

# Optimization for Deep Learning

**Kun Yuan**

**Center for Machine Learning Research @ Peking University**





深度学习最优化课程微信群

- **Basics in Deep Learning and Large Language Model**

Linear regression; Logistic regression; Multi-class regression; Neural network

Word embedding; Language models; RNN; Seq2seq; Transformer; Decoder-only LLMs

- **Foundations in Optimization**

Basics in optimization; Gradient descent; Forward-backward propagation

Accelerated gradient descent; Proximal gradient descent; Zeroth-order method; Block-coordinate descent

- **Foundations in Stochastic Optimization**

Stochastic gradient descent; Momentum SGD; Adaptive SGD;

- **Optimization for DL and LLM**

Mixed-precision training; memory-efficient training; distributed training; communication-efficient training

Low-rank fine-tuning; block-wise fine-tuning; zeroth-order finetuning

# Grading policy

---

- Homework (~30%)
- Mid-term (~30%)
- Final project (~40%)

# Teaching assistants

---



孔博傲



林子凌



刘禹希



乔永琦



宋奕龙



吴百濠



杨光正奥



张源

# A Brief Introduction to Deep Learning

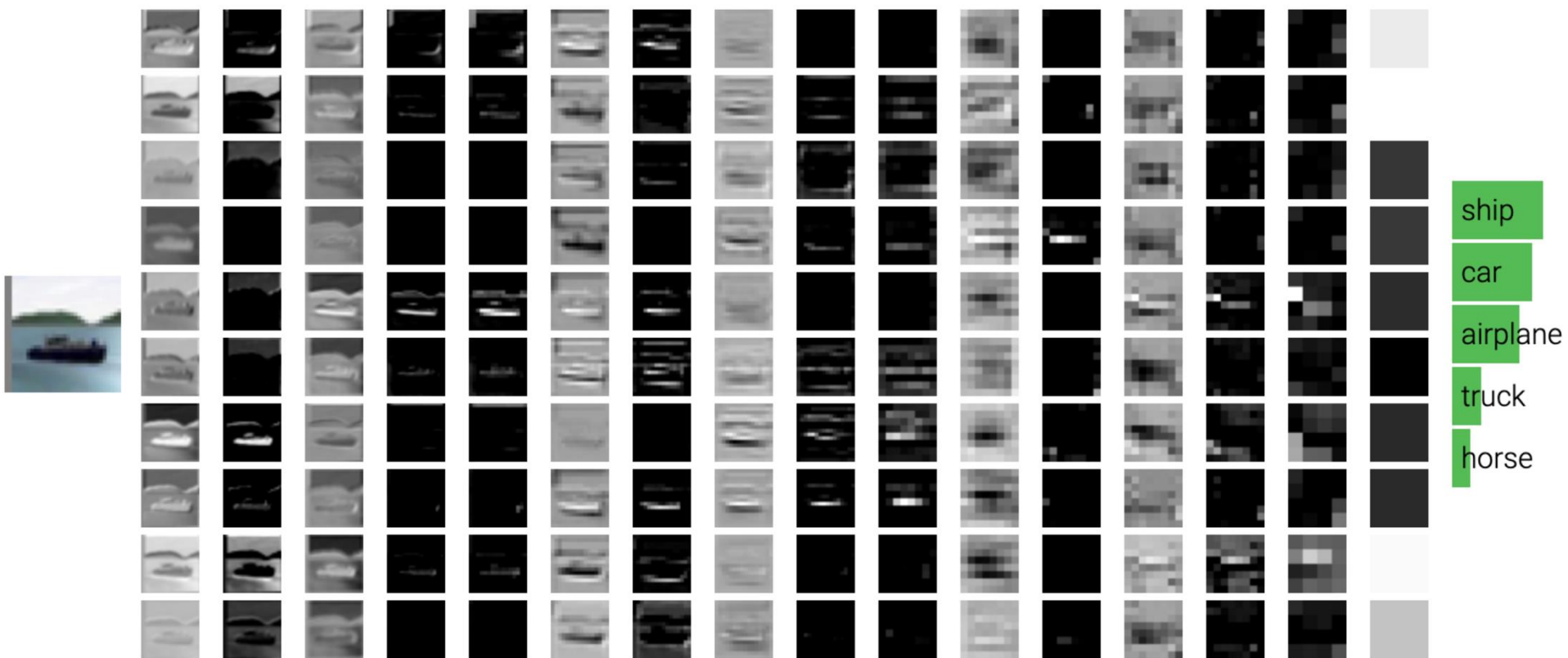
**Kun Yuan**

**Center for Machine Learning Research @ Peking University**



# Multi-classification

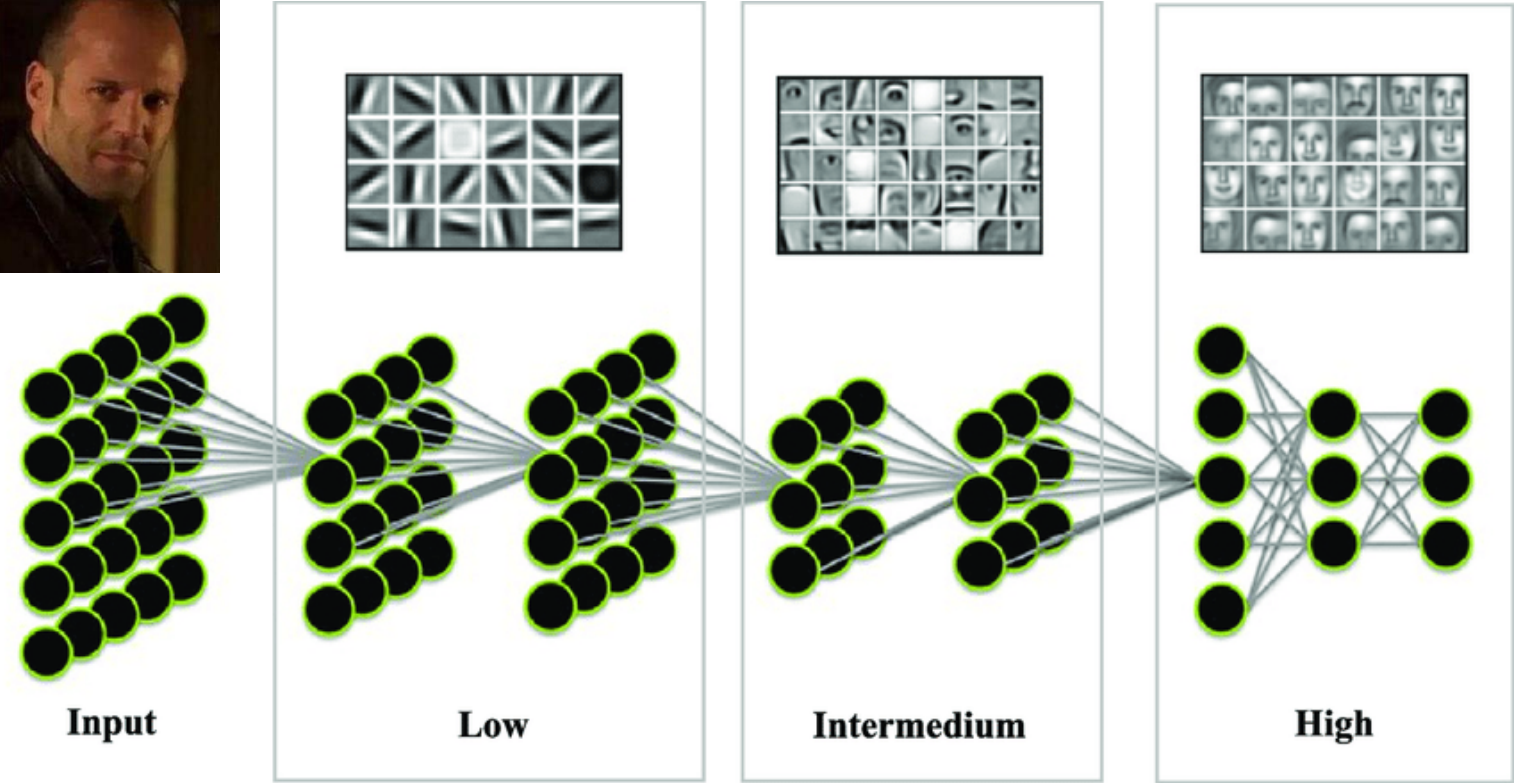
- Multi-classification is very common in real life



[CS231n: Deep Learning for Computer Vision]

# Multi-classification

- Face recognition is one of the most successful multi-classification tasks



- Elon Musk
- Dwayne Johnson
- 郭达
- Adam Lambert
- Jason Statham



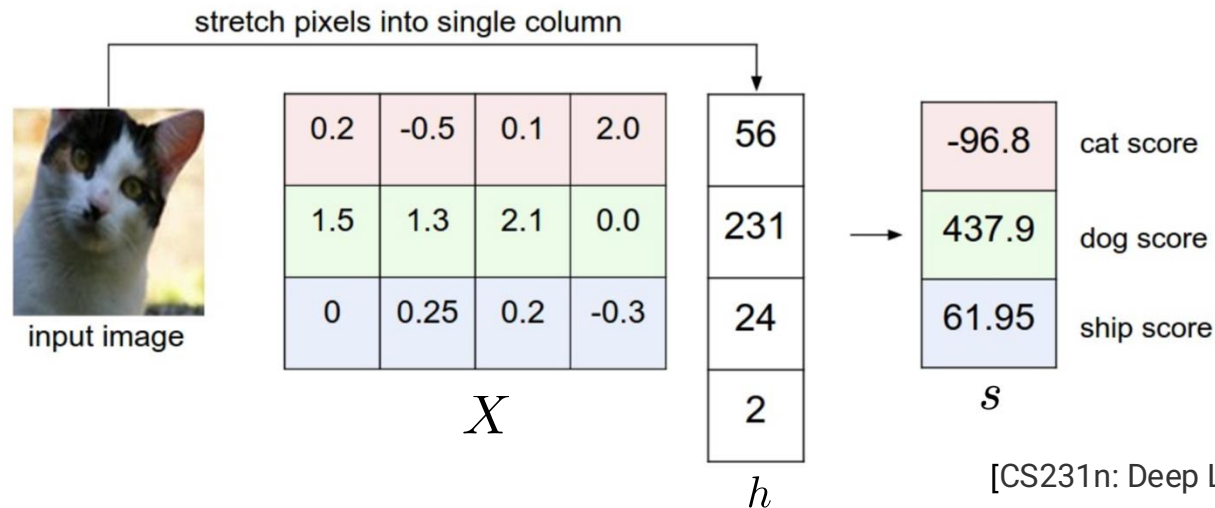


# Predict the score

- We first collect the dataset  $\{h_i, y_i\}_{i=1}^N$  where  $h_i \in \mathbb{R}^d$  is the feature and  $y_i \in \{1, \dots, C\}$  is the label
- We consider a linear model to predict the score of each class

$$s = Xh \in \mathbb{R}^C$$

where  $X \in \mathbb{R}^{C \times d}$  is the model parameters to learn and  $s_i$  is the score that  $h$  belongs to class  $i$

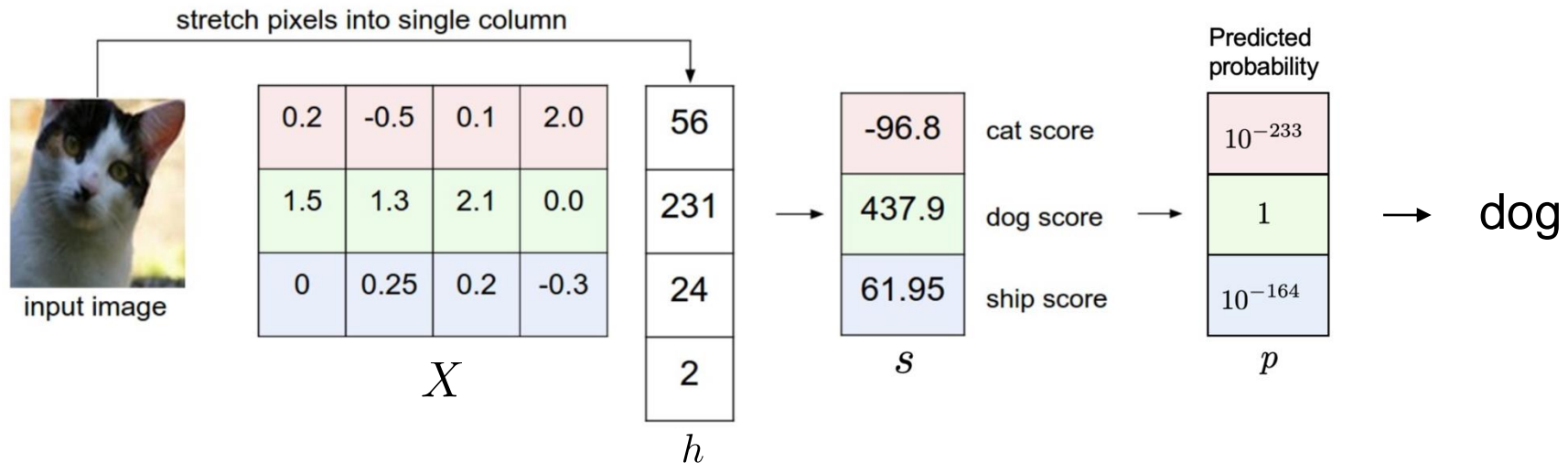


[CS231n: Deep Learning for Computer Vision]

# Predict the probability

- Given the score vector  $s$ , the probability of each class with the softmax function is as follows

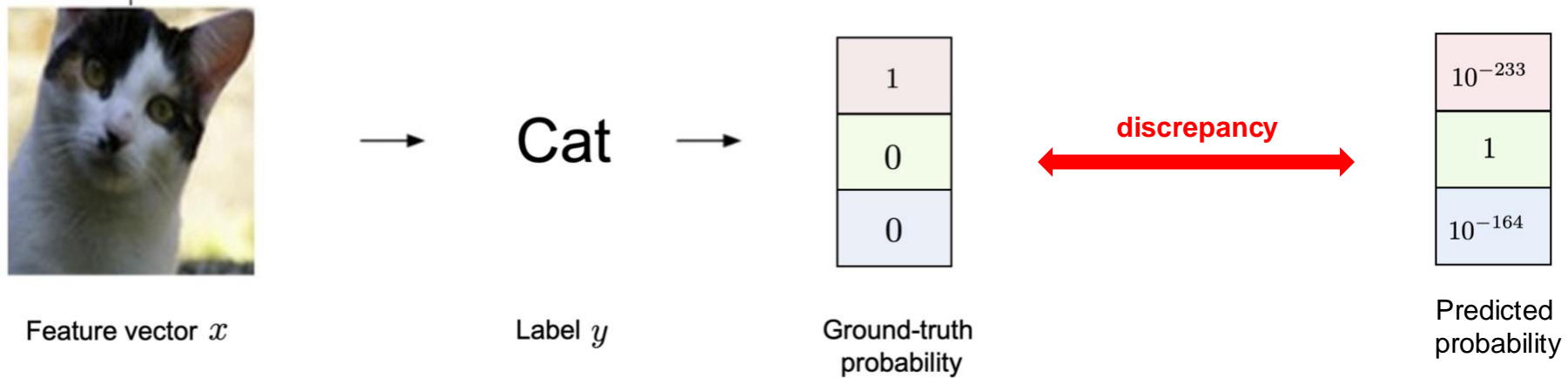
$$p_i = \frac{\exp(s_i)}{\sum_{j=1}^C \exp(s_j)} \in (0, 1)$$



- Given the model parameters  $X$ , we can predict the class that feature  $h$  belongs to
- But a bad model  $X$  will result in incorrect predictions

# Cross-entropy

- How to achieve a good model  $X$  that can provide accurate predictions?
- We need to train the model so that the discrepancy between ground-truth and predictions are minimized



- How to measure the discrepancy?

- Cross entropy can measure the difference between two distributions  $p \in R^d$  and  $q \in R^d$

$$H(\mathbf{p}, \mathbf{q}) = - \sum_{j=1}^d p_j \log(q_j)$$

Smaller cross entropy indicates smaller difference between  $p$  and  $q$

- Examples:

$$\mathbf{p} = (1, 0, 0, 0) \quad \mathbf{q} = (0.25, 0.25, 0.25, 0.25) \quad \longrightarrow \quad H(\mathbf{p}, \mathbf{q}) = 2$$

$$\mathbf{p} = (1, 0, 0, 0) \quad \mathbf{q} = (0.91, 0.03, 0.03, 0.03) \quad \longrightarrow \quad H(\mathbf{p}, \mathbf{q}) = 0.136$$

# Multi-classification: Loss function

- A good model  $X$  will minimize the discrepancy between predictions and ground-truth

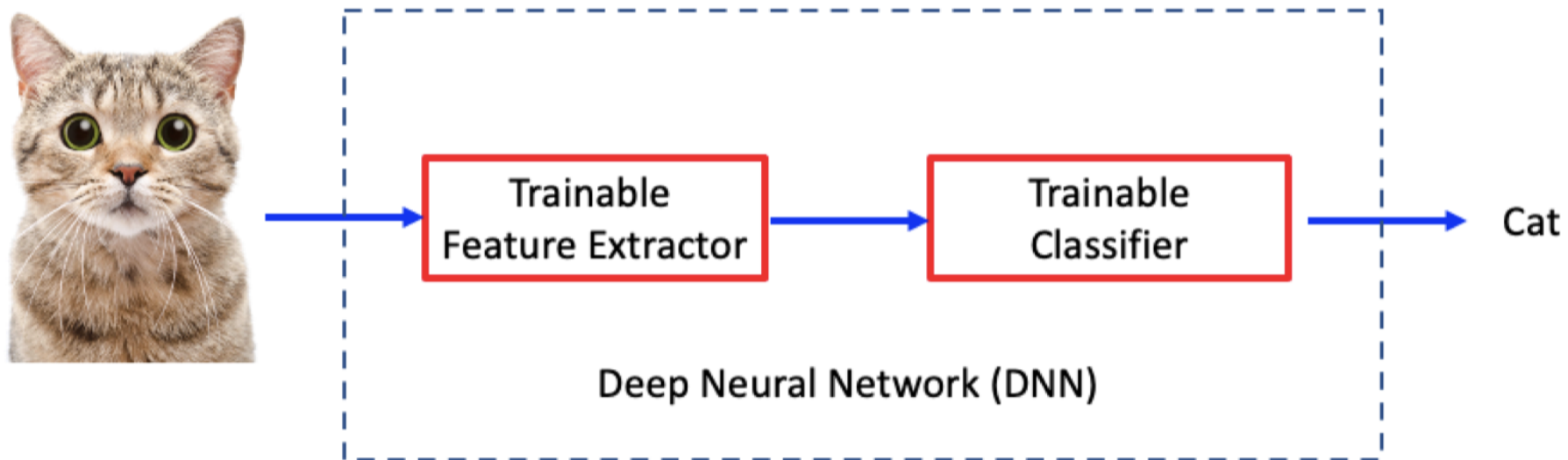
$$X^* = \arg \min_X \left\{ \frac{1}{N} \sum_{i=1}^N H(\underset{\text{prediction}}{p_i}, \underset{\text{ground-truth}}{q_i}) \right\}, \quad \text{where} \quad p_i = \text{softmax}(\underset{\text{feature of the } i\text{-th sample}}{X h_i})$$

- Once a good model  $X^*$  is achieved, we can make predictions as follows

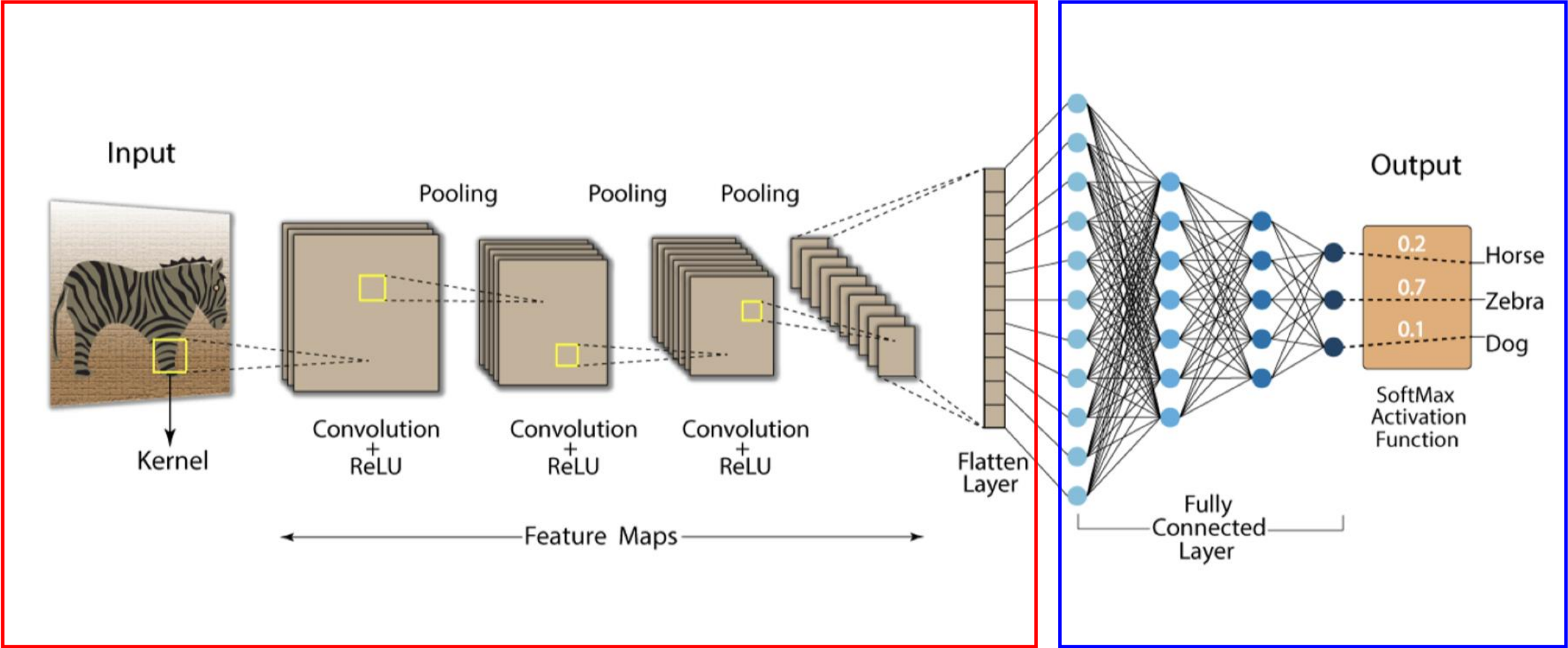
$$p = \text{softmax}(X^* h) \quad \longrightarrow \quad h \text{ belongs to class } j^* = \arg \max_j \{p_j\}$$

# Deep neural network (DNN)

- A deep neural network (DNN) typically includes a **feature extractor** and a **classifier**
- Well-trained DNN can make precise predictions



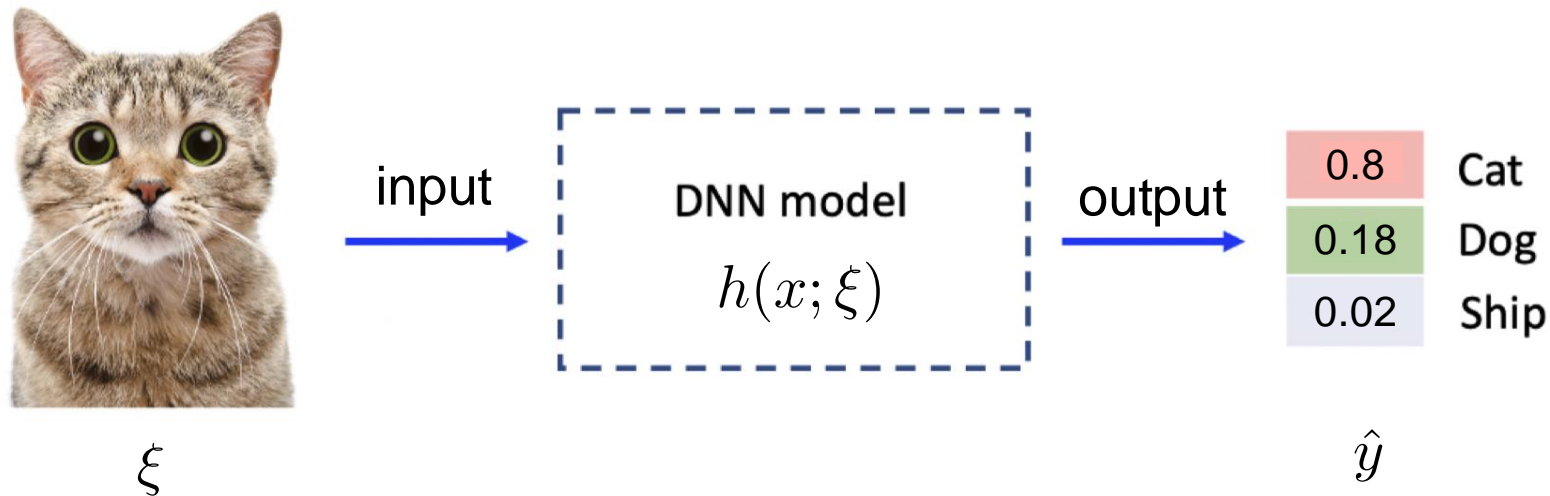
# Example: convolutional neural network



Feature extractor

Multi-Classifier

- We model DNN as  $h(x; \xi) : \mathbb{R}^d \rightarrow \mathbb{R}^c$  that maps input data  $\xi$  to a probability  $\hat{y}$ 
  - $x \in \mathbb{R}^d$  is the DNN model parameter to be trained
  - $\xi$  is a random input data sample
  - $c$  is the number of classes






# Training DNN can be formulated into an optimization problem

- Define  $L(\hat{y}, y) = - \sum_{j=1}^d y_{[j]} \log(\hat{y}_{[j]})$  as the loss function to measure the difference between predictions and the ground-truth label, where  $y_{[j]}$  is the j-th element in  $y$
- The model parameter  $x^*$  can be achieved by solving the following optimization problem

$$x^* = \arg \min_{x \in \mathbb{R}^d} \left\{ \mathbb{E}_{(\xi, y) \sim \mathcal{D}} \left[ L(h(x; \xi), y) \right] \right\}$$



data distribution   prediction   real label

# Training DNN can be formulated into an optimization problem

- The model parameter  $x^*$  can be achieved by solving the following optimization problem

$$x^* = \arg \min_{x \in \mathbb{R}^d} \left\{ \mathbb{E}_{(\xi, y) \sim \mathcal{D}} \left[ L(\underbrace{h(x; \xi)}_{\text{prediction}}, \underbrace{y}_{\text{real label}}) \right] \right\}$$

data distribution    prediction    real label

- If we define  $\xi = (\xi, y)$  and  $F(x; \xi) = L(h(x; \xi), y)$ , the above problem becomes

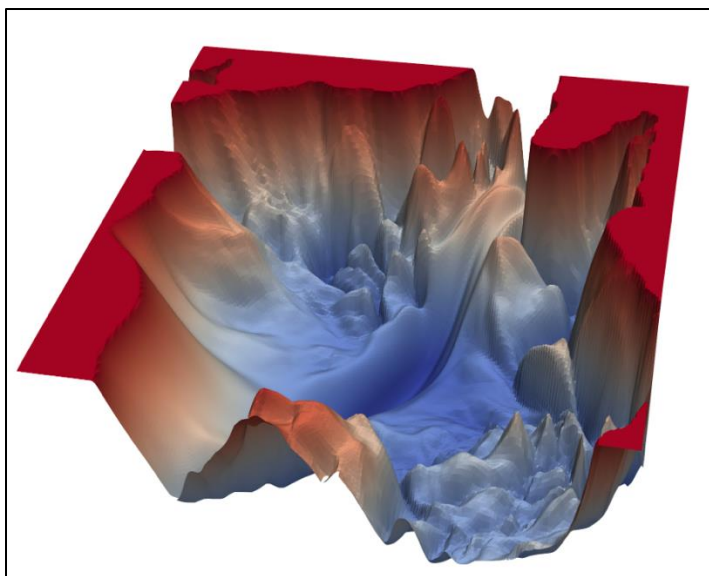
$$x^* = \arg \min_{x \in \mathbb{R}^d} \{ \mathbb{E}_{\xi \sim \mathcal{D}} [F(x; \xi)] \}$$

- Most optimization researchers use the second formulation as the starting point to develop algorithms

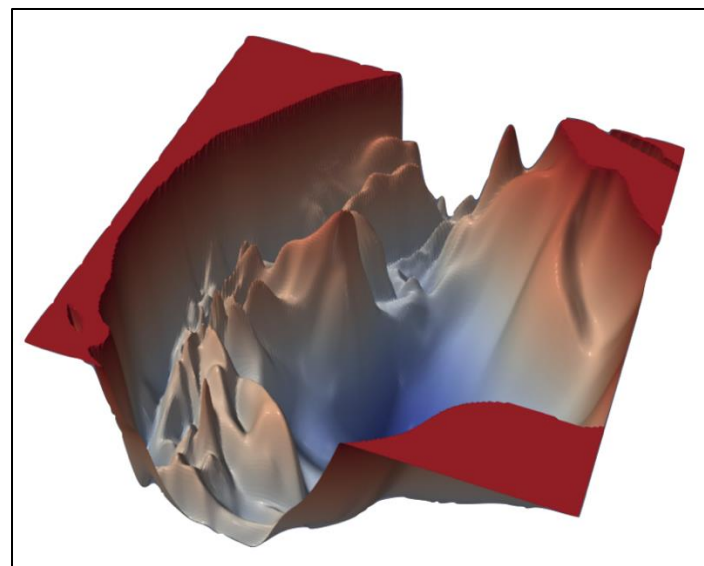
# DNN model is notoriously difficult to train

- Highly-nonconvex cost functions; cannot find global minima; trapped into local minimum

VGG-56



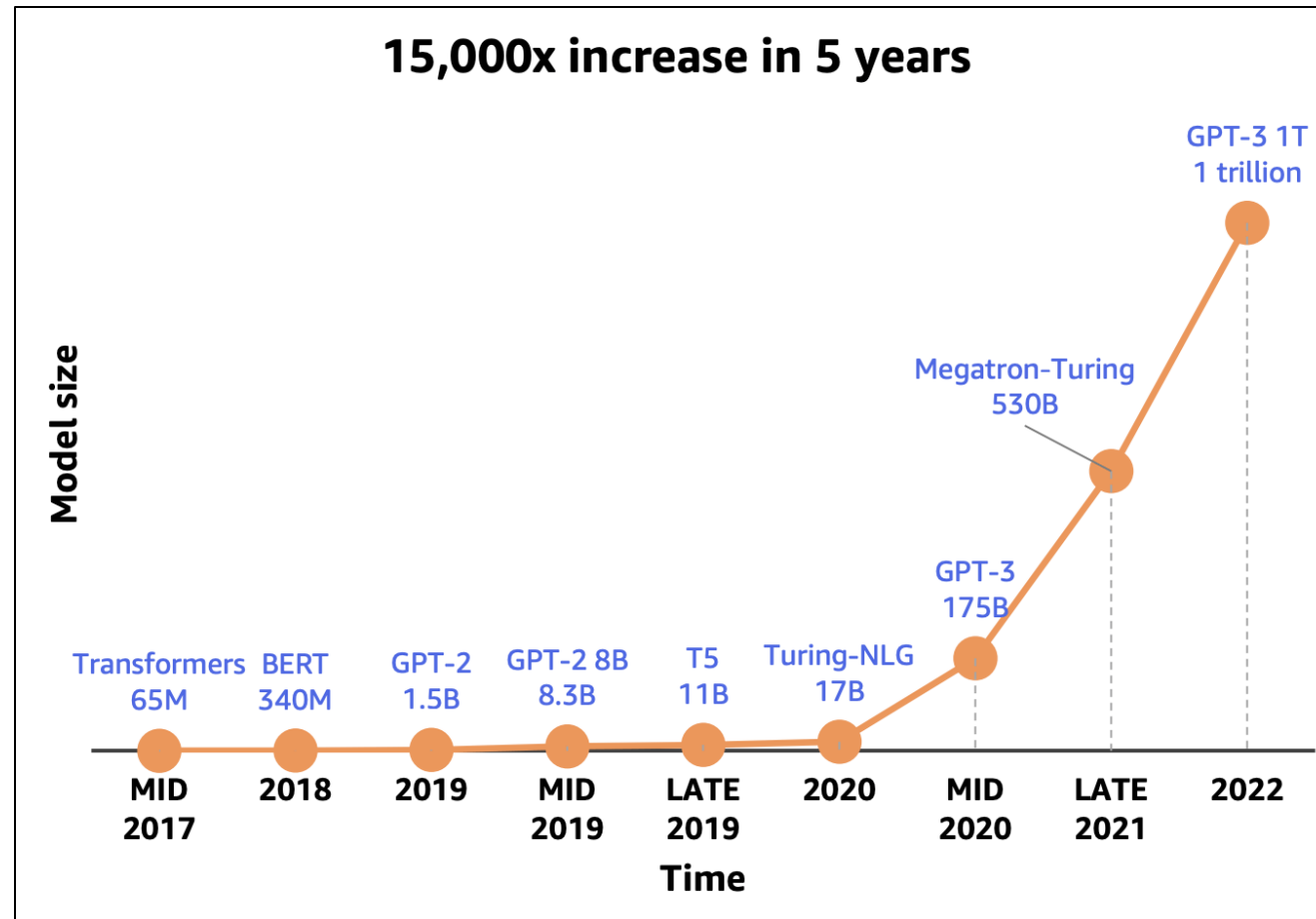
VGG-110



[Visualizing the loss landscape of neural nets]

# DNN model is notoriously difficult to train

- The model size is large, i.e.,  $x \in \mathbb{R}^d$  is of extremely high dimensions



[Train and deploy large language models on Amazon SageMaker]

# DNN model is notoriously difficult to train

- The size of the dataset is huge

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

} Crawled data from websites; in both high quality and low quality

} High-quality data

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

# DNN model is notoriously difficult to train

---

DNN training = Non-convex training + Huge dimensions + Huge dataset

**Efficient** and **scalable** optimization approaches are in urgent need