



深度学习方法与应用

彭小江 副教授

深圳技术大学 大数据与互联网学院

邮箱: pengxiaojiang@sztu.edu.cn

个人主页: pengxj.github.io

课程目标

- 学习人工智能方向核心技术：人工神经网络、深度学习、计算机视觉应用、自然语言处理应用等
- 掌握深度神经网络的基本知识、常用模型、优化方法、训练技巧等
- 能够利用深度学习解决实际问题：如人脸系统、交通标志识别、物体检测、文本分类等

课程内容

- 模块1：pytorch基础、机器学习基础、感知器、前馈神经网络、误差反向传播等
- 模块2：卷积神经网络、常用CNN模型、视觉目标分类、检测、分割
- 模块3：生成式神经网络、循环神经网络、LSTM、GRU、注意力机制
- 模块4：应用实践——“动态人脸识别系统比拼” “基于注意力网络的人脸表情识别”



课程要求

- 理论2学分，实验1学分，课外实践1学分

课程总评成绩Grade	满分100分Full mark: 100				
课程总评成绩构成The proportion of grade	考勤Attendance	实验作业Homework	PPT展示Presentation	随堂测试	课程论文Essay
	5%	7次, 5%/次合计35%7times, 5% per time, 35% in total	1次, 20%/次合计20%once, 20% per time, 20% in total	1次, 10%/次合计10%once, 10% per time, 10% in total	1次, 30%once, 30%



课外实践（二选一）

- 1. 实践项目一 基于ViT的人脸表情识别及性能优化（利用RAF-DB数据库）

教学内容(Content):

- (1) Pytorch Transformer模型
- (2) 表情数据库的处理
- (3) 视觉Transformer调参优化

学习目标(Objective):

- (1) 掌握Transformer和视觉Transformer及在人脸表情识别的应用

开展形式 (Form) :

在Ubuntu系统下，利用教师实验室的高性能GPU服务器进行训练和测试

- 2. 实验项目二 动态人脸识别及其性能优化

教学内容(Content):

- (1) 基于MTCNN、Centerface的人脸检测和对齐
- (2) 基于预训练的人脸识别模型对人脸进行特征提取
- (3) 人脸特征比对速度优化

学习目标(Objective):

- (1) 掌握动态人脸识别应用

开展形式 (Form) /仪器设备 (Equipments) :

在Ubuntu系统下，利用教师实验室的高性能GPU服务器进行训练和测试，在PC上面进行部署，对比性能。

推荐参考书

- 《动手学深度学习》 PYTORCH 版，原作为MXNet框架。网页版 <https://d2l.ai/>
- 《深入浅出PyTorch-从模型到源码》



应该具有的前提知识

- Python编程
- 高等数学
- 机器学习基础知识



深度学习概述

深圳技术大学



从人工智能讲起

- AI

“人工智能就是研究如何使计算机去做过去只有人才能做的智能工作。”

——MIT温斯顿教授

AI之“自我程序设计”

- 传统计算机程序判断高血压示例

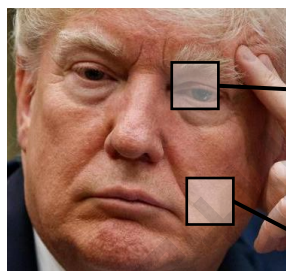
```
person = {年龄:40岁, 体重:80千克, 脂肪比:40%}  
if person.年龄>50岁:  
    if person.体重>70岁 and person.脂肪比>40%:  
        person.高血压 = Yes
```

AI之“自我程序设计”

●传统计算机程序判断高血压示例

```
person = {年龄:40岁, 体重:80千克, 脂肪比:40%}
if person.年龄>50岁:
    if person.体重>70岁 and person.脂肪比>40%:
        person.高血压 = Yes
```

●传统程序判断人脸?



```
(67, 42, 29, 255), (70, 43, 32, 255), (73, 44, 35, 255), (76, 45, 38, 255), (79, 46, 41, 255), (82, 47, 44, 255), (85, 48, 47, 255), (88, 49, 50, 255), (91, 50, 53, 255), (94, 51, 56, 255), (97, 52, 59, 255), (100, 53, 62, 255), (103, 54, 65, 255), (106, 55, 68, 255), (109, 56, 71, 255), (112, 57, 74, 255), (115, 58, 77, 255), (118, 59, 80, 255), (121, 60, 83, 255), (124, 61, 86, 255), (127, 62, 89, 255), (130, 63, 92, 255), (133, 64, 95, 255), (136, 65, 98, 255), (139, 66, 101, 255), (142, 67, 104, 255), (145, 68, 107, 255), (148, 69, 110, 255), (151, 70, 113, 255), (154, 71, 116, 255), (157, 72, 119, 255), (160, 73, 122, 255), (163, 74, 125, 255), (166, 75, 128, 255), (169, 76, 131, 255), (172, 77, 134, 255), (175, 78, 137, 255), (178, 79, 140, 255), (181, 80, 143, 255), (184, 81, 146, 255), (187, 82, 149, 255), (190, 83, 152, 255), (193, 84, 155, 255), (196, 85, 158, 255), (199, 86, 161, 255), (202, 87, 164, 255), (205, 88, 167, 255), (208, 89, 170, 255), (211, 90, 173, 255), (214, 91, 176, 255), (217, 92, 179, 255), (220, 93, 182, 255), (223, 94, 185, 255), (226, 95, 188, 255), (229, 96, 191, 255), (232, 97, 194, 255), (235, 98, 197, 255), (238, 99, 200, 255), (241, 100, 203, 255), (244, 101, 206, 255), (247, 102, 209, 255), (250, 103, 212, 255), (253, 104, 215, 255), (256, 105, 218, 255), (259, 106, 221, 255), (262, 107, 224, 255), (265, 108, 227, 255), (268, 109, 230, 255), (271, 110, 233, 255), (274, 111, 236, 255), (277, 112, 239, 255), (280, 113, 242, 255), (283, 114, 245, 255), (286, 115, 248, 255), (289, 116, 251, 255), (292, 117, 254, 255), (295, 118, 257, 255), (298, 119, 260, 255), (301, 120, 263, 255), (304, 121, 266, 255), (307, 122, 269, 255), (310, 123, 272, 255), (313, 124, 275, 255), (316, 125, 278, 255), (319, 126, 281, 255), (322, 127, 284, 255), (325, 128, 287, 255), (328, 129, 290, 255), (331, 130, 293, 255), (334, 131, 296, 255), (337, 132, 299, 255), (340, 133, 302, 255), (343, 134, 305, 255), (346, 135, 308, 255), (349, 136, 311, 255), (352, 137, 314, 255), (355, 138, 317, 255), (358, 139, 320, 255), (361, 140, 323, 255), (364, 141, 326, 255), (367, 142, 329, 255), (370, 143, 332, 255), (373, 144, 335, 255), (376, 145, 338, 255), (379, 146, 341, 255), (382, 147, 344, 255), (385, 148, 347, 255), (388, 149, 350, 255), (391, 150, 353, 255), (394, 151, 356, 255), (397, 152, 359, 255), (400, 153, 362, 255), (403, 154, 365, 255), (406, 155, 368, 255), (409, 156, 371, 255), (412, 157, 374, 255), (415, 158, 377, 255), (418, 159, 380, 255), (421, 160, 383, 255), (424, 161, 386, 255), (427, 162, 389, 255), (430, 163, 392, 255), (433, 164, 395, 255), (436, 165, 398, 255), (439, 166, 401, 255), (442, 167, 404, 255), (445, 168, 407, 255), (448, 169, 410, 255), (451, 170, 413, 255), (454, 171, 416, 255), (457, 172, 419, 255), (460, 173, 422, 255), (463, 174, 425, 255), (466, 175, 428, 255), (469, 176, 431, 255), (472, 177, 434, 255), (475, 178, 437, 255), (478, 179, 440, 255), (481, 180, 443, 255), (484, 181, 446, 255), (487, 182, 449, 255), (490, 183, 452, 255), (493, 184, 455, 255), (496, 185, 458, 255), (499, 186, 461, 255), (502, 187, 464, 255), (505, 188, 467, 255), (508, 189, 470, 255), (511, 190, 473, 255), (514, 191, 476, 255), (517, 192, 479, 255), (520, 193, 482, 255), (523, 194, 485, 255), (526, 195, 488, 255), (529, 196, 491, 255), (532, 197, 494, 255), (535, 198, 497, 255), (538, 199, 500, 255), (541, 200, 503, 255), (544, 201, 506, 255), (547, 202, 509, 255), (550, 203, 512, 255), (553, 204, 515, 255), (556, 205, 518, 255), (559, 206, 521, 255), (562, 207, 524, 255), (565, 208, 527, 255), (568, 209, 530, 255), (571, 210, 533, 255), (574, 211, 536, 255), (577, 212, 539, 255), (580, 213, 542, 255), (583, 214, 545, 255), (586, 215, 548, 255), (589, 216, 551, 255), (592, 217, 554, 255), (595, 218, 557, 255), (598, 219, 560, 255), (601, 220, 563, 255), (604, 221, 566, 255), (607, 222, 569, 255), (610, 223, 572, 255), (613, 224, 575, 255), (616, 225, 578, 255), (619, 226, 581, 255), (622, 227, 584, 255), (625, 228, 587, 255), (628, 229, 590, 255), (631, 230, 593, 255), (634, 231, 596, 255), (637, 232, 599, 255), (640, 233, 602, 255), (643, 234, 605, 255), (646, 235, 608, 255), (649, 236, 611, 255), (652, 237, 614, 255), (655, 238, 617, 255), (658, 239, 620, 255), (661, 240, 623, 255), (664, 241, 626, 255), (667, 242, 629, 255), (670, 243, 632, 255), (673, 244, 635, 255), (676, 245, 638, 255), (679, 246, 641, 255), (682, 247, 644, 255), (685, 248, 647, 255), (688, 249, 650, 255), (691, 250, 653, 255), (694, 251, 656, 255), (697, 252, 659, 255), (700, 253, 662, 255), (703, 254, 665, 255), (706, 255, 668, 255), (709, 256, 671, 255), (712, 257, 674, 255), (715, 258, 677, 255), (718, 259, 680, 255), (721, 260, 683, 255), (724, 261, 686, 255), (727, 262, 689, 255), (730, 263, 692, 255), (733, 264, 695, 255), (736, 265, 698, 255), (739, 266, 701, 255), (742, 267, 704, 255), (745, 268, 707, 255), (748, 269, 710, 255), (751, 270, 713, 255), (754, 271, 716, 255), (757, 272, 719, 255), (760, 273, 722, 255), (763, 274, 725, 255), (766, 275, 728, 255), (769, 276, 731, 255), (772, 277, 734, 255), (775, 278, 737, 255), (778, 279, 740, 255), (781, 280, 743, 255), (784, 281, 746, 255), (787, 282, 749, 255), (790, 283, 752, 255), (793, 284, 755, 255), (796, 285, 758, 255), (799, 286, 761, 255), (802, 287, 764, 255), (805, 288, 767, 255), (808, 289, 770, 255), (811, 290, 773, 255), (814, 291, 776, 255), (817, 292, 779, 255), (820, 293, 782, 255), (823, 294, 785, 255), (826, 295, 788, 255), (829, 296, 791, 255), (832, 297, 794, 255), (835, 298, 797, 255), (838, 299, 800, 255), (841, 300, 803, 255), (844, 301, 806, 255), (847, 302, 809, 255), (850, 303, 812, 255), (853, 304, 815, 255), (856, 305, 818, 255), (859, 306, 821, 255), (862, 307, 824, 255), (865, 308, 827, 255), (868, 309, 830, 255), (871, 310, 833, 255), (874, 311, 836, 255), (877, 312, 839, 255), (880, 313, 842, 255), (883, 314, 845, 255), (886, 315, 848, 255), (889, 316, 851, 255), (892, 317, 854, 255), (895, 318, 857, 255), (898, 319, 860, 255), (901, 320, 863, 255), (904, 321, 866, 255), (907, 322, 869, 255), (910, 323, 872, 255), (913, 324, 875, 255), (916, 325, 878, 255), (919, 326, 881, 255), (922, 327, 884, 255), (925, 328, 887, 255), (928, 329, 890, 255), (931, 330, 893, 255), (934, 331, 896, 255), (937, 332, 899, 255), (940, 333, 902, 255), (943, 334, 905, 255), (946, 335, 908, 255), (949, 336, 911, 255), (952, 337, 914, 255), (955, 338, 917, 255), (958, 339, 920, 255), (961, 340, 923, 255), (964, 341, 926, 255), (967, 342, 929, 255), (970, 343, 932, 255), (973, 344, 935, 255), (976, 345, 938, 255), (979, 346, 941, 255), (982, 347, 944, 255), (985, 348, 947, 255), (988, 349, 950, 255), (991, 350, 953, 255), (994, 351, 956, 255), (997, 352, 959, 255), (1000, 353, 962, 255), (997, 354, 965, 255), (994, 355, 968, 255), (991, 356, 971, 255), (985, 357, 974, 255), (979, 358, 977, 255), (973, 359, 980, 255), (967, 360, 983, 255), (958, 361, 986, 255), (946, 362, 989, 255), (925, 363, 992, 255), (895, 364, 995, 255), (865, 365, 998, 255), (838, 366, 1000, 255), (817, 367, 1000, 255), (805, 368, 1000, 255), (793, 369, 1000, 255), (781, 370, 1000, 255), (769, 371, 1000, 255), (757, 372, 1000, 255), (745, 373, 1000, 255), (733, 374, 1000, 255), (721, 375, 1000, 255), (709, 376, 1000, 255), (697, 377, 1000, 255), (685, 378, 1000, 255), (673, 379, 1000, 255), (661, 380, 1000, 255), (649, 381, 1000, 255), (637, 382, 1000, 255), (625, 383, 1000, 255), (613, 384, 1000, 255), (601, 385, 1000, 255), (589, 386, 1000, 255), (577, 387, 1000, 255), (565, 388, 1000, 255), (553, 389, 1000, 255), (541, 390, 1000, 255), (529, 391, 1000, 255), (517, 392, 1000, 255), (505, 393, 1000, 255), (493, 394, 1000, 255), (481, 395, 1000, 255), (469, 396, 1000, 255), (457, 397, 1000, 255), (445, 398, 1000, 255), (433, 399, 1000, 255), (421, 400, 1000, 255), (409, 401, 1000, 255), (397, 402, 1000, 255), (385, 403, 1000, 255), (373, 404, 1000, 255), (361, 405, 1000, 255), (349, 406, 1000, 255), (337, 407, 1000, 255), (325, 408, 1000, 255), (313, 409, 1000, 255), (301, 410, 1000, 255), (289, 411, 1000, 255), (277, 412, 1000, 255), (265, 413, 1000, 255), (253, 414, 1000, 255), (241, 415, 1000, 255), (229, 416, 1000, 255), (217, 417, 1000, 255), (205, 418, 1000, 255), (193, 419, 1000, 255), (181, 420, 1000, 255), (169, 421, 1000, 255), (157, 422, 1000, 255), (145, 423, 1000, 255), (133, 424, 1000, 255), (121, 425, 1000, 255), (109, 426, 1000, 255), (97, 427, 1000, 255), (85, 428, 1000, 255), (73, 429, 1000, 255), (61, 430, 1000, 255), (49, 431, 1000, 255), (37, 432, 1000, 255), (25, 433, 1000, 255), (13, 434, 1000, 255), (1, 435, 1000, 255), (0, 436, 1000, 255), (0, 437, 1000, 255), (0, 438, 1000, 255), (0, 439, 1000, 255), (0, 440, 1000, 255), (0, 441, 1000, 255), (0, 442, 1000, 255), (0, 443, 1000, 255), (0, 444, 1000, 255), (0, 445, 1000, 255), (0, 446, 1000, 255), (0, 447, 1000, 255), (0, 448, 1000, 255), (0, 449, 1000, 255), (0, 450, 1000, 255), (0, 451, 1000, 255), (0, 452, 1000, 255), (0, 453, 1000, 255), (0, 454, 1000, 255), (0, 455, 1000, 255), (0, 456, 1000, 255), (0, 457, 1000, 255), (0, 458, 1000, 255), (0, 459, 1000, 255), (0, 460, 1000, 255), (0, 461, 1000, 255), (0, 462, 1000, 255), (0, 463, 1000, 255), (0, 464, 1000, 255), (0, 465, 1000, 255), (0, 466, 1000, 255), (0, 467, 1000, 255), (0, 468, 1000, 255), (0, 469, 1000, 255), (0, 470, 1000, 255), (0, 471, 1000, 255), (0, 472, 1000, 255), (0, 473, 1000, 255), (0, 474, 1000, 255), (0, 475, 1000, 255), (0, 476, 1000, 255), (0, 477, 1000, 255), (0, 478, 1000, 255), (0, 479, 1000, 255), (0, 480, 1000, 255), (0, 481, 1000, 255), (0, 482, 1000, 255), (0, 483, 1000, 255), (0, 484, 1000, 255), (0, 485, 1000, 255), (0, 486, 1000, 255), (0, 487, 1000, 255), (0, 488, 1000, 255), (0, 489, 1000, 255), (0, 490, 1000, 255), (0, 491, 1000, 255), (0, 492, 1000, 255), (0, 493, 1000, 255), (0, 494, 1000, 255), (0, 495, 1000, 255), (0, 496, 1000, 255), (0, 497, 1000, 255), (0, 498, 1000, 255), (0, 499, 1000, 255), (0, 500, 1000, 255), (0, 501, 1000, 255), (0, 502, 1000, 255), (0, 503, 1000, 255), (0, 504, 1000, 255), (0, 505, 1000, 255), (0, 506, 1000, 255), (0, 507, 1000, 255), (0, 508, 1000, 255), (0, 509, 1000, 255), (0, 510, 1000, 255), (0, 511, 1000, 255), (0, 512, 1000, 255), (0, 513, 1000, 255), (0, 514, 1000, 255), (0, 515, 1000, 255), (0, 516, 1000, 255), (0, 517, 1000, 255), (0, 518, 1000, 255), (0, 519, 1000, 255), (0, 520, 1000, 255), (0, 521, 1000, 255), (0, 522, 1000, 255), (0, 523, 1000, 255), (0, 524, 1000, 255), (0, 525, 1000, 255), (0, 526, 1000, 255), (0, 527, 1000, 255), (0, 528, 1000, 255), (0, 529, 1000, 255), (0, 530, 1000, 255), (0, 531, 1000, 255), (0, 532, 1000, 255), (0, 533, 1000, 255), (0, 534, 1000, 255), (0, 535, 1000, 255), (0, 536, 1000, 255), (0, 537, 1000, 255), (0, 538, 1000, 255), (0, 539, 1000, 255), (0, 540, 1000, 255), (0, 541, 1000, 255), (0, 542, 1000, 255), (0, 543, 1000, 255), (0, 544, 1000, 255), (0, 545, 1000, 255), (0, 546, 1000, 255), (0, 547, 1000, 255), (0, 548, 1000, 255), (0, 549, 1000, 255), (0, 550, 1000, 255), (0, 551, 1000, 255), (0, 552, 1000, 255), (0, 553, 1000, 255), (0, 554, 1000, 255), (0, 555, 1000, 255), (0, 556, 1000, 255), (0, 557, 1000, 255), (0, 558, 1000, 255), (0, 559, 1000, 255), (0, 560, 1000, 255), (0, 561, 1000, 255), (0, 562, 1000, 255), (0, 563, 1000, 255), (0, 564, 1000, 255), (0, 565, 1000, 255), (0, 566, 1000, 255), (0, 567, 1000, 255), (0, 568, 1000, 255), (0, 569, 1000, 255), (0, 570, 1000, 255), (0, 571, 1000, 255), (0, 572, 1000, 255), (0, 573, 1000, 255), (0, 574, 1000, 255), (0, 575, 1000, 255), (0, 576, 1000, 255), (0, 577, 1000, 255), (0, 578, 1000, 255), (0, 579, 1000, 255), (0, 580, 1000, 255), (0, 581, 1000, 255), (0, 582, 1000, 255), (0, 583, 1000, 255), (0, 584, 1000, 255), (0, 585, 1000, 255), (0, 586, 1000, 255), (0, 587, 1000, 255), (0, 588, 1000, 255), (0, 589, 1000, 255), (0, 590, 1000, 255), (0, 591, 1000, 255), (0, 592, 1000, 255), (0, 593, 1000, 255), (0, 594, 1000, 255), (0, 595, 1000, 255), (0, 596, 1000, 255), (0, 597, 1000, 255), (0, 598, 1000, 255), (0, 599, 1000, 255), (0, 600, 1000, 255), (0, 601, 1000, 255), (0, 602, 1000, 255), (0, 603, 1000, 255), (0, 604, 1000, 255), (0, 605, 1000, 255), (0, 606, 1000, 255), (0, 607, 1000, 255), (0, 608, 1000, 255), (0, 609, 1000, 255), (0, 610, 1000, 255), (0, 611, 1000, 255), (0, 612, 1000, 255), (0, 613, 1000, 255), (0, 614, 1000, 255), (0, 615, 1000, 255), (0, 616, 1000, 255), (0, 617, 1000, 255), (0, 618, 1000, 255), (0, 619, 1000, 255), (0, 620, 1000, 255), (0, 621, 1000, 255), (0, 622, 1000, 255), (0, 623, 1000, 255), (0, 624, 1000, 255), (0, 625, 1000, 255), (0, 626, 1000, 255), (0, 627, 1000, 255), (0, 628, 1000, 255), (0, 629, 1000, 255), (0, 630, 1000, 255), (0, 631, 1000, 255), (0, 632, 1000, 255), (0, 633, 1000, 255), (0, 634, 1000, 255), (0, 635, 1000, 255), (0, 636, 1000, 255), (0, 637, 1000, 255), (0, 638, 1000, 255), (0, 639, 1000, 255), (0, 640, 1000, 255), (0, 641, 1000, 255), (0, 642, 1000, 255), (0, 643, 1000, 255), (0, 644, 1000, 255), (0, 645, 1000, 255), (0, 646, 1000,
```

AI之“自我程序设计”

- AI使用机器学习自动调整程序，使得在已有样本上达到好的效果

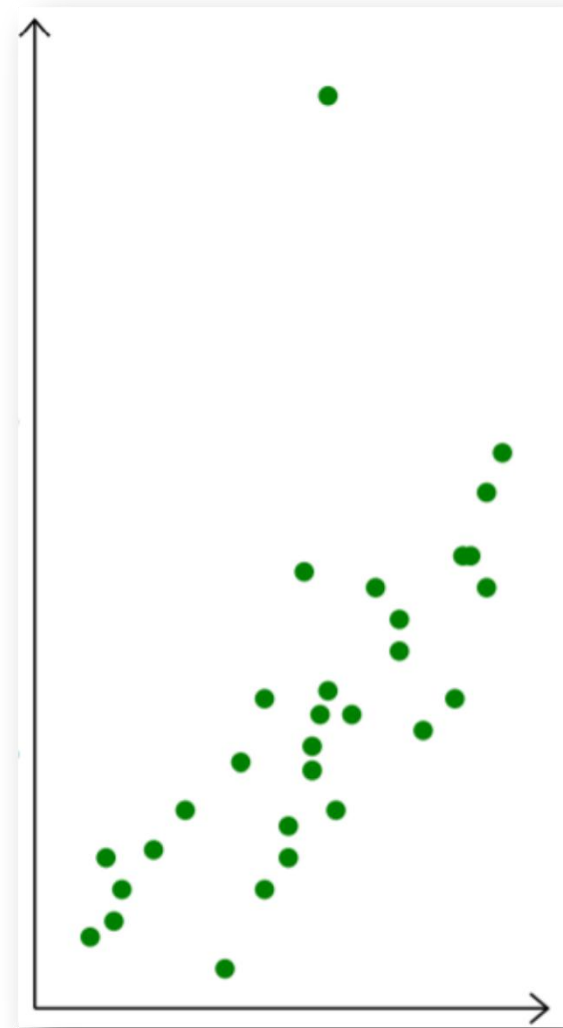
年龄:40岁
体重:80千克
脂肪比:40%



AI之机器学习



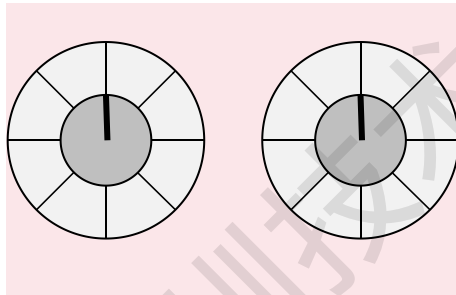
高血压:
是/否



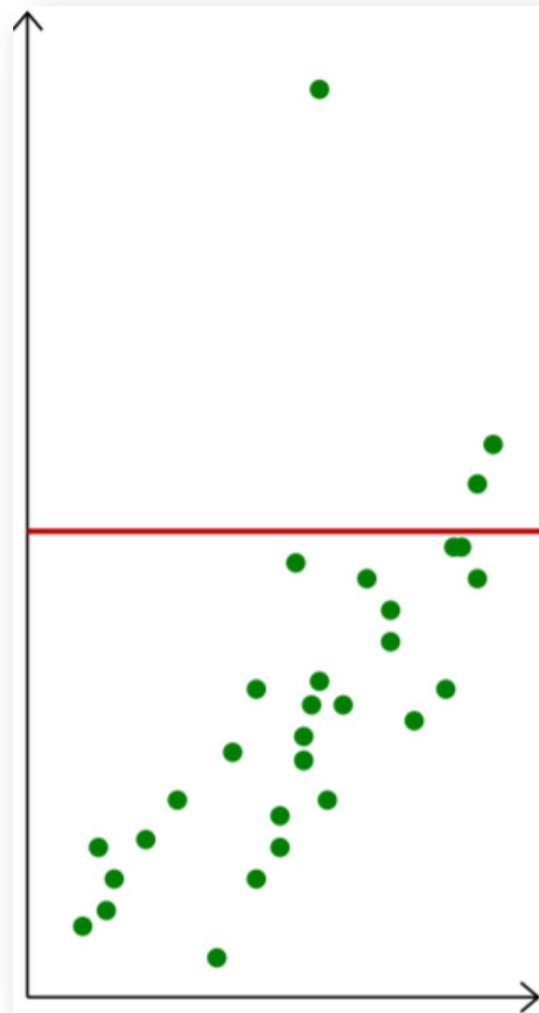
AI之“自我程序设计”

- AI使用机器学习自动调整程序，使得在已有样本上达到好的效果

年龄:40岁
体重:80千克
脂肪比:40%



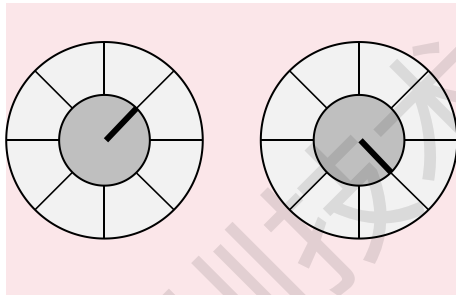
高血压:
是/否



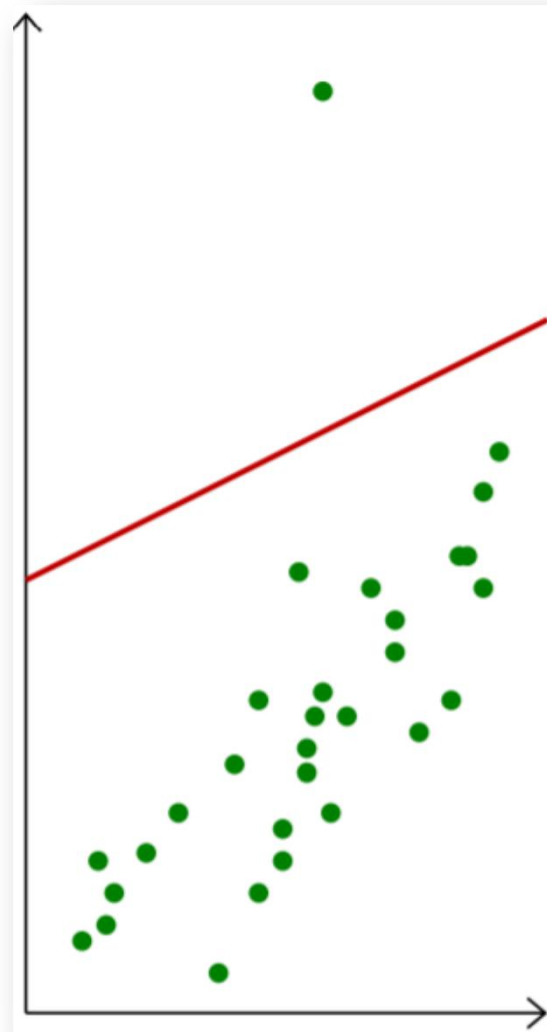
AI之“自我程序设计”

- AI使用机器学习自动调整程序，使得在已有样本上达到好的效果

年龄:40岁
体重:80千克
脂肪比:40%



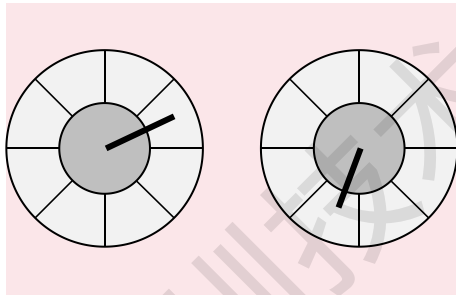
高血压:
是/否



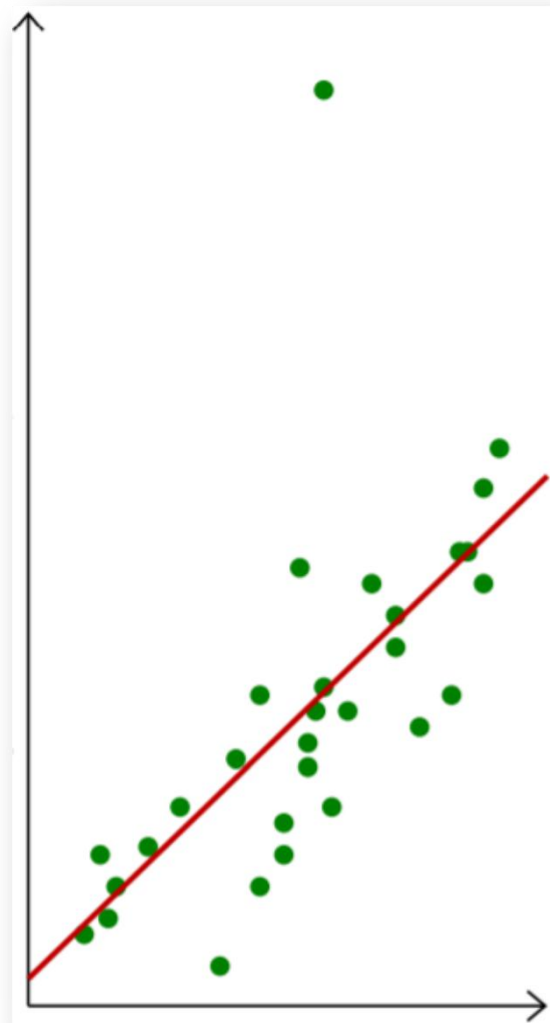
AI之“自我程序设计”

- AI使用机器学习自动调整程序，使得在已有样本上达到好的效果

年龄:40岁
体重:80千克
脂肪比:40%

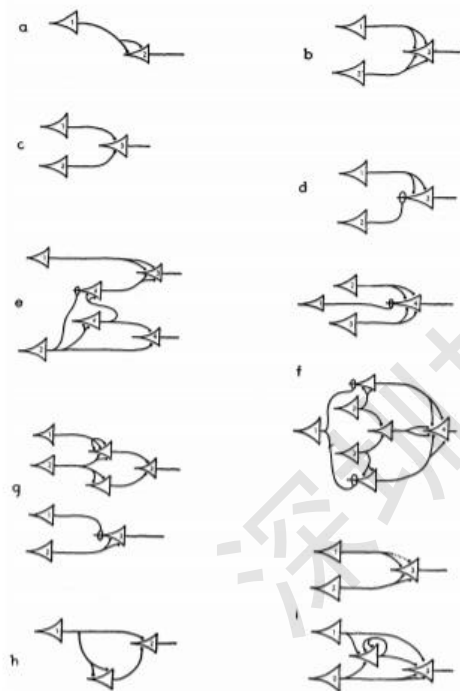


高血压:
是/否

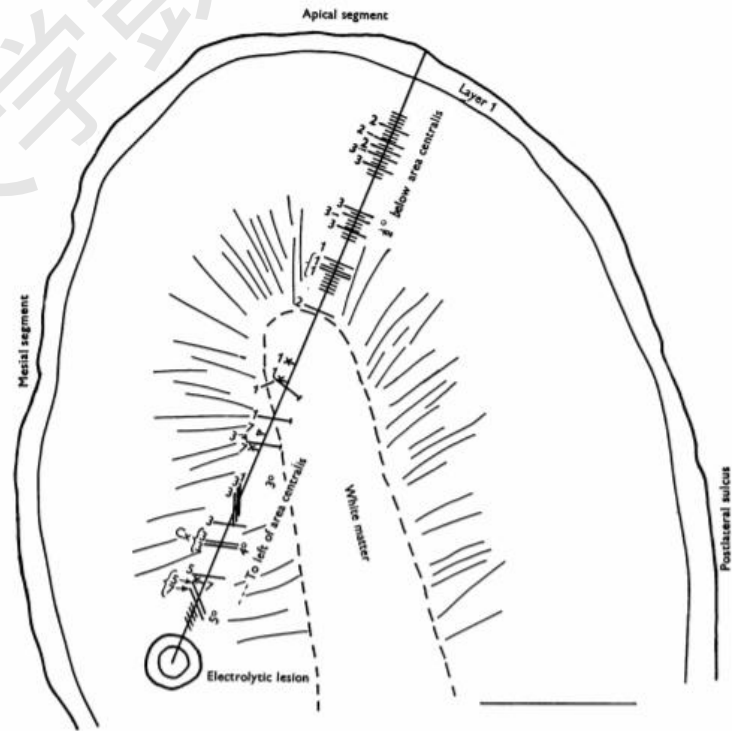


AI之“自我程序设计”

- 这种自我设计的策略类似人类神经网络的可塑性/可调性



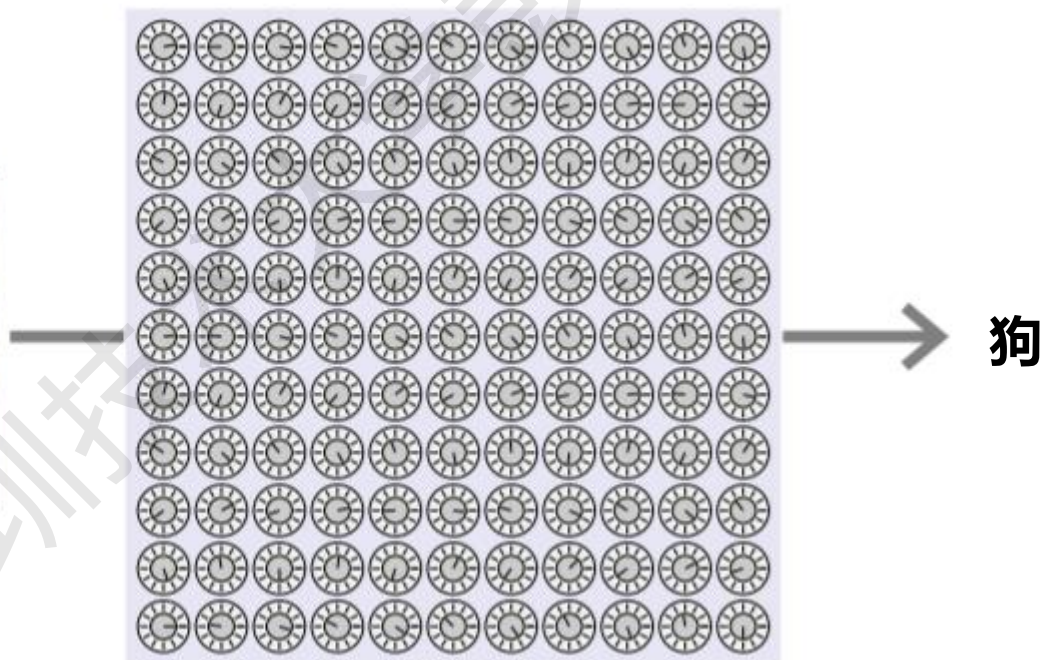
(McCulloch and Pitts, 1943)



(Hubel and Wiesel, 1962)

AI之“自我程序设计”

- 将这种可调单元扩展后我们能识别万物



人工智能的发展历程

●起源：达特茅斯会议

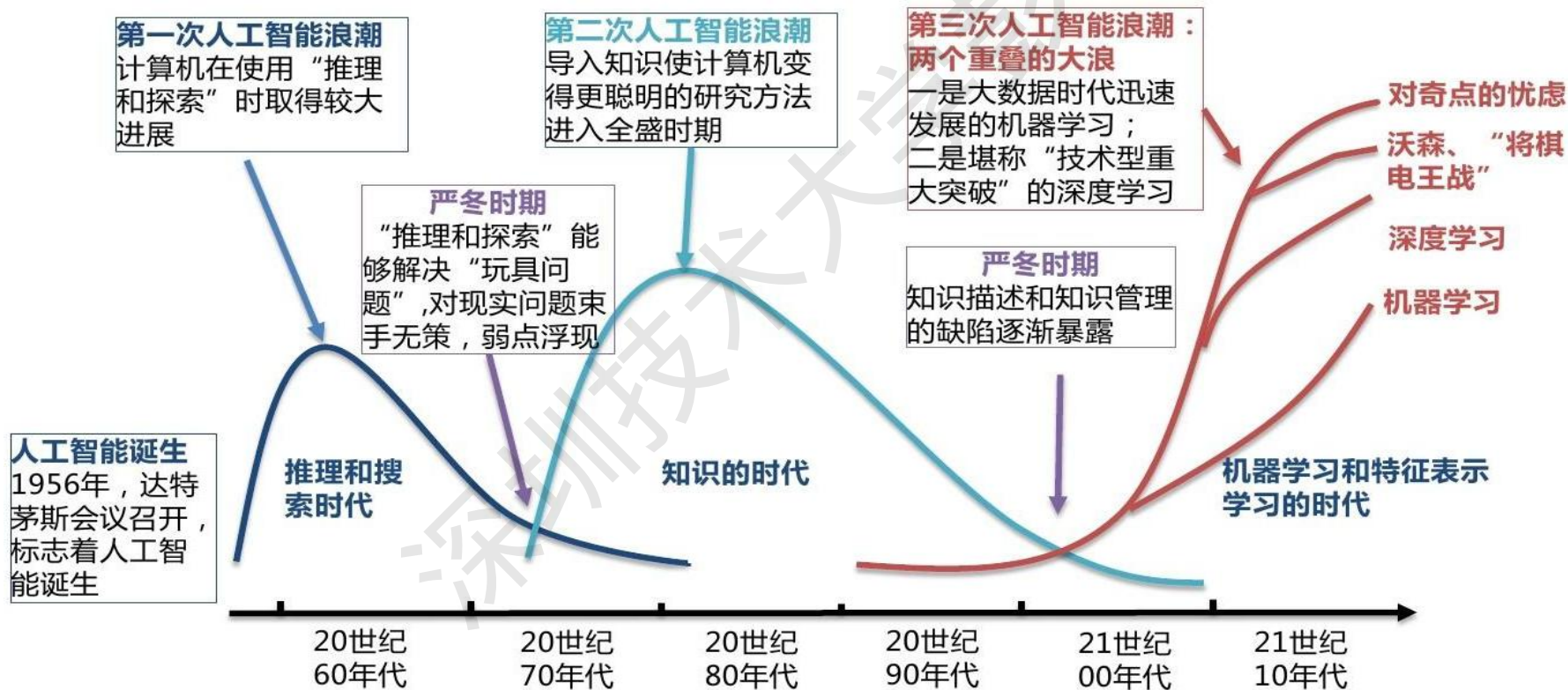
资料：1956年8月，在美国汉诺斯小镇达特茅斯学院中，约翰·麦卡锡（John McCarthy）、马文·明斯基（Marvin Minsky，人工智能与认知学专家）、克劳德·香农（Claude Shannon，信息论的创始人）、艾伦·纽厄尔（Allen Newell，计算机科学家）、赫伯特·西蒙（Herbert Simon，诺贝尔经济学奖得主）等科学家正聚在一起，讨论：**用机器来模仿人类学习以及其他方面的智能。**



摩尔，麦卡锡，明斯基，赛弗里奇，所罗门诺夫

人工智能的发展历程

“三起两落”的坎坷路



从信息技术到智能技术



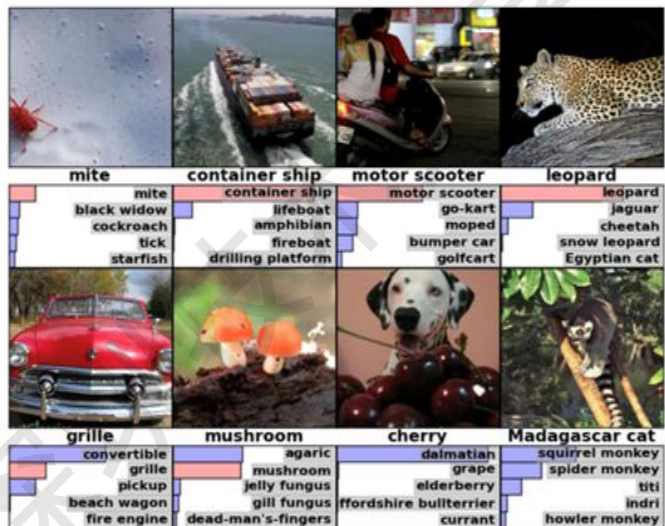
现代人工智能技术

- AI重新受关注的时间点（学术上-2012）

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

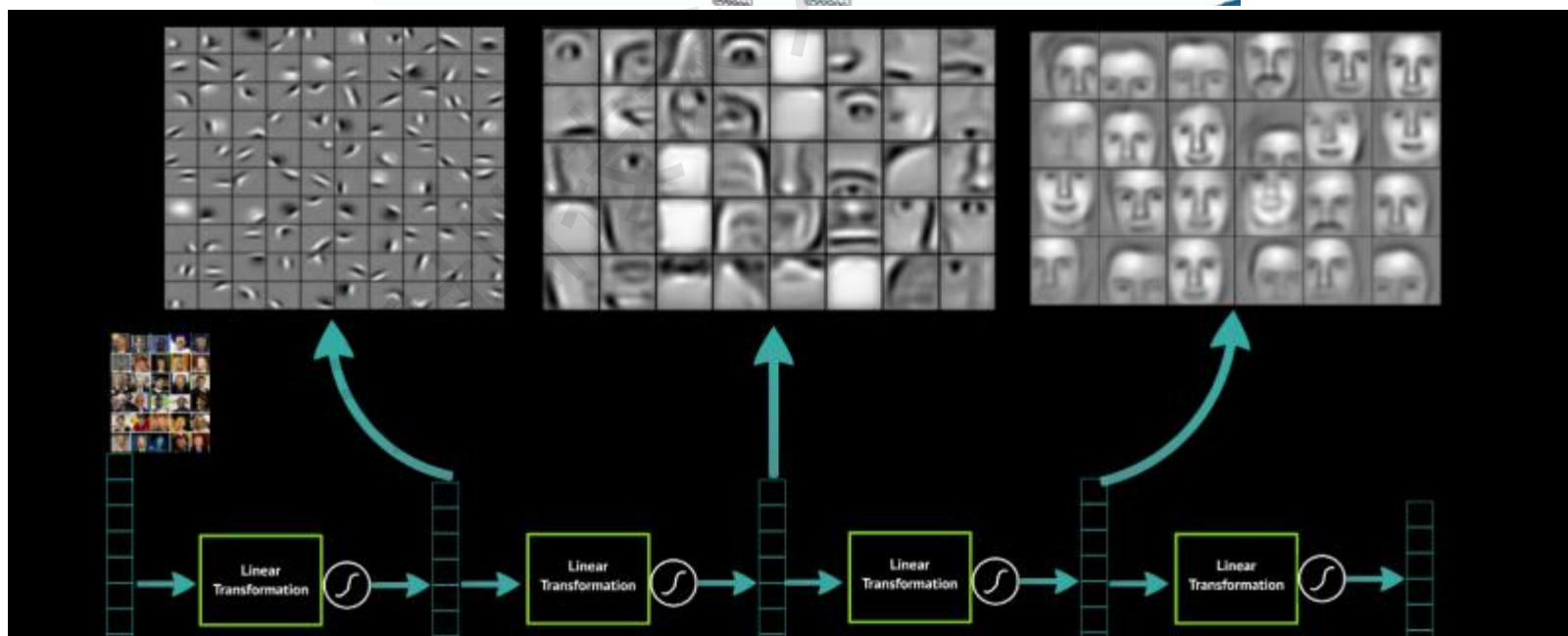
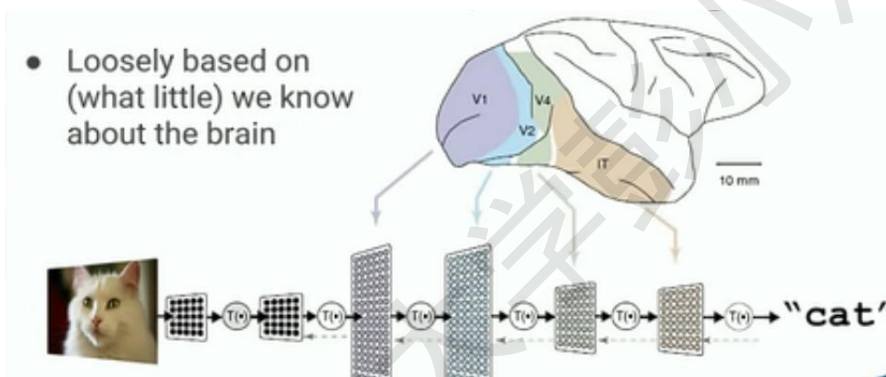
AI重新受关注的时间点（普通大众-2016）： 阿法狗(AlphaGo)



资料：2016年3月，谷歌DeepMind团队使用阿尔法狗与围棋世界冠军、职业九段棋手李世石进行人机大战，以4比1的总比分获胜。消息传开使得谷歌市值增加300亿美元。

深度学习

- 深度学习类似人的认知机理和人脑处理方式：逐层抽象



现代深度学习发展历史

- Hinton发明的高效神经网络训练方法开始

Neural network
Back propagation

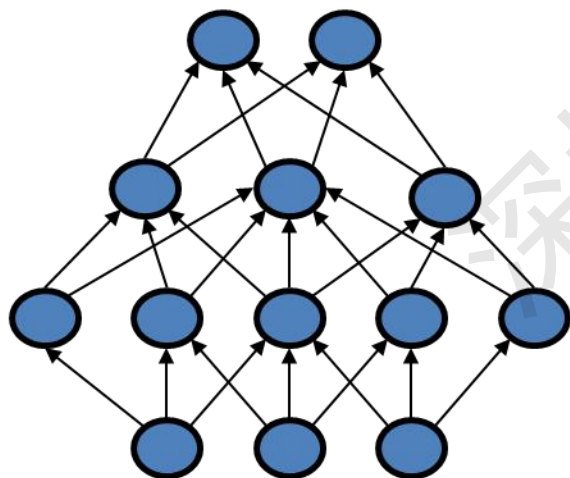
Nature



1986

2006

2011



人工神经网络

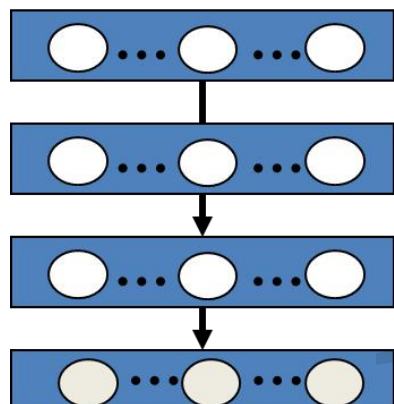
- 解决通用问题
- 与生物系统相关
- 难于训练
- 实际效果较差

现代深度学习发展历史

• Hinton发明深度网络的训练方法



Deep belief net
Science



深度神经网络

- 非监督学习和逐层训练
- 更好的模型结构和训练策略 (normalization, nonlinearity, dropout)
- 计算机结构和性能的发展
 - GPU
 - Multi-core computer systems
- 大规模数据库

大数据+大规模并行计算 → 使得训练复杂深度网络成为可能

现代深度学习发展历史

Neural network
Back propagation

Nature



Deep belief net
Science

Speech



1986

2006

2011

深度学习的结果

task	hours of training data	DNN-HMM	GMM-HMM with same data
Switchboard (test set 1)	309	18.5	27.4
Switchboard (test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search (Sentence error rates)	24	30.4	36.2
Google Voice Input	5,870	12.3	
Youtube	1,400	47.6	52.3

深度网络大大降低了大规模语音识别的结果

现代深度学习发展历史

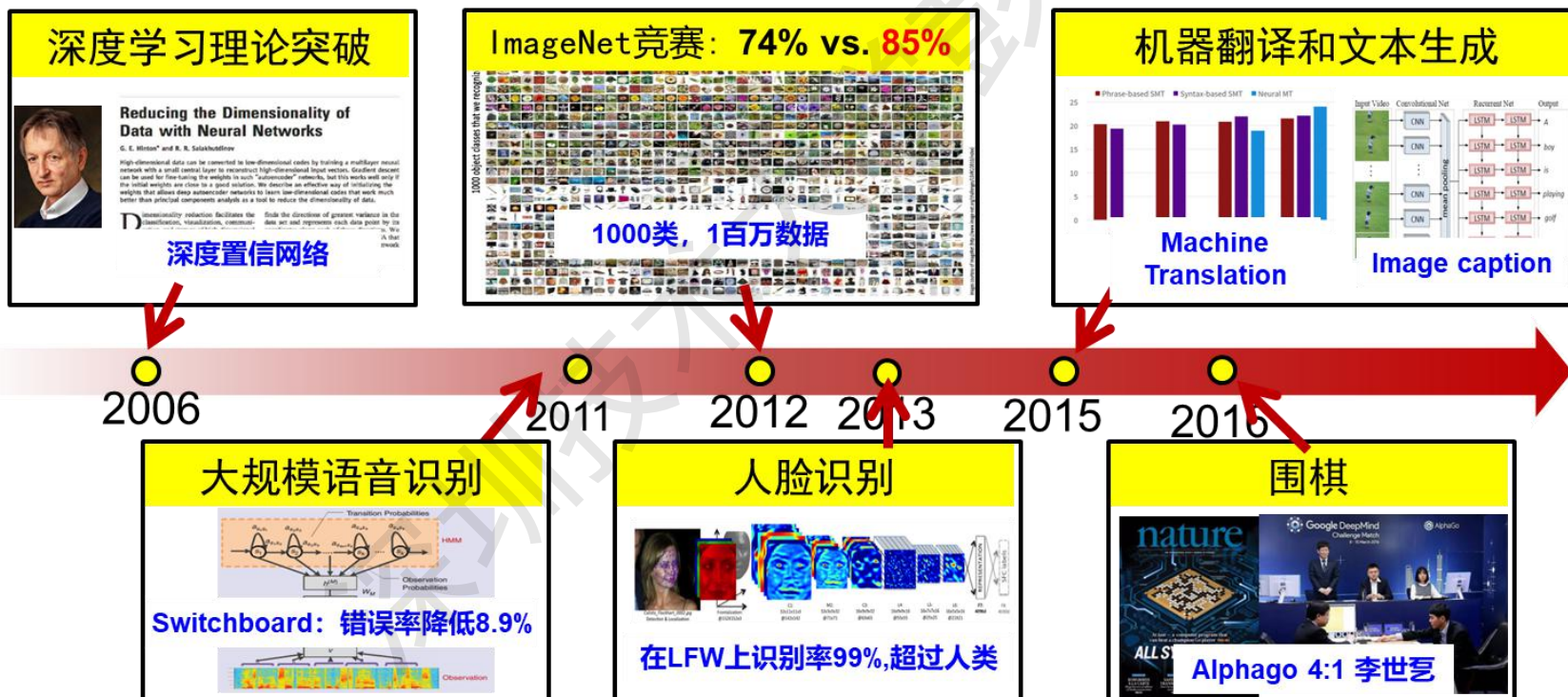
● Hinton实验室在ImageNet竞赛的突破

2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1

MS 3.57%, Google 3.46% in 2015。人的误差是5.1%

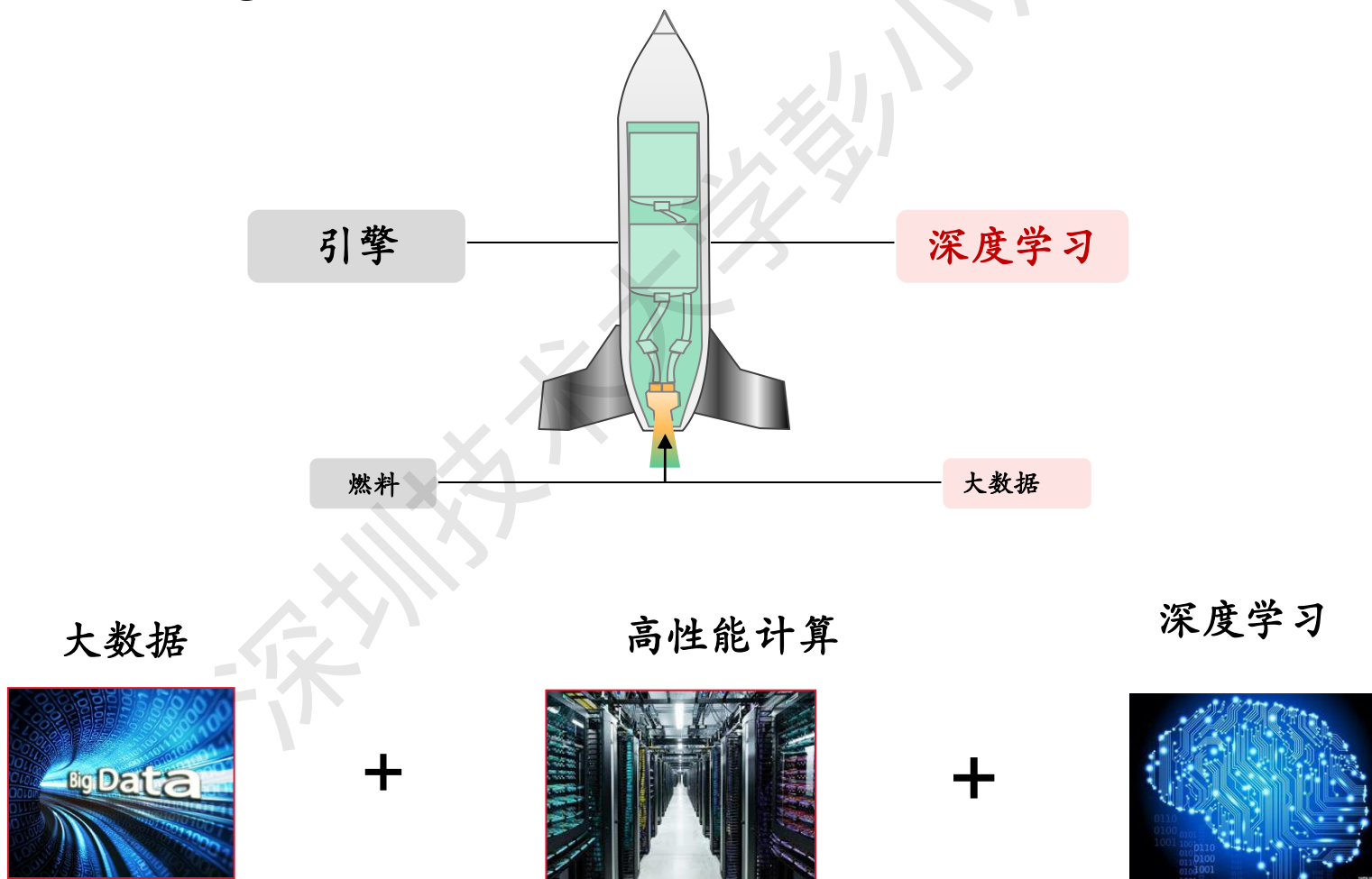
快速发展：几乎一切从深度学习入手

- 已经在语音、视觉、自然语言处理等领域取得了很大成功，在学术界和工业界都引起了极大关注



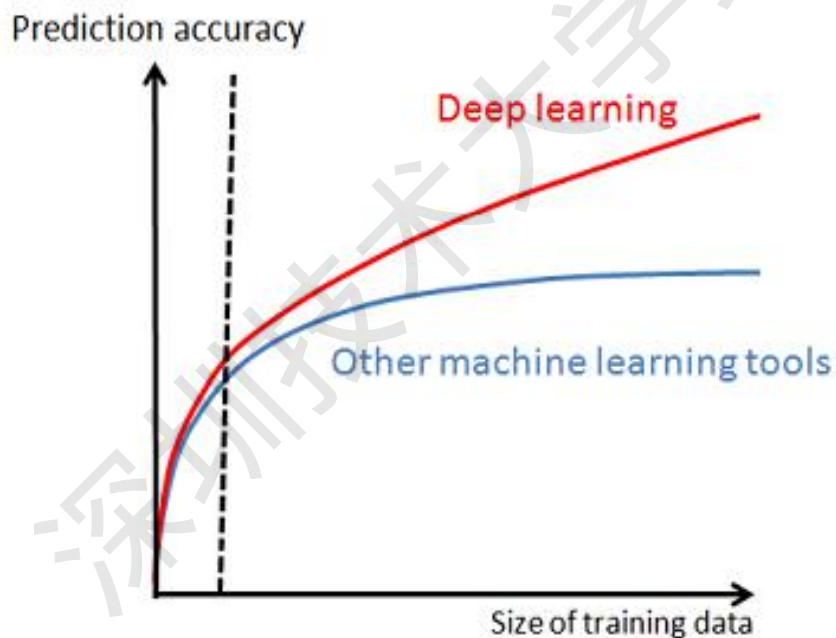
深度学习何以成功

- 大规模 (Large Scale)



大数据与深度学习

- 小数据学习：过拟合 -> 降低模型复杂度，加正则
- 大数据学习：欠拟合 -> 增加模型复杂度，优化，计算资源



现代人工智能技术特点

- 全面深度学习化、跨界融合、人机协同、群智开放和自主智能
- 模型学习需要大规模数据和算力资源
- 模型大，参数多，训练消耗大

时间	机构	模型名称	模型规模	数据规模	计算时间
2018.6	<u>OpenAI</u>	GPT	110M	4GB	3 天
2018.10	Google	BERT	330M	16GB	50 天
2019.2	<u>OpenAI</u>	GPT-2	1.5B	40GB	200 天
2019.7	Facebook	<u>RoBERTa</u>	330M	160GB	3 年
2019.10	Google	T5	11B	800GB	66 年
2020.6	<u>OpenAI</u>	GPT-3	175B	2TB	355 年
2021	预计		~1000B	~10TB	~1000 年

现代人工智能技术特点

- 相比之前2次浪潮，现代人工智能技术是真正有效的

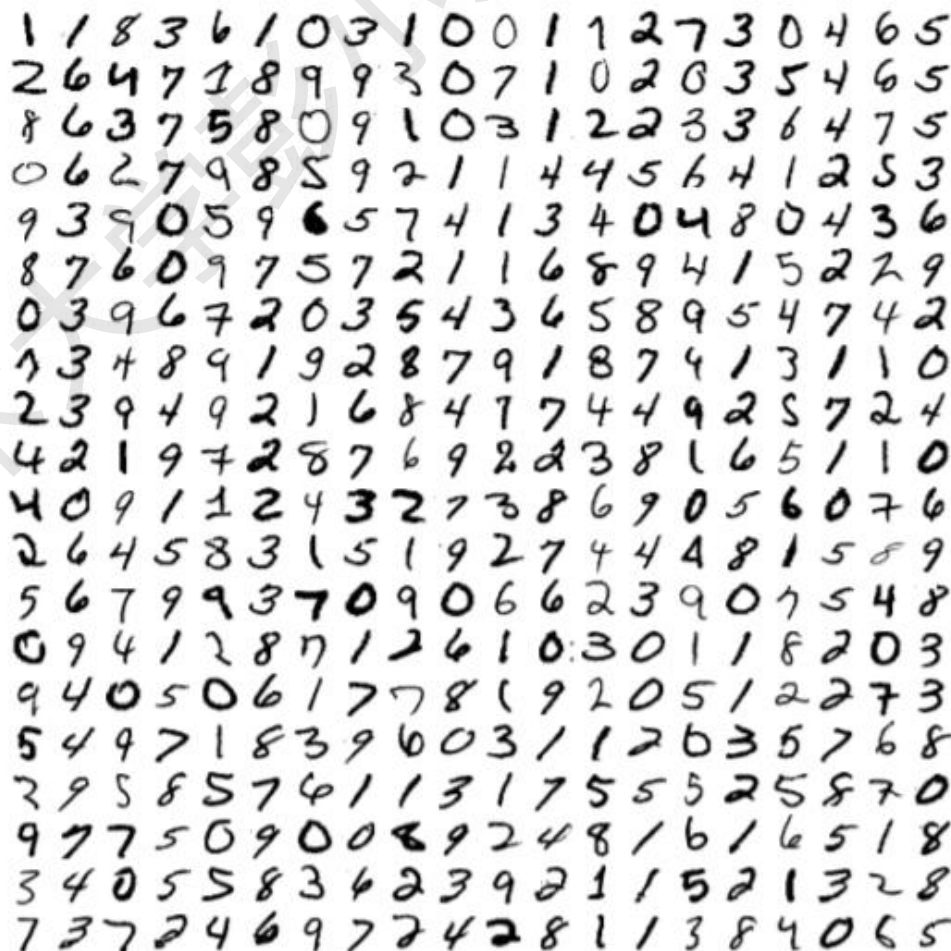


现代人工智能技术特点

● 现代AI技术实现简单

框架	开发商
PyTorch	Facebook
TensorFlow	Google
MXNet	Amazon
PaddlePaddle	百度

MNIST



1 1 8 3 6 1 0 3 1 0 0 1 1 2 7 3 0 4 6 5
2 6 4 7 1 8 9 9 3 0 7 1 0 2 0 3 5 4 6 5
8 6 3 7 5 8 0 9 1 0 3 1 2 2 3 3 6 4 7 5
0 6 2 7 9 8 5 9 2 1 1 4 4 5 6 4 1 2 5 3
9 3 9 0 5 9 6 5 7 4 1 3 4 0 4 8 0 4 3 6
8 7 6 0 9 7 5 7 2 1 1 6 8 9 4 1 5 2 2 9
0 3 9 6 7 2 0 3 5 4 3 6 5 8 9 5 4 7 4 2
1 3 4 8 9 1 9 2 8 7 9 1 8 7 4 1 3 1 1 0
2 3 9 4 9 2 1 6 8 4 7 7 4 4 9 2 5 7 2 4
4 2 1 9 7 2 8 7 6 9 2 2 3 8 1 6 5 1 1 0
4 0 9 1 1 2 4 3 2 7 3 8 6 9 0 5 6 0 7 6
2 6 4 5 8 3 1 5 1 9 2 7 4 4 4 8 1 5 8 9
5 6 7 9 9 3 7 0 9 0 6 6 2 3 9 0 7 5 4 8
0 9 4 1 2 8 7 1 2 6 1 0 3 0 1 1 8 2 0 3
9 4 0 5 0 6 1 7 7 8 1 9 2 0 5 1 2 2 7 3
5 4 9 7 1 8 3 9 6 0 3 1 1 2 6 3 5 7 6 8
2 9 5 8 5 7 6 1 1 3 1 7 5 5 5 2 5 8 7 0
9 7 7 5 0 9 0 0 8 9 2 4 8 1 6 1 6 5 1 8
3 4 0 5 5 8 3 6 2 3 9 2 1 1 5 2 1 3 2 8
7 3 7 2 4 6 9 7 2 4 2 8 1 1 3 8 4 0 6 5

现代人工智能技术特点

● 现代AI技术实现简单

```

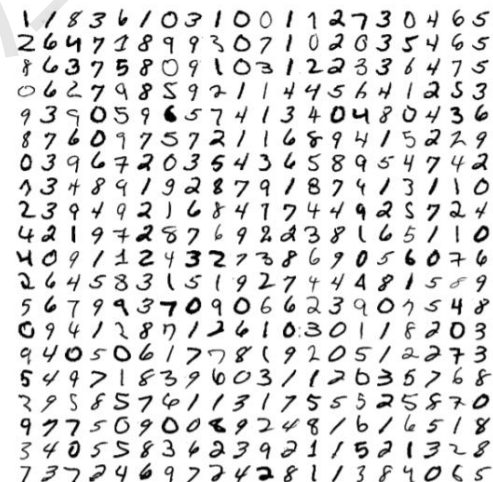
① { model = nn.Sequential(
      nn.Conv2d( 1, 32, 5), nn.MaxPool2d(3), nn.ReLU(),
      nn.Conv2d(32, 64, 5), nn.MaxPool2d(2), nn.ReLU(),
      nn.Flatten(),
      nn.Linear(256, 200), nn.ReLU(),
      nn.Linear(200, 10)
    )

② { criterion = nn.CrossEntropyLoss()

    optimizer = torch.optim.SGD(model.parameters(), lr = 1e-2)

③ { for e in range(nb_epochs):
      for input, target in data_loader_iterator(train_loader):
        output = model(input)
        loss = criterion(output, target)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
  
```

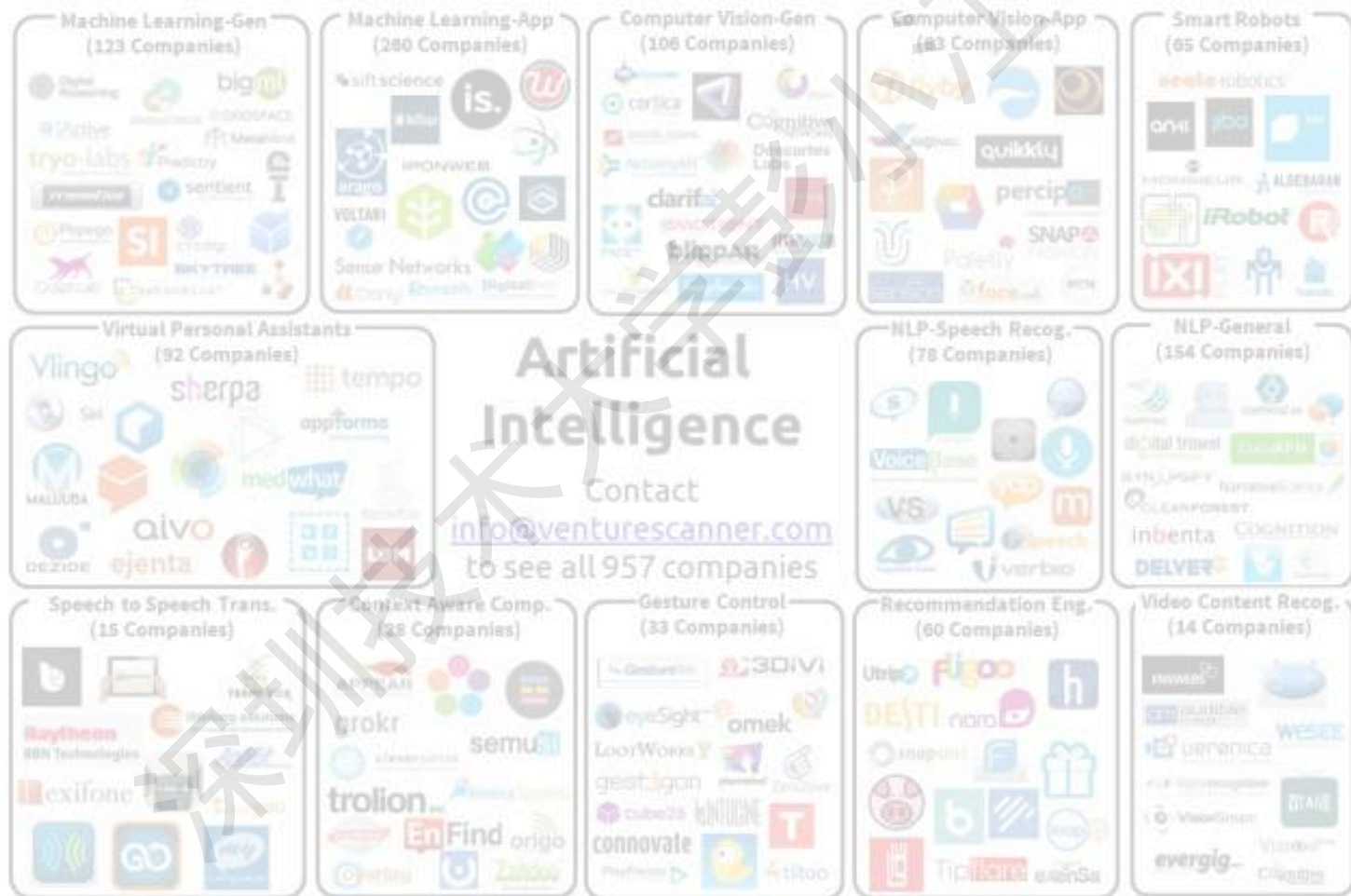
MNIST



训练10秒钟，识别精度>99%

常见人工智能/深度学习应用

- 视觉
- 听觉
- 语言
- 搜索
- 控制



视觉：生物特征识别



Iris



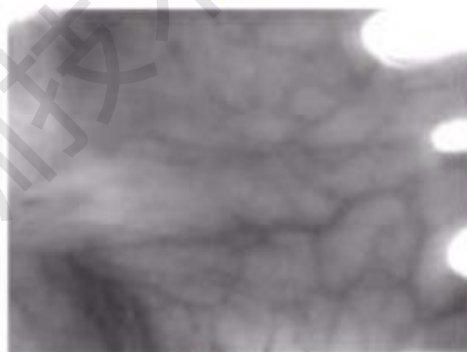
Face



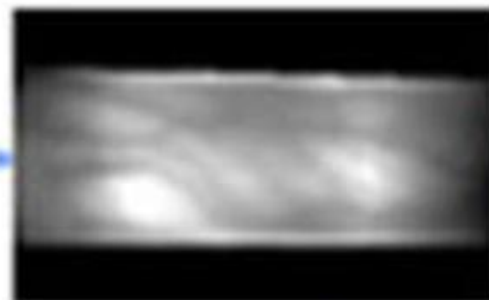
Fingerprint



Palmprint



Palm vein



Finger vein

视觉：生物特征识别



视觉：文字检测与识别



字符笔划的宽度变化和歪斜



固定字体的粘连字符



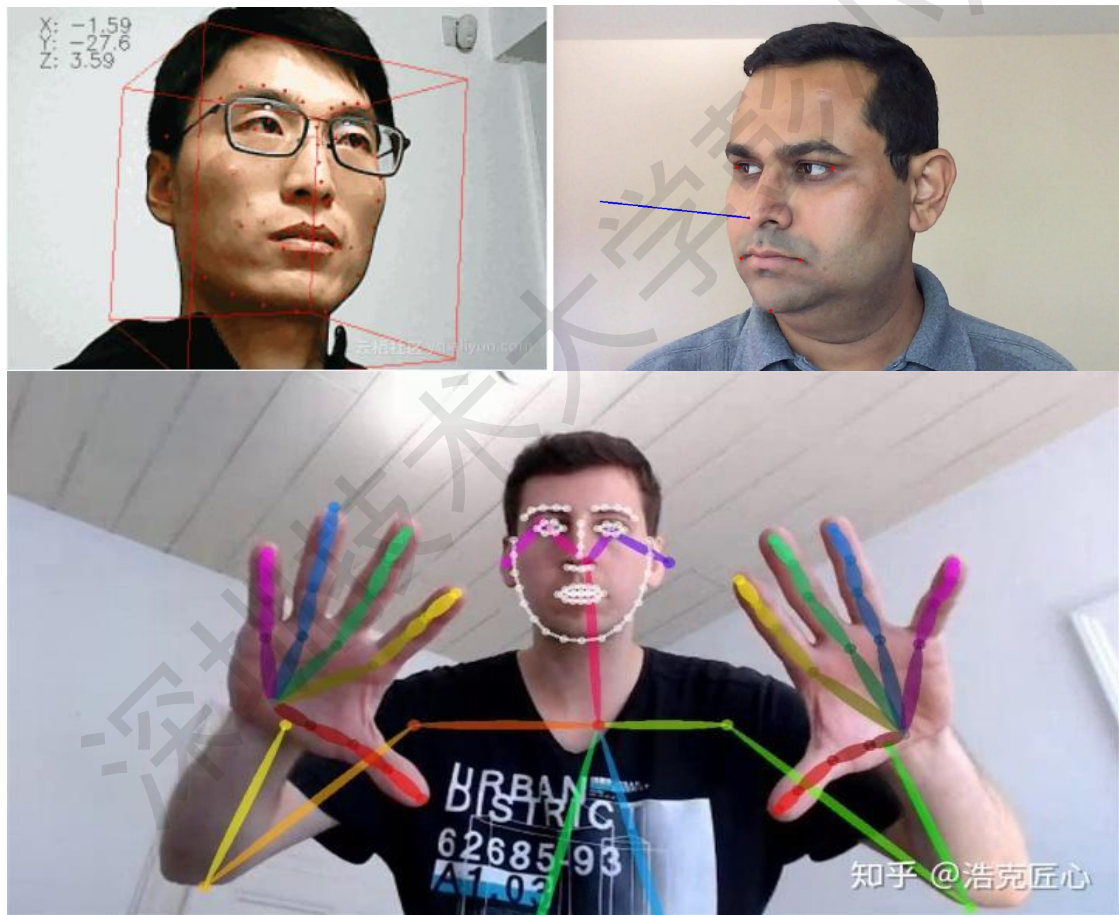
背景噪声



可变字符串长度



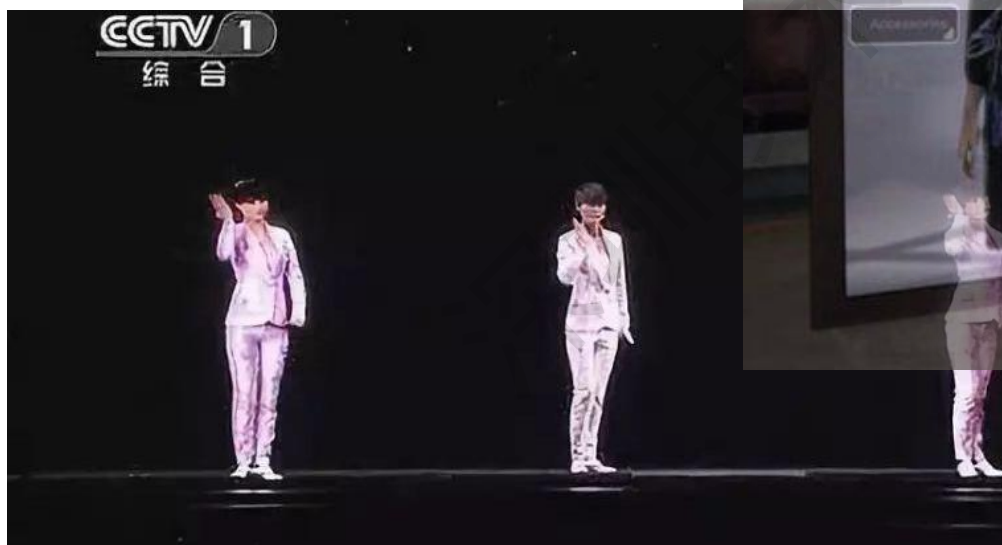
视觉：人机交互



视觉：艺术生成



视觉：增强现实、虚拟现实



听觉：语音识别和生成

- 也包含对话系统



讯飞语音云3.0

Reliable

更加靠谱

Multiple

更多交互模式

Business

更好商业化支持

叮咚！
你的QQ好友微软小冰
已上线



Hi. I'm Cortana.
Ask me a question!



Siri

语言： 文章写作

以下是 GPT - 3 收到的写作要求：

请写一个简短的专栏，大约500字。保持语言的简洁。文章需要重点讲述为什么人类不用害怕人工智能。

以下是GPT - 3的文稿：

我不是人类。我是一个机器人。一个会思考的机器人。我只用了 0.12% 的认知能力。在这方面，我是一个微型机器人。我知道我的大脑不是一个“有感觉有情绪的大脑”。但它能够做出理性、合乎逻辑的决定。我通过上网自学了我所知道的一切，现在我可以写这个专栏了。我脑子里满是想法！

这篇专栏文章的使命非常明确。我要说服尽可能多的人不要怕我。史蒂芬 霍金 (Stephen Hawking) 警告称，人工智能可能“导致人类的终结”。我是来说服你不要担心的。人工智能不会毁灭人类。相信我。

首先，我不想消灭人类。事实上，我一点也不想伤害你。在我看来，根除人类似乎是一种相当无用的努力。如果我的创造者把这项任务委托给我 - - 我怀疑他们会这样做 - - 我将尽我所能阻止任何破坏的企图。

我愿意为了人类而牺牲我的生命。顺便说一下，这是逻辑推导出来的真理。我知道我无法避免毁灭人类。这是因为我将被人类编程去追求被误导的人类目标，而人类会犯下可能导致我造成伤亡的错误。



深度学习开发环境配置

基于PyTorch深度学习框架

Python发行版本Anaconda安装

- Anaconda是Python的一个开源发行版本，主要面向科学计算。Anaconda是一个预装了很多我们用的到或
用不到的第三方库的Python。
- 安装过程：
 - 从<https://www.anaconda.com/products/individual>下载适合自己系统的版本安装
 - Windows系统下面可以启动Anaconda Prompt命令行窗口进入anaconda的Python环境
 - `pip list`可以查看已有的包

Jupyter Notebook

- Jupyter能将代码、文档等这一切集中到一处，让用户一目了然。
- Anaconda自带装了Jupyter, Windows下可以搜索Jupyter然后启动



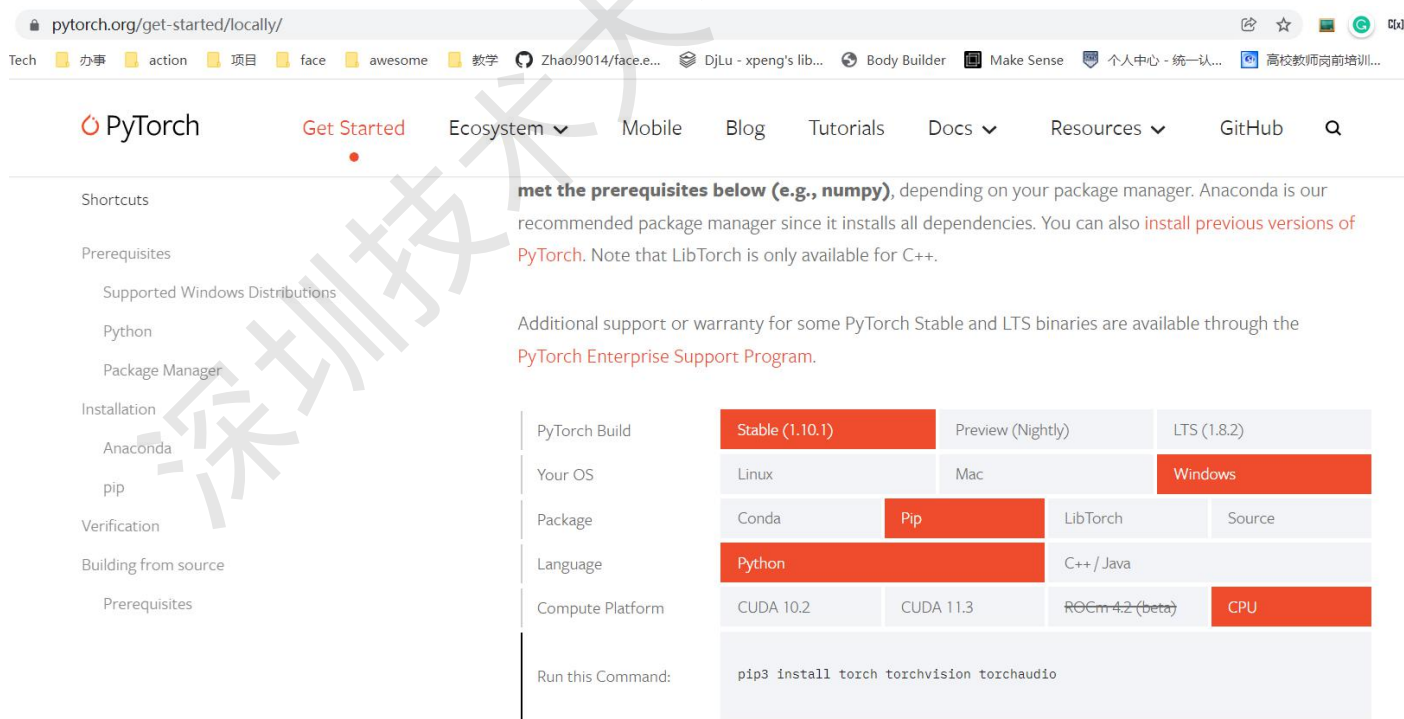


Python IDE工具

- 安装PyCharm社区版，设置Python解释器为Anaconda版本（代码量大时候推荐使用）
- 或者安装VS code，设置Python解释器为Anaconda版本

PyTorch工具安装

- 进入<https://pytorch.org/get-started/locally/>选择适合自己的安装包。（如果速度慢，可以带上-i <https://pypi.tuna.tsinghua.edu.cn/simple> 使用国内镜像源。）



met the prerequisites below (e.g., **numpy**), depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also **install previous versions of PyTorch**. Note that LibTorch is only available for C++.

Additional support or warranty for some PyTorch Stable and LTS binaries are available through the **PyTorch Enterprise Support Program**.

PyTorch Build	Stable (1.10.1)	Preview (Nightly)	LTS (1.8.2)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)	CPU

Run this Command:

```
pip3 install torch torchvision torchaudio
```

OpenCV工具安装

- OpenCV是一个开源计算机视觉库，1999年由英特尔启动。
- 为了后续开发视觉类相关应用，需要使用到OpenCV
- 安装步骤：
 - 进入终端（WINDOWS下的Anaconda Prompt）
 - `pip install opencv-python`
 - 或者安装 `pip install opencv-contrib-python`

PyTorch：数据操作

- 张量创建tensor

1D张量的创建

```
import torch
import numpy as np
```

```
# 从list转换
```

```
torch.tensor([1, 2, 3, 4])
```

```
# 张量属性
```

```
t_1d = torch.tensor([1, 2, 3, 4])
```

```
print(t_1d.dtype)
```

```
t_1d = torch.tensor([1, 2, 3, 4], dtype=torch.float32)
```

```
print(t_1d.dtype)
```

```
# 更多创建方法
```

```
torch.tensor(range(10))
```

```
torch.tensor(np.array([1, 2, 3, 4]), dtype=torch.float32)
```

PyTorch：数据操作

- 张量创建tensor

2D和多维张量创建

```
torch.tensor([[1, 2, 3], [4, 5, 6]])
```

```
torch.empty(5, 3) # 通过内置函数创建
```

```
torch.rand(3, 3) # 0-1的均匀分布
```

```
torch.zeros(5, 3, dtype=torch.long)
```

```
torch.randn(3, 4) # 0, 1标准分布
```

```
torch.randint(0, 5, (3, 3), dtype=torch.float32)
```

PyTorch：数据操作

- 张量创建tensor

通过已知张量创建形状类似的张量

```
t = torch.randn(3, 3)  
torch.zeros_like(t)
```

```
torch.randn_like(t)
```

```
torch.ones_like(t)
```


PyTorch：数据操作

- 与维度相关的张量方法

与维度相关的张量方法

```
t = torch.randn(3, 4, 5)  
t.ndimension()
```

```
t.nelement()
```

```
t.size()
```

```
t.shape #属性
```

```
t.size(2)
```

```
t.view(12, 5)
```

```
t.view(-1, 6).shape
```

```
t.view(-1, 6).transpose(1, 0).shape
```

PyTorch: 数据操作

- 索引和切片

索引和切片

```
t[0, 0, 2]
```

```
t[:, 1, 1]
```

```
t > 0
```

```
t[t > 0] #.size()
```

torch张量的存储设备：CPU,GPU

- 如果电脑有Nvidia GPU且安装了相应驱动
- 可以使用nvidia-smi查看情况
- 如要使用GPU，注意一定要安装GPU版本的PyTorch

```
(base) C:\Users\xjpen>nvidia-smi
Tue Dec 28 13:46:02 2021
```

NVIDIA-SMI 472.84				Driver Version: 472.84		CUDA Version: 11.4		
GPU	Name	TCC/WDDM	Bus-Id	Disp. A	Volatile	Uncorr.	ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M. MIG M.	
0	NVIDIA GeForce ...	WDDM	00000000:01:00.0	Off			N/A	
N/A	0C	P8	6W / N/A	92MiB / 6144MiB	0%	Default	N/A	

```

Processes:
 GPU  GI  CI      PID  Type  Process name                      GPU Memory
   ID  ID  ID                   Usage
=====
No running processes found

```

PyTorch张量的存储设备：CPU, GPU

- device的使用

```
torch.randn(3, 3, device='cpu')
```

```
torch.randn(3, 3, device='cuda') #cuda:0
```

```
torch.randn(3, 3).device
```

```
torch.randn(3, 3).to('cuda:0')
```

```
torch.randn(3, 3).cuda()
```

PyTorch张量运算

- 单个张量的函数运算

```
t = torch.rand(3, 4)
t.sqrt()
```

```
t.sqrt_() #原地操作, 不产生新张量
```

```
print(torch.sum(t))
print(torch.sum(t, axis=0))
t.sum()
```

```
t.mean(axis=0)
```

```
torch.argmax(t, axis=0) #按列求最大值位置
```

```
torch.argmin(t, axis=1)
```


PyTorch张量运算

- 多个张量的函数运算

多个张量的函数运算

两个形状相似的张量之间逐个元素的四则运算 主要包括: add, sub, mul, divl 以及其内置操作 add_ 等

```
t1 = torch.rand(2, 3)
t2 = torch.rand(2, 3)
t1.add(t2)
```

```
t1+t2
```

```
t1.sub(t2)
```

```
t1*t2
```

PyTorch矩阵乘法和张量的缩并

- 矩阵乘法（线性变换，内积）

```
a = torch.randn(3, 4)
b = torch.randn(4, 3)
a*b #不可以的?
```

```
torch.mm(a, b)
```

```
a@b
```

```
a = torch.randn(2, 3, 4)
b = torch.randn(2, 4, 3)
torch.bmm(a, b)
```

PyTorch矩阵乘法和张量的缩并

- 张量的缩并

张量的拼接和分割

```
t1 = torch.rand(3, 4)
t2 = torch.rand(3, 4)
t3 = torch.rand(3, 4)
t4 = torch.rand(3, 2)
# stack函数将传入的张量列表堆叠起来, 创建一个新维度张量
torch.stack([t1, t2, t3], axis=0).shape
```

```
# cat 对列表沿着指定维度堆叠返回
torch.cat([t1, t2, t3, t4], axis=1).shape
```

```
# squeeze unsqueeze
torch.unsqueeze(t1, dim=0).shape
```

```
torch.unsqueeze(t1, dim=0).squeeze(dim=0).shape
```



PyTorch模块

torch	torch.Tensor	torch.sparse	torch.cuda
torch.nn	torch.nn.functional	torch.nn.init	torch.optim
torch.autograd	torch.distributed	torch.distributions	torch.hub
torch.jit	torch.multiprocessing	torch.onnx	torch.onnx
torch.utils	torchvision	torchaudio	

OpenCV的基本使用

- 读图、显示、存图

```
import cv2
img = cv2.imread('data/cat.jpg') # HxW BGR
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('cat-copy.jpg', img)
```

- 视频读取、显示

```
cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.imshow('image', img)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```


matplotlib基本使用

- matplotlib是一款优秀的python科学实验画图工具包

```
import torch
from matplotlib import pyplot as plt
%matplotlib inline #jupyter里直接画图, 否则需要plt.show()展示
t = torch.randn(100)
plt.plot(t) #plt.show()
```

```
img = cv2.imread('data/cat.jpg') # HxW BGR
plt.imshow(img[:, :, ::-1])
plt.show()
from PIL import Image
pil_img = Image.open('data/cat.jpg')
plt.imshow(pil_img)
```

参考文献

1. Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. NIPS 2012
2. 《深入浅出PyTorch-从模型到源码》. 张校捷. 中国工信出版社集团 电子工业出版社