

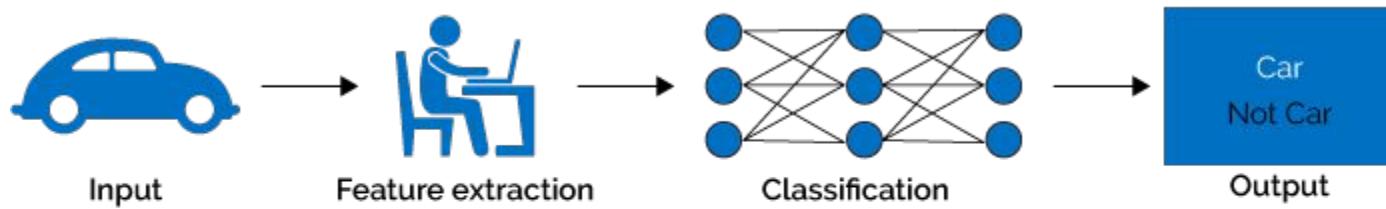
# REDES NEURAIS CONVOLUCIONAIS

Thaís de Almeida Ratis Ramos

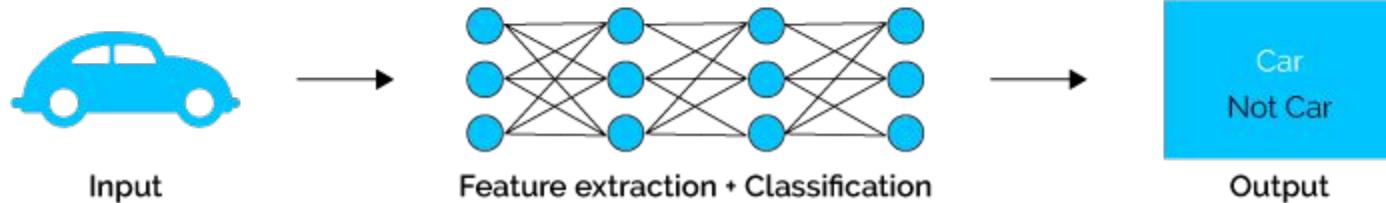
Baseado nas aulas de Thaís Gaudencio e Yuri Malheiros – UFPB

# MACHINE LEARNING X DEEP LEARNING

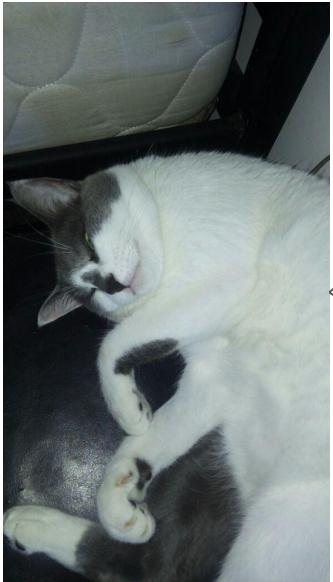
## Machine Learning



## Deep Learning



# VISÃO



- Gato
- Branco e cinza
- Dormindo

```
>>> a = cv2.imread('img.jpg')
>>> a
```

array([[[ 87, 93, 82], [ 88, 94, 83], [ 89, 95, 84],  
 [ 90, 96, 85], [ 91, 97, 86], [ 92, 98, 87], [ 93, 99, 88], [ 94, 100, 89], [ 95, 101, 90], [ 96, 102, 91], [ 97, 103, 92], [ 98, 104, 93], [ 99, 105, 94], [100, 106, 95], [101, 107, 96], [102, 108, 97], [103, 109, 98], [104, 110, 99], [105, 111, 100], [106, 112, 101], [107, 113, 102], [108, 114, 103], [109, 115, 104], [110, 116, 105], [111, 117, 106], [112, 118, 107], [113, 119, 108], [114, 120, 109], [115, 121, 110], [116, 122, 111], [117, 123, 112], [118, 124, 113], [119, 125, 114], [120, 126, 115], [121, 127, 116], [122, 128, 117], [123, 129, 118], [124, 130, 119], [125, 131, 120], [126, 132, 121], [127, 133, 122], [128, 134, 123], [129, 135, 124], [130, 136, 125], [131, 137, 126], [132, 138, 127], [133, 139, 128], [134, 140, 129], [135, 141, 130], [136, 142, 131], [137, 143, 132], [138, 144, 133], [139, 145, 134], [140, 146, 135], [141, 147, 136], [142, 148, 137], [143, 149, 138], [144, 150, 139], [145, 151, 140], [146, 152, 141], [147, 153, 142], [148, 154, 143], [149, 155, 144], [150, 156, 145], [151, 157, 146], [152, 158, 147], [153, 159, 148], [154, 160, 149], [155, 161, 150], [156, 162, 151], [157, 163, 152], [158, 164, 153], [159, 165, 154], [160, 166, 155], [161, 167, 156], [162, 168, 157], [163, 169, 158], [164, 170, 159], [165, 171, 160], [166, 172, 161], [167, 173, 162], [168, 174, 163], [169, 175, 164], [170, 176, 165], [171, 177, 166], [172, 178, 167], [173, 179, 168], [174, 180, 169], [175, 181, 170], [176, 182, 171], [177, 183, 172], [178, 184, 173], [179, 185, 174], [180, 186, 175], [181, 187, 176], [182, 188, 177], [183, 189, 178], [184, 190, 179], [185, 191, 180], [186, 192, 181], [187, 193, 182], [188, 194, 183], [189, 195, 184], [190, 196, 185], [191, 197, 186], [192, 198, 187], [193, 199, 188], [194, 200, 189], [195, 201, 190], [196, 202, 191], [197, 203, 192], [198, 204, 193], [199, 205, 194], [200, 206, 195], [201, 207, 196], [202, 208, 197], [203, 209, 198], [204, 210, 199], [205, 211, 200], [206, 212, 201], [207, 213, 202], [208, 214, 203], [209, 215, 204], [210, 216, 205], [211, 217, 206], [212, 218, 207], [213, 219, 208], [214, 220, 209], [215, 221, 210], [216, 222, 211], [217, 223, 212], [218, 224, 213], [219, 225, 214], [220, 226, 215], [221, 227, 216], [222, 228, 217], [223, 229, 218], [224, 230, 219], [225, 231, 220], [226, 232, 221], [227, 233, 222], [228, 234, 223], [229, 235, 224], [230, 236, 225], [231, 237, 226], [232, 238, 227], [233, 239, 228], [234, 240, 229], [235, 241, 230], [236, 242, 231], [237, 243, 232], [238, 244, 233], [239, 245, 234], [240, 246, 235], [241, 247, 236], [242, 248, 237], [243, 249, 238], [244, 250, 239], [245, 251, 240], [246, 252, 241], [247, 253, 242], [248, 254, 243], [249, 255, 244], [250, 256, 245], [251, 257, 246], [252, 258, 247], [253, 259, 248], [254, 260, 249], [255, 261, 250], [256, 262, 251], [257, 263, 252], [258, 264, 253], [259, 265, 254], [260, 266, 255], [261, 267, 256], [262, 268, 257], [263, 269, 258], [264, 270, 259], [265, 271, 260], [266, 272, 261], [267, 273, 262], [268, 274, 263], [269, 275, 264], [270, 276, 265], [271, 277, 266], [272, 278, 267], [273, 279, 268], [274, 280, 269], [275, 281, 270], [276, 282, 271], [277, 283, 272], [278, 284, 273], [279, 285, 274], [280, 286, 275], [281, 287, 276], [282, 288, 277], [283, 289, 278], [284, 290, 279], [285, 291, 280], [286, 292, 281], [287, 293, 282], [288, 294, 283], [289, 295, 284], [290, 296, 285], [291, 297, 286], [292, 298, 287], [293, 299, 288], [294, 300, 289], [295, 301, 290], [296, 302, 291], [297, 303, 292], [298, 304, 293], [299, 305, 294], [300, 306, 295], [301, 307, 296], [302, 308, 297], [303, 309, 298], [304, 310, 299], [305, 311, 300], [306, 312, 301], [307, 313, 302], [308, 314, 303], [309, 315, 304], [310, 316, 305], [311, 317, 306], [312, 318, 307], [313, 319, 308], [314, 320, 309], [315, 321, 310], [316, 322, 311], [317, 323, 312], [318, 324, 313], [319, 325, 314], [320, 326, 315], [321, 327, 316], [322, 328, 317], [323, 329, 318], [324, 330, 319], [325, 331, 320], [326, 332, 321], [327, 333, 322], [328, 334, 323], [329, 335, 324], [330, 336, 325], [331, 337, 326], [332, 338, 327], [333, 339, 328], [334, 340, 329], [335, 341, 330], [336, 342, 331], [337, 343, 332], [338, 344, 333], [339, 345, 334], [340, 346, 335], [341, 347, 336], [342, 348, 337], [343, 349, 338], [344, 350, 339], [345, 351, 340], [346, 352, 341], [347, 353, 342], [348, 354, 343], [349, 355, 344], [350, 356, 345], [351, 357, 346], [352, 358, 347], [353, 359, 348], [354, 360, 349], [355, 361, 350], [356, 362, 351], [357, 363, 352], [358, 364, 353], [359, 365, 354], [360, 366, 355], [361, 367, 356], [362, 368, 357], [363, 369, 358], [364, 370, 359], [365, 371, 360], [366, 372, 361], [367, 373, 362], [368, 374, 363], [369, 375, 364], [370, 376, 365], [371, 377, 366], [372, 378, 367], [373, 379, 368], [374, 380, 369], [375, 381, 370], [376, 382, 371], [377, 383, 372], [378, 384, 373], [379, 385, 374], [380, 386, 375], [381, 387, 376], [382, 388, 377], [383, 389, 378], [384, 390, 379], [385, 391, 380], [386, 392, 381], [387, 393, 382], [388, 394, 383], [389, 395, 384], [390, 396, 385], [391, 397, 386], [392, 398, 387], [393, 399, 388], [394, 400, 389], [395, 401, 390], [396, 402, 391], [397, 403, 392], [398, 404, 393], [399, 405, 394], [400, 406, 395], [401, 407, 396], [402, 408, 397], [403, 409, 398], [404, 410, 399], [405, 411, 400], [406, 412, 401], [407, 413, 402], [408, 414, 403], [409, 415, 404], [410, 416, 405], [411, 417, 406], [412, 418, 407], [413, 419, 408], [414, 420, 409], [415, 421, 410], [416, 422, 411], [417, 423, 412], [418, 424, 413], [419, 425, 414], [420, 426, 415], [421, 427, 416], [422, 428, 417], [423, 429, 418], [424, 430, 419], [425, 431, 420], [426, 432, 421], [427, 433, 422], [428, 434, 423], [429, 435, 424], [430, 436, 425], [431, 437, 426], [432, 438, 427], [433, 439, 428], [434, 440, 429], [435, 441, 430], [436, 442, 431], [437, 443, 432], [438, 444, 433], [439, 445, 434], [440, 446, 435], [441, 447, 436], [442, 448, 437], [443, 449, 438], [444, 450, 439], [445, 451, 440], [446, 452, 441], [447, 453, 442], [448, 454, 443], [449, 455, 444], [450, 456, 445], [451, 457, 446], [452, 458, 447], [453, 459, 448], [454, 460, 449], [455, 461, 450], [456, 462, 451], [457, 463, 452], [458, 464, 453], [459, 465, 454], [460, 466, 455], [461, 467, 456], [462, 468, 457], [463, 469, 458], [464, 470, 459], [465, 471, 460], [466, 472, 461], [467, 473, 462], [468, 474, 463], [469, 475, 464], [470, 476, 465], [471, 477, 466], [472, 478, 467], [473, 479, 468], [474, 480, 469], [475, 481, 470], [476, 482, 471], [477, 483, 472], [478, 484, 473], [479, 485, 474], [480, 486, 475], [481, 487, 476], [482, 488, 477], [483, 489, 478], [484, 490, 479], [485, 491, 480], [486, 492, 481], [487, 493, 482], [488, 494, 483], [489, 495, 484], [490, 496, 485], [491, 497, 486], [492, 498, 487], [493, 499, 488], [494, 500, 489], [495, 501, 490], [496, 502, 491], [497, 503, 492], [498, 504, 493], [499, 505, 494], [500, 506, 495], [501, 507, 496], [502, 508, 497], [503, 509, 498], [504, 510, 499], [505, 511, 500], [506, 512, 501], [507, 513, 502], [508, 514, 503], [509, 515, 504], [510, 516, 505], [511, 517, 506], [512, 518, 507], [513, 519, 508], [514, 520, 509], [515, 521, 510], [516, 522, 511], [517, 523, 512], [518, 524, 513], [519, 525, 514], [520, 526, 515], [521, 527, 516], [522, 528, 517], [523, 529, 518], [524, 530, 519], [525, 531, 520], [526, 532, 521], [527, 533, 522], [528, 534, 523], [529, 535, 524], [530, 536, 525], [531, 537, 526], [532, 538, 527], [533, 539, 528], [534, 540, 529], [535, 541, 530], [536, 542, 531], [537, 543, 532], [538, 544, 533], [539, 545, 534], [540, 546, 535], [541, 547, 536], [542, 548, 537], [543, 549, 538], [544, 550, 539], [545, 551, 540], [546, 552, 541], [547, 553, 542], [548, 554, 543], [549, 555, 544], [550, 556, 545], [551, 557, 546], [552, 558, 547], [553, 559, 548], [554, 560, 549], [555, 561, 550], [556, 562, 551], [557, 563, 552], [558, 564, 553], [559, 565, 554], [560, 566, 555], [561, 567, 556], [562, 568, 557], [563, 569, 558], [564, 570, 559], [565, 571, 560], [566, 572, 561], [567, 573, 562], [568, 574, 563], [569, 575, 564], [570, 576, 565], [571, 577, 566], [572, 578, 567], [573, 579, 568], [574, 580, 569], [575, 581, 570], [576, 582, 571], [577, 583, 572], [578, 584, 573], [579, 585, 574], [580, 586, 575], [581, 587, 576], [582, 588, 577], [583, 589, 578], [584, 590, 579], [585, 591, 580], [586, 592, 581], [587, 593, 582], [588, 594, 583], [589, 595, 584], [590, 596, 585], [591, 597, 586], [592, 598, 587], [593, 599, 588], [594, 600, 589], [595, 601, 590], [596, 602, 591], [597, 603, 592], [598, 604, 593], [599, 605, 594], [600, 606, 595], [601, 607, 596], [602, 608, 597], [603, 609, 598], [604, 610, 599], [605, 611, 600], [606, 612, 601], [607, 613, 602], [608, 614, 603], [609, 615, 604], [610, 616, 605], [611, 617, 606], [612, 618, 607], [613, 619, 608], [614, 620, 609], [615, 621, 610], [616, 622, 611], [617, 623, 612], [618, 624, 613], [619, 625, 614], [620, 626, 615], [621, 627, 616], [622, 628, 617], [623, 629, 618], [624, 630, 619], [625, 631, 620], [626, 632, 621], [627, 633, 622], [628, 634, 623], [629, 635, 624], [630, 636, 625], [631, 637, 626], [632, 638, 627], [633, 639, 628], [634, 640, 629], [635, 641, 630], [636, 642, 631], [637, 643, 632], [638, 644, 633], [639, 645, 634], [640, 646, 635], [641, 647, 636], [642, 648, 637], [643, 649, 638], [644, 650, 639], [645, 651, 640], [646, 652, 641], [647, 653, 642], [648, 654, 643], [649, 655, 644], [650, 656, 645], [651, 657, 646], [652, 658, 647], [653, 659, 648], [654, 660, 649], [655, 661, 650], [656, 662, 651], [657, 663, 652], [658, 664, 653], [659, 665, 654], [660, 666, 655], [661, 667, 656], [662, 668, 657], [663, 669, 658], [664, 670, 659], [665, 671, 660], [666, 672, 661], [667, 673, 662], [668, 674, 663], [669, 675, 664], [670, 676, 665], [671, 677, 666], [672, 678, 667], [673, 679, 668], [674, 680, 669], [675, 681, 670], [676, 682, 671], [677, 683, 672], [678, 684, 673], [679, 685, 674], [680, 686, 675], [681, 687, 676], [682, 688, 677], [683, 689, 678], [684, 690, 679], [685, 691, 680], [686, 692, 681], [687, 693, 682], [688, 694, 683], [689, 695, 684], [690, 696, 685], [691, 697, 686], [692, 698, 687], [693, 699, 688], [694, 700, 689], [695, 701, 690], [696, 702, 691], [697, 703, 692], [698, 704, 693], [699, 705, 694], [700, 706, 695], [701, 707, 696], [702, 708, 697], [703, 709, 698], [704, 710, 699], [705, 711, 700], [706, 712, 701], [707, 713, 702], [708, 714, 703], [709, 715, 704], [710, 716, 705], [711, 717, 706], [712, 718, 707], [713, 719, 708], [714, 720, 709], [715, 721, 710], [716, 722, 711], [717, 723, 712], [718, 724, 713], [719, 725, 714], [720, 726, 715], [721, 727, 716], [722, 728, 717], [723, 729, 718], [724, 730, 719], [725, 731, 720], [726, 732, 721], [727, 733, 722], [728, 734, 723], [729, 735, 724], [730, 736, 725], [731, 737, 726], [732, 738, 727], [733, 739, 728], [734, 740, 729], [735, 741, 730], [736, 742, 731], [737, 743, 732], [738, 744, 733], [739, 745, 734], [740, 746, 735], [741, 747, 736], [742, 748, 737], [743, 749, 738], [744, 750, 739], [745, 751, 740], [746, 752, 741], [747, 753, 742], [748, 754, 743], [749, 755, 744], [750, 756, 745], [751, 757, 746], [752, 758, 747], [753, 759, 748], [754, 760, 749], [755, 761, 750], [756, 762, 751], [757, 763, 752], [758, 764, 753], [759, 765, 754], [760, 766, 755], [761, 767, 756], [762, 768, 757], [763, 769, 758], [764, 770, 759], [765, 771, 760], [766, 772, 761], [767, 773, 762], [768, 774, 763], [769, 775, 764], [770, 776, 765], [771, 777, 766], [772, 778, 767], [773, 779, 768], [774, 780, 769], [775, 781, 770], [776, 782, 771], [777, 783, 772], [778, 784, 773], [779, 785, 774], [780, 786, 775], [781, 787, 776], [782, 788, 777], [783, 789, 778], [784, 790, 779], [785, 791, 780], [786, 792, 781], [787, 793, 782], [788, 794, 783], [789, 795, 784], [790, 796, 785], [791, 797, 786], [792, 798, 787], [793, 799, 788], [794, 800, 789], [795, 801, 790], [796, 802, 791], [797, 803, 792], [798, 804, 793], [799, 805, 794], [800, 806, 795], [801, 807, 796], [802, 808, 797], [803, 809, 798], [804, 810, 799], [805, 811, 800], [806, 812, 801], [807, 813, 802], [808, 814, 803], [809, 815, 804], [810, 816, 805], [811, 817, 806], [812, 818, 807], [813, 819, 808], [814, 820, 809], [815, 821, 810], [816, 822, 811], [817, 823, 812], [818, 824, 813], [819, 825, 814], [820, 826, 815], [821, 827, 816], [822, 828, 817], [823, 829, 818], [824, 830, 819], [825, 831, 820], [826, 832, 821], [827, 833, 822], [828, 834, 823], [829, 835, 824], [830, 836, 825], [831, 837, 826], [832, 838, 827], [833, 839, 828], [834, 840, 829], [835, 841, 830], [836, 842, 831], [837, 843, 832], [838, 844, 833], [839, 845, 834], [840, 846, 835], [841, 847, 836], [842, 848, 837], [843, 849, 838], [844, 850, 839], [845, 851, 840], [846, 852, 841], [847, 853, 842], [848, 854, 843], [849, 855, 844], [850, 856, 845], [851, 857, 846], [852, 858, 847], [853, 859, 848], [854, 860, 849], [855, 861, 850], [856, 862, 851], [857, 863, 852], [858, 864, 853], [859, 865, 854], [860, 866, 855], [861, 867, 856], [862, 868, 857], [863, 869, 858], [864, 870, 859], [865, 871, 860], [866, 872, 861], [867, 873, 862], [868, 874, 863], [869, 875, 864], [870, 876, 865], [871, 877, 866], [872, 878, 867], [873, 879, 868], [874, 880, 869], [875, 881, 870], [876, 882, 871], [877, 883, 872], [878, 884, 873], [879, 885, 874], [880, 886, 875], [881, 887, 876], [882, 888, 877], [883, 889, 878], [884, 890, 879], [885, 891, 880], [886, 892, 881], [887, 893, 882], [888, 894, 883], [889, 895, 884],

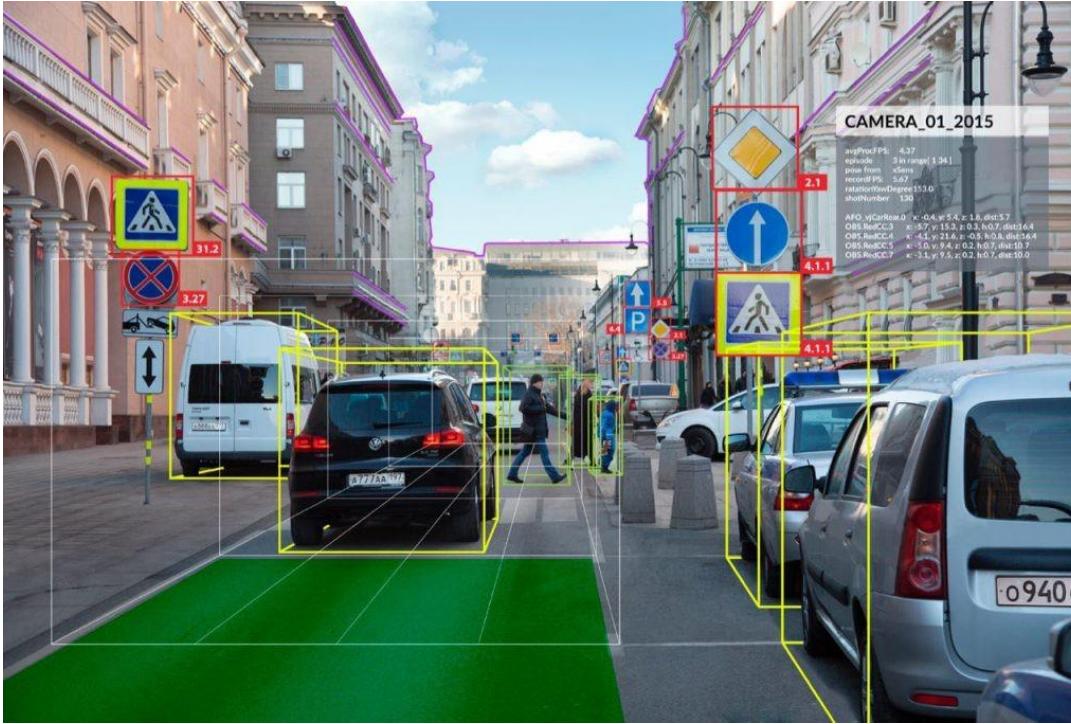
# VISÃO COMPUTACIONAL

Visão computacional é o campo da ciência que estuda como os computadores podem enxergar e entender o conteúdo de imagens e vídeos

Isto inclui adquirir, processar e analisar imagens ou vídeos

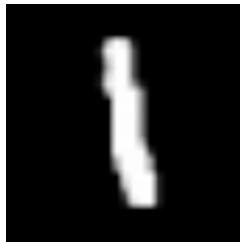
Problemas ligados à visão computacional muitas vezes são resolvidos facilmente por pessoas, mas não pelas máquinas

# VISÃO COMPUTACIONAL



# VISÃO COMPUTACIONAL

Uma imagem é representada como uma matriz



# IMAGENS RGB

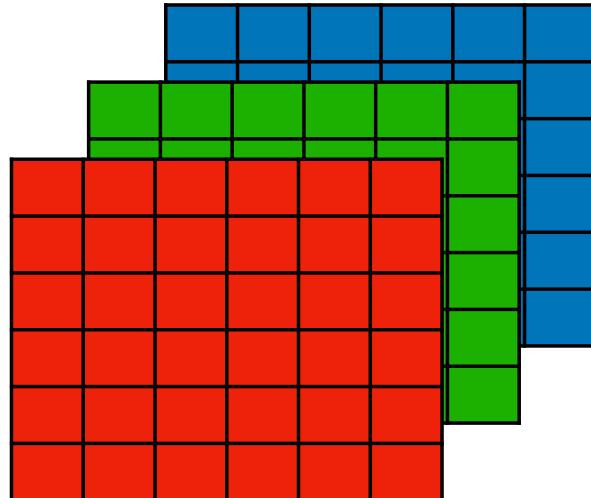
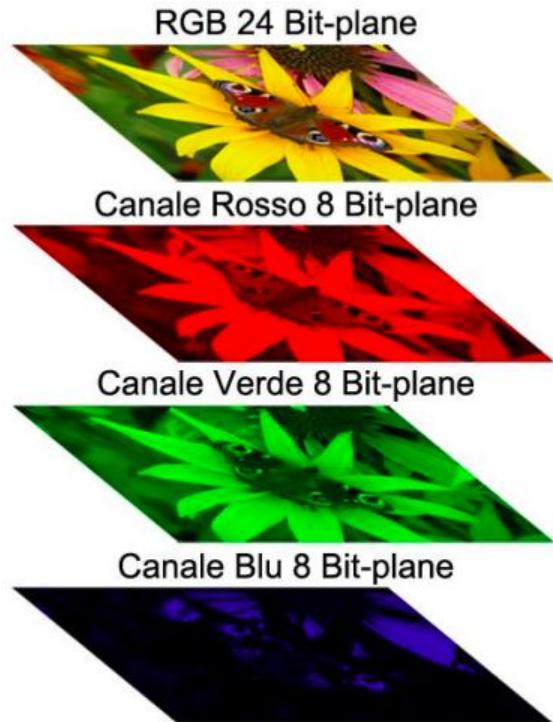


Imagen  $A \times L \times 3$

Imagens  $A \times L \times C$

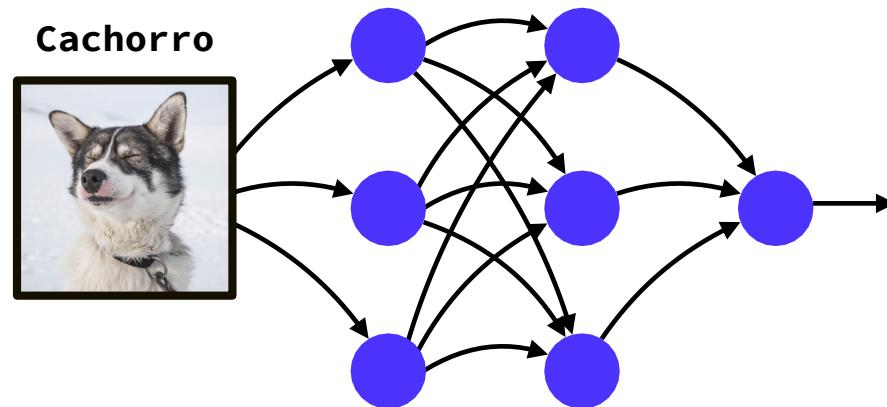
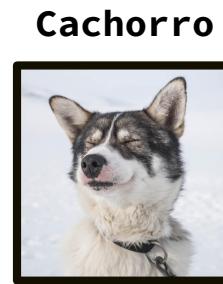
$C$ : no. de canais, ou profundidade

# CLASSIFICAÇÃO DE IMAGENS



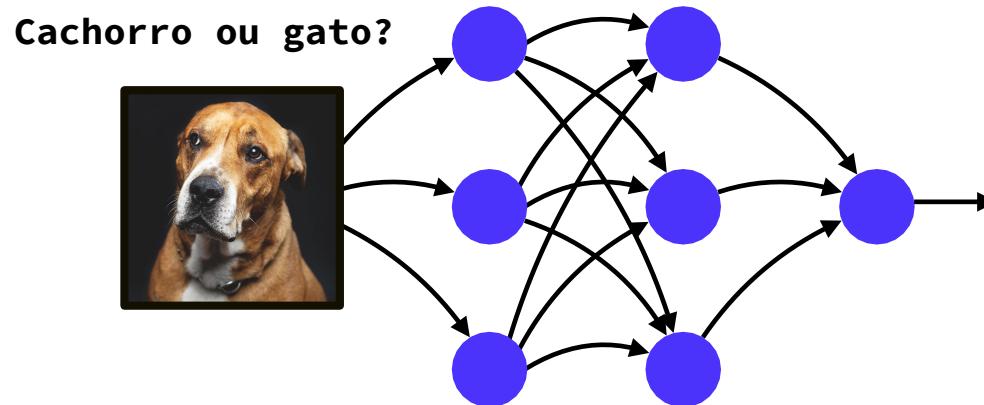
# CLASSIFICAÇÃO DE IMAGENS

Treinamos uma rede neural com várias imagens de cachorros e gatos



# CLASSIFICAÇÃO DE IMAGENS

Para que possamos classificar novas entradas



# CLASSIFICAÇÃO DE IMAGENS

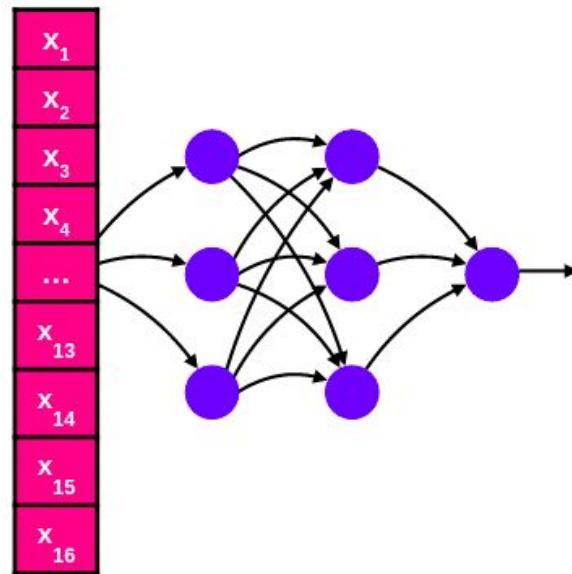
Como passar uma imagem para uma rede neural?

A rede recebe entradas numéricas, ela não recebe uma matriz

# CLASSIFICAÇÃO DE IMAGENS

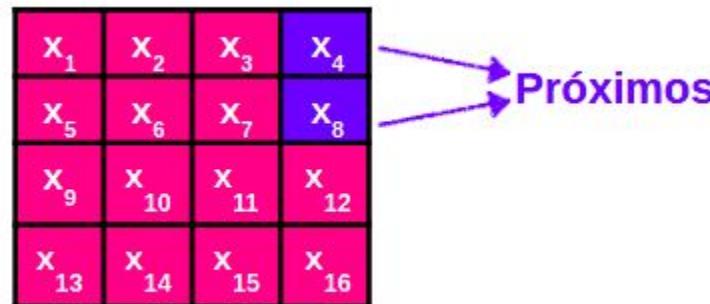
Podemos passar cada valor da matriz como uma entrada

$x_1$	$x_2$	$x_3$	$x_4$
$x_5$	$x_6$	$x_7$	$x_8$
$x_9$	$x_{10}$	$x_{11}$	$x_{12}$
$x_{13}$	$x_{14}$	$x_{15}$	$x_{16}$



# CLASSIFICAÇÃO DE IMAGENS

Uma limitação dessa abordagem é que perdemos a noção espacial da imagem



# CLASSIFICAÇÃO DE IMAGENS

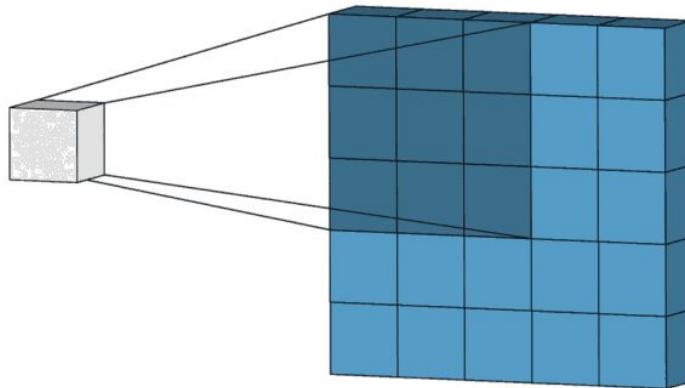
Outra limitação é que o número de entradas e pesos será muito grande

Uma imagem colorida 1000x1000 tem 3 milhões de valores  
(A x L x 3)

Se a primeira camada intermediária possuir 10 neurônios,  
teremos 30 milhões de pesos só na primeira camada!

# CLASSIFICAÇÃO DE IMAGENS - VISÃO COMPUTACIONAL

- Uma operação linear para reconhecer padrões
- Operação do filtro/kernel sobre a imagem/sinal
- Disposição espacial dos dados se transforma em informação



# CLASSIFICAÇÃO DE IMAGENS

Podemos processar previamente as imagens e depois passar para rede neural:

- Detectar arestas
- Detectar cantos
- Detectar regiões

# DETECÇÃO DE ARESTAS



# DETECÇÃO DE ARESTAS

Imagen

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

Convolução

\*

1	0	-1
1	0	-1
1	0	-1

Filtro

=


# OPERAÇÃO COM FILTROS

O mesmo filtro é “aplicado” sobre o campo receptivo (CR) que desliza sobre a imagem

A cada iteração, são somados os produtos dos pares ( $\text{sum}(\text{CR} \times \text{filtro})$ ) gerando um novo valor para a matriz resultante

# DETEÇÃO DE ARESTAS

Campo receptivo

Imagem

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

Convolução

\*

1	0	-1
1	0	-1
1	0	-1

Filtro

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

3x1

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$3 \times 1 + 1 \times 1$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3 \times 1 + 1 \times 1 + 2 \times 1$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=


# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1 = -5$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=

-5			

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$0x1 + 5x1 + 7x1 + 1x0 + 8x0 + 2x0 + 2x-1 + 9x-1 + 5x-1 = -4$$

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

\*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4		

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

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

\*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

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

\*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

\*

1	0	-1
1	0	-1
1	0	-1

=

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

\*

1	0	-1
1	0	-1
1	0	-1

=

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$10x1 + 10x1 + 10x1 + 10x0 + 10x0 + 10x0 + 10x-1 + 10x-1 + 10x-1 = 0$$

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

\*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

$$10 \times 1 + 10 \times 1 + 10 \times 1 + 10 \times 0 = 30$$

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

\*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

# DETEÇÃO DE ARESTAS - CONVOLUÇÃO 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

\*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

# DETEÇÃO DE ARESTAS

Filtros diferentes detectam características diferentes:

Arestas verticais

1	0	-1
1	0	-1
1	0	-1

Arestas horizontais

1	1	1
0	0	0
-1	-1	-1

# REDES NEURAIS CONVOLUCIONAIS

$$D_{(0,0)} = (S_{(0,0)} * W_{(0,0)}) + (S_{(1,0)} * W_{(1,0)}) + (S_{(2,0)} * W_{(2,0)}) +$$

$$(S_{(0,1)} * W_{(0,1)}) + (S_{(1,1)} * W_{(1,1)}) + (S_{(2,1)} * W_{(2,1)}) +$$

$$(S_{(0,2)} * W_{(0,2)}) + (S_{(1,2)} * W_{(1,2)}) + (S_{(2,2)} * W_{(2,2)})$$

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} * \begin{matrix} 6 & & \\ & & \\ & & \end{matrix} = \begin{matrix} & & \\ & & \\ & & \end{matrix}$$

$$\begin{aligned} 7 \times 1 + 4 \times 1 + 3 \times 1 + \\ 2 \times 0 + 5 \times 0 + 3 \times 0 + \\ 3 \times -1 + 3 \times -1 + 2 \times -1 \\ = 6 \end{aligned}$$

Tamanho Final = (Img - W) + 1  $\rightarrow$  (5 - 3) + 1 = 3

↑  
**Filtro**

# PADDING

Padding = 1; Imagem (32 x 32) -> (33 x 33)

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & & & 0 \\ \hline 0 & & 32 \times 32 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 1 & 1 & -1 \\ \hline 0 & 0 & 2 \\ \hline -2 & -2 & -1 \\ \hline \end{array} \quad = \quad (32 \times 32)$$

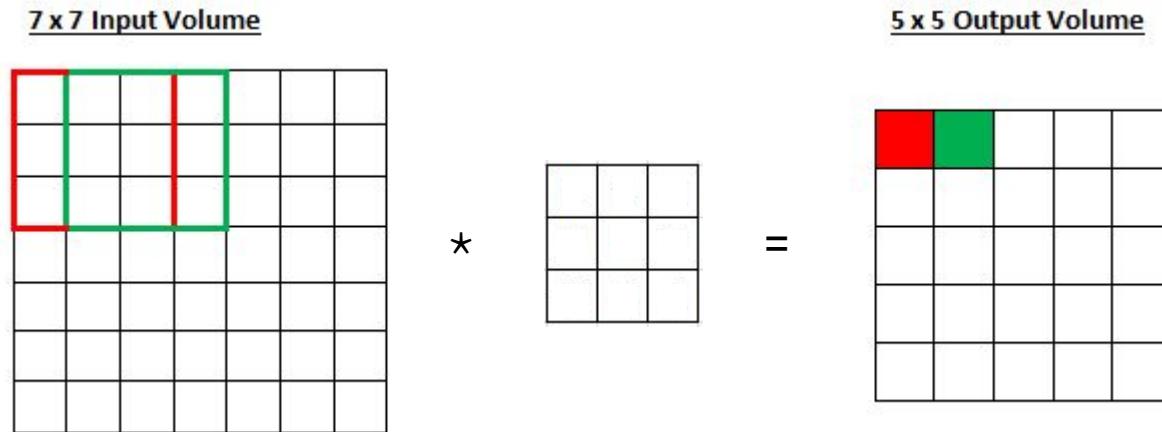
33x33    3x3

Tamanho Final = (Img - W + 2\*P) + 1

Tamanho Final = (32 - 3 + 2\*1) + 1 = 32

# STRIDE

Stride = 1; Filtro 3x3

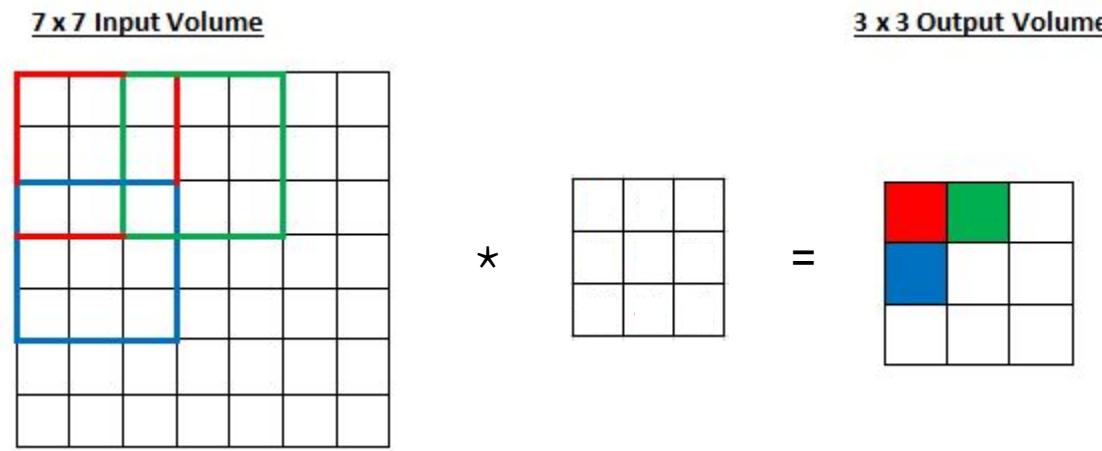


$$\text{Tamanho Final} = [(Img - W + 2*P) / S] + 1$$

$$\text{Tamanho Final} = [(7 - 3 + 2*0) / 1] + 1 = 5$$

# STRIDE

Stride = 2; Filtro 3x3



$$\text{Tamanho Final} = [( \text{Img} - W + 2*P ) / S] + 1$$

$$\text{Tamanho Final} = [( 7 - 3 + 2*0 ) / 2] + 1 = 3$$

# REDES NEURAIS CONVOLUCIONAIS

Podemos tentar criar filtros manualmente escolhendo valores

Quais filtros criar?

Como saber se os valores escolhidos são bons?

Não sabemos previamente!

# REDES NEURAIS CONVOLUCIONAIS

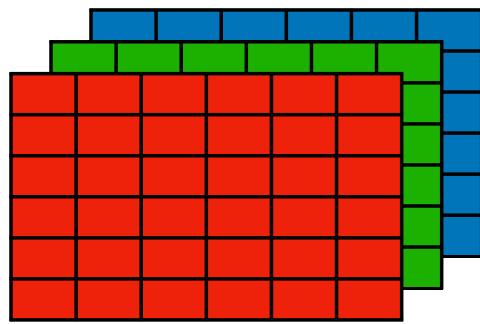
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

\*

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

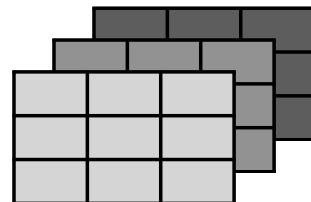
Vamos tratar os valores dos filtros como pesos que serão aprendidos pela rede neural através do backpropagation

# CONVOLUÇÃO 3D



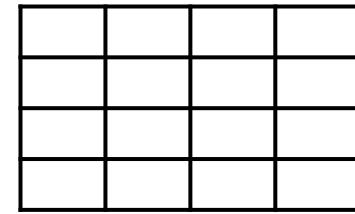
$6 \times 6 \times 3$

\*



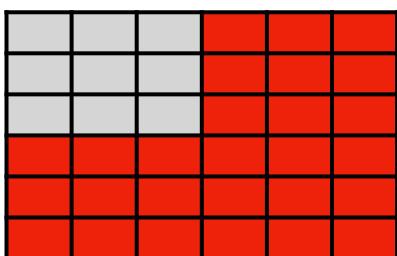
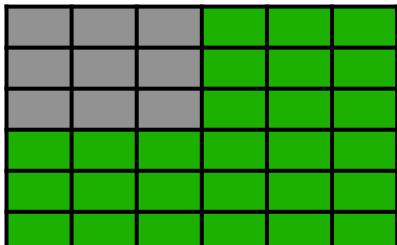
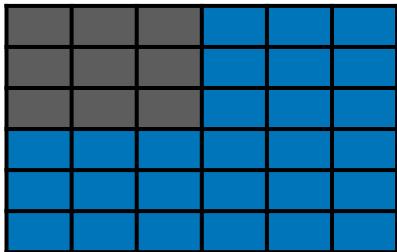
$3 \times 3 \times 3$

=

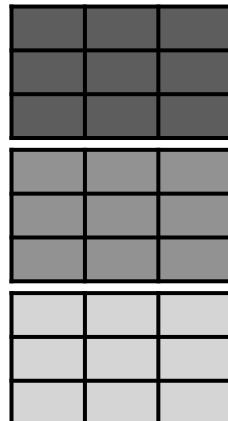


$4 \times 4$

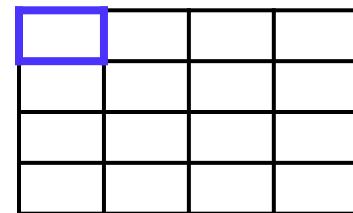
# CONVOLUÇÃO 3D



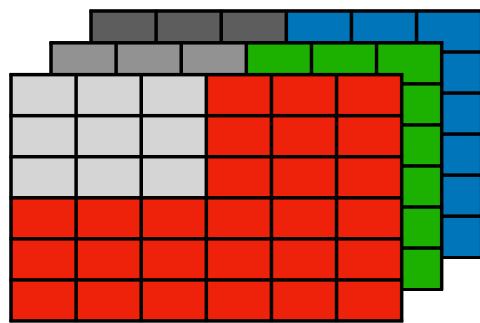
\*



=

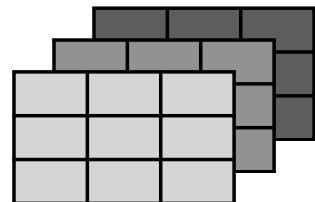


# CONVOLUÇÃO 3D



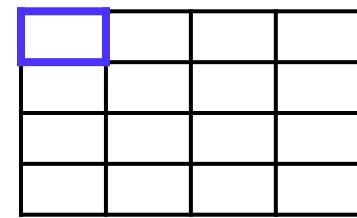
$6 \times 6 \times 3$

\*



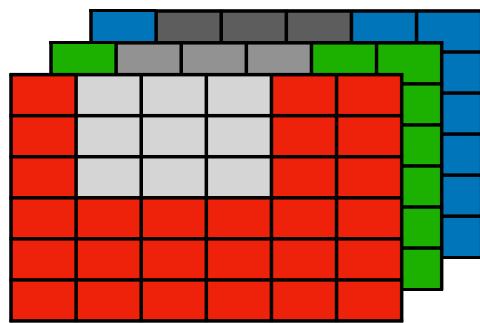
$3 \times 3 \times 3$

=



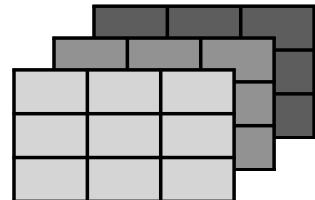
$4 \times 4$

# CONVOLUÇÃO 3D



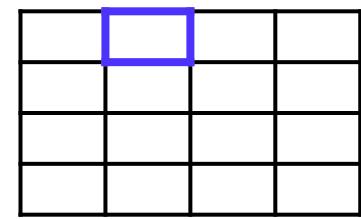
$6 \times 6 \times 3$

\*



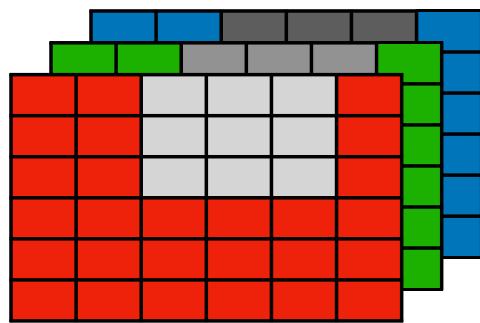
$3 \times 3 \times 3$

=



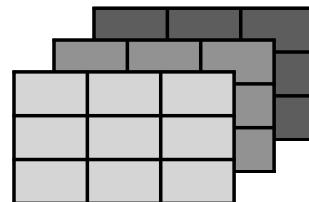
$4 \times 4$

# CONVOLUÇÃO 3D



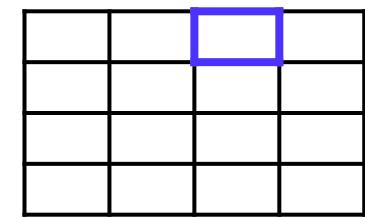
$6 \times 6 \times 3$

\*



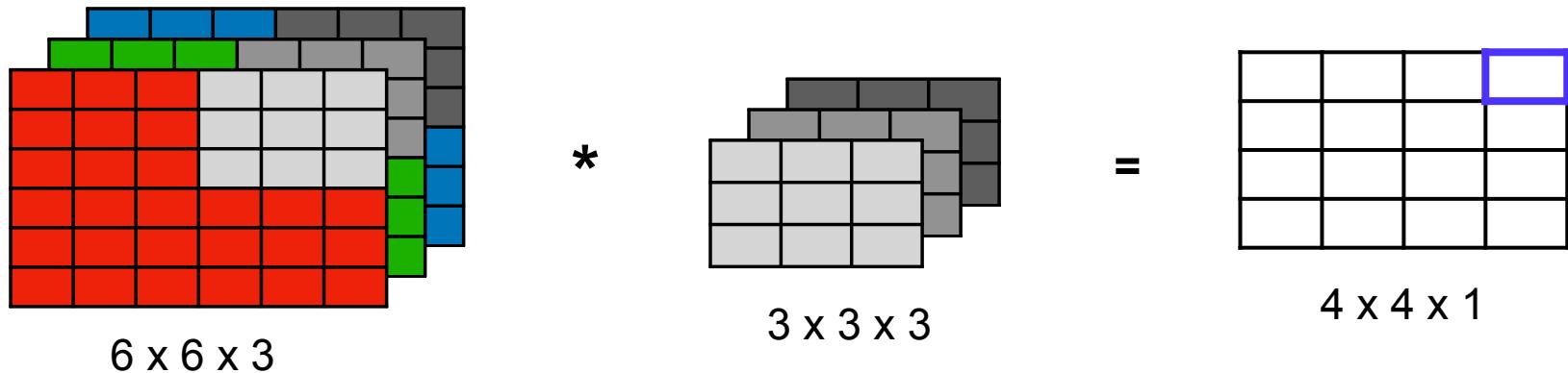
$3 \times 3 \times 3$

=



$4 \times 4$

# CONVOLUÇÃO 3D



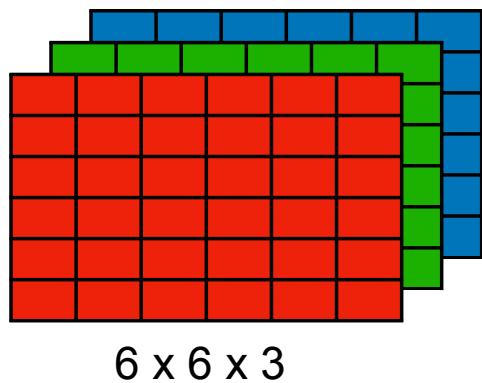
Tamanho Final 3D: Imagem<sub>(x, y, c)</sub> o N\*Filtros<sub>(x', y', c)</sub> = Saída<sub>(x'', y'', N)</sub>

Importante: x'' e y'' são calculados como se fosse 2D: (P=0; S=1)

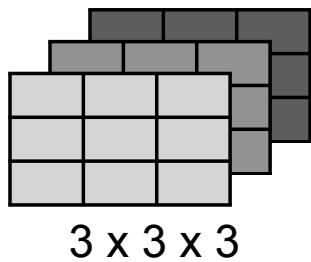
$$TF = [(Img - W + 2P) / S] + 1 \implies [(6 - 3 + 0) / 1] + 1 = 4$$

Tamanho Final 3D: Img<sub>(6, 6, 3)</sub> o 1\*W<sub>(3, 3, 3)</sub> = Saída<sub>(4, 4, 1)</sub>

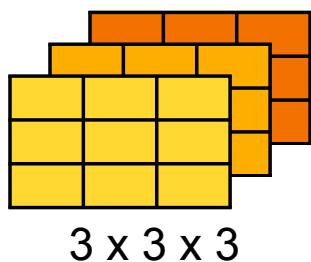
# CONVOLUÇÃO 3D



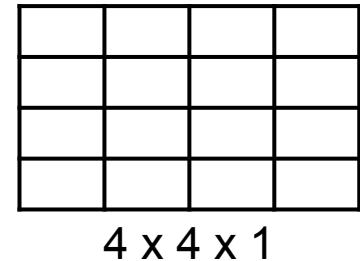
\*



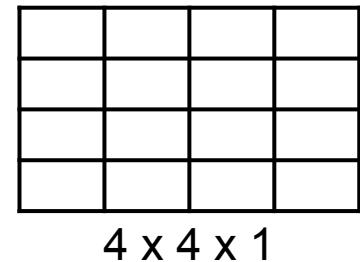
\*



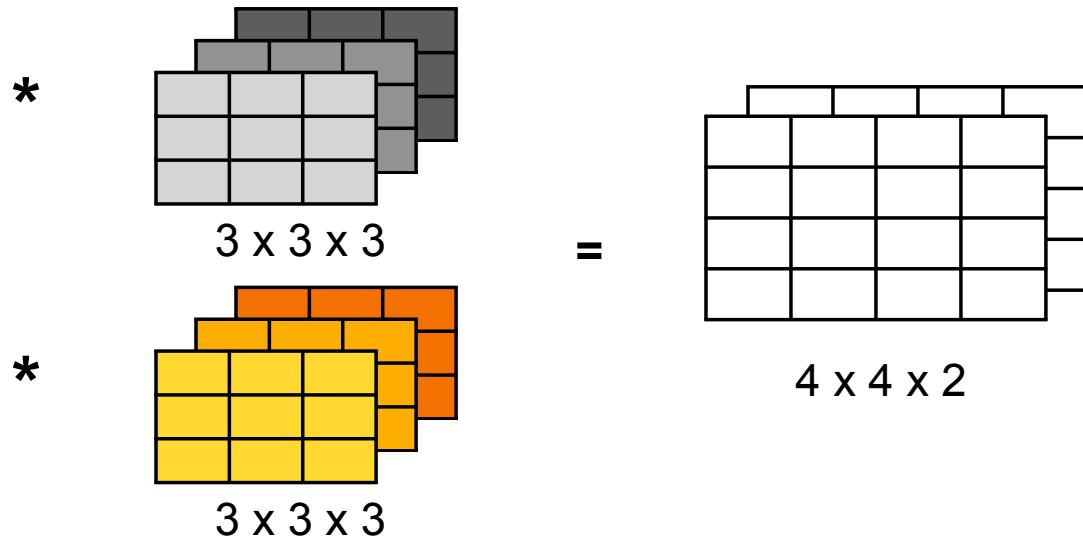
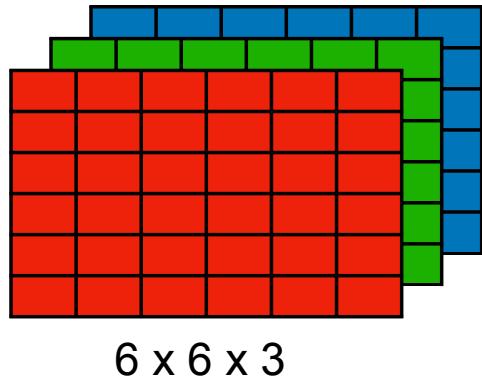
=



=



# CONVOLUÇÃO 3D



Tamanho Final 3D: Imagem<sub>(x,y,c)</sub> o N\*Filtros<sub>(x',y',c)</sub> = Saída<sub>(x'',y'',N)</sub>

$x''$  e  $y''$  são definidos pelo mesmo cálculo anterior cujo resultado foi 4

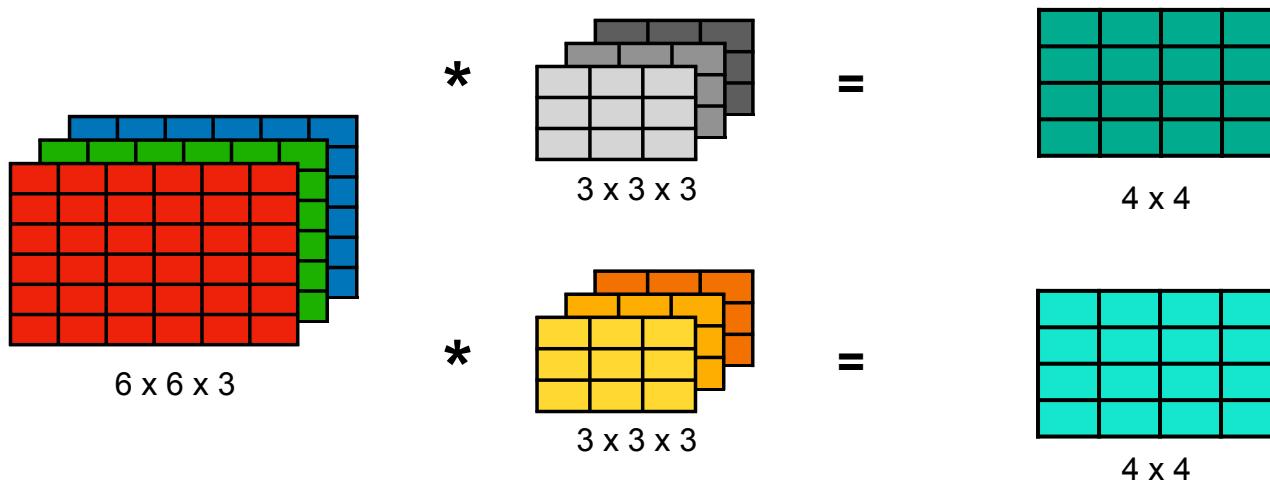
Tamanho Final 3D: Img<sub>(6, 6, 3)</sub> o 2\*W<sub>(3, 3, 3)</sub> = Saída<sub>(4, 4, 2)</sub>

# REDES NEURAIS CONVOLUCIONAIS

Temos todas as peças para entender e construir uma rede neural convolucional

Iniciaremos com uma camada

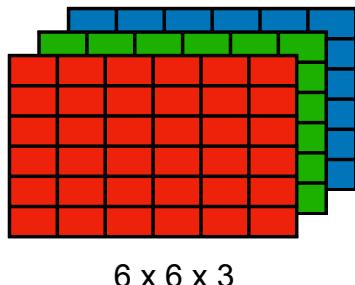
# REDES NEURAIS CONVOLUCIONAIS



# REDES NEURAIS CONVOLUCIONAIS

$$\begin{array}{c} \text{Input: } 6 \times 6 \times 3 \\ \text{Convolution: } * \quad 3 \times 3 \times 3 \\ \text{Output: } F_a \left( \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \right) \\ \quad \quad \quad 4 \times 4 \end{array}$$
  
$$\begin{array}{c} \text{Input: } 6 \times 6 \times 3 \\ \text{Convolution: } * \quad 3 \times 3 \times 3 \\ \text{Output: } F_a \left( \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \right) \\ \quad \quad \quad 4 \times 4 \end{array}$$

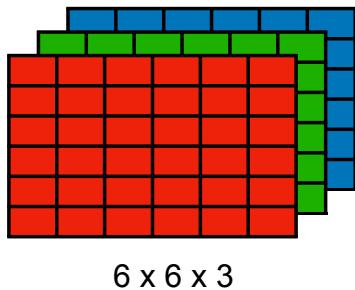
# REDES NEURAIS CONVOLUCIONAIS



$$\ast \quad \begin{array}{c} \text{3D tensor diagram} \\ \text{3} \times 3 \times 3 \end{array} = F_a \left( \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \right) = \begin{array}{c} \text{3D tensor diagram} \\ \text{4} \times 4 \end{array}$$

$$\ast \quad \begin{array}{c} \text{3D tensor diagram} \\ \text{3} \times 3 \times 3 \end{array} = F_a \left( \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \right) = \begin{array}{c} \text{3D tensor diagram} \\ \text{4} \times 4 \end{array}$$

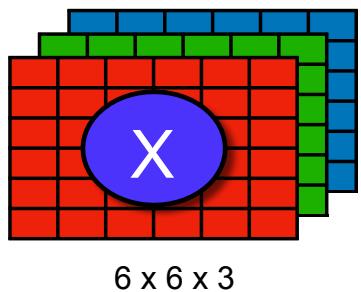
# REDES NEURAIS CONVOLUCIONAIS



$$\begin{array}{c} * \\ \text{---} \\ \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \\ \text{---} \\ 3 \times 3 \times 3 \end{matrix} = F_a \left( \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \right)_{4 \times 4} = \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \\ \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \\ \text{---} \\ 3 \times 3 \times 3 \end{matrix} = F_a \left( \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \right)_{4 \times 4} = \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \\ \text{---} \\ 4 \times 4 \times 2 \end{array}$$

The diagram illustrates the convolutional operation. It shows two input tensors being multiplied by weight matrices  $F_a$ . The first input is a  $6 \times 6 \times 3$  tensor (red, green, blue) and the second is a  $3 \times 3 \times 3$  tensor (yellow, orange). Both are multiplied by weight matrices  $F_a$  of size  $4 \times 4$ , resulting in two output tensors of size  $4 \times 4$  (cyan).

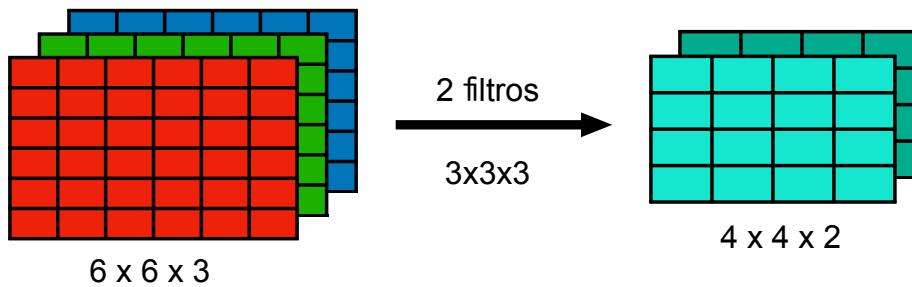
# REDES NEURAIS CONVOLUCIONAIS



$$\begin{array}{c} * \\ \text{---} \\ \begin{matrix} \text{---} & \text{---} \\ \text{---} & \text{---} \end{matrix} \end{array} \quad \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \quad = F_a \left( \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \right) \quad = \quad \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array}$$

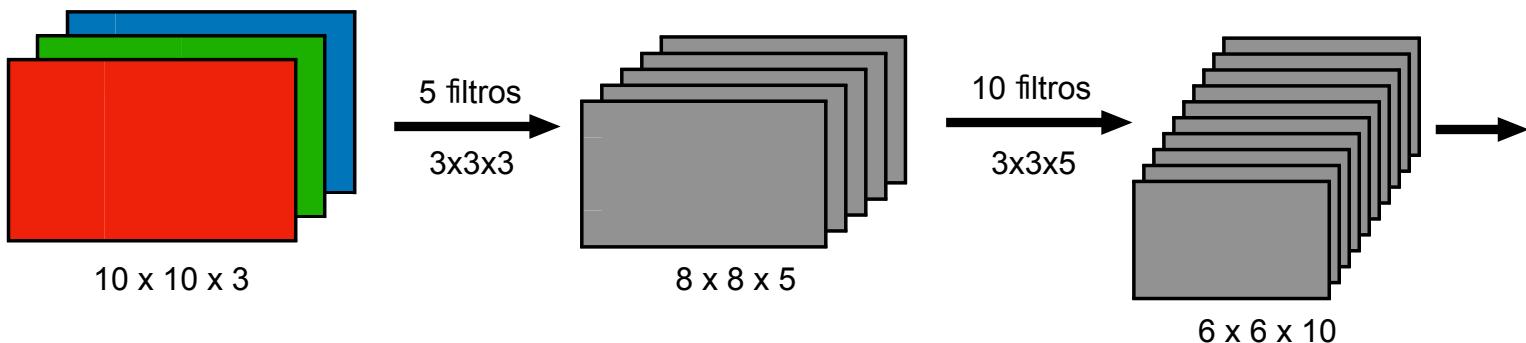
The diagram illustrates two convolutional operations. The first operation shows input  $X$  (dimensions  $6 \times 6 \times 3$ ) being multiplied by weight  $W$  (dimensions  $3 \times 3 \times 3$ ) to produce output  $Z$  (dimensions  $4 \times 4$ ). The second operation shows input  $Z$  being multiplied by weight  $W$  (dimensions  $3 \times 3 \times 3$ ) to produce output  $A$  (dimensions  $4 \times 4 \times 2$ ). The weights  $W$  are shown as 3D tensors, and the outputs  $Z$  and  $A$  are shown as 2D grids. The activation function  $F_a$  is applied to each output channel.

# REDES NEURAIS CONVOLUCIONAIS



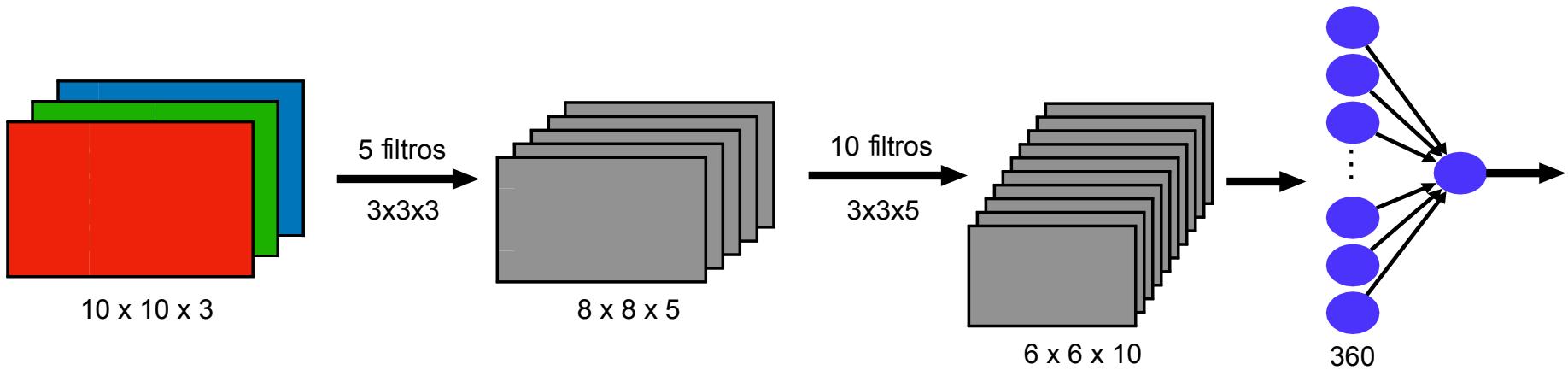
# REDES NEURAIS CONVOLUCIONAIS

Uma rede convolucional com duas camadas:



# REDES NEURAIS CONVOLUCIONAIS

No final, usamos uma camada totalmente conectada:



# POOLING

A operação de pooling é usada para aumentar a velocidade e diminuir a quantidade de memória usada.

Ela também realça características encontradas pelos filtros.

# MAX POOLING

Stride = 2

Filtro 2x2

Tamanho Final = Img / S

3	2	3	3
1	8	4	3
5	1	1	2
6	2	3	1

4 x 4



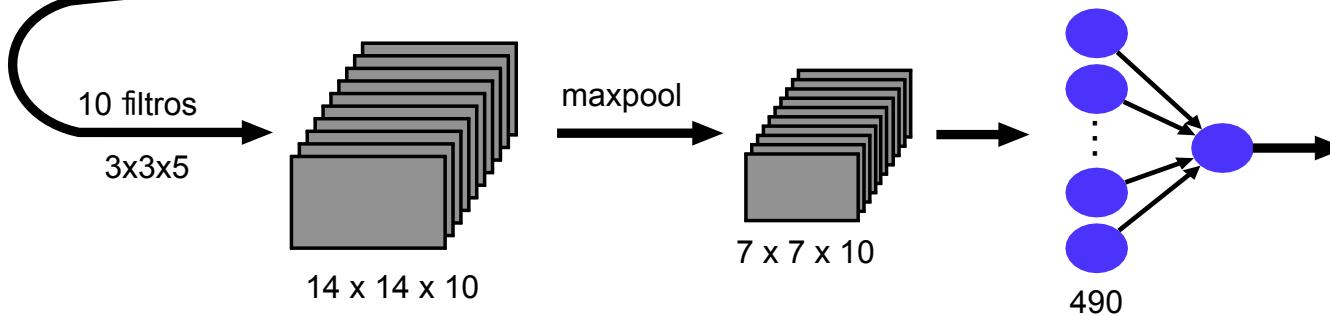
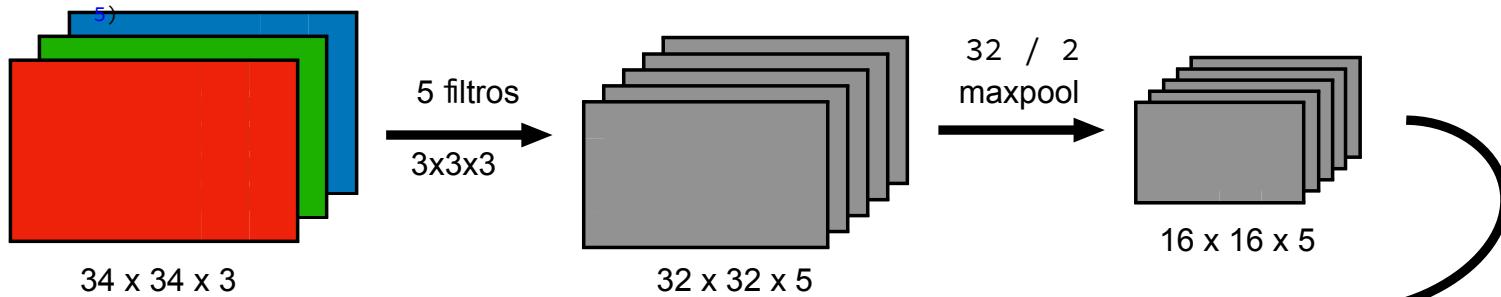
8	4
6	3

2 x 2

# REDES NEURAIS CONVOLUCIONAIS

$$\text{TF 2D: } [(34 - 3 + 0) / 1] + 1 = 32$$

$$\text{TF 3D: } \text{Img}_{(34, 34, 3)} \circ 5 * W_{(3, 3, 3)} = \text{Saída}_{(32, 32, 5)}$$

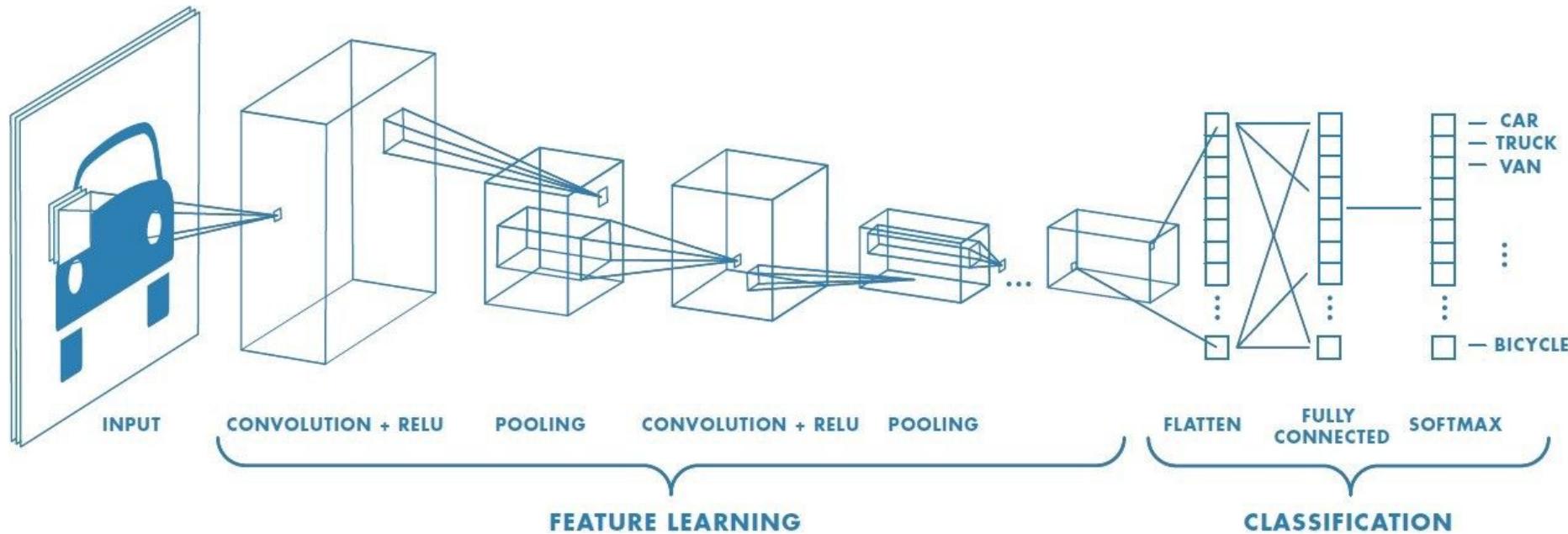


# REDES NEURAIS CONVOLUCIONAIS

Nas Redes Neurais Convolucionais é muito comum usarmos o padrão:

Convolução ➔ pooling ➔ convolução ➔ pooling ➔ ...  
➔ convolução ➔ pooling ➔ camada totalmente conectada

# REDES NEURAIS CONVOLUCIONAIS



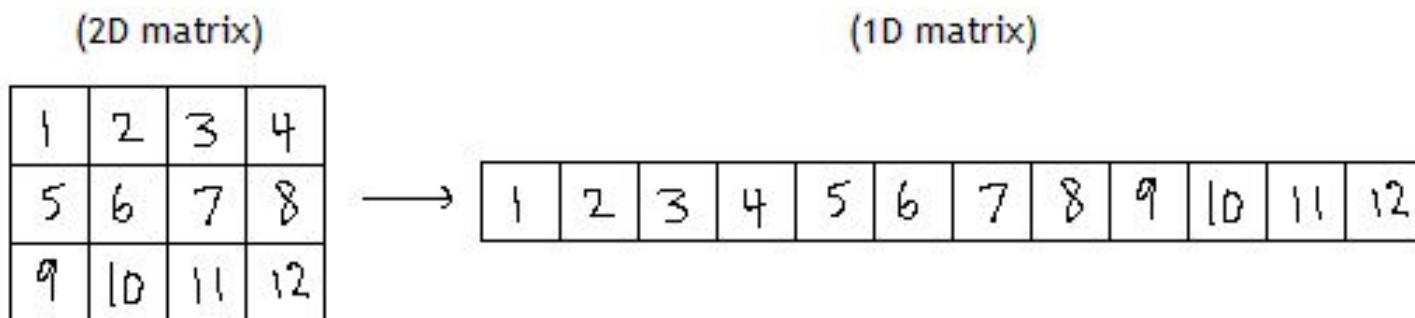
# REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais, ao longo de suas camadas, vão aprendendo a detectar padrões cada vez mais complexos.

É possível visualizar esses padrões através da criação de imagens artificialmente para maximizar uma das saídas de uma determinada camada.

# REDES NEURAIS CONVOLUCIONAIS EM DADOS CATEGÓRICOS

# REDES NEURAIS CONVOLUCIONAIS 1D



# REDES NEURAIS CONVOLUCIONAIS 1D

$$4 \times 3 + 1 \times 7 + 3 \times 5 = 34$$

# REDES NEURAIS CONVOLUCIONAIS 1D

$$1 \times 3 + 3 \times 7 + 8 \times 5 = 64$$

# REDES NEURAIS CONVOLUCIONAIS 1D



Tamanho Final = [(Matriz - W + 2\*P) / S] + 1 (Mesma fórmula da 2D)

$$\text{Tamanho Final} = [(6 - 3 + 2 \cdot 0) / 1] + 1 = 4$$

## Cálculos:

$$1^{\circ} \text{ Elemento: } 1X2 + 3X0 + 3X1 = 5$$

2º Elemento:  $3x_2 + 3x_0 + 0x_1 = 6$

3º Elemento: 3x2 + 0x0 + 1x1 = 7

$$4^{\circ} \text{ Elemento: } 0 \times 2 + 1 \times 0 + 2 \times 1 = 2$$

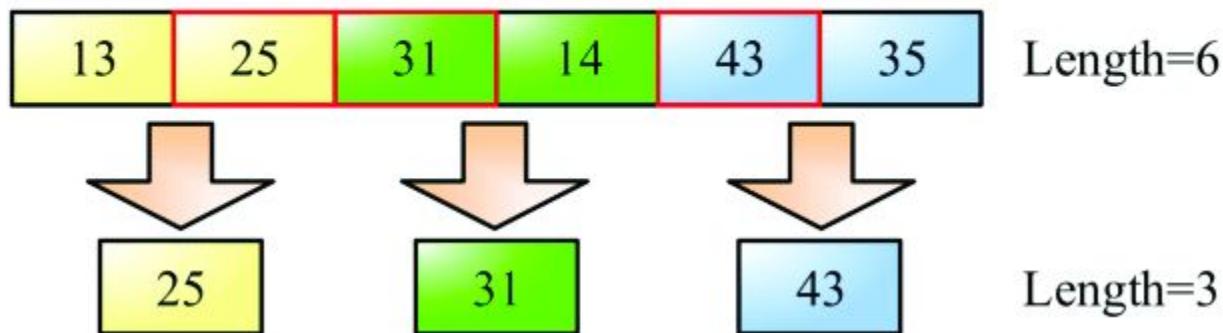
# MAX POOLING 1D

Stride = 2

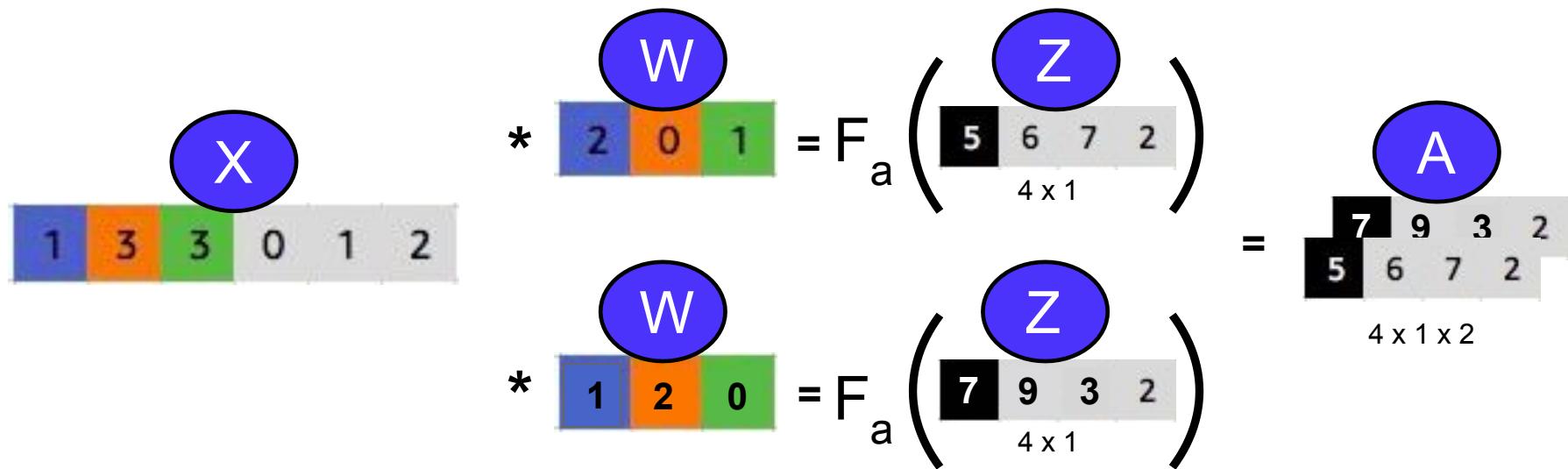
Filtro 2x1

Tamanho Final = Matriz / S

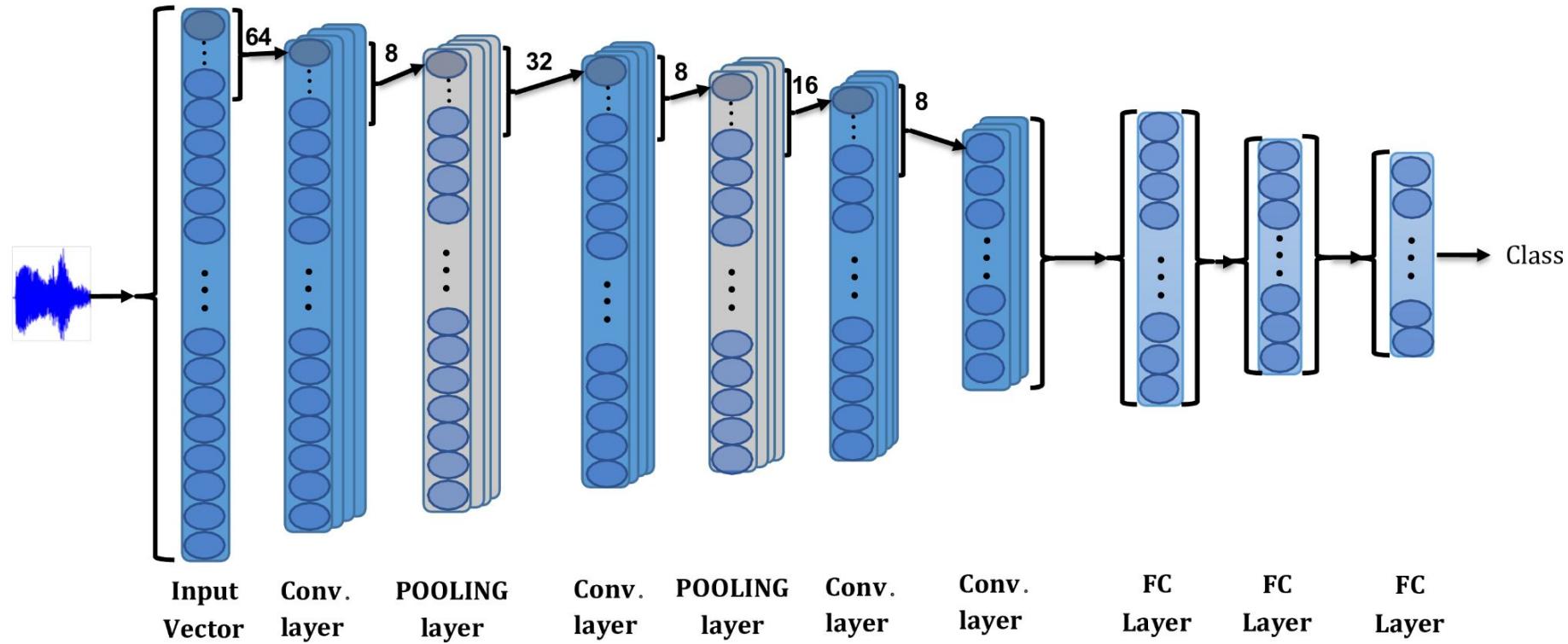
Tamanho Final = 6 / 2 = 3



# REDES NEURAIS CONVOLUCIONAIS 1D



# REDES NEURAIS CONVOLUCIONAIS 1D



# APLICAÇÕES NA BIOINFORMÁTICA E BIOMEDICINA

nature > articles > article

Article | Published: 15 January 2020

## Improved protein structure prediction using potentials from deep learning

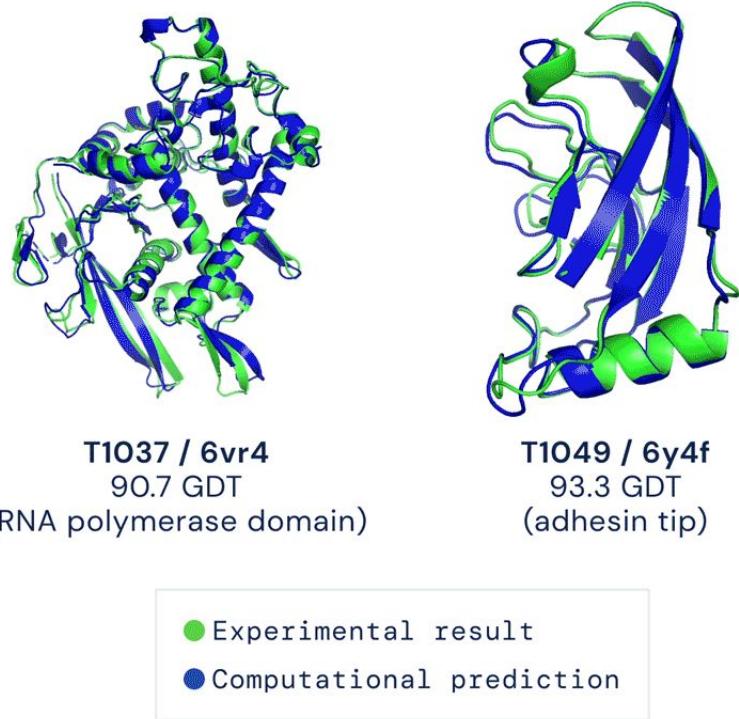
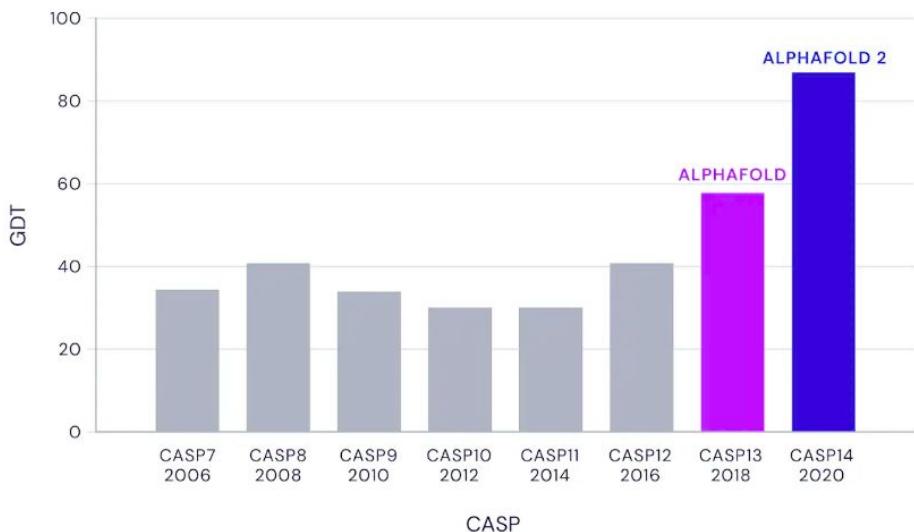
Andrew W. Senior , Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu & Demis Hassabis

*Nature* **577**, 706–710(2020) | Cite this article

**93k** Accesses | **259** Citations | **611** Altmetric | Metrics

# ALPHAFOLD

Median Free-Modelling Accuracy



Published online 13 January 2020

*NAR Genomics and Bioinformatics*, 2020, Vol. 2, No. 1 1

doi: 10.1093/nargab/lqz024

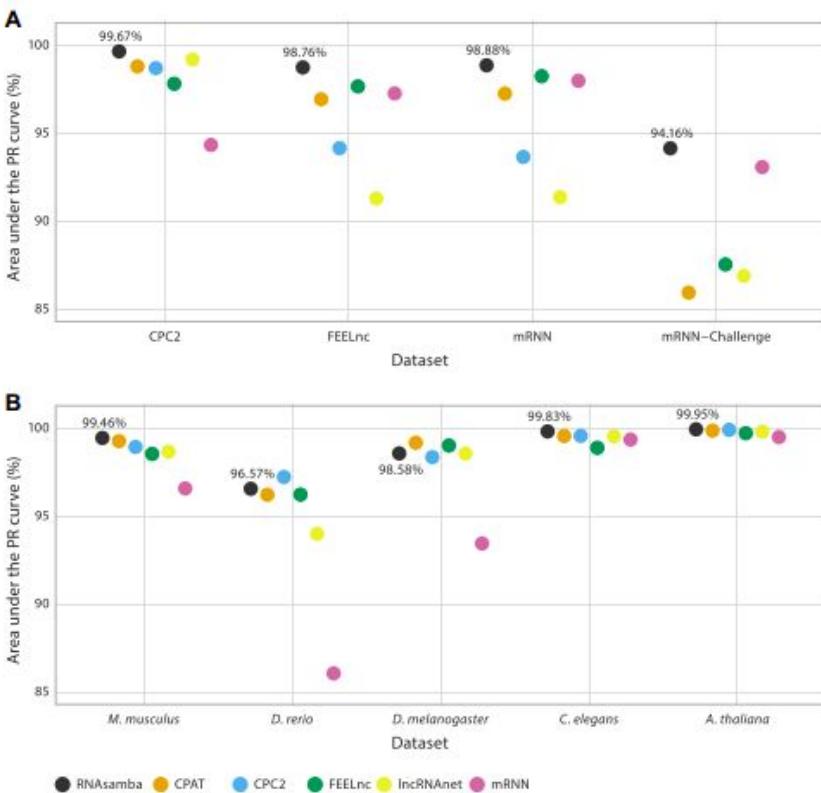
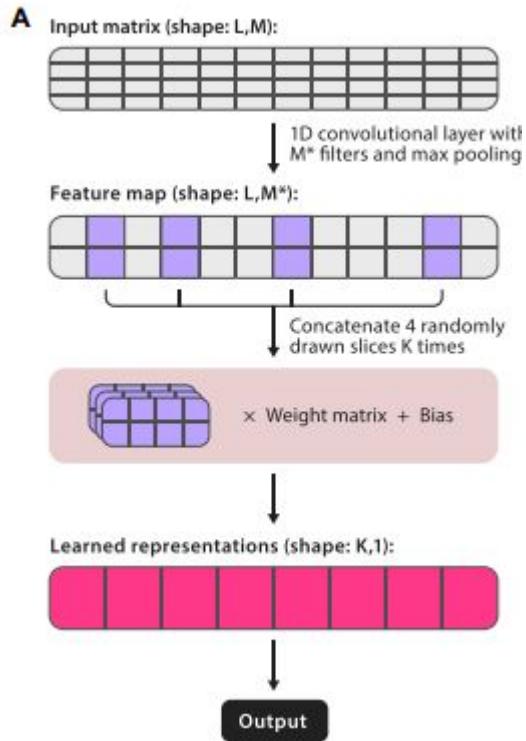
## RNAsamba: neural network-based assessment of the protein-coding potential of RNA sequences

Antonio P. Camargo<sup>1</sup>, Vsevolod Sourkov<sup>2</sup>, Gonçalo A. G. Pereira<sup>1</sup> and  
Marcelo F. Carazzolle<sup>1,\*</sup>

<sup>1</sup>Department of Genetics, Evolution, Microbiology and Immunology, Institute of Biology, University of Campinas, Campinas, SP, 13083-862, Brazil and <sup>2</sup>Department of Computer Science, ReDNA Labs, Pattaya, Chonburi, 20150, Thailand

Received August 09, 2019; Revised November 15, 2019; Editorial Decision December 10, 2019; Accepted December 17, 2019

# RNASAMBA



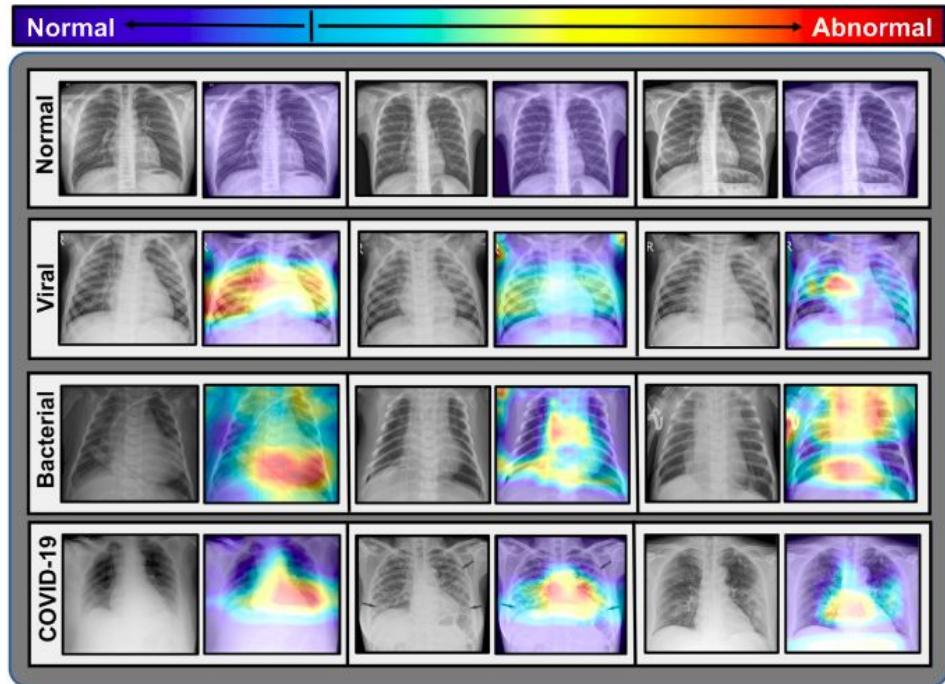
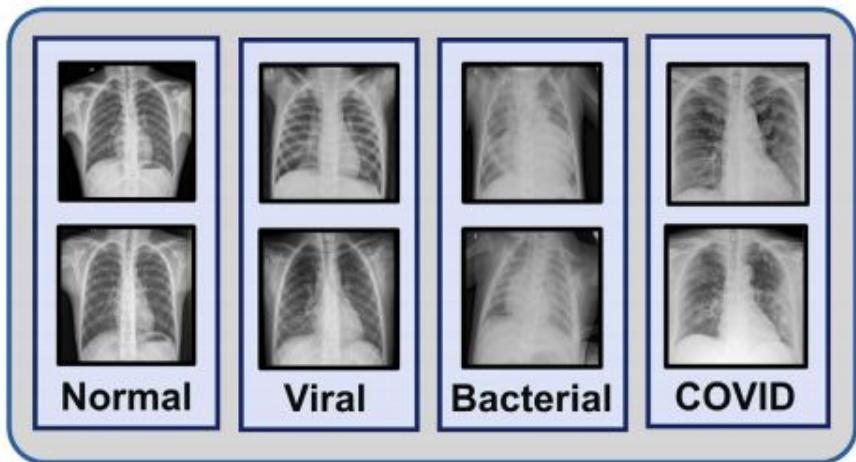


# CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization<sup>☆</sup>

Tanvir Mahmud, Md Awsafur Rahman, Shaikh Anowarul Fattah \*

*Department of EEE, BUET, ECE Building, West Palashi, Dhaka 1205, Bangladesh*

# COVXNET



# TIPOS DE CÂNCER ATRAVÉS DA EXPRESSÃO GÊNICA

Research | [Open Access](#) | Published: 03 April 2020

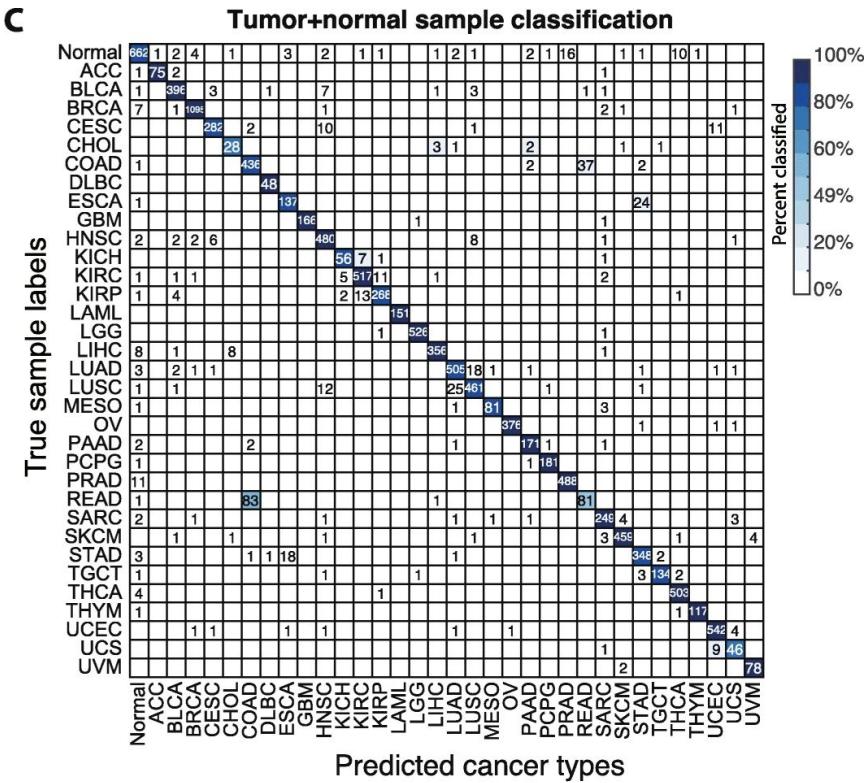
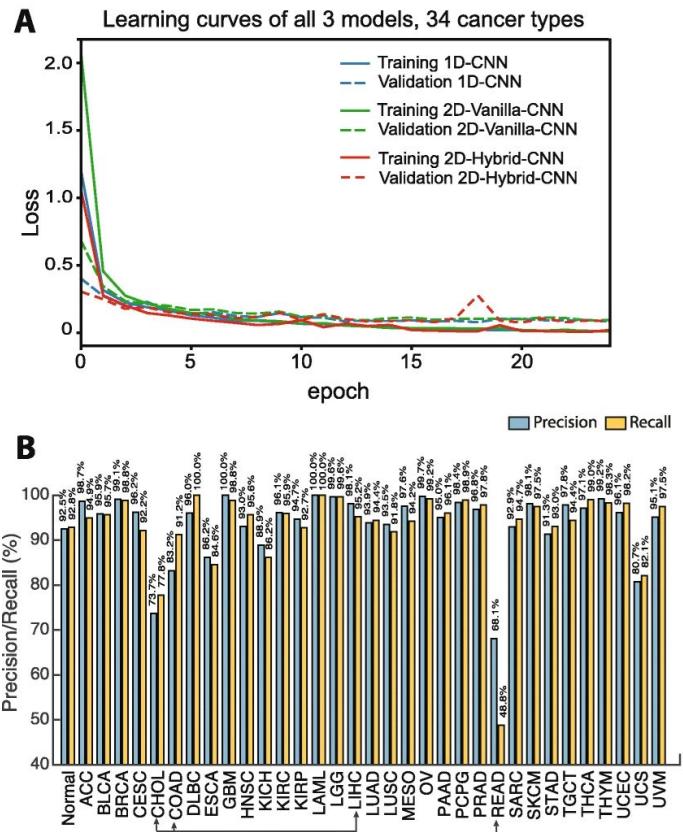
## Convolutional neural network models for cancer type prediction based on gene expression

[Milad Mostavi](#), [Yu-Chiao Chiu](#), [Yufei Huang](#)  & [Yidong Chen](#) 

[BMC Medical Genomics](#) **13**, Article number: 44 (2020) | [Cite this article](#)

**3727** Accesses | **10** Citations | **8** Altmetric | [Metrics](#)

# TIPOS DE CÂNCER ATRAVÉS DA EXPRESSÃO GÊNICA



# CÂNCER DE MAMA

S.I. : Healthcare Analytics | Published: 09 October 2020

## **Convolutional neural network-based models for diagnosis of breast cancer**

[Mehedi Masud](#), [Amr E. Eldin Rashed](#) & [M. Shamim Hossain](#) 

[Neural Computing and Applications](#) (2020) | [Cite this article](#)

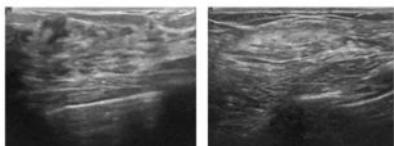
1525 Accesses | 1 Citations | 1 Altmetric | [Metrics](#)

# CÂNCER DE MAMA

Normal

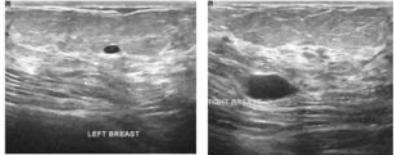
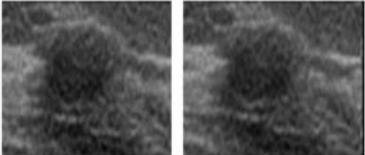
Dataset1

Not available      Not available

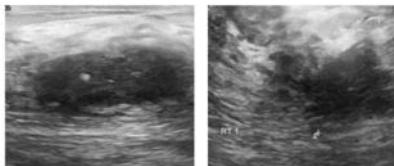
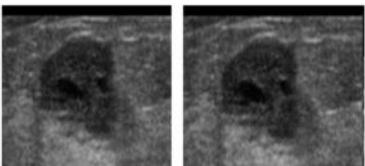


Dataset2

Benign



Malignant



Compare Pretrained Networks

