

L'apprentissage profond avec Python de François Chollet

<https://github.com/fchollet/deep-learning-with-python-notebooks/tree/master>

Loïc ALAVOINE

3 janvier 2026

Table des matières

1 Les éléments mathématiques constitutifs des réseaux neuronaux	2
1.1 Code utilisé dans le terminal Windows du dossier dans lequel on travaille. On suppose que Python et Pandoc sont installés.	2
1.2 Un premier aperçu d'un réseau neuronal	2
1.2.1 Chargement du jeu de données MNIST dans Keras	2
1.2.2 L'architecture du réseau	3
1.2.3 L'étape de compilation	3
1.2.4 Préparation des données (image)	3
1.2.5 Ajustement du modèle	4
1.2.6 Utiliser le modèle pour faire des prédictions	4
1.2.7 Évaluation du modèle sur de nouvelles données	5
1.3 Représentations de données pour les réseaux neuronaux	5
1.3.1 Scalaires (tenseurs de rang 0)	5
1.3.2 Vecteurs (tenseurs de rang 1)	5
1.3.3 Matrices (tenseurs de rang 2)	5
1.3.4 Tenseurs de rang 3 et tenseurs de rang supérieur	6
1.3.5 Caractéristiques principales	6
1.3.5.1 Affichage du cinquième chiffre	7
1.3.6 Manipulation des tenseurs dans NumPy	8
1.3.6.1 Équivalent à l'exemple précédent	8
1.3.6.2 Egalement équivalent à l'exemple précédent	9
1.3.6.3 Saucissonnage	10
1.3.7 La notion de lots de données	12
1.3.8 Exemples concrets de tenseurs de données	12

Chapitre 1

Les éléments mathématiques constitutifs des réseaux neuronaux

1.1 Code utilisé dans le terminal Windows du dossier dans lequel on travaille. On suppose que Python et Pandoc sont installés.

```
python -m venv DeepLearningBook
.\DeepLearningBook\Scripts\activate
python.exe -m pip install --upgrade pip
pip install notebook
pip install ipykernel
pip install keras
pip install tensorflow
pip install matplotlib
jupyter notebook
```

1.2 Un premier aperçu d'un réseau neuronal

1.2.1 Chargement du jeu de données MNIST dans Keras

```
[1]: from keras.datasets import mnist
```

```
C:\Users\loica\OneDrive\Desktop\Deep Learning Chollet\DeepLearningBook\Lib\site-packages\keras\src\export\tf2onnx_lib.py:8: FutureWarning: In the future
`np.object` will be defined as the corresponding NumPy scalar.
  if not hasattr(np, "object"):
```

```
[2]: (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
[3]: train_images.shape
```

```
[3]: (60000, 28, 28)
```

```
[4]: len(train_labels)
[4]: 60000
[5]: train_labels
[5]: array([5, 0, 4, ..., 5, 6, 8], shape=(60000,), dtype=uint8)
[6]: test_images.shape
[6]: (10000, 28, 28)
[7]: len(test_labels)
[7]: 10000
[8]: test_labels
[8]: array([7, 2, 1, ..., 4, 5, 6], shape=(10000,), dtype=uint8)
```

1.2.2 L'architecture du réseau

```
[9]: import keras
[10]: from keras import layers
[11]: model = keras.Sequential(
[   layers.Dense(512, activation="relu"),
    layers.Dense(10, activation="softmax"),
]
)
```

1.2.3 L'étape de compilation

```
[12]: model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
```

1.2.4 Préparation des données (image)

```
[13]: train_images = train_images.reshape((60000, 28 * 28))
[14]: train_images = train_images.astype("float32") / 255
[15]: test_images = test_images.reshape((10000, 28 * 28))
```

```
[16]: test_images = test_images.astype("float32") / 255
```

1.2.5 Ajustement du modèle

```
[17]: model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
Epoch 1/5  
469/469 4s 6ms/step -  
accuracy: 0.9248 - loss: 0.2671  
Epoch 2/5  
469/469 3s 6ms/step -  
accuracy: 0.9685 - loss: 0.1099  
Epoch 3/5  
469/469 3s 7ms/step -  
accuracy: 0.9789 - loss: 0.0714  
Epoch 4/5  
469/469 3s 7ms/step -  
accuracy: 0.9849 - loss: 0.0516  
Epoch 5/5  
469/469 3s 7ms/step -  
accuracy: 0.9879 - loss: 0.0398
```

```
[17]: <keras.src.callbacks.history.History at 0x19cc3e74980>
```

1.2.6 Utiliser le modèle pour faire des prédictions

```
[18]: test_digits = test_images[0:10]
```

```
[19]: predictions = model.predict(test_digits)
```

```
1/1 0s 76ms/step
```

```
[20]: predictions[0]
```

```
[20]: array([6.1895157e-08, 7.3658493e-09, 1.2752834e-05, 1.0990108e-05,  
           1.7984386e-10, 2.7613081e-07, 9.1607633e-12, 9.9997544e-01,  
           1.2313654e-07, 3.2242909e-07], dtype=float32)
```

```
[21]: predictions[0].argmax()
```

```
[21]: np.int64(7)
```

```
[22]: predictions[0][7]
```

```
[22]: np.float32(0.99997544)
```

```
[23]: test_labels[0]
```

```
[23]: np.uint8(7)
```

1.2.7 Évaluation du modèle sur de nouvelles données

```
[24]: test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
313/313 1s 2ms/step -  
accuracy: 0.9779 - loss: 0.0692
```

```
[25]: print(f"test_acc: {test_acc}")
```

```
test_acc: 0.9779000282287598
```

1.3 Représentations de données pour les réseaux neuronaux

1.3.1 Scalaires (tenseurs de rang 0)

```
[26]: import numpy as np
```

```
[27]: x = np.array(12)
```

```
[28]: x
```

```
[28]: array(12)
```

```
[29]: x.ndim
```

```
[29]: 0
```

1.3.2 Vecteurs (tenseurs de rang 1)

```
[30]: x = np.array([12, 3, 6, 14, 7])
```

```
[31]: x
```

```
[31]: array([12, 3, 6, 14, 7])
```

```
[32]: x.ndim
```

```
[32]: 1
```

1.3.3 Matrices (tenseurs de rang 2)

```
[33]: x = np.array([[5, 78, 2, 34, 0],  
                  [6, 79, 3, 35, 1],  
                  [7, 80, 4, 36, 2]])
```

```
[34]: x
```

```
[34]: array([[ 5, 78,  2, 34,  0],  
           [ 6, 79,  3, 35,  1],  
           [ 7, 80,  4, 36,  2],  
           [ 8, 81,  5, 37,  3],  
           [ 9, 82,  6, 38,  4],  
           [10, 83,  7, 39,  5],  
           [11, 84,  8, 40,  6],  
           [12, 85,  9, 41,  7],  
           [13, 86, 10, 42,  8],  
           [14, 87, 11, 43,  9],  
           [15, 88, 12, 44, 10],  
           [16, 89, 13, 45, 11],  
           [17, 90, 14, 46, 12],  
           [18, 91, 15, 47, 13],  
           [19, 92, 16, 48, 14],  
           [20, 93, 17, 49, 15],  
           [21, 94, 18, 50, 16],  
           [22, 95, 19, 51, 17],  
           [23, 96, 20, 52, 18],  
           [24, 97, 21, 53, 19],  
           [25, 98, 22, 54, 20],  
           [26, 99, 23, 55, 21],  
           [27, 100, 24, 56, 22],  
           [28, 101, 25, 57, 23],  
           [29, 102, 26, 58, 24],  
           [30, 103, 27, 59, 25],  
           [31, 104, 28, 60, 26],  
           [32, 105, 29, 61, 27],  
           [33, 106, 30, 62, 28],  
           [34, 107, 31, 63, 29],  
           [35, 108, 32, 64, 30],  
           [36, 109, 33, 65, 31],  
           [37, 110, 34, 66, 32],  
           [38, 111, 35, 67, 33],  
           [39, 112, 36, 68, 34],  
           [40, 113, 37, 69, 35],  
           [41, 114, 38, 70, 36],  
           [42, 115, 39, 71, 37],  
           [43, 116, 40, 72, 38],  
           [44, 117, 41, 73, 39],  
           [45, 118, 42, 74, 40],  
           [46, 119, 43, 75, 41],  
           [47, 120, 44, 76, 42],  
           [48, 121, 45, 77, 43],  
           [49, 122, 46, 78, 44],  
           [50, 123, 47, 79, 45],  
           [51, 124, 48, 80, 46],  
           [52, 125, 49, 81, 47],  
           [53, 126, 50, 82, 48],  
           [54, 127, 51, 83, 49],  
           [55, 128, 52, 84, 50],  
           [56, 129, 53, 85, 51],  
           [57, 130, 54, 86, 52],  
           [58, 131, 55, 87, 53],  
           [59, 132, 56, 88, 54],  
           [60, 133, 57, 89, 55],  
           [61, 134, 58, 90, 56],  
           [62, 135, 59, 91, 57],  
           [63, 136, 60, 92, 58],  
           [64, 137, 61, 93, 59],  
           [65, 138, 62, 94, 60],  
           [66, 139, 63, 95, 61],  
           [67, 140, 64, 96, 62],  
           [68, 141, 65, 97, 63],  
           [69, 142, 66, 98, 64],  
           [70, 143, 67, 99, 65],  
           [71, 144, 68, 100, 66],  
           [72, 145, 69, 101, 67],  
           [73, 146, 70, 102, 68],  
           [74, 147, 71, 103, 69],  
           [75, 148, 72, 104, 70],  
           [76, 149, 73, 105, 71],  
           [77, 150, 74, 106, 72],  
           [78, 151, 75, 107, 73],  
           [79, 152, 76, 108, 74],  
           [80, 153, 77, 109, 75],  
           [81, 154, 78, 110, 76],  
           [82, 155, 79, 111, 77],  
           [83, 156, 80, 112, 78],  
           [84, 157, 81, 113, 79],  
           [85, 158, 82, 114, 80],  
           [86, 159, 83, 115, 81],  
           [87, 160, 84, 116, 82],  
           [88, 161, 85, 117, 83],  
           [89, 162, 86, 118, 84],  
           [90, 163, 87, 119, 85],  
           [91, 164, 88, 120, 86],  
           [92, 165, 89, 121, 87],  
           [93, 166, 90, 122, 88],  
           [94, 167, 91, 123, 89],  
           [95, 168, 92, 124, 90],  
           [96, 169, 93, 125, 91],  
           [97, 170, 94, 126, 92],  
           [98, 171, 95, 127, 93],  
           [99, 172, 96, 128, 94],  
           [100, 173, 97, 129, 95],  
           [101, 174, 98, 130, 96],  
           [102, 175, 99, 131, 97],  
           [103, 176, 100, 132, 98],  
           [104, 177, 101, 133, 99],  
           [105, 178, 102, 134, 100],  
           [106, 179, 103, 135, 101],  
           [107, 180, 104, 136, 102],  
           [108, 181, 105, 137, 103],  
           [109, 182, 106, 138, 104],  
           [110, 183, 107, 139, 105],  
           [111, 184, 108, 140, 106],  
           [112, 185, 109, 141, 107],  
           [113, 186, 110, 142, 108],  
           [114, 187, 111, 143, 109],  
           [115, 188, 112, 144, 110],  
           [116, 189, 113, 145, 111],  
           [117, 190, 114, 146, 112],  
           [118, 191, 115, 147, 113],  
           [119, 192, 116, 148, 114],  
           [120, 193, 117, 149, 115],  
           [121, 194, 118, 150, 116],  
           [122, 195, 119, 151, 117],  
           [123, 196, 120, 152, 118],  
           [124, 197, 121, 153, 119],  
           [125, 198, 122, 154, 120],  
           [126, 199, 123, 155, 121],  
           [127, 200, 124, 156, 122],  
           [128, 201, 125, 157, 123],  
           [129, 202, 126, 158, 124],  
           [130, 203, 127, 159, 125],  
           [131, 204, 128, 160, 126],  
           [132, 205, 129, 161, 127],  
           [133, 206, 130, 162, 128],  
           [134, 207, 131, 163, 129],  
           [135, 208, 132, 164, 130],  
           [136, 209, 133, 165, 131],  
           [137, 210, 134, 166, 132],  
           [138, 211, 135, 167, 133],  
           [139, 212, 136, 168, 134],  
           [140, 213, 137, 169, 135],  
           [141, 214, 138, 170, 136],  
           [142, 215, 139, 171, 137],  
           [143, 216, 140, 172, 138],  
           [144, 217, 141, 173, 139],  
           [145, 218, 142, 174, 140],  
           [146, 219, 143, 175, 141],  
           [147, 220, 144, 176, 142],  
           [148, 221, 145, 177, 143],  
           [149, 222, 146, 178, 144],  
           [150, 223, 147, 179, 145],  
           [151, 224, 148, 180, 146],  
           [152, 225, 149, 181, 147],  
           [153, 226, 150, 182, 148],  
           [154, 227, 151, 183, 149],  
           [155, 228, 152, 184, 150],  
           [156, 229, 153, 185, 151],  
           [157, 230, 154, 186, 152],  
           [158, 231, 155, 187, 153],  
           [159, 232, 156, 188, 154],  
           [160, 233, 157, 189, 155],  
           [161, 234, 158, 190, 156],  
           [162, 235, 159, 191, 157],  
           [163, 236, 160, 192, 158],  
           [164, 237, 161, 193, 159],  
           [165, 238, 162, 194, 160],  
           [166, 239, 163, 195, 161],  
           [167, 240, 164, 196, 162],  
           [168, 241, 165, 197, 163],  
           [169, 242, 166, 198, 164],  
           [170, 243, 167, 199, 165],  
           [171, 244, 168, 200, 166],  
           [172, 245, 169, 201, 167],  
           [173, 246, 170, 202, 168],  
           [174, 247, 171, 203, 169],  
           [175, 248, 172, 204, 170],  
           [176, 249, 173, 205, 171],  
           [177, 250, 174, 206, 172],  
           [178, 251, 175, 207, 173],  
           [179, 252, 176, 208, 174],  
           [180, 253, 177, 209, 175],  
           [181, 254, 178, 210, 176],  
           [182, 255, 179, 211, 177],  
           [183, 256, 180, 212, 178],  
           [184, 257, 181, 213, 179],  
           [185, 258, 182, 214, 180],  
           [186, 259, 183, 215, 181],  
           [187, 260, 184, 216, 182],  
           [188, 261, 185, 217, 183],  
           [189, 262, 186, 218, 184],  
           [190, 263, 187, 219, 185],  
           [191, 264, 188, 220, 186],  
           [192, 265, 189, 221, 187],  
           [193, 266, 190, 222, 188],  
           [194, 267, 191, 223, 189],  
           [195, 268, 192, 224, 190],  
           [196, 269, 193, 225, 191],  
           [197, 270, 194, 226, 192],  
           [198, 271, 195, 227, 193],  
           [199, 272, 196, 228, 194],  
           [200, 273, 197, 229, 195],  
           [201, 274, 198, 230, 196],  
           [202, 275, 199, 231, 197],  
           [203, 276, 200, 232, 198],  
           [204, 277, 201, 233, 199],  
           [205, 278, 202, 234, 200],  
           [206, 279, 203, 235, 201],  
           [207, 280, 204, 236, 202],  
           [208, 281, 205, 237, 203],  
           [209, 282, 206, 238, 204],  
           [210, 283, 207, 239, 205],  
           [211, 284, 208, 240, 206],  
           [212, 285, 209, 241, 207],  
           [213, 286, 210, 242, 208],  
           [214, 287, 211, 243, 209],  
           [215, 288, 212, 244, 210],  
           [216, 289, 213, 245, 211],  
           [217, 290, 214, 246, 212],  
           [218, 291, 215, 247, 213],  
           [219, 292, 216, 248, 214],  
           [220, 293, 217, 249, 215],  
           [221, 294, 218, 250, 216],  
           [222, 295, 219, 251, 217],  
           [223, 296, 220, 252, 218],  
           [224, 297, 221, 253, 219],  
           [225, 298, 222, 254, 220],  
           [226, 299, 223, 255, 221],  
           [227, 300, 224, 256, 222],  
           [228, 301, 225, 257, 223],  
           [229, 302, 226, 258, 224],  
           [230, 303, 227, 259, 225],  
           [231, 304, 228, 260, 226],  
           [232, 305, 229, 261, 227],  
           [233, 306, 230, 262, 228],  
           [234, 307, 231, 263, 229],  
           [235, 308, 232, 264, 230],  
           [236, 309, 233, 265, 231],  
           [237, 310, 234, 266, 232],  
           [238, 311, 235, 267, 233],  
           [239, 312, 236, 268, 234],  
           [240, 313, 237, 269, 235],  
           [241, 314, 238, 270, 236],  
           [242, 315, 239, 271, 237],  
           [243, 316, 240, 272, 238],  
           [244, 317, 241, 273, 239],  
           [245, 318, 242, 274, 240],  
           [246, 319, 243, 275, 241],  
           [247, 320, 244, 276, 242],  
           [248, 321, 245, 277, 243],  
           [249, 322, 246, 278, 244],  
           [250, 323, 247, 279, 245],  
           [251, 324, 248, 280, 246],  
           [252, 325, 249, 281, 247],  
           [253, 326, 250, 282, 248],  
           [254, 327, 251, 283, 249],  
           [255, 328, 252, 284, 250],  
           [256, 329, 253, 285, 251],  
           [257, 330, 254, 286, 252],  
           [258, 331, 255, 287, 253],  
           [259, 332, 256, 288, 254],  
           [260, 333, 257, 289, 255],  
           [261, 334, 258, 290, 256],  
           [262, 335, 259, 291, 257],  
           [263, 336, 260, 292, 258],  
           [264, 337, 261, 293, 259],  
           [265, 338, 262, 294, 260],  
           [266, 339, 263, 295, 261],  
           [267, 340, 264, 296, 262],  
           [268, 341, 265, 297, 263],  
           [269, 342, 266, 298, 264],  
           [270, 343, 267, 299, 265],  
           [271, 344, 268, 300, 266],  
           [272, 345, 269, 301, 267],  
           [273, 346, 270, 302, 268],  
           [274, 347, 271, 303, 269],  
           [275, 348, 272, 304, 270],  
           [276, 349, 273, 305, 271],  
           [277, 350, 274, 306, 272],  
           [278, 351, 275, 307, 273],  
           [279, 352, 276, 308, 274],  
           [280, 353, 277, 309, 275],  
           [281, 354, 278, 310, 276],  
           [282, 355, 279, 311, 277],  
           [283, 356, 280, 312, 278],  
           [284, 357, 281, 313, 279],  
           [285, 358, 282, 314, 280],  
           [286, 359, 283, 315, 281],  
           [287, 360, 284, 316, 282],  
           [288, 361, 285, 317, 283],  
           [289, 362, 286, 318, 284],  
           [290, 363, 287, 319, 285],  
           [291, 364, 288, 320, 286],  
           [292, 365, 289, 321, 287],  
           [293, 366, 290, 322, 288],  
           [294, 367, 291, 323, 289],  
           [295, 368, 292, 324, 290],  
           [296, 369, 293, 325, 291],  
           [297, 370, 294, 326, 292],  
           [298, 371, 295, 327, 293],  
           [299, 372, 296, 328, 294],  
           [300, 373, 297, 329, 295],  
           [301, 374, 298, 330, 296],  
           [302, 375, 299, 331, 297],  
           [303, 376, 300, 332, 298],  
           [304, 377, 301, 333, 299],  
           [305, 378, 302, 334, 300],  
           [306, 379, 303, 335, 301],  
           [307, 380, 304, 336, 302],  
           [308, 381, 305, 337, 303],  
           [309, 382, 306, 338, 304],  
           [310, 383, 307, 339, 305],  
           [311, 384, 308, 340, 306],  
           [312, 385, 309, 341, 307],  
           [313, 386, 310, 342, 308],  
           [314, 387, 311, 343, 309],  
           [315, 388, 312, 344, 310],  
           [316, 389, 313, 345, 311],  
           [317, 390, 314, 346, 312],  
           [318, 391, 315, 347, 313],  
           [319, 392, 316, 348, 314],  
           [320, 393, 317, 349, 315],  
           [321, 394, 318, 350, 316],  
           [322, 395, 319, 351, 317],  
           [323, 396, 320, 352, 318],  
           [324, 397, 321, 353, 319],  
           [325, 398, 322, 354, 320],  
           [326, 399, 323, 355, 321],  
           [327, 400, 324, 356, 322],  
           [328, 401, 325, 357, 323],  
           [329, 402, 326, 358, 324],  
           [330, 403, 327, 359, 325],  
           [331, 404, 328, 360, 326],  
           [332, 405, 329, 361, 327],  
           [333, 406, 330, 362, 328],  
           [334, 407, 331, 363, 329],  
           [335, 408, 332, 364, 330],  
           [336, 409, 333, 365, 331],  
           [337, 410, 334, 366, 332],  
           [338, 411, 335, 367, 333],  
           [339, 412, 336, 368, 334],  
           [340, 413, 337, 369, 335],  
           [341, 414, 338, 370, 336],  
           [342, 415, 339, 371, 337],  
           [343, 416, 340, 372, 338],  
           [344, 417, 341, 373, 339],  
           [345, 418, 342, 374, 340],  
           [346, 419, 343, 375, 341],  
           [347, 420, 344, 376, 342],  
           [348, 421, 345, 377, 343],  
           [349, 422, 346, 378, 344],  
           [350, 423, 347, 379, 345],  
           [351, 424, 348, 380, 346],  
           [352, 425, 349, 381, 347],  
           [353, 426, 350, 382, 348],  
           [354, 427, 351, 383, 349],  
           [355, 428, 352, 384, 350],  
           [356, 429, 353, 385, 351],  
           [357, 430, 354, 386, 352],  
           [358, 431, 355, 387, 353],  
           [359, 432, 356, 388, 354],  
           [360, 433, 357, 389, 355],  
           [361, 434, 358, 390, 356],  
           [362, 435, 359, 391, 357],  
           [363, 436, 360, 392, 358],  
           [364, 437, 361, 393, 359],  
           [365, 438, 362, 394, 360],  
           [366, 439, 363, 395, 361],  
           [367, 440, 364, 396, 362],  
           [368, 441, 365, 397, 363],  
           [369, 442, 366, 398, 364],  
           [370, 443, 367, 399, 365],  
           [371, 444, 368, 400, 366],  
           [372, 445, 369, 401, 367],  
           [373, 446,
```

```
[ 7, 80, 4, 36, 2]])
```

```
[35]: x.ndim
```

```
[35]: 2
```

1.3.4 Tenseurs de rang 3 et tenseurs de rang supérieur

```
[36]: x = np.array([[[5, 78, 2, 34, 0],  
                   [6, 79, 3, 35, 1],  
                   [7, 80, 4, 36, 2]],  
                  [[5, 78, 2, 34, 0],  
                   [6, 79, 3, 35, 1],  
                   [7, 80, 4, 36, 2]],  
                  [[5, 78, 2, 34, 0],  
                   [6, 79, 3, 35, 1],  
                   [7, 80, 4, 36, 2]]])
```

```
[37]: x
```

```
[37]: array([[[ 5, 78, 2, 34, 0],  
              [ 6, 79, 3, 35, 1],  
              [ 7, 80, 4, 36, 2]],  
  
             [[ 5, 78, 2, 34, 0],  
              [ 6, 79, 3, 35, 1],  
              [ 7, 80, 4, 36, 2]],  
  
             [[ 5, 78, 2, 34, 0],  
              [ 6, 79, 3, 35, 1],  
              [ 7, 80, 4, 36, 2]])
```

```
[38]: x.ndim
```

```
[38]: 3
```

1.3.5 Caractéristiques principales

```
[39]: from keras.datasets import mnist
```

```
[40]: (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
[41]: train_images.ndim
```

```
[41]: 3
```

```
[42]: train_images.shape
```

```
[42]: (60000, 28, 28)
```

```
[43]: train_images.dtype
```

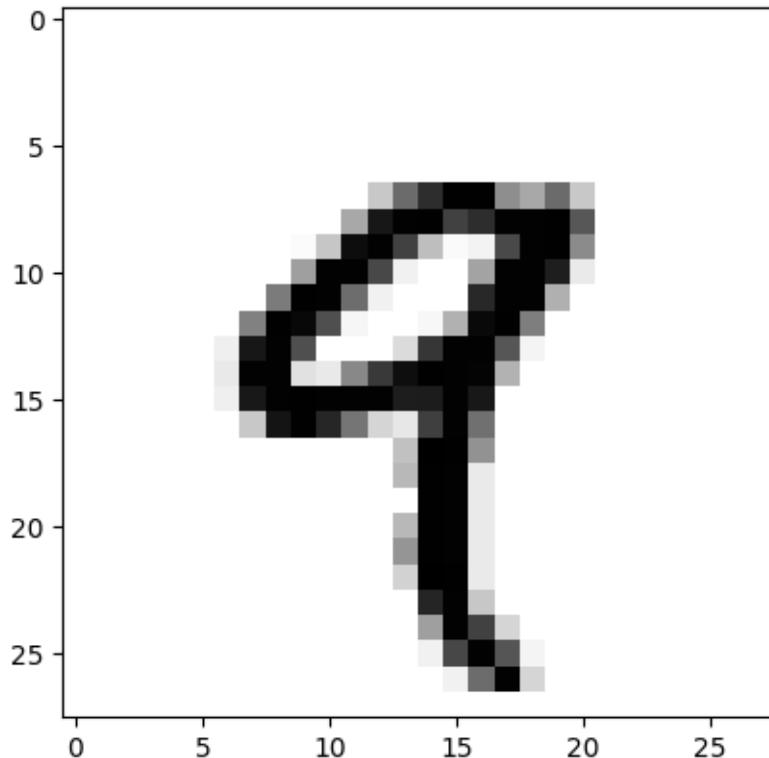
```
[43]: dtype('uint8')
```

1.3.5.1 Affichage du cinquième chiffre

```
[44]: import matplotlib.pyplot as plt
```

```
[45]: digit = train_images[4]
```

```
[46]: plt.imshow(digit, cmap=plt.cm.binary)  
plt.show()
```



```
[47]: train_labels[4]
```

```
[47]: np.uint8(9)
```

1.3.6 Manipulation des tenseurs dans NumPy

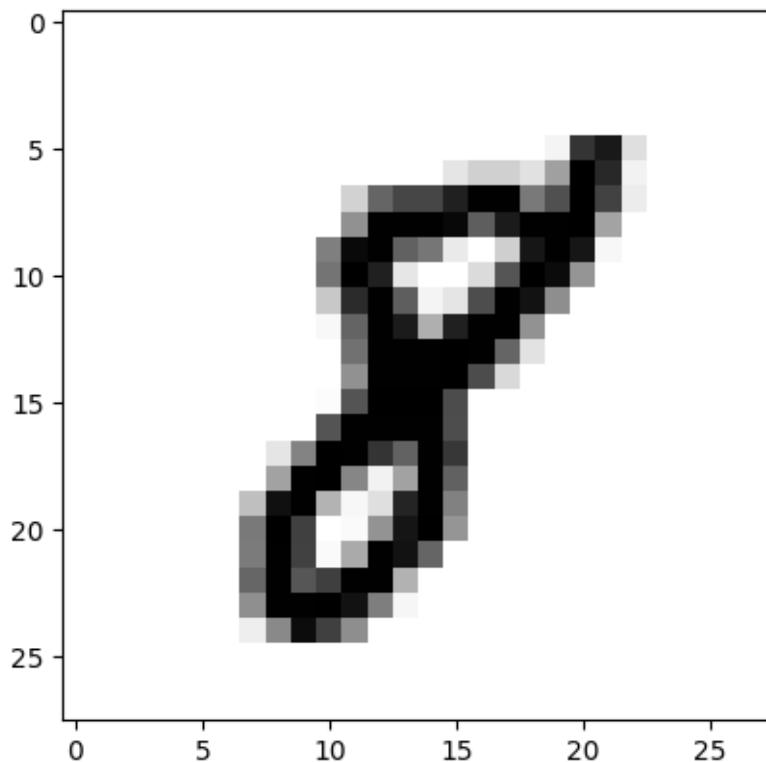
```
[48]: my_slice = train_images[10:100]
```

```
[49]: my_slice.shape
```

```
[49]: (90, 28, 28)
```

```
[50]: import matplotlib.pyplot as plt
```

```
[51]: plt.imshow(my_slice[7], cmap=plt.cm.binary)  
plt.show()
```



1.3.6.1 Équivalent à l'exemple précédent

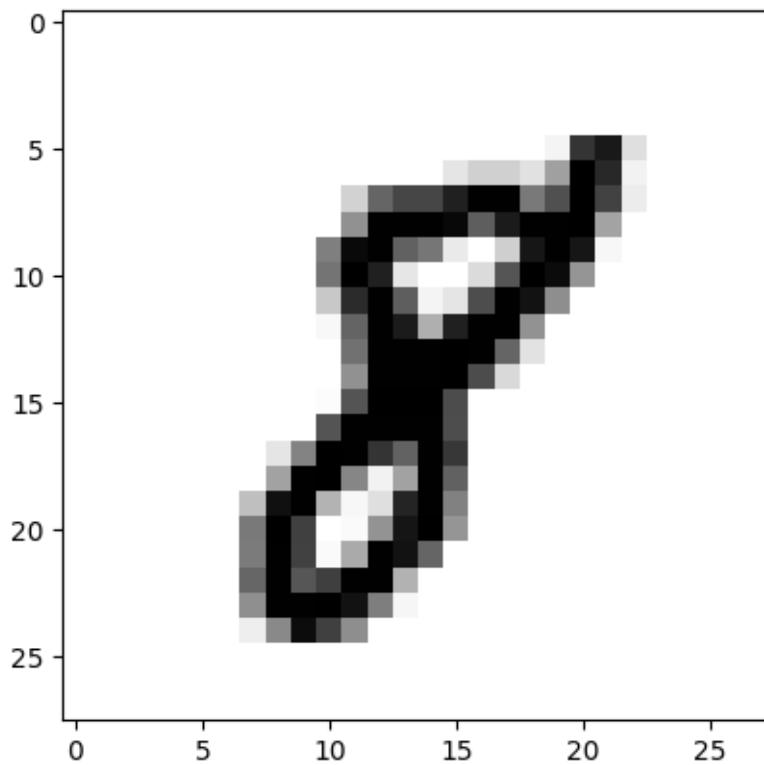
```
[52]: my_slice = train_images[10:100, :, :]
```

```
[53]: my_slice.shape
```

```
[53]: (90, 28, 28)
```

```
[54]: import matplotlib.pyplot as plt
```

```
[55]: plt.imshow(my_slice[7], cmap=plt.cm.binary)
plt.show()
```



1.3.6.2 Egalement équivalent à l'exemple précédent

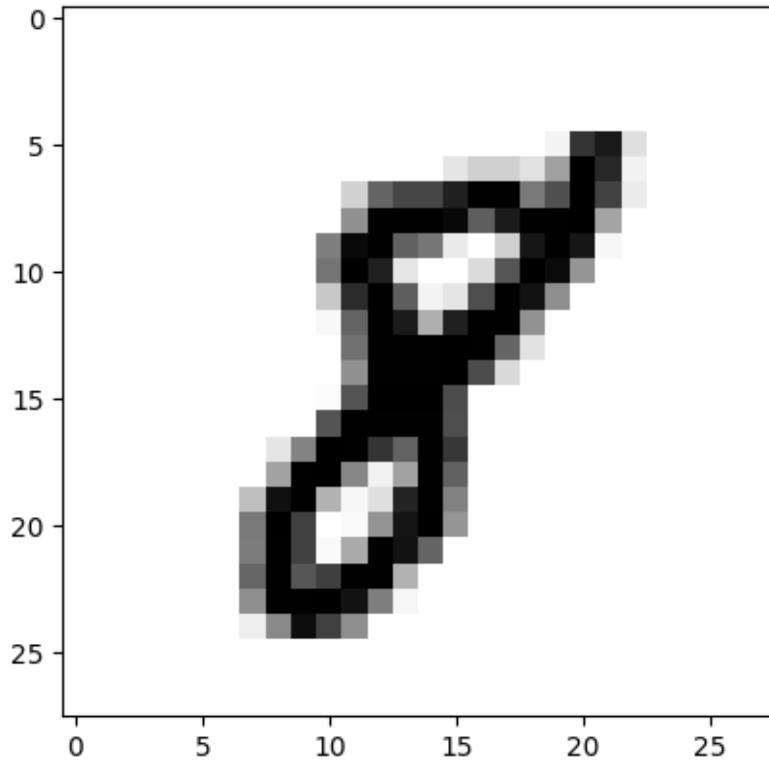
```
[56]: my_slice = train_images[10:100, 0:28, 0:28]
```

```
[57]: my_slice.shape
```

```
[57]: (90, 28, 28)
```

```
[58]: import matplotlib.pyplot as plt
```

```
[59]: plt.imshow(my_slice[7], cmap=plt.cm.binary)
plt.show()
```

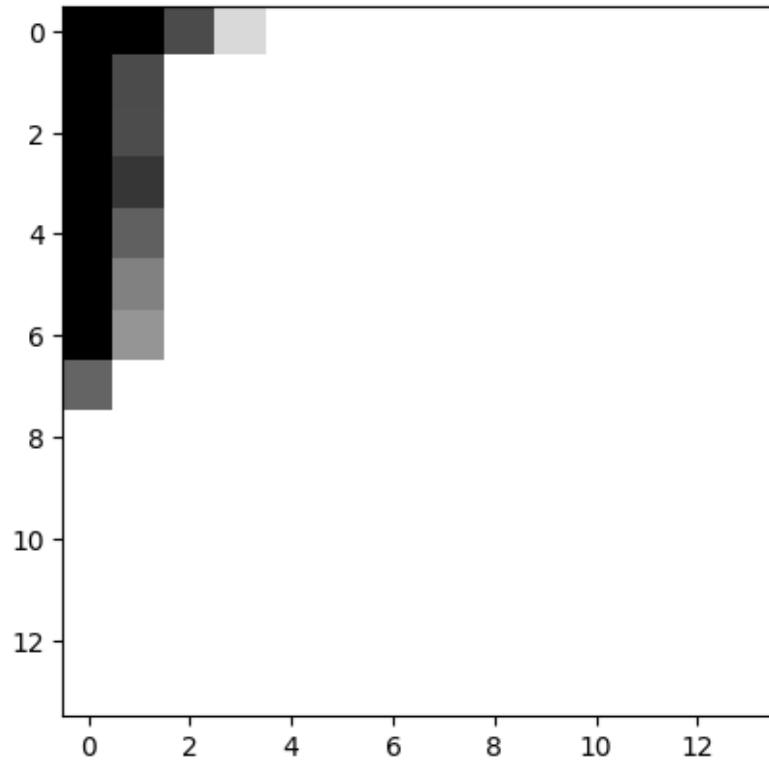


1.3.6.3 Saucissonnage

```
[60]: my_slice = train_images[:, 14:, 14:]
```

```
[61]: import matplotlib.pyplot as plt
```

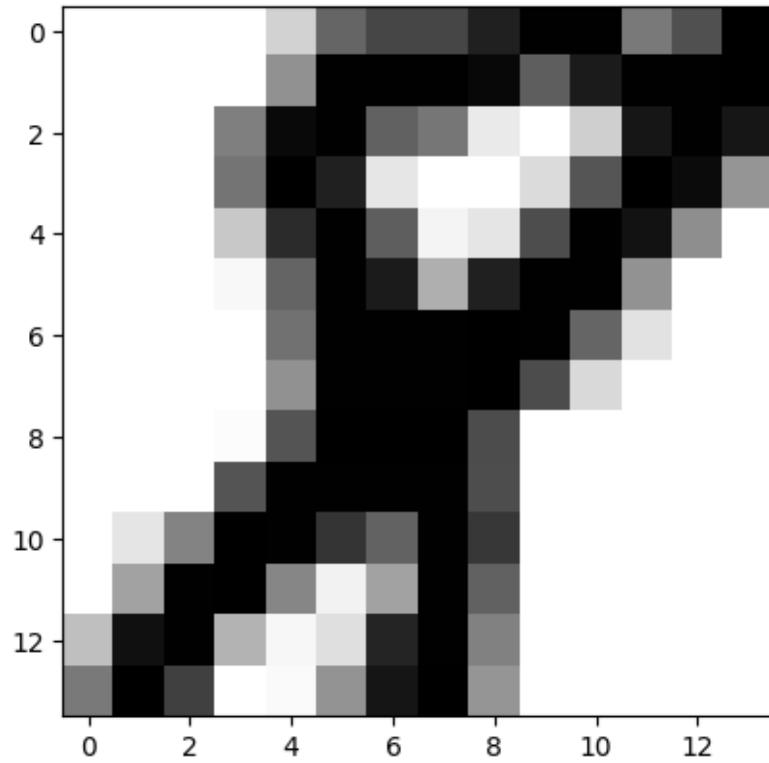
```
[62]: plt.imshow(my_slice[17], cmap=plt.cm.binary)  
plt.show()
```



```
[63]: my_slice = train_images[:, 7:-7, 7:-7]
```

```
[64]: import matplotlib.pyplot as plt
```

```
[65]: plt.imshow(my_slice[17], cmap=plt.cm.binary)
plt.show()
```



1.3.7 La notion de lots de données

```
[66]: batch = train_images[:128]
```

```
[67]: batch = train_images[128:256]
```

```
[68]: n = 3  
batch = train_images[128 * n : 128 * (n + 1)]
```

1.3.8 Exemples concrets de tenseurs de données

```
[ ]:
```