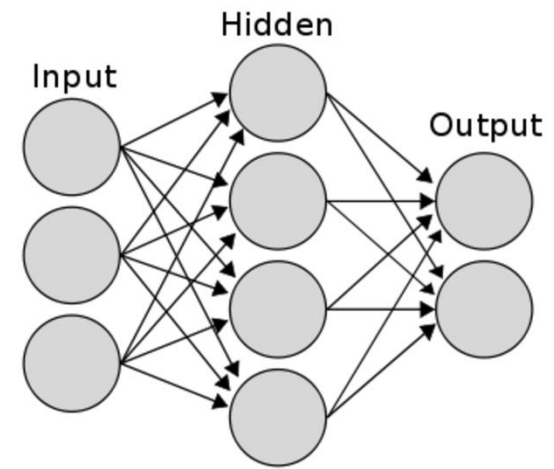
Solving specialized problem with **human ingenuity:**

Feature engineering / specify hypothesis class/ optimization

# So what's so special about deep learning?



# Two-layer neural networks



- Linear neural network:  $S(x) = w_2^T (W_1 x + \mathbf{b}_1) + b_2$ 
  - Still a linear model at the end of the day, so let's add a nonlinearity  $\sigma$ !
- Two-layer MLP:  $S(x) = w_2^T \sigma(W_1 x + \mathbf{b}_1) + b_2$ 
  - Linear model w.r.t. to a learnable feature map
- RBF-kernel:  $S(x) = w_2^T \exp(i (W_1 x + \mathbf{b}_1)) + b_2$ 
  - Kernels are infinite width neural networks with fixed weights
  - By Bochner's theorem, see, e.g., [\[Rahimi and Recht, 2007\]](#)

# Neural networks: Example: AlexNet (2012)

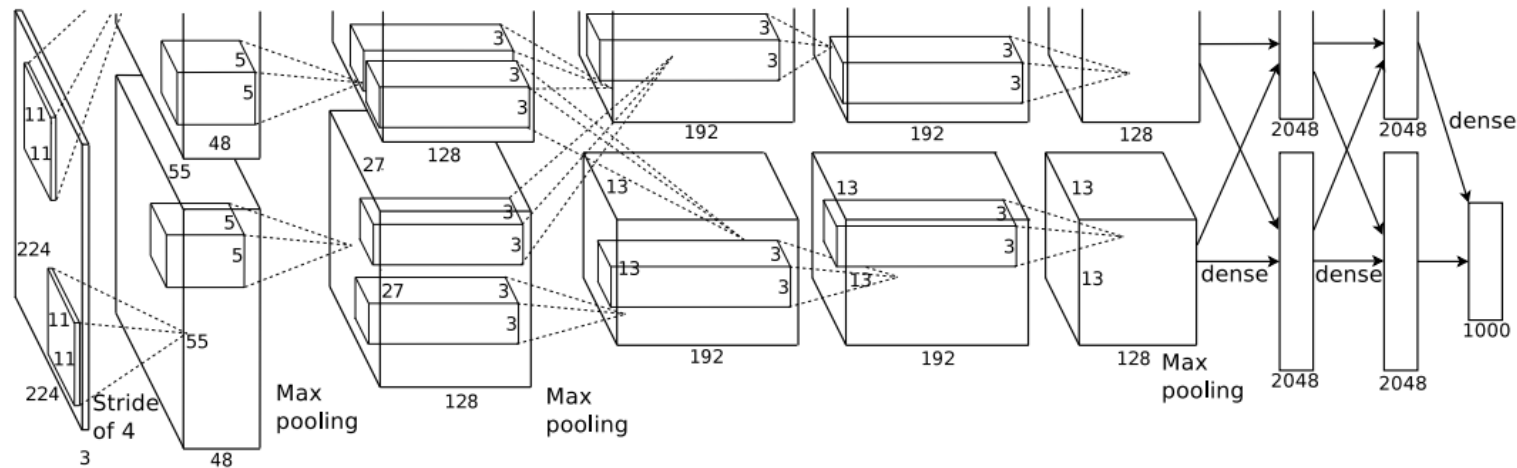


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

## Imagenet classification with deep convolutional neural networks

[A Krizhevsky, I Sutskever...](#) - *Advances in neural ...*, 2012 - [proceedings.neurips.cc](#)

... a large, **deep convolutional neural network** to **classify** the 1.2 million high-resolution images in the **ImageNet** ... The **neural network**, which has 60 million parameters and 650,000 neurons, ...

☆ Save 📄 Cite Cited by 122248 Related articles All 111 versions Import into BibTeX 🔗



# Modern neural networks are very complicated

## Attention is all you need

[A Vaswani, N Shazeer, N Parmar...](#) - Advances in neural ..., 2017 - proceedings.neurips.cc

... to attend to **all** positions in the decoder up to and including that position. **We need** to prevent

... **We** implement this inside of scaled dot-product **attention** by masking out (setting to  $-\infty$ ) ...

☆ Save 📄 Cite Cited by 97503 Related articles All 62 versions Import into BibTeX 🔗

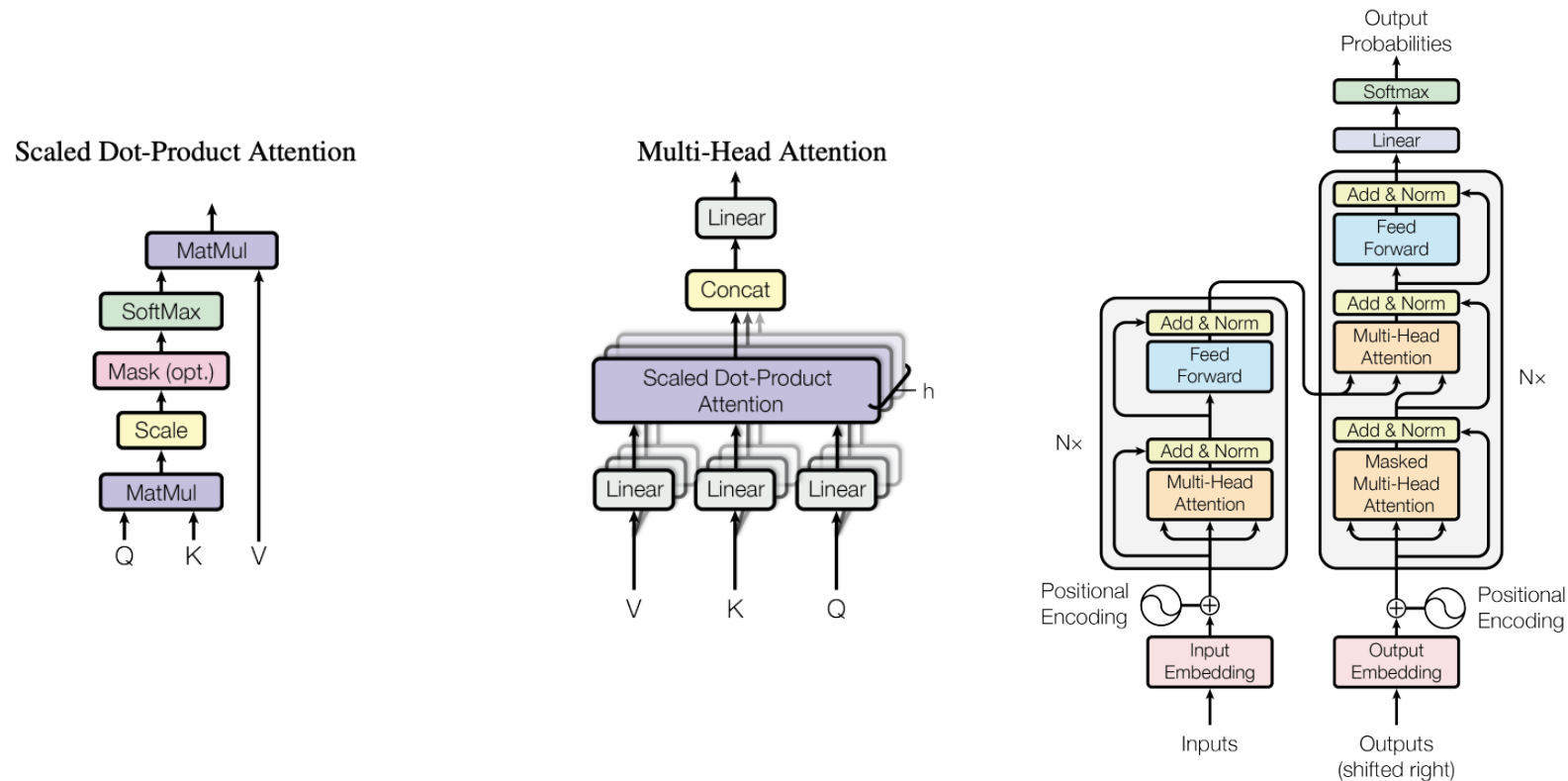


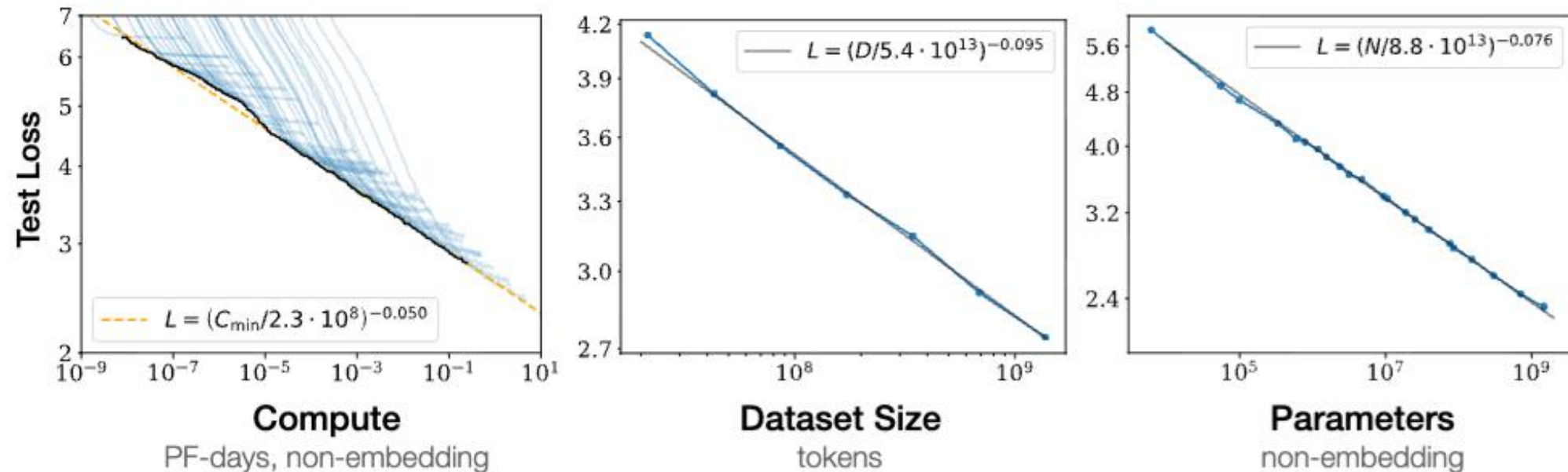
Figure 1: The Transformer - model architecture.

# Solution to this? Brute-force computation with autograd and GPUs

- **Autograd:** basically chain rules, can be automated.
  - Design networks such that every block is differentiable.
- **Faster Computation:**
  - Well-packaged Deep Learning Farmework: Write Python wrapper code but running C++ underneath
  - Parallel computing: Numerical linear algebra and GPUs, scientific computing, supercomputing centers.
  - Distributed computing: Cloud computing, Map-Reduce, federated learning
- Popular tools (there are many more of these):



# Scale compute / data / parameters --- test loss always gets smaller.



**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

# Example of autograd and implementation of a neural network

```
import jax.numpy as jnp
from jax import grad
#from jax import jit
import numpy as np

# What if we want to do a neural network?

def predict_nn(params, inputs):

    W1 = params[0]
    b1 = params[1]
    W2 = params[2]
    b2 = params[3]
    hidden = jnp.dot(W1, inputs) + b1
    activated = jnp.tanh(hidden)
    scores = jnp.dot(W2, activated) + b2

    outputs = soft_argmax(scores)
    return outputs

def loss_fun_nn(params, inputs, targets):
    preds = predict_nn(params, inputs)
    # You can use logistic loss instead
    return jnp.sum((preds - targets)**2)

grad_fun_nn = grad(loss_fun_nn) # gradient evaluation function
```

Now with LLMs we don't even need to write Jax / pytorch code... We may get away with vibe code.

- Let's try it out!
- AI resources available to students:
  - Google Gemini 2.5 (free for one year)
  - Google Colab (Jupyter Notebook, GPU + TPU)
- Feel free to use your favorite LLM for coding.

# Next lecture

- Make sure you complete the reading materials
  - **Required:** *The Foundation Model* (Read the intro and at least one subsection from Section 2, 3, 4, or 5.)
  - **Optional Reading:** Sutton's *The Bitter Lesson*; GPT-2 *Multitask Learners*; GPT-3 *Few-Shot Learners*; *Sparks of AGI*; *Emergent Abilities*; ; *Are Emergent Abilities a Mirage?*
- Get ready for the discussion.