PEKING UNIVERSITY

Introduction to Foundation Models

Kun Yuan

Center for Machine Learning Research @ Peking University

深度学习最优化课程微信群

Contents

- **Foundation Models**

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

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

  CNN; ViT; Clip; Dalle; Diffusion; DiT;

- **Pre-training**

  Basics in optimization; Gradient descent; Forward-backward propagation; Stochastic gradient descent;

  Adaptive gradient descent; Distributed pretraining; Mixed-precision pre-training; FlashAttention

- **Post-training and Inference**

  SFT; RLHF; PEFT; Alignment; KV Cache

- **Application**

  Chain of thought; Tree of though; Prompt Engineering; Retrieval -agmented generation; LLM agents

Grading policy

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

Teaching assistants

宋奕龙

Leading TA

商秋林

胡婕

胡奕鹏

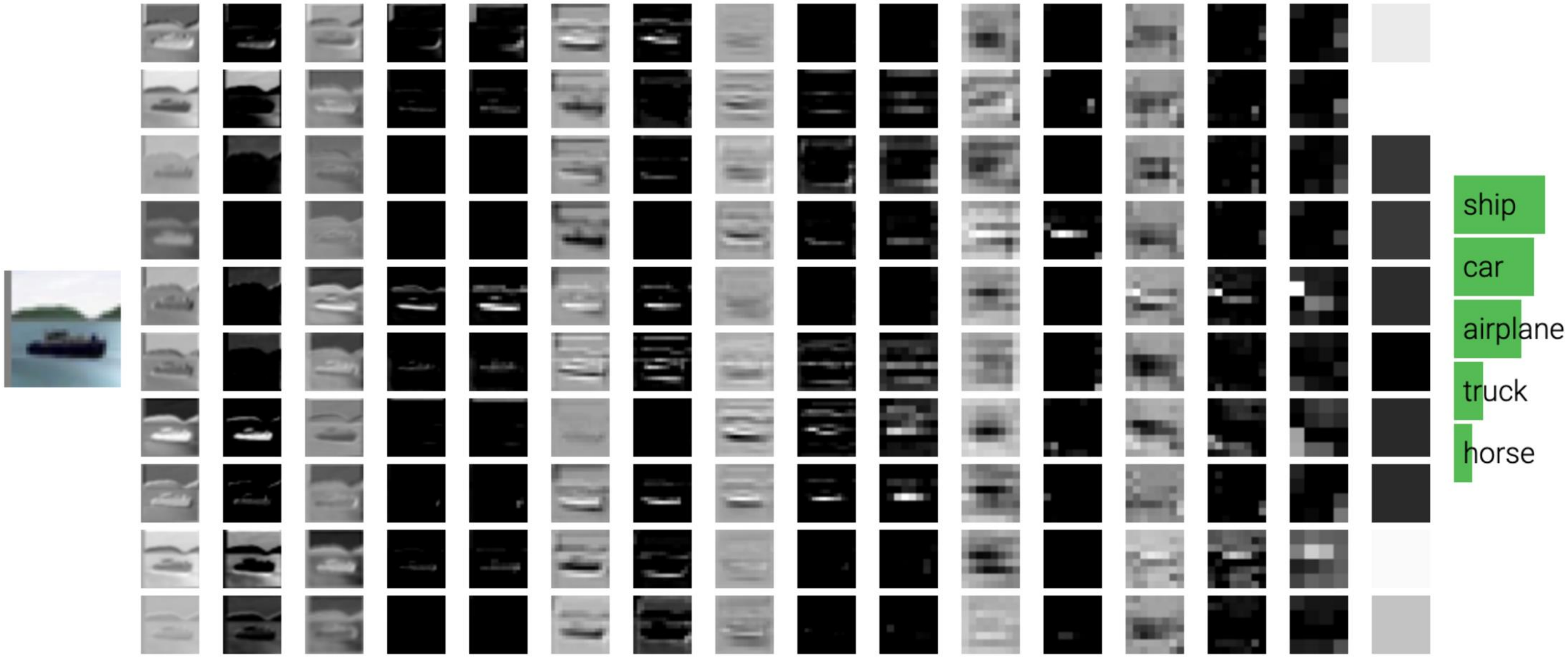
A Brief Introduction to Deep Learning

Kun Yuan

Center for Machine Learning Research @ Peking University

Multi-classification

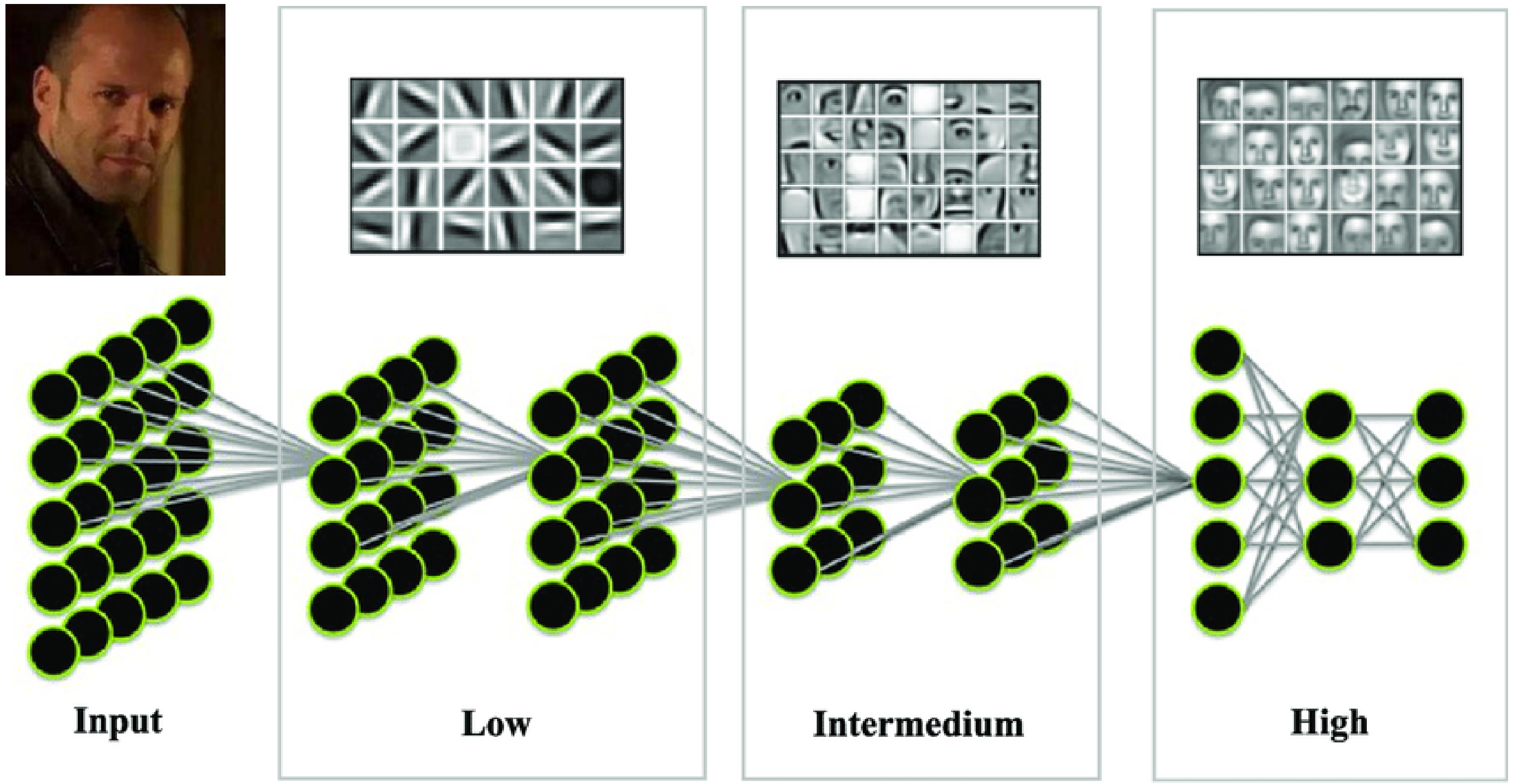
- Multi-classification is very common in real life

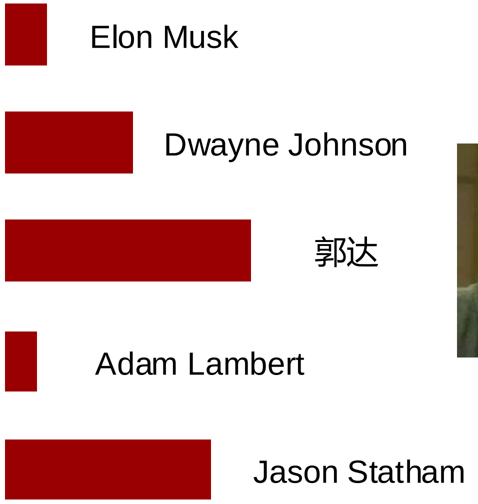


[CS231n: Deep Learning for Computer Vision]

Multi-classification

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





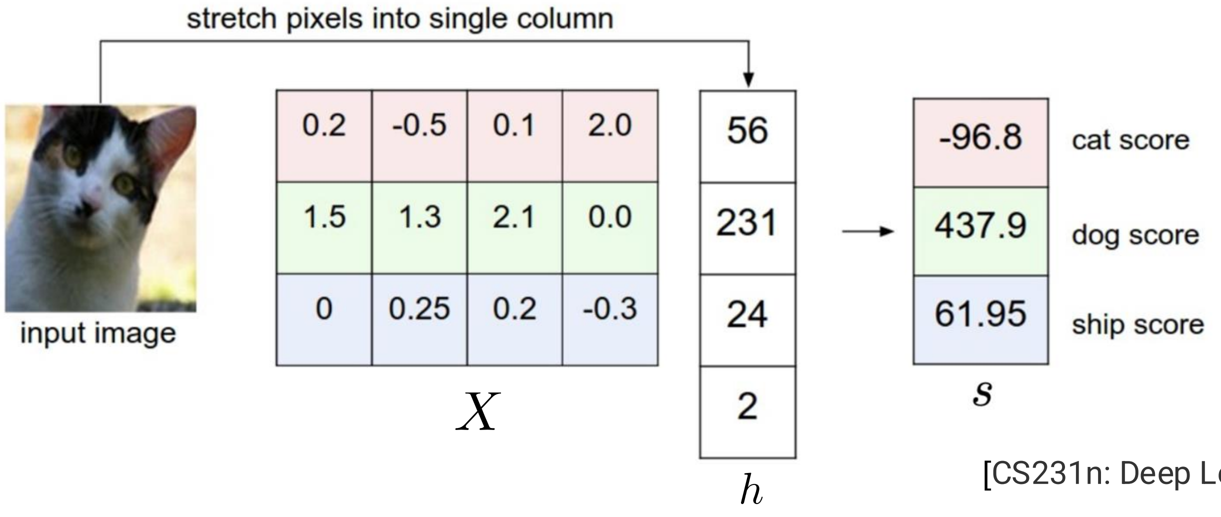
Predict the score

- We first collect the dataset $\{h_i, y_i\}_{i=1}^N$ where $h_i \in R^d$ is the feature and $y_i \in \{1, \cdots, C\}$ 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

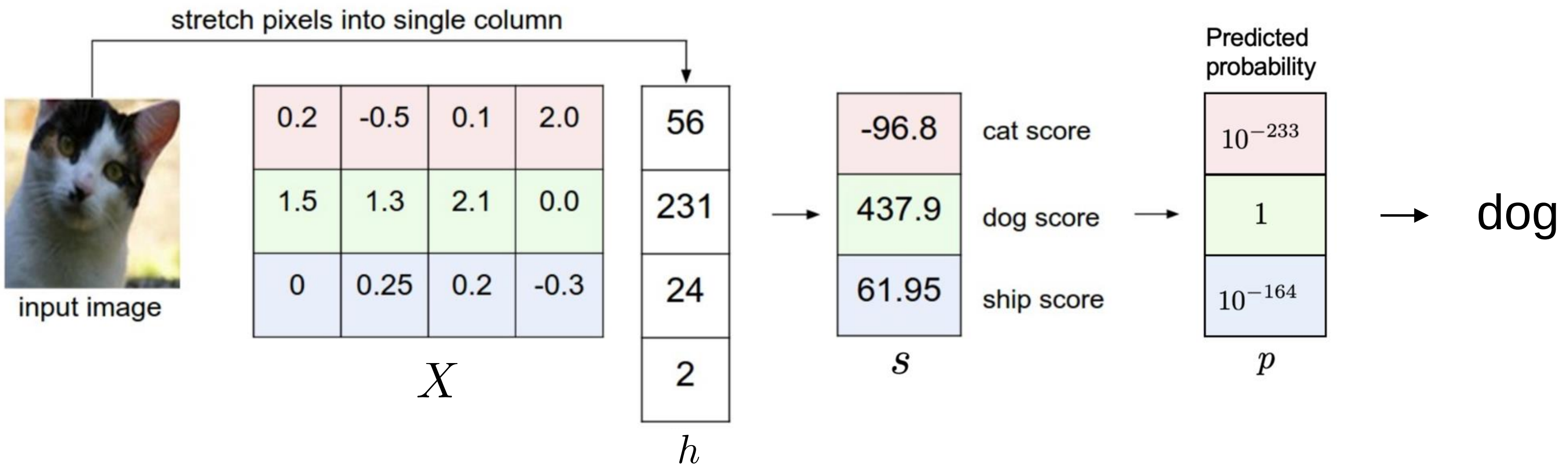


[CS231n: Deep Learning for Computer Vision]

Predict the probability

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

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



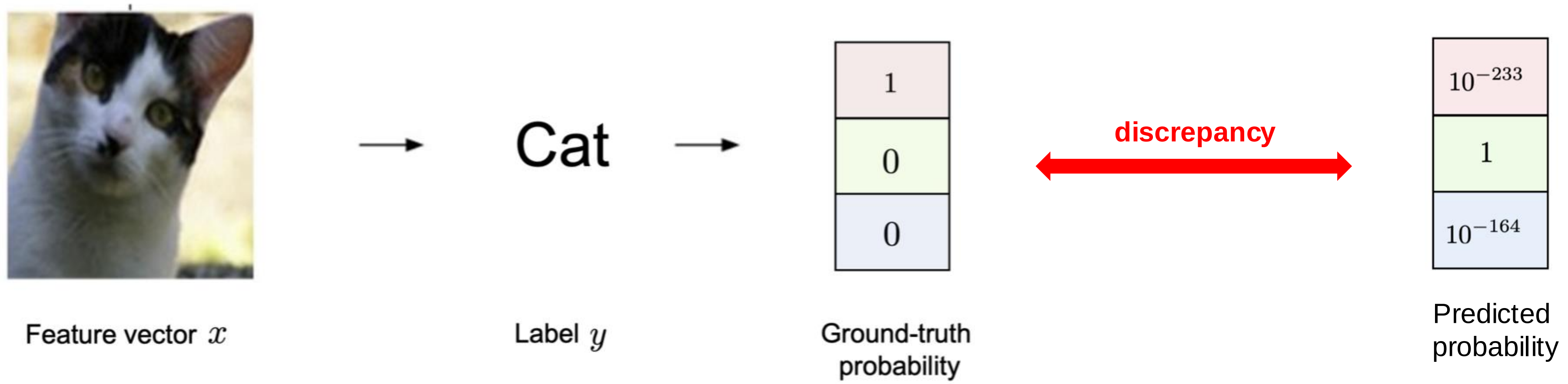
- Given the model parameters X, we can predict the class that feature h belongs to

- 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

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

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

Smaller cross entropy indicates smaller difference between p and q

- Examples:

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

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

Multi-classification: Loss function

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

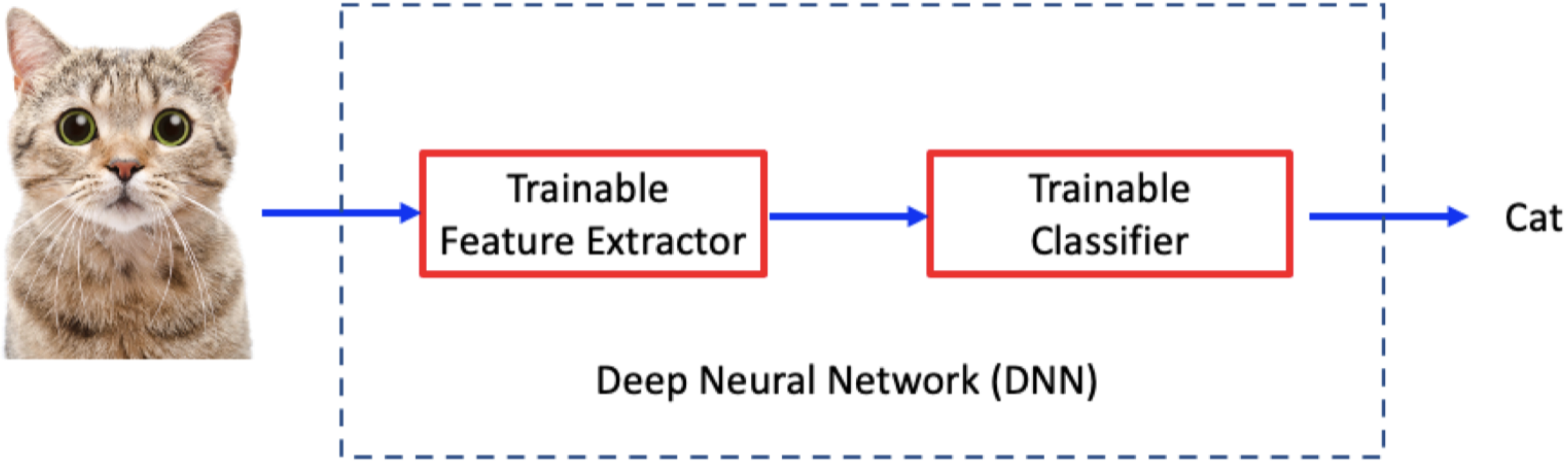
$$X^\star = \arg\min_X \left\{ \frac{1}{N} \sum_{i=1}^{N} H(\underbrace{p_i}_{\text{prediction}}, \underbrace{q_i}_{\text{ground-truth}}) \right\}, \quad \text{where} \quad p_i = \text{softmax}(X \underbrace{h_i}_{\text{feature of the i-th sample}})$$

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

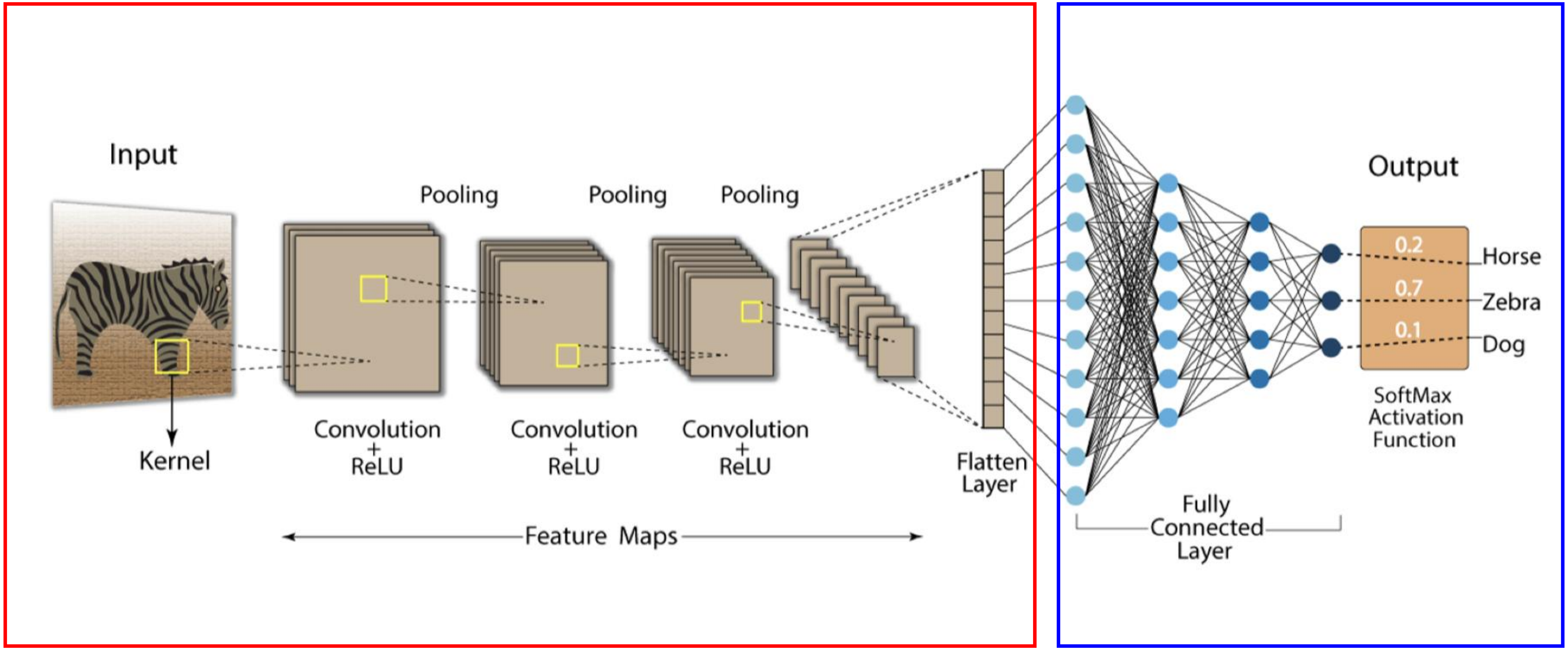
$$p = \text{softmax}(X^\star h) \longrightarrow h \text{ belongs to class } j^\star = \arg\max_j \{p_j\}$$

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

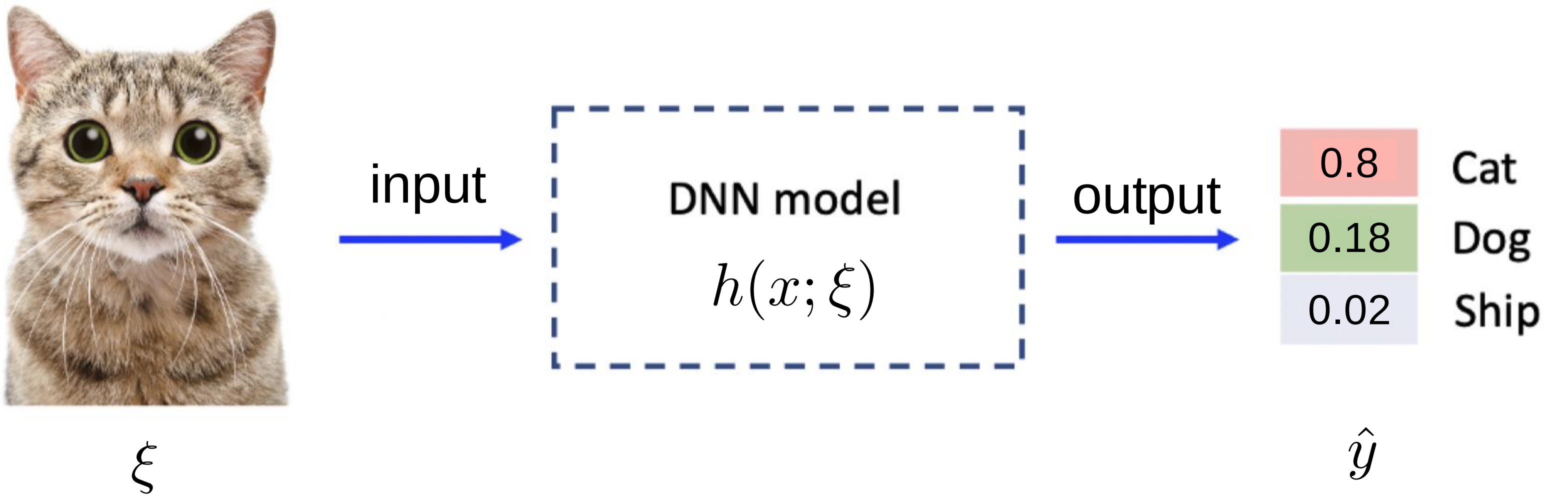


Feature extractor

Multi-Classifier

DNN model

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



Training DNN can be formulated into an optimization problem

PEKING UNIVERSITY

- 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 $x^\star$ can be achieved by solving the following optimization problem

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

Training DNN can be formulated into an optimization problem

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

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

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

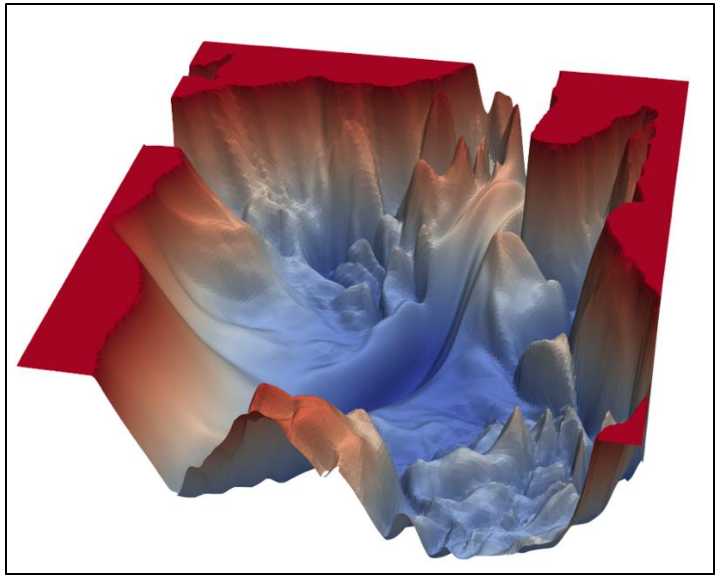
$$x^\star = \arg\min_{x \in \mathbb{R}^d} \{ \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{D}} [F(x;\boldsymbol{\xi})] \}$$

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

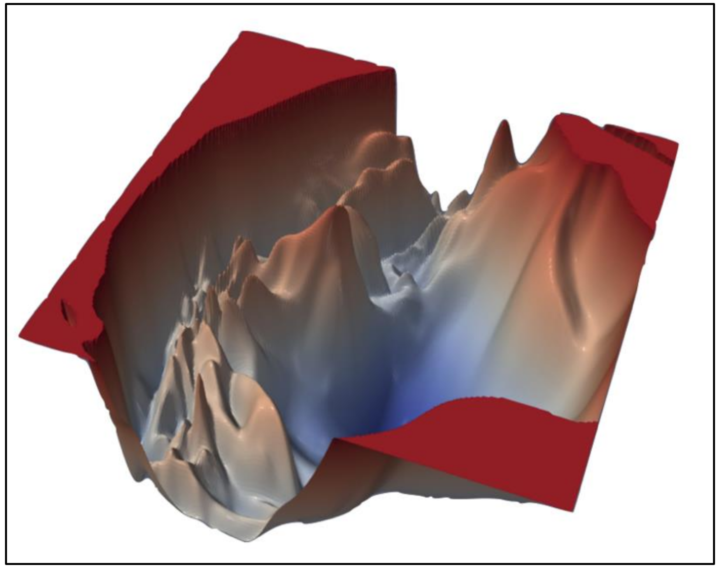
DNN model is notoriously difficult to train

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

VGG-56



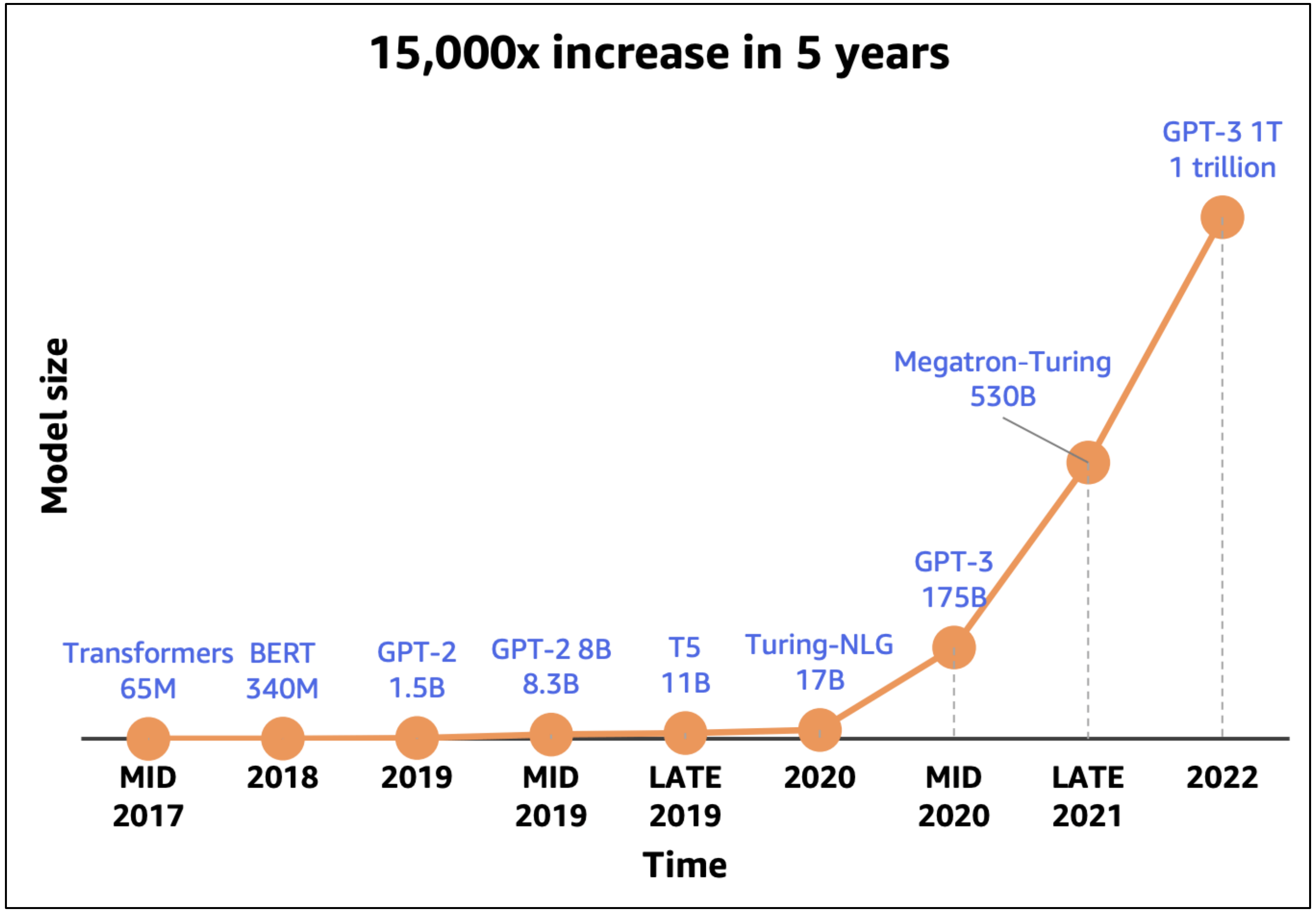
VGG-110



[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

PEKING UNIVERSITY

- 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

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.

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

High-quality data

[LLaMA: Open and Efficient Foundation Language Models]

DNN model is notoriously difficult to train

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

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