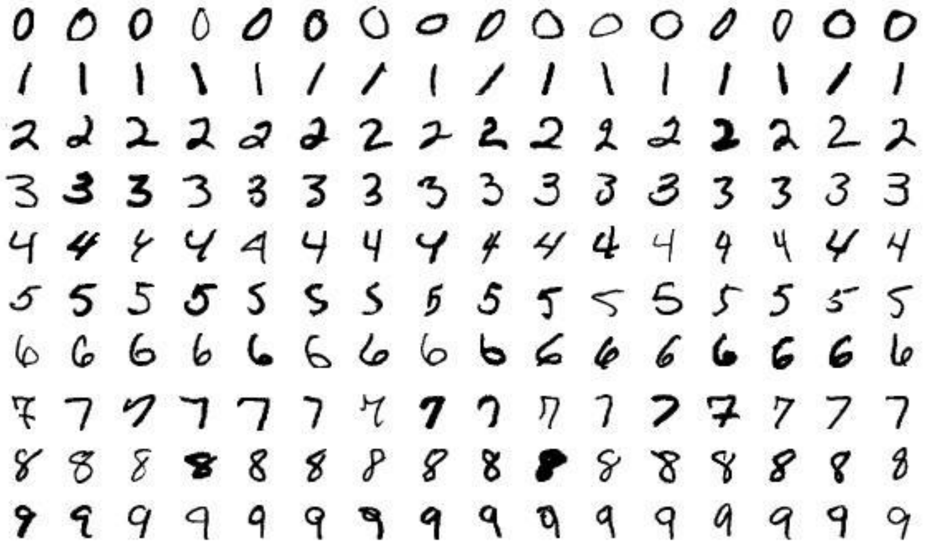
PyTorch

Libraries to install

- Numpy
- Torch
- Sklearn
- How?
  - pip install numpy, pip install scikit-learn, pip install torch
  - conda install numpy, conda install scikit-learn, conda install scikit-learn
- If pip is not installed:
  - python -m pip install --upgrade pip

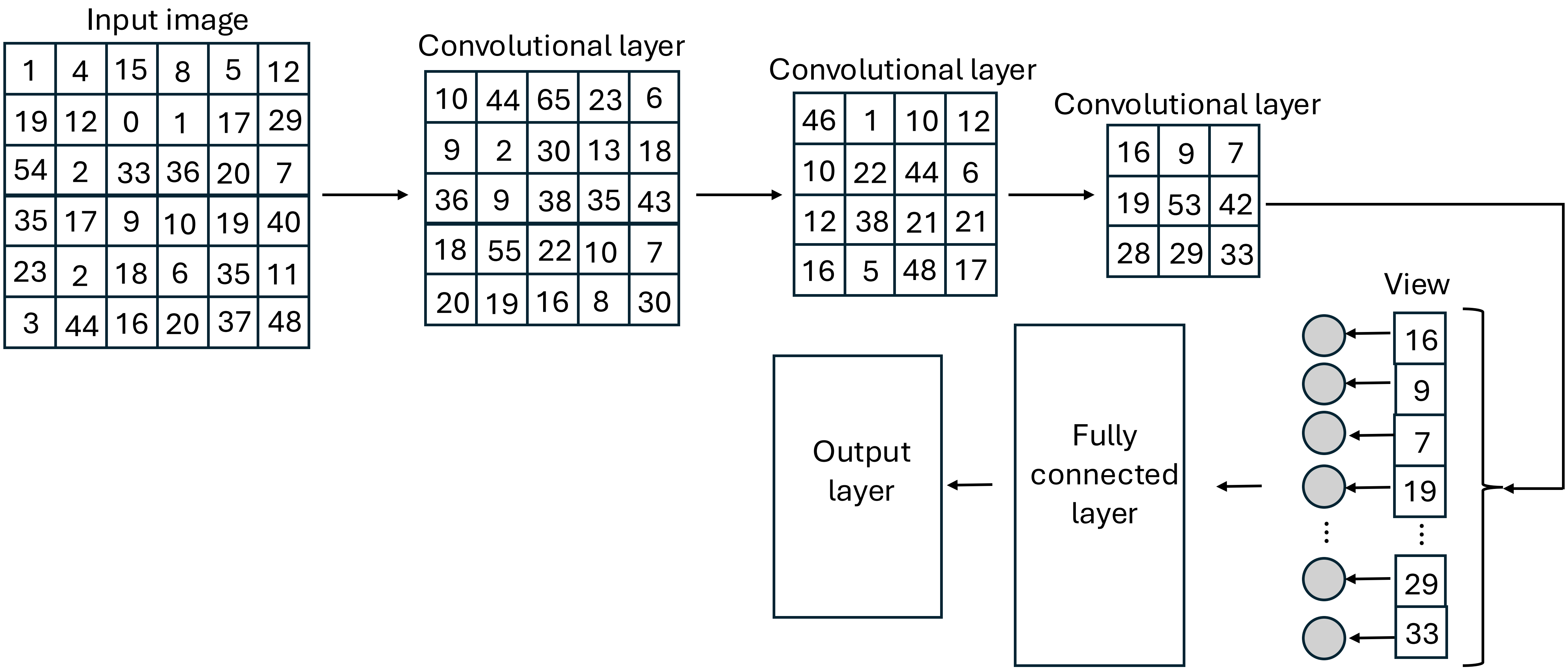
Problem

- MNIST Database: Handwritten Digits.
- The goal is to build a neural network that can **recognize digits from 0 to 9**.
- We will use **two types of networks**:
  - CNN (Convolutional Neural Network).
  - Fully-connected Neural Network.



Convolutional layers

- The resulting feature map size decreases with each convolutional layer.



Input image

1	4	15	8	5	12
19	12	0	1	17	29
54	2	33	36	20	7
35	17	9	10	19	40
23	2	18	6	35	11
3	44	16	20	37	48

Convolutional layer

10	44	65	23	6
9	2	30	13	18
36	9	38	35	43
18	55	22	10	7
20	19	16	8	30

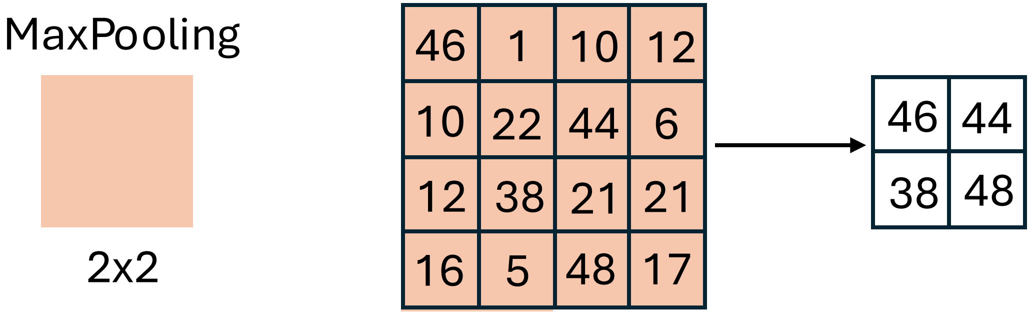
Convolutional layer

46	1	10	12
10	22	44	6
12	38	21	21
16	5	48	17

Convolutional layer

16	9	7
19	53	42
28	29	33

Max Pooling



MaxPooling

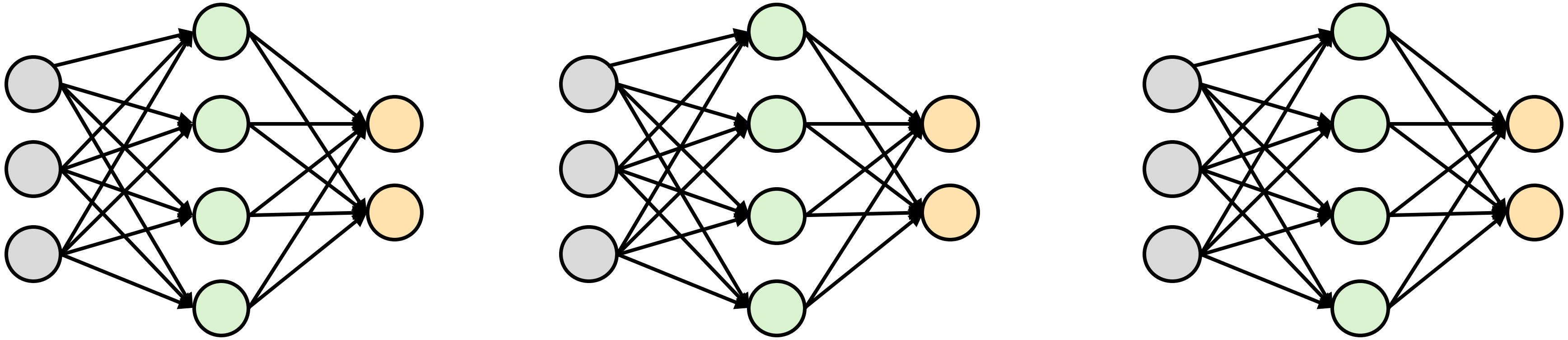
2x2

46	1	10	12
10	22	44	6
12	38	21	21
16	5	48	17

46	44
38	48

Dropout

- Assigns a weight of 0 (no contribution) to the specified percentage of neurons.
- Example:



Hands-on exercises

- Instead of building a model to classify all 10 digits, convert the task into a **binary classification** problem — for example, distinguishing between **0 and 8**. To do this:
  - **Select only the images** corresponding to digits **0** and **8** from the dataset.
  - **Modify the neural network accordingly** to handle binary output.

Hands-on exercises

- To use BCELoss() instead of CrossEntropyLoss()
    - criterion = nn.BCELoss()
        - onehot_encoder = OneHotEncoder(sparse_output=False)
        - labels_oh = onehot_encoder.fit_transform(labels.reshape(-1,1))
        - labels = labels_oh.astype('float32')
    - self.sig = nn.Sigmoid()
    - bal_acc = balanced_accuracy_score(onehot_encoder.inverse_transform(np.reshape(labels_test,[labels_test.shape[0],labels_test.shape[2]])),predictions)