Tutorial on Improving Deep Learning by Exploiting Synthetic Images

Jiayun Liu, Ontology Engineering Group
Universidad Politécnica de Madrid, Spain
Manuel Castillo-Cara, Universidad Nacional de Educación a Distancia
Raúl García-Castro, Ontology Engineering Group, Universidad Politécnica de Madrid

jiayun.liu@upm.es DCY1117 18/10/24 Santiago de Compostela

Materials

Materials: https://gooev.ai/2/9pqz

Index

1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Index

1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Tabular data

preg	plas	pres	skin	insu	mass	pedi	age	class
1	85	66	29	0	26.6	351	31	tested_negative
5	116	74	0	0	25.6	201	30	tested_negative
10	115	0	0	0	35.3	134	29	tested_negative
4	110	92	0	0	37.6	191	30	tested_negative
10	139	80	0	0	27.1	1441	57	tested_negative
8	99	84	0	0	35.4	388	50	tested_negative
5	117	92	0	0	34.1	337	38	tested_negative
5	109	75	26	0	36	546	60	tested_negative

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	class
0.00632	18	2.31	0	538	6575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	469	6421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	469	7185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	458	6998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	458	7147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.02985	0	2.18	0	458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

Tabular data

Classification

preg	plas	pres	skin	insu	mass	pedi	age	class
1	85	66	29	0	26.6	351	31	tested_negative
5	116	74	0	0	25.6	201	30	tested_negative
10	115	0	0	0	35.3	134	29	tested_negative
4	110	92	0	0	37.6	191	30	tested_negative
10	139	80	0	0	27.1	1441	57	tested_negative
8	99	84	0	0	35.4	388	50	tested_negative
5	117	92	0	0	34.1	337	38	tested_negative
5	109	75	26	0	36	546	60	tested_negative

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	class
0.00632	18	2.31	0	538	6575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	469	6421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	469	7185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	458	6998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	458	7147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.02985	0	2.18	0	458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

Tabular data

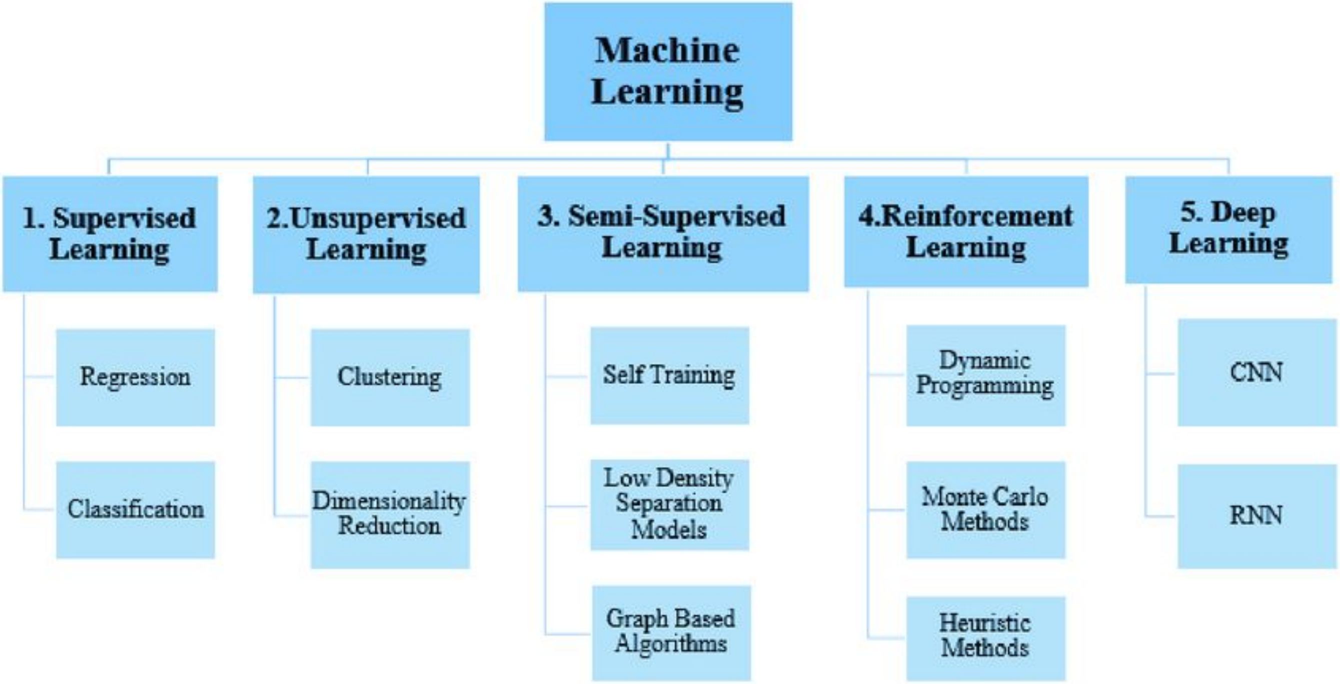
preg	plas	pres	skin	insu	mass	pedi	age	class
1	85	66	29	0	26.6	351	31	tested_negative
5	116	74	0	0	25.6	201	30	tested_negative
10	115	0	0	0	35.3	134	29	tested_negative
4	110	92	0	0	37.6	191	30	tested_negative
10	139	80	0	0	27.1	1441	57	tested_negative
8	99	84	0	0	35.4	388	50	tested_negative
5	117	92	0	0	34.1	337	38	tested_negative
5	109	75	26	0	36	546	60	tested_negative

Classification

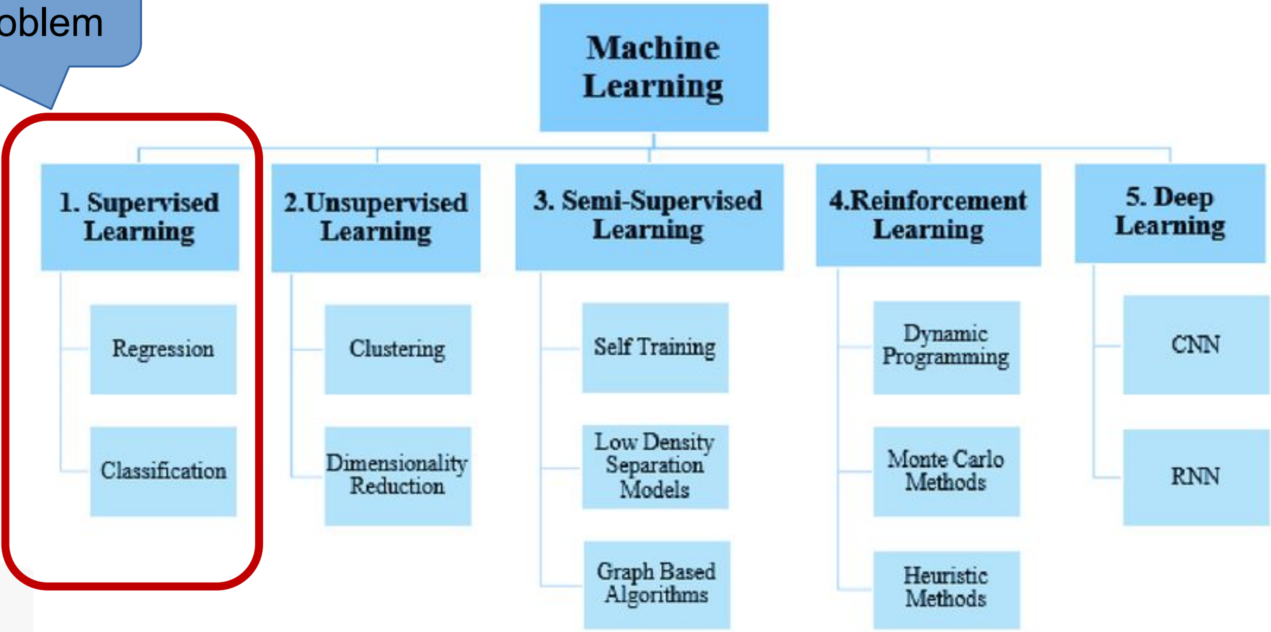
CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	class
0.00632	18	2.31	0	538	6575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	469	6421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	469	7185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	458	6998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	458	7147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.02985	0	2.18	0	458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

Regression

Tabular data

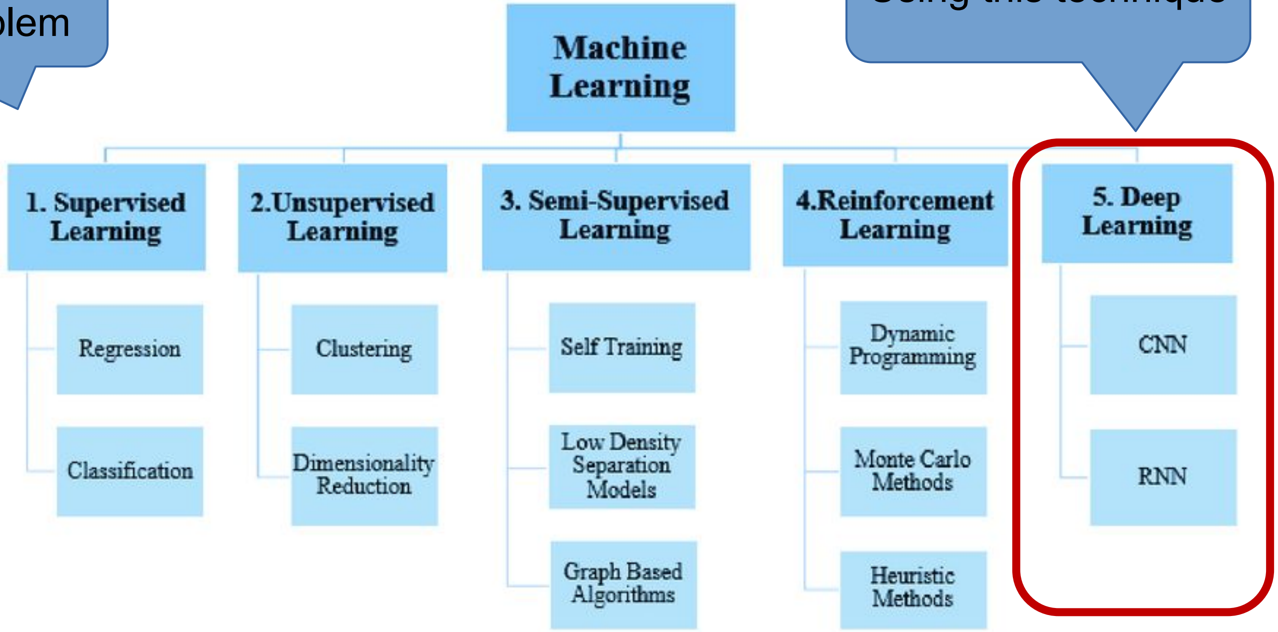


We are going to resolve this problem



We are going to resolve this problem

Using this technique



The open challenge in Deep Learning for Tabular Data

- Kadra et al. referred to tabular datasets as the "last unconquered castle" for models based on Deep Neural Networks (DNNs).
- Applying DNNs to Tabular Data (TD) for inference or data generation remains a significant challenge.

The references or papers are included as embedded links within the text.

The open challenge in Deep Learning for Tabular Data

- Kadra et al. referred to tabular datasets as the "last unconquered castle" for models based on Deep Neural Networks (DNNs).
- Applying DNNs to Tabular Data (TD) for inference or data generation remains a significant challenge.
- Vadim Borísov et al. conducted benchmarks comparing ensemble methods and DNNs across various datasets, stating that "Progress in research on competitive deep learning models for tabular data is stagnating." and "It is an open research area."

The references or papers are included as embedded links within the text.

The open challenge in Deep Learning for Tabular Data

- Kadra et al. referred to tabular datasets as the "last unconquered castle" for models based on Deep Neural Networks (DNNs).
- Applying DNNs to Tabular Data (TD) for inference or data generation remains a significant challenge.
- Vadim Borísov et al. conducted benchmarks comparing ensemble methods and DNNs across various datasets, stating that "Progress in research on competitive deep learning models for tabular data is stagnating." and "It is an open research area."
- Shwartz Ziv & Armon In their paper, "Tabular Data: Deep Learning is Not All You Need," they compared DNN approaches with Gradient Boosted Decision Trees (GBDT). GBDTs challenged DNNs, indicating that modeling tabular data using DNNs is still an open research problem.

The references or papers are included as embedded links within the text.

Index

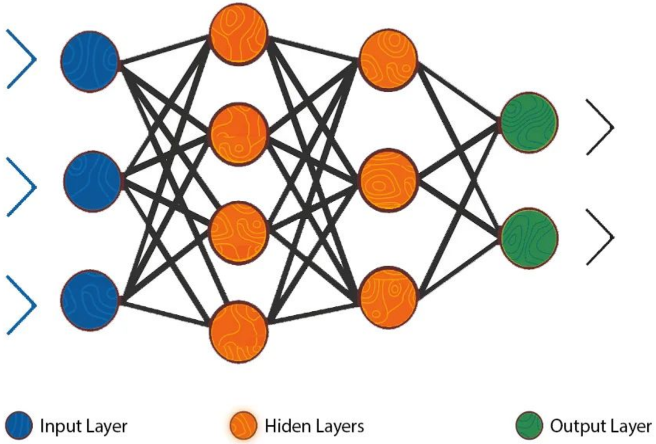
1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Tabular data + Deep Learning

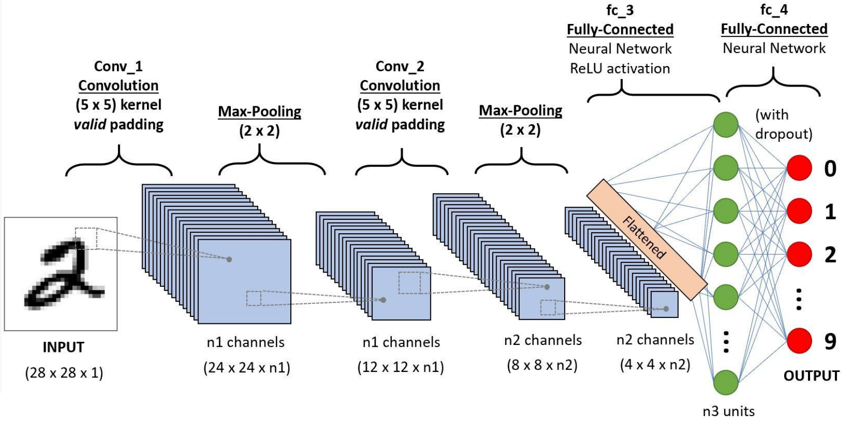
Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
…	…	…	…	…	…
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15

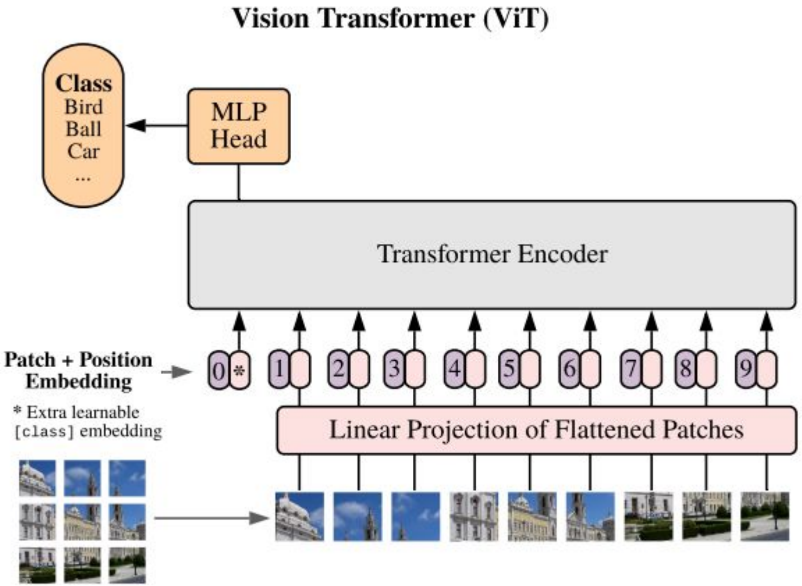
Tabular data + Deep Learning

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15



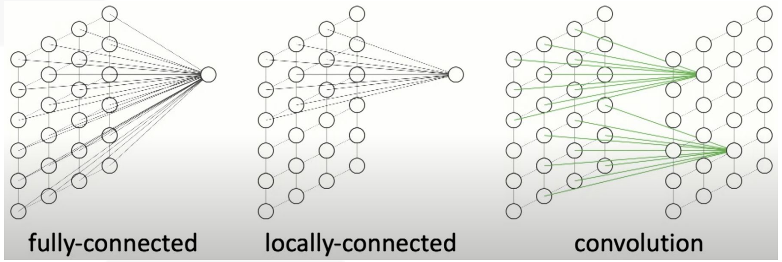
Tabular data + Deep Learning

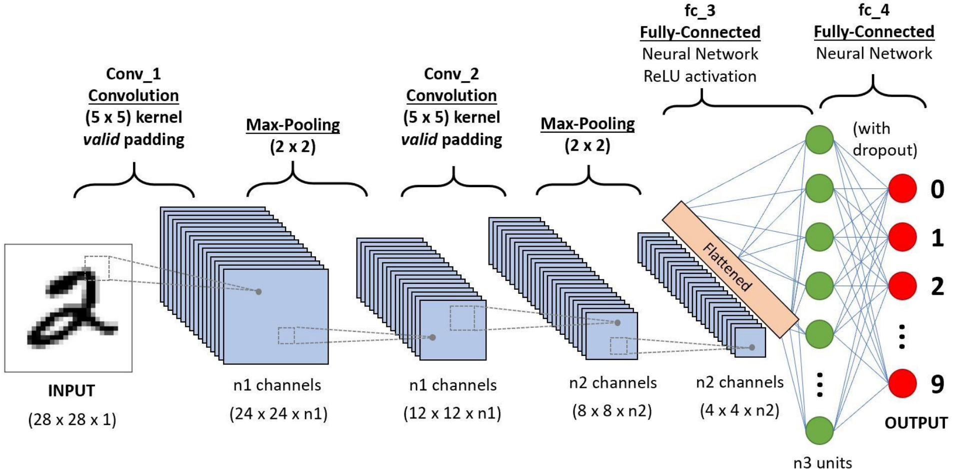




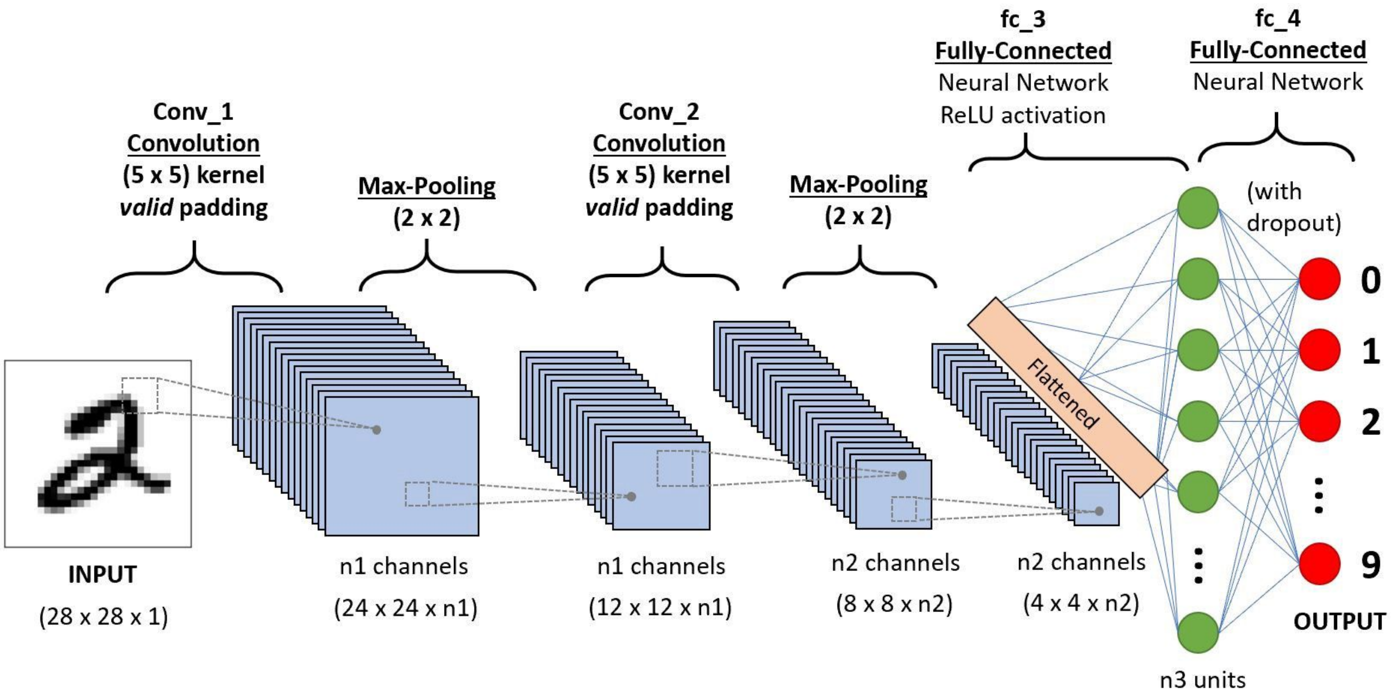
Convolutional Neural Network

- Convolution
  - locally-connected
  - spatial weight sharing (kernel)
- Pooling
- Fully-connected

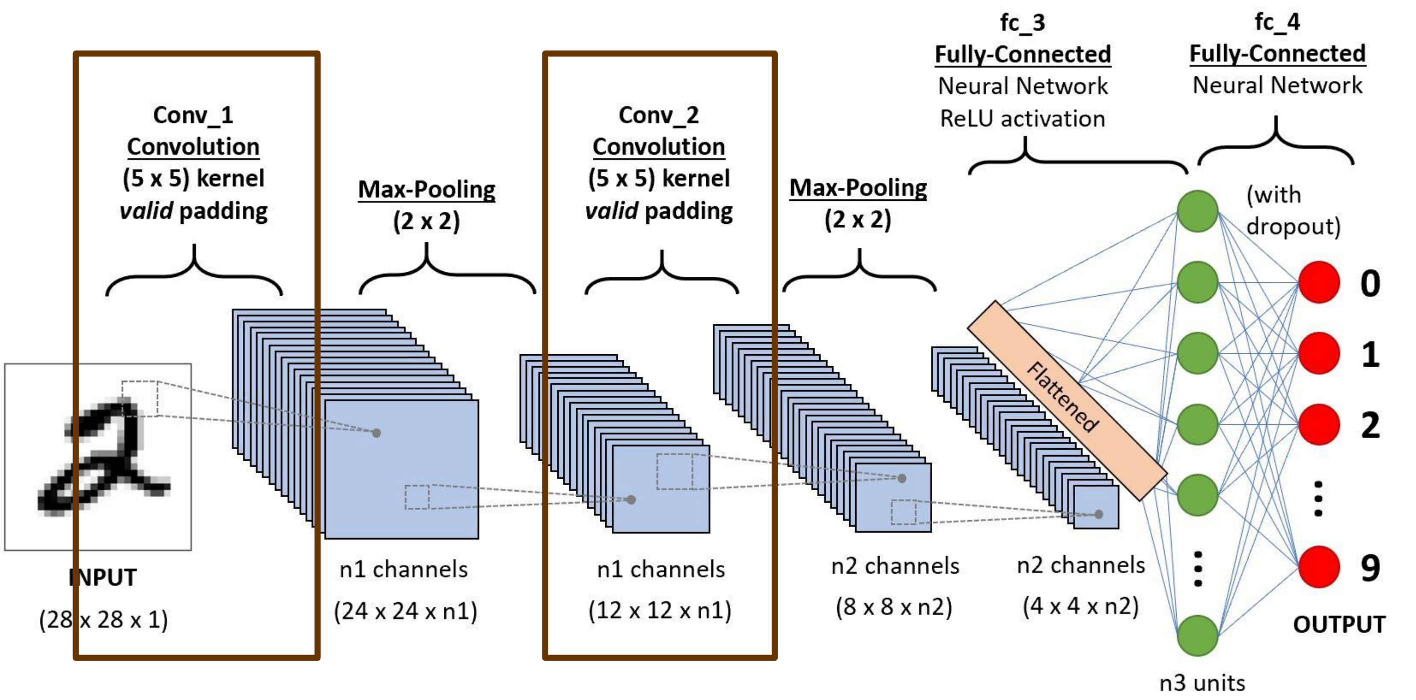




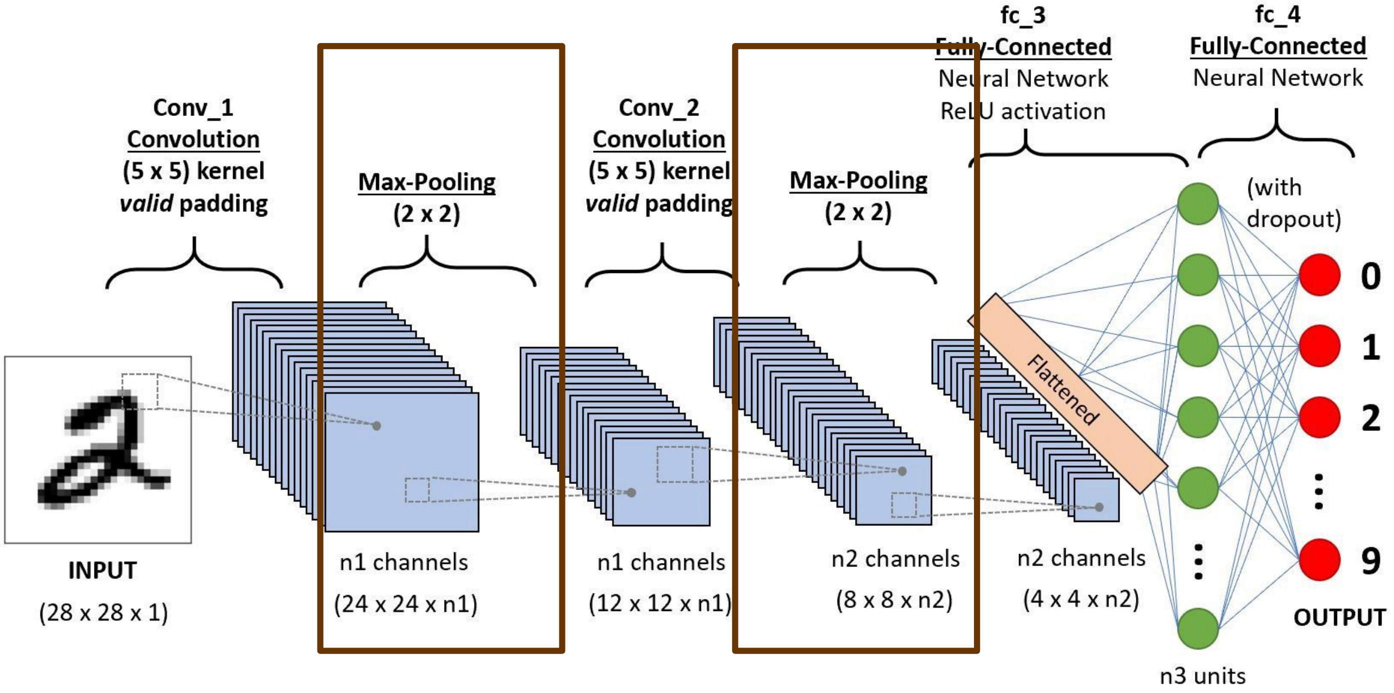
Convolutional Neural Network



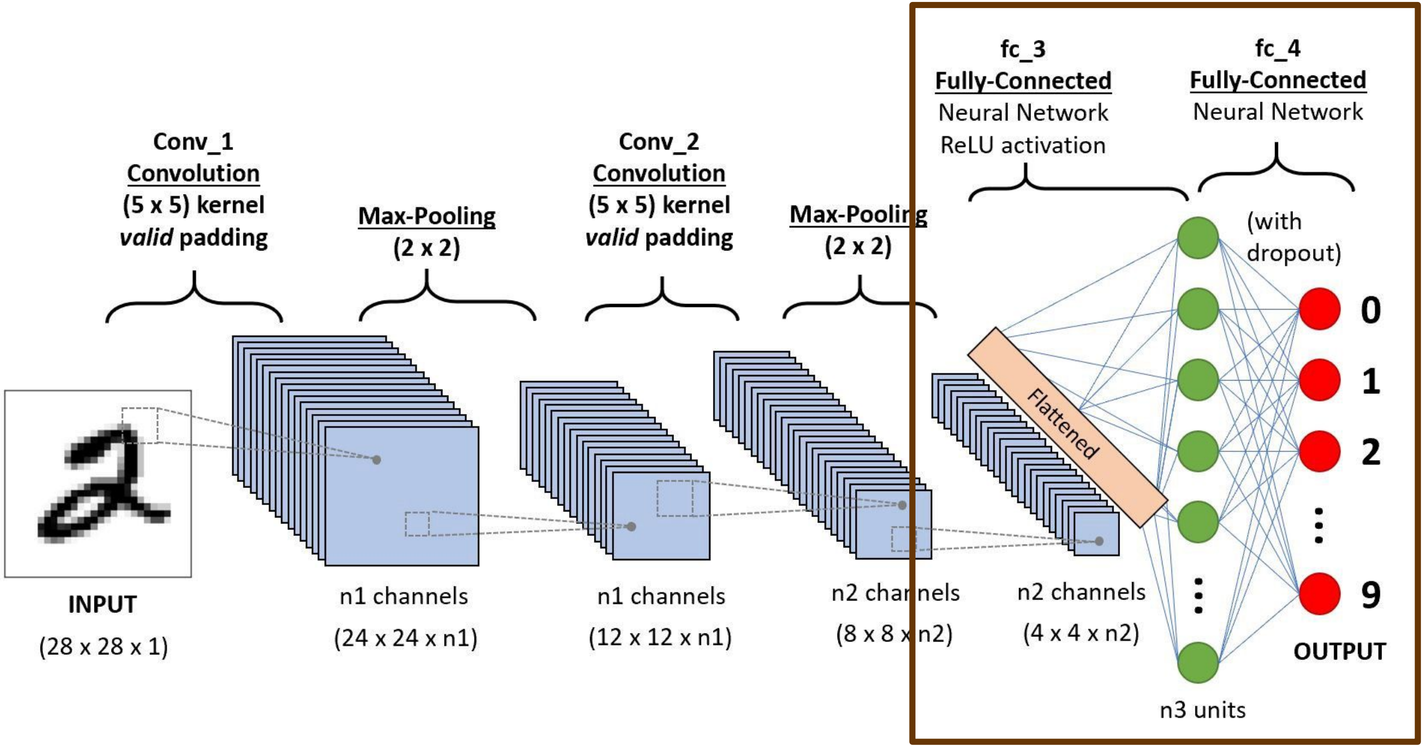
Convolutional Neural Network



Convolutional Neural Network



Convolutional Neural Network



Vision Transformer (ViT)

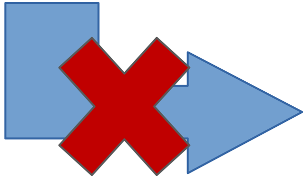
Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

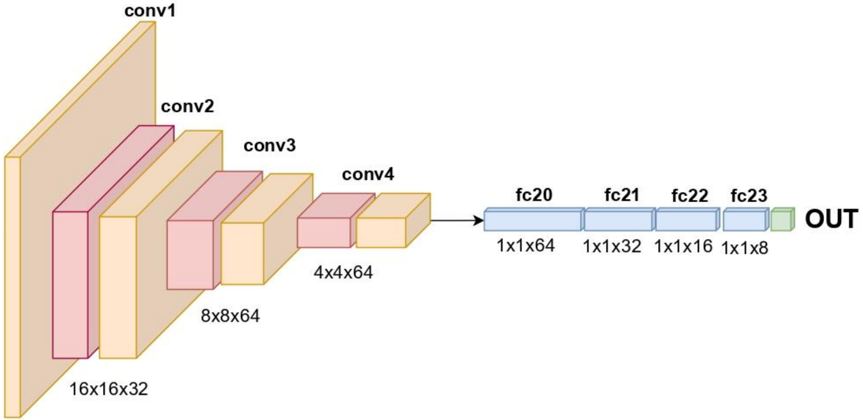
Tabular data + CNN

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15

Tabular data + CNN

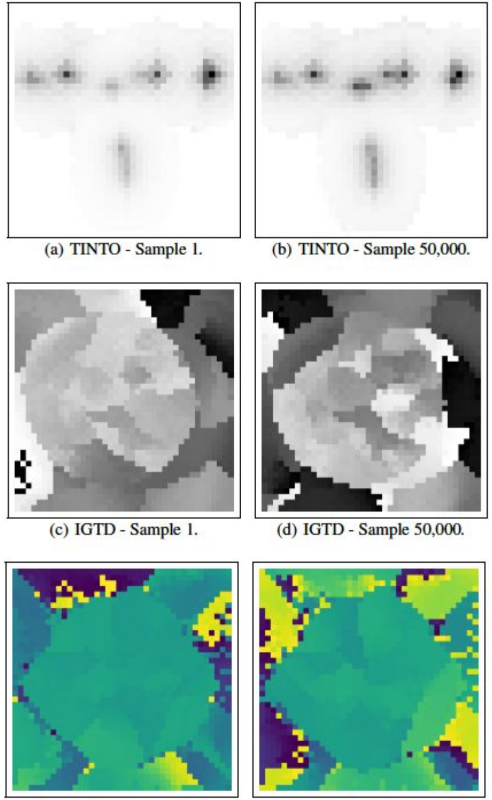
Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15





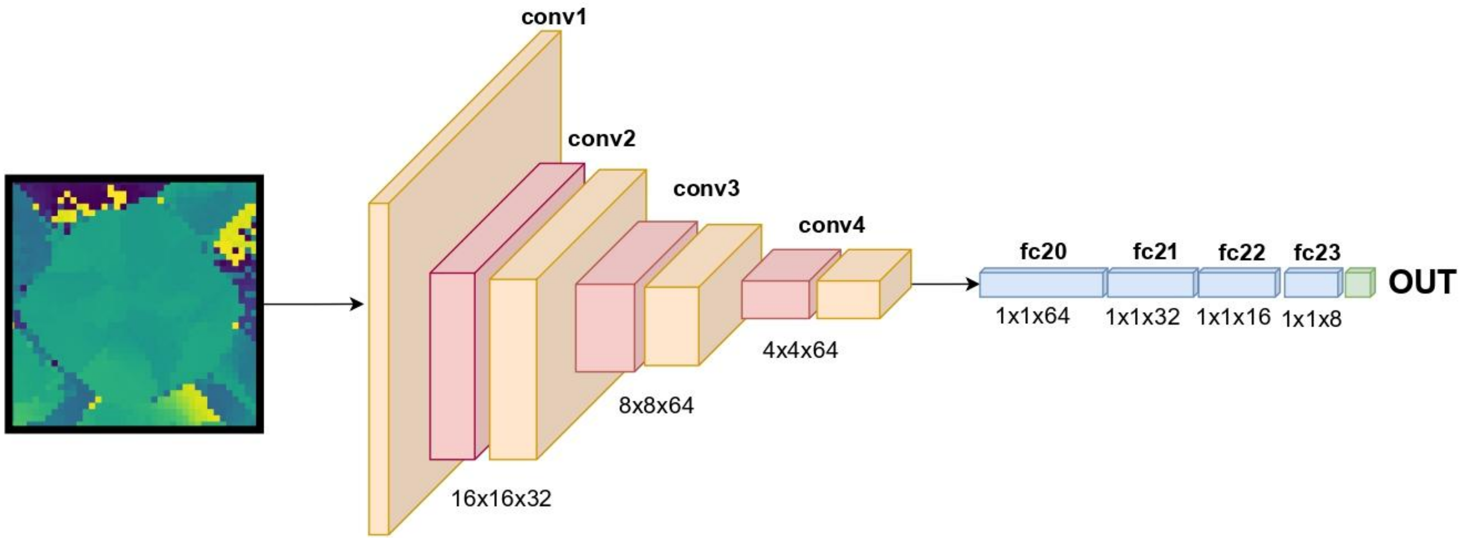
Synthetic images

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15



(a) TINTO - Sample 1. (b) TINTO - Sample 50,000.
(c) IGTD - Sample 1. (d) IGTD - Sample 50,000.
(e) REFINED - Sample 1. (f) REFINED - Sample 50,000.

Tabular data + CNN



Index

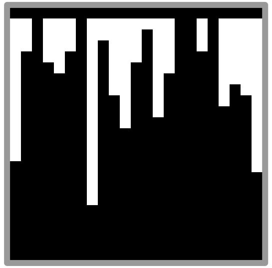
1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Non-parametric

Bar Graph, DistanceMatrix and Combination

Equidistant Bar Graph:

- The values of the normalized variables, scaled between [0,1], are represented in a bar graph.
- The resulting images are provided in a single channel (black and white).

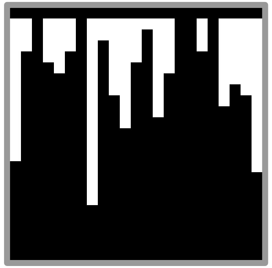


Sharma, A., & Kumar, D. (2022). Classification with 2-D Convolutional Neural Networks for breast cancer diagnosis. Scientific Reports, 12(1), 21857.

Bar Graph, DistanceMatrix and Combination

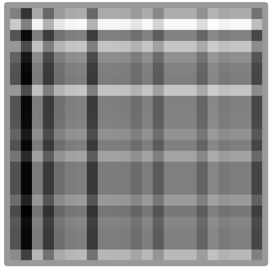
Equidistant Bar Graph:

- The values of the normalized variables, scaled between [0,1], are represented in a bar graph.
- The resulting images are provided in a single channel (black and white).



Normalized Distance Matrix:

- Represents a distance matrix of all the normalized variables scaled between [0,1].
- The values are displayed in grayscale.
- The resulting images are provided in a single channel (black and white).



Sharma, A., & Kumar, D. (2022). Classification with 2-D Convolutional Neural Networks for breast cancer diagnosis. Scientific Reports, 12(1), 21857.

Bar Graph, DistanceMatrix and Combination

Equidistant Bar Graph:

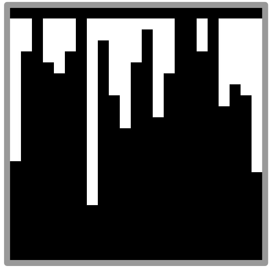
- The values of the normalized variables, scaled between [0,1], are represented in a bar graph.
- The resulting images are provided in a single channel (black and white).

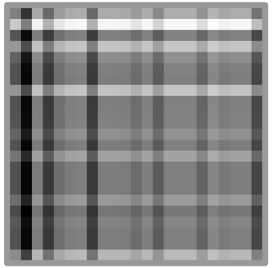
Normalized Distance Matrix:

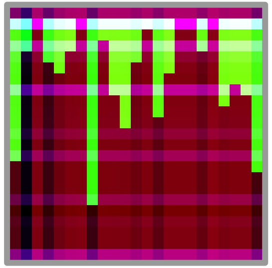
- Represents a distance matrix of all the normalized variables scaled between [0,1].
- The values are displayed in grayscale.
- The resulting images are provided in a single channel (black and white).

Combination of options:

- This method is a combination of the two previous ones.
- The resulting images are provided in 3 channels (RGB).







Sharma, A., & Kumar, D. (2022). Classification with 2-D Convolutional Neural Networks for breast cancer diagnosis. Scientific Reports, 12(1), 21857.

Bar Graph, DistanceMatrix and Combination

Equidistant Bar Graph:

- The values of the normalized variables, scaled between [0,1], are represented in a bar graph.
- The resulting images are provided in a single channel (black and white).

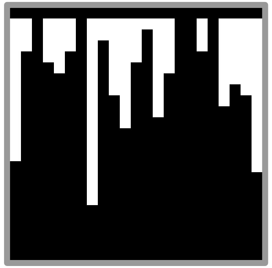
Normalized Distance Matrix:

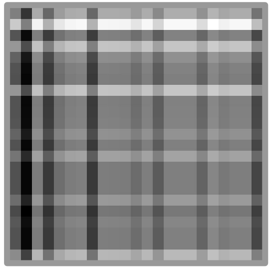
- Represents a distance matrix of all the normalized variables scaled between [0,1].
- The values are displayed in grayscale.
- The resulting images are provided in a single channel (black and white).

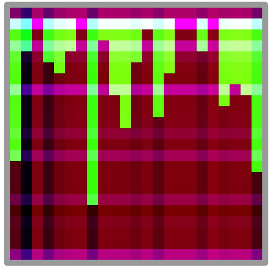
Combination of options:

- This method is a combination of the two previous ones.
- The resulting images are provided in 3 channels (RGB).

They perform very simple calculations with a very low computational cost.
However, they do not take into account the spatial distribution of the variables.







Sharma, A., & Kumar, D. (2022). Classification with 2-D Convolutional Neural Networks for breast cancer diagnosis. Scientific Reports, 12(1), 21857.

SuperTML

- The data values are directly drawn onto the image in a single channel (black and white). A region of the image and a size are assigned to represent each feature in the form of text or a written number.
- There are two approaches:
  - SuperTML_EF: In this approach, each feature is given a region and font size of the same size."

(b) SuperTML_EF

Sun, B., Yang, L., Zhang, W., Lin, M., Dong, P., Young, C., & Dong, J. (2019). Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops

SuperTML

- The data values are directly drawn onto the image in a single channel (black and white). A region of the image and a size are assigned to represent each feature in the form of text or a written number.
- There are two approaches:
    - SuperTML_EF: In this approach, each feature is given a region and font size of the same size."
    - SuperTML_VF: Each feature is given a region and font size based on its relevance. The more important the feature, the larger its font size and region.

(b) SuperTML_EF

(a) SuperTML_VF

Sun, B., Yang, L., Zhang, W., Lin, M., Dong, P., Young, C., & Dong, J. (2019). Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops

SuperTML

- The data values are directly drawn onto the image in a single channel (black and white). A region of the image and a size are assigned to represent each feature in the form of text or a written number.
- There are two approaches:
  - SuperTML_EF: In this approach, each feature is given a region and font size of the same size."
  - SuperTML_VF: Each feature is given a region and font size based on its relevance. The more important the feature, the larger its font size and region.

- The method does not take into account the spatial distribution of the variables, beyond the variation in size and font in SuperTML_VF.
- Another drawback is the sensitivity to small variations in the variables.
  - For example, 3.9999 and 4.0 are numerically almost identical.

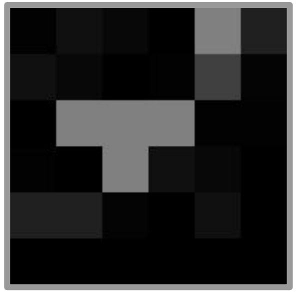
(b) SuperTML_EF

(a) SuperTML_VF

Sun, B., Yang, L., Zhang, W., Lin, M., Dong, P., Young, C., & Dong, J. (2019). Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops

FeatureWrap

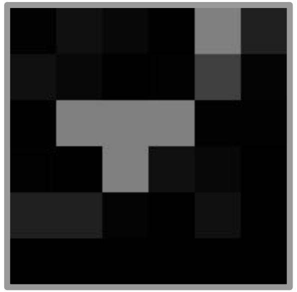
- FeatureWrap transforms each feature into a binary vector using binning and one-hot encoding, mapping groups of 8 bits to grayscale pixel values to represent feature information.
- Feature to Binary Vector Transformation:
  - Categorical Data: Uses one-hot encoding. Each unique category is represented by a 1D vector.
  - Numerical Data: Normalized using min-max scaling to the range [0, 1]. Discretized into groups according to the number of bins. Transformed into one-hot encoded vectors.



Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion detection using convolutional neural networks for representation learning,” in International conference on neural information processing. Springer, 2017, pp. 858–866.

FeatureWrap

- FeatureWrap transforms each feature into a binary vector using binning and one-hot encoding, mapping groups of 8 bits to grayscale pixel values to represent feature information.
- Feature to Binary Vector Transformation:
  - Categorical Data: Uses one-hot encoding. Each unique category is represented by a 1D vector.
  - Numerical Data: Normalized using min-max scaling to the range [0, 1]. Discretized into groups according to the number of bins. Transformed into one-hot encoded vectors.



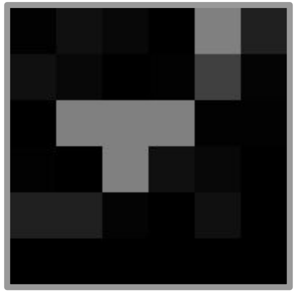
Age (Num)	Gender (Cat)	Salary (Num)
25	Male	30000

→

Age (Bin)	Gender (One-Hot)	Salary (Bin)
[1, 0, 0, 0, 0]	[1, 0]	[1, 0, 0, 0, 0]

Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in International conference on neural information processing. Springer, 2017, pp. 858–866.

FeatureWrap



- FeatureWrap transforms each feature into a binary vector using binning and one-hot encoding, mapping groups of 8 bits to grayscale pixel values to represent feature information.
- Feature to Binary Vector Transformation:
  - Categorical Data: Uses one-hot encoding. Each unique category is represented by a 1D vector.
  - Numerical Data: Normalized using min-max scaling to the range [0, 1]. Discretized into groups according to the number of bins. Transformed into one-hot encoded vectors.
  - Concatenates binary vectors into a single vector and pads the vector with zeros if necessary to reach the required image length.
  - Group the bits into bytes and converts them into numerical values between 0-255.

Age (Num)	Gender (Cat)	Salary (Num)
25	Male	30000

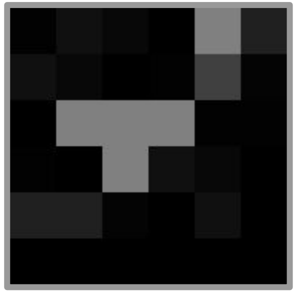
→

Age (Bin)	Gender (One-Hot)	Salary (Bin)
[1, 0, 0, 0, 0]	[1, 0]	[1, 0, 0, 0, 0]

→ **Concatenated Binary Vector**: [1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0] + [0,0,0…]

Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion detection using convolutional neural networks for representation learning,” in International conference on neural information processing. Springer, 2017, pp. 858–866.

FeatureWrap



- FeatureWrap transforms each feature into a binary vector using binning and one-hot encoding, mapping groups of 8 bits to grayscale pixel values to represent feature information.
- Feature to Binary Vector Transformation:
  - Categorical Data: Uses one-hot encoding. Each unique category is represented by a 1D vector.
  - Numerical Data: Normalized using min-max scaling to the range [0, 1]. Discretized into groups according to the number of bins. Transformed into one-hot encoded vectors.
  - Concatenates binary vectors into a single vector and pads the vector with zeros if necessary to reach the required image length.
  - Group the bits into bytes and converts them into numerical values between 0-255.

Age (Num)	Gender (Cat)	Salary (Num)
25	Male	30000

→

Age (Bin)	Gender (One-Hot)	Salary (Bin)
[1, 0, 0, 0, 0]	[1, 0]	[1, 0, 0, 0, 0]

→ **Concatenated Binary Vector**: [1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0] + [0,0,0…]

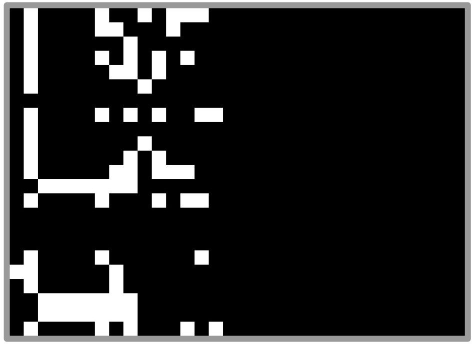
Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion detection using convolutional neural networks for representation learning,” in International conference on neural information processing. Springer, 2017, pp. 858–866.

Binary Image Encoding

The **Binary Image Encoding (BIE)** method takes numerical data and converts it into its **floating-point binary representation**.

There are two main steps to this method:

- First, we convert the numeric values into their **floating-point binary representation**. This binary format consists of three key components: the **sign bit**, which indicates whether the number is positive or negative, the **exponent**, which shows the position of the decimal point, and the **mantissa**, which stores the significant digits of the number.



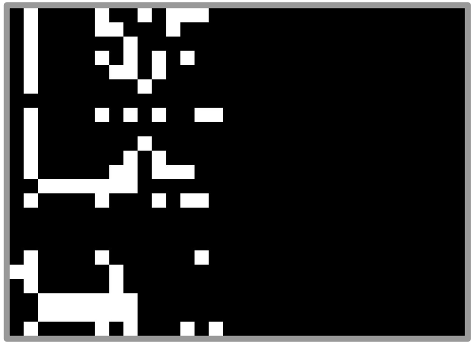
N. Briner, D. Cullen, J. Halladay, D. Miller, R. Primeau, A. Avila, R. Basnet, and T. Doleck, “Tabular-to-image transformations for the classification of anonymous network traffic using deep residual networks,” IEEE Access, vol. 11, pp.113 100–113 113, 2023.

Binary Image Encoding

The **Binary Image Encoding (BIE)** method takes numerical data and converts it into its **floating-point binary representation**.

There are two main steps to this method:

- First, we convert the numeric values into their **floating-point binary representation**. This binary format consists of three key components: the **sign bit**, which indicates whether the number is positive or negative, the **exponent**, which shows the position of the decimal point, and the **mantissa**, which stores the significant digits of the number.



Age: 25

- Sign bit: **0** (since it's positive)
- Exponent: **10000011**
- Mantissa: **10010000000000000000000**
- Binary representation: **0 10000011 10010000000000000000000**

Age	Salary
25	30000

N. Briner, D. Cullen, J. Halladay, D. Miller, R. Primeau, A. Avila, R. Basnet, and T. Doleck, “Tabular-to-image transformations for the classification of anonymous network traffic using deep residual networks,” IEEE Access, vol. 11, pp.113 100–113 113, 2023.

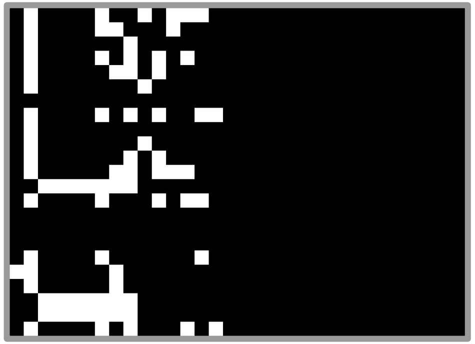
Binary Image Encoding

The **Binary Image Encoding (BIE)** method takes numerical data and converts it into its **floating-point binary representation**.

There are two main steps to this method:

- First, we convert the numeric values into their **floating-point binary representation**. This binary format consists of three key components: the **sign bit**, which indicates whether the number is positive or negative, the **exponent**, which shows the position of the decimal point, and the **mantissa**, which stores the significant digits of the number.
- Once we have the floating-point encoding, each bit is used to generate the image. A bit value of 0 becomes **black** (0), and a bit value of 1 becomes **white** (255). This is done bit by bit, creating the final black-and-white image.

Age	Salary
25	30000



Age: 25

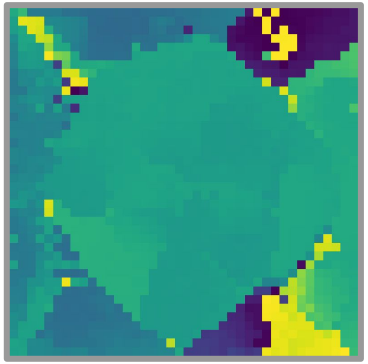
- Sign bit: **0** (since it's positive)
- Exponent: **10000011**
- Mantissa: **10010000000000000000000**
- Binary representation: **0 10000011 10010000000000000000000**

N. Briner, D. Cullen, J. Halladay, D. Miller, R. Primeau, A. Avila, R. Basnet, and T. Doleck, “Tabular-to-image transformations for the classification of anonymous network traffic using deep residual networks,” IEEE Access, vol. 11, pp.113 100–113 113, 2023.

Parametric

REFINED

REFINED stands for REpresentation of Features as Images with NEighborhood Dependencies, it transforms tabular data into images by **preserving the original distances** between features from the high-dimensional space. It uses a **hill climbing** algorithm for optimization.



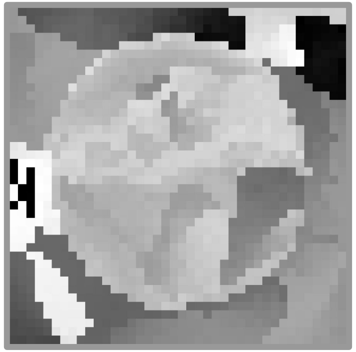
Steps:

1. **Dimensionality Reduction**:
    - Uses **Multidimensional Scaling (MDS)** to project the features into a **2D space**, preserving distances between features.
2. **Initial Feature Assignment**:
    - Places features on a grid based on their 2D coordinates using **Bayesian variation of MDS (BMDS)** , but this initial layout may not guarantee that each feature is mapped perfectly. This can result in a **dispersed image**.
3. **Optimization with Hill Climbing**:
    - Iteratively swaps the positions of features on the grid using Hill Climbing to minimize the difference between their original distances (euclidean distance) and the distances in the grid, optimizing the arrangement.

O. Bazgir *et al*-, “Representation of features as images with neighborhood dependencias for compatibility with convolutional neural networks,” Nature Communications, vol. 11, 2020.

IGTD

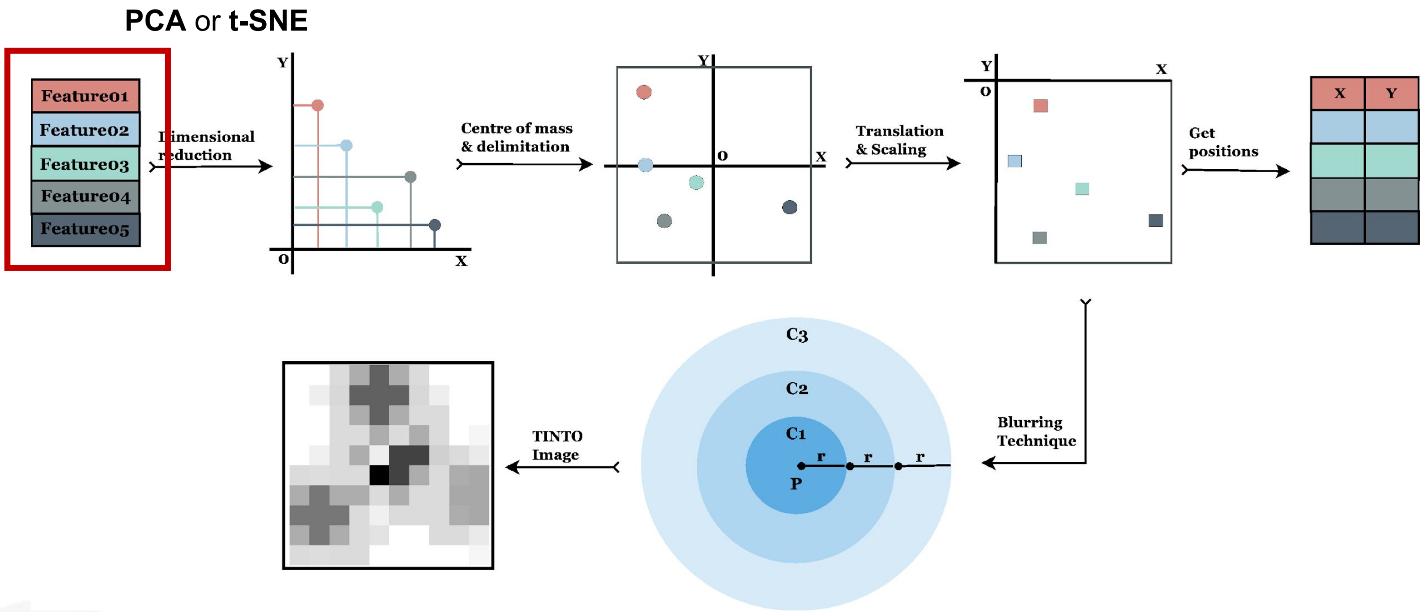
IGTD transforms tabular data into images by arranging variables in a way that **similar features** are placed close to each other on the image, and dissimilar ones are placed farther apart.



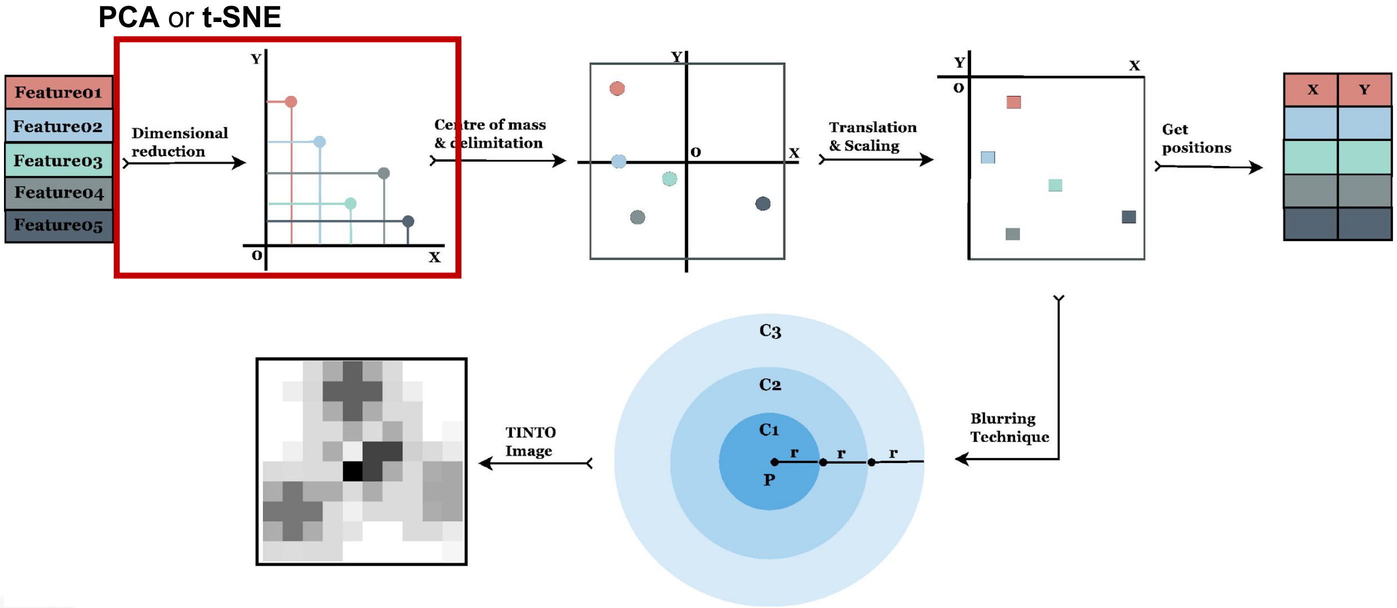
Steps:

1. **Generate the Similarity Matrix**:
   - Calculates the similarity between features using methods like **Pearson**, **Spearman**, or **Euclidean distance**.
2. **Generate the Distance Matrix**:
   - Computes the distance between pixels in the image, based on their positions in the 2D grid.
3. **Reorganize the Data**:
   - Reorders the features so that the arrangement of features in the image reflects their similarities, minimizing the difference between the **similarity matrix** and the **distance matrix** using an optimization process. In each iteration, the pixel that has gone the longest without being swapped is permuted with the pixel that minimizes the difference between the rankings.

Zhu, Y., Brettin, T., Xia, F. et al. Converting tabular data into images for deep learning with convolutional neural networks. Sci Rep 11, 11325 (2021). https://doi.org/10.1038/s41598-021-90923-y

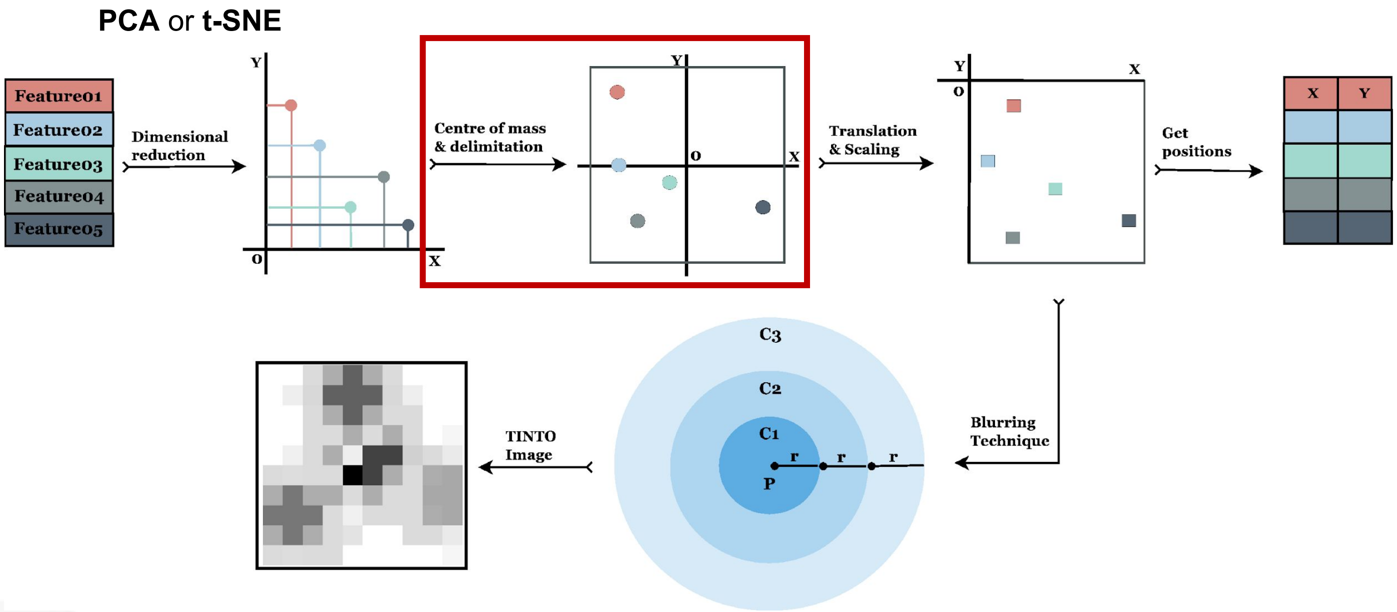


1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, *91*, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional Neural Networks. *SoftwareX*, *22*, 101391.
3. A. Sharma, et al. “Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” Scientific Reports, vol. 9, 2019.



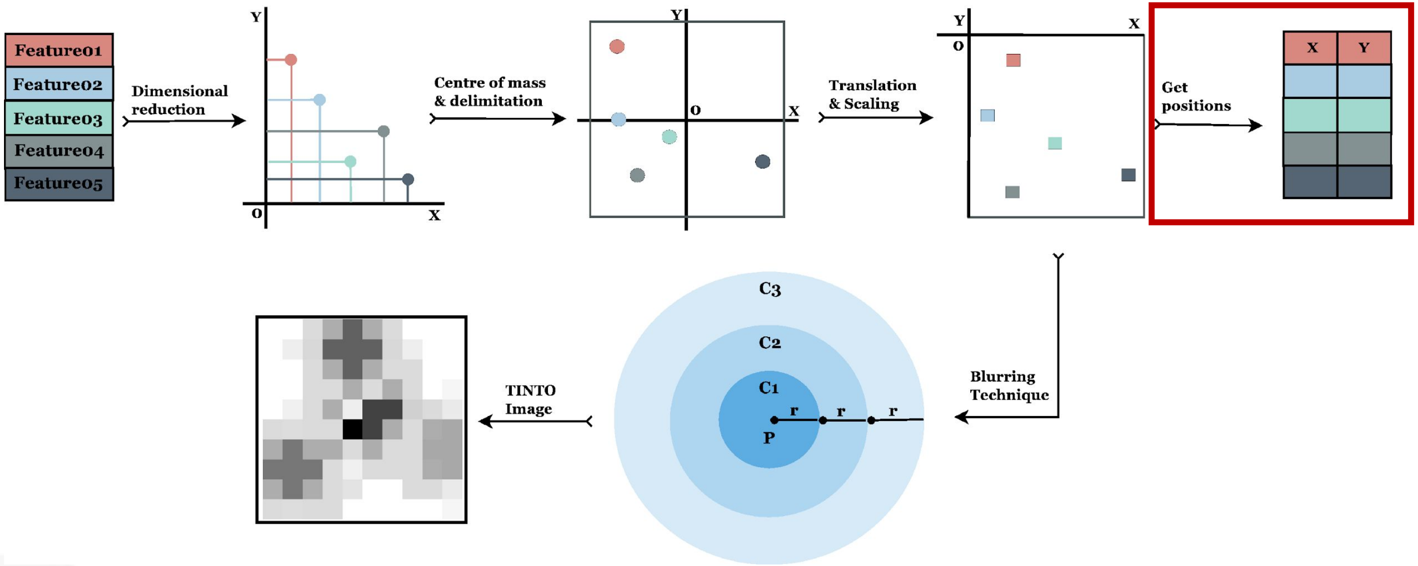
1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, *91*, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional Neural Networks. *SoftwareX*, *22*, 101391.
3. A. Sharma, et al. “Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” Scientific Reports, vol. 9, 2019.

TINTO



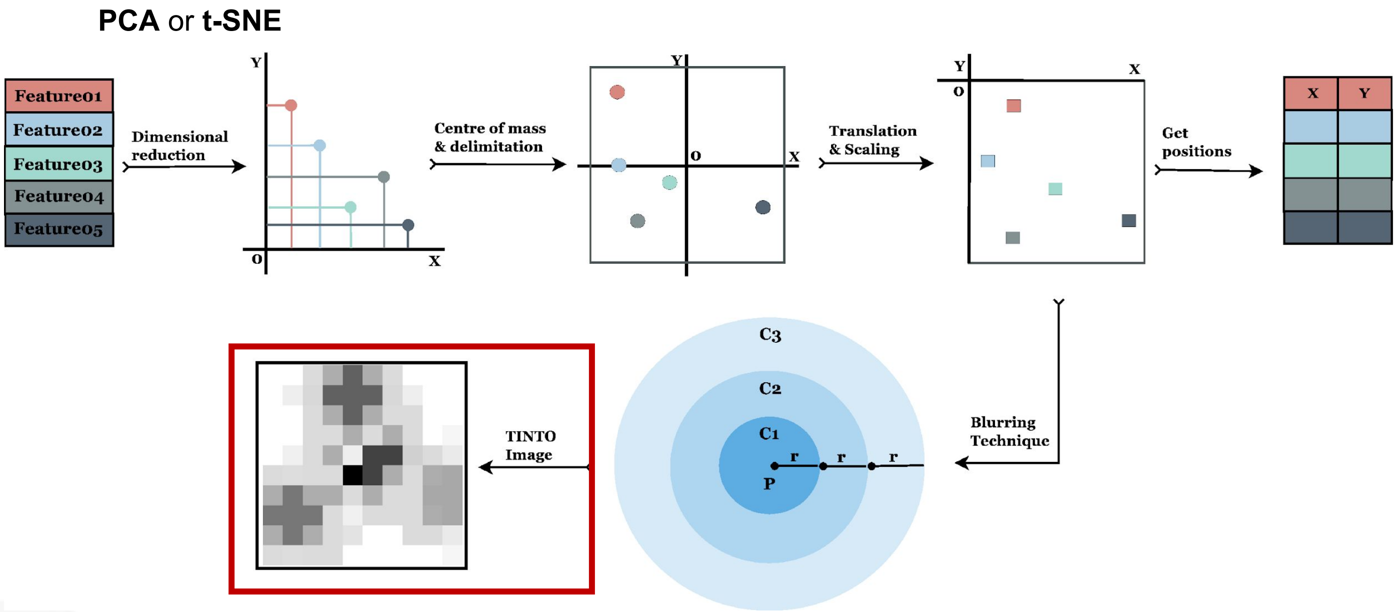
1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, *91*, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional Neural Networks. *SoftwareX*, *22*, 101391.
3. A. Sharma, et al. “Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” Scientific Reports, vol. 9, 2019.

PCA or **t-SNE**



1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, *91*, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional Neural Networks. *SoftwareX*, *22*, 101391.
3. A. Sharma, et al. "Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture," Scientific Reports, vol. 9, 2019.

TINTO



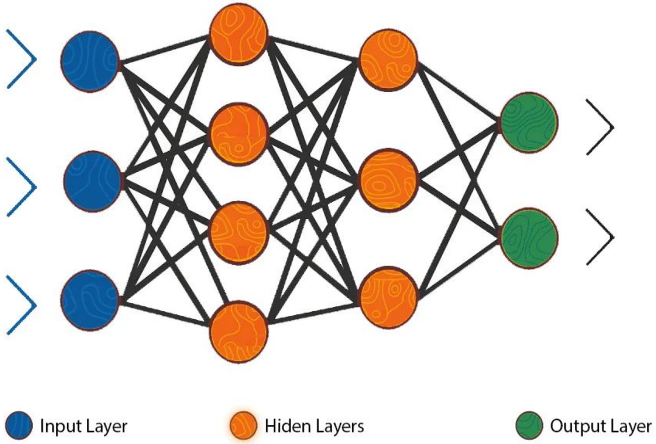
1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, *91*, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional Neural Networks. *SoftwareX*, *22*, 101391.
3. A. Sharma, et al. “Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” Scientific Reports, vol. 9, 2019.

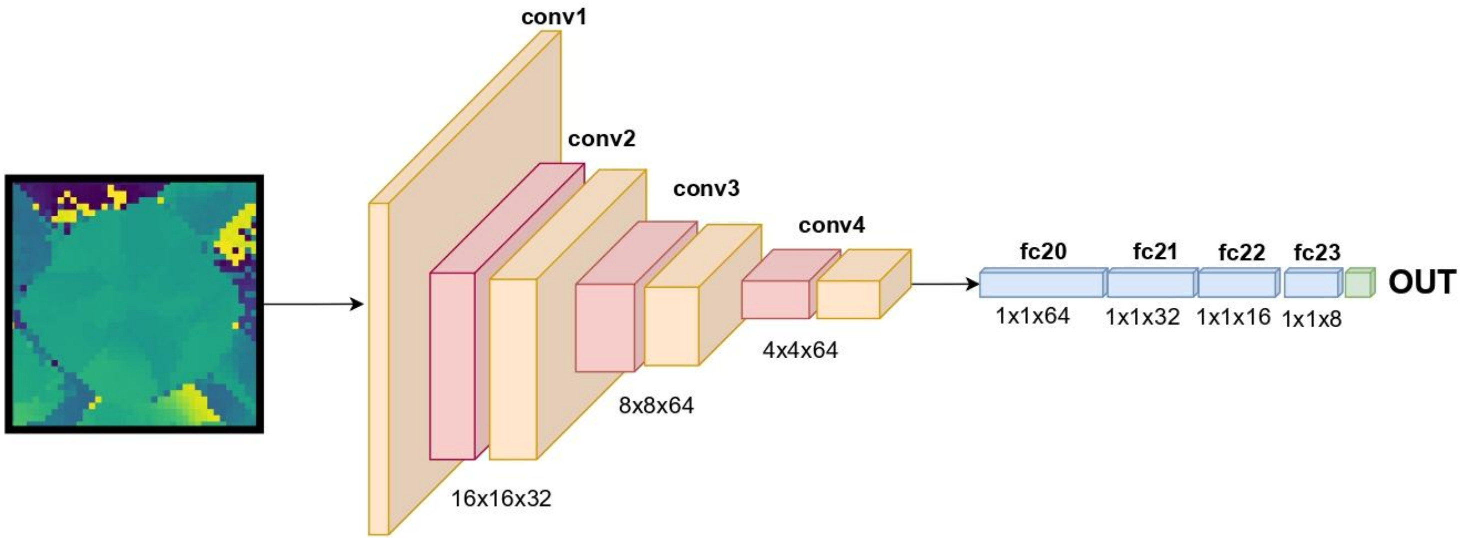
Index

1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

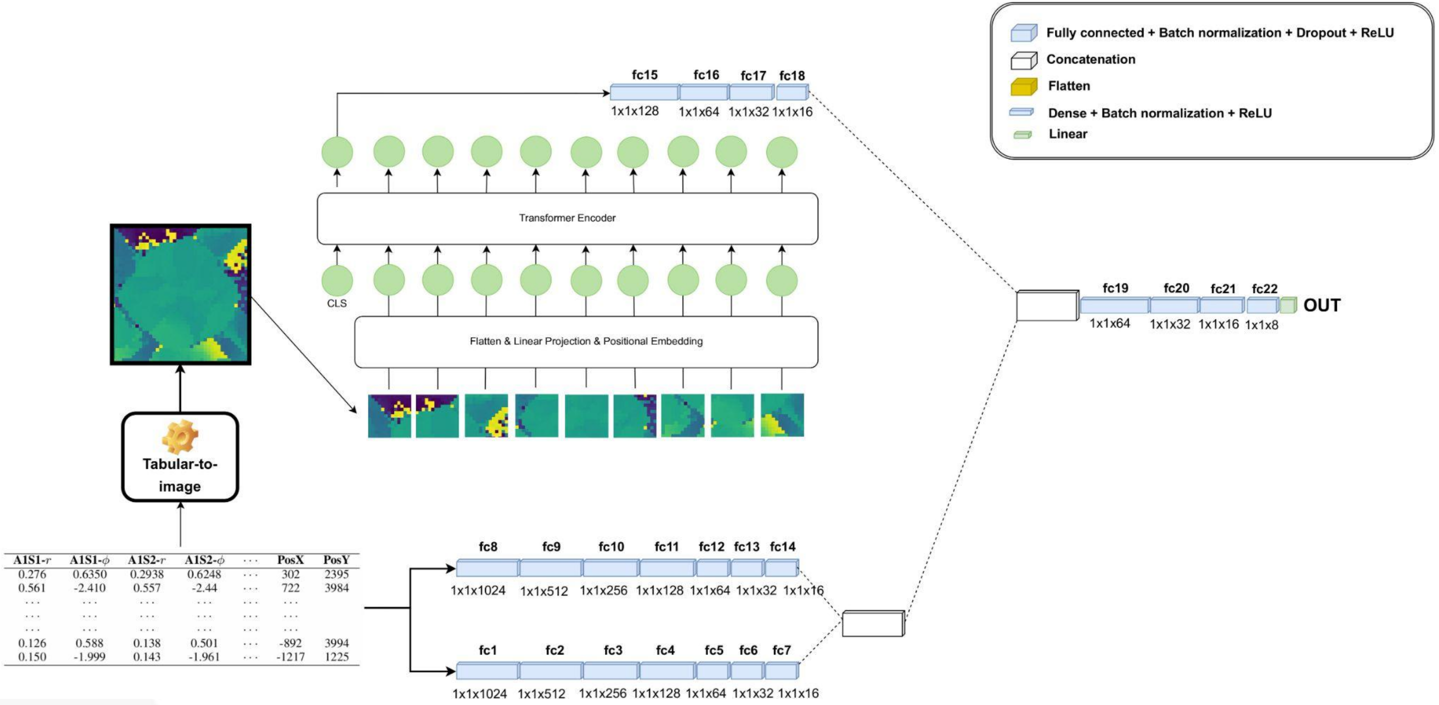
Tabular data + Deep Learning

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15




conv1
conv2
conv3
conv4
16x16x32
8x8x64
4x4x64
fc20
fc21
fc22
fc23
OUT
1x1x64
1x1x32
1x1x16
1x1x8

Hybrid Model



A1S1-r	A1S1-ϕ	A1S2-r	A1S2-ϕ	$\cdots$	PosX	PosY
0.276	0.6350	0.2938	0.6248	$\cdots$	302	2395
0.561	-2.410	0.557	-2.44	$\cdots$	722	3984
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
0.126	0.588	0.138	0.501	$\cdots$	-892	3994
0.150	-1.999	0.143	-1.961	$\cdots$	-1217	1225

Index

1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Example use case

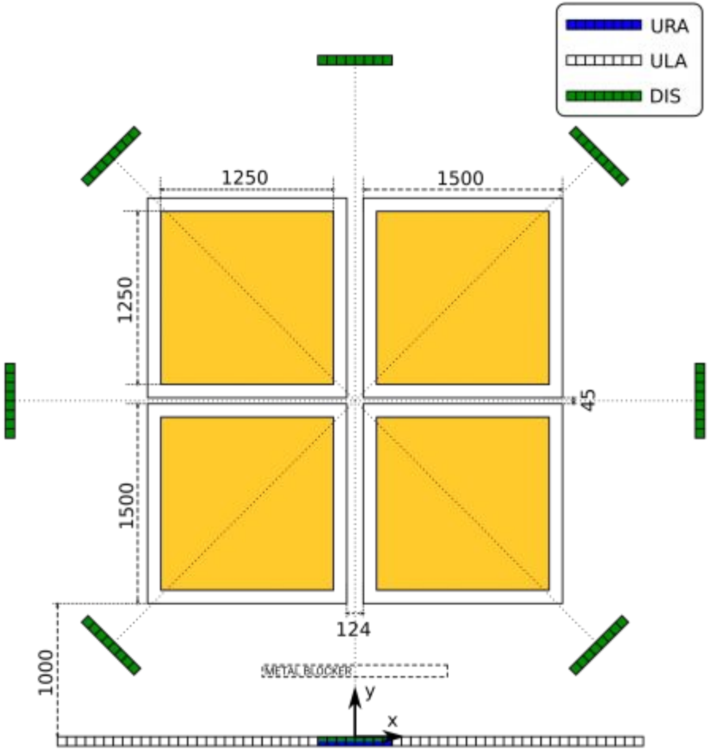


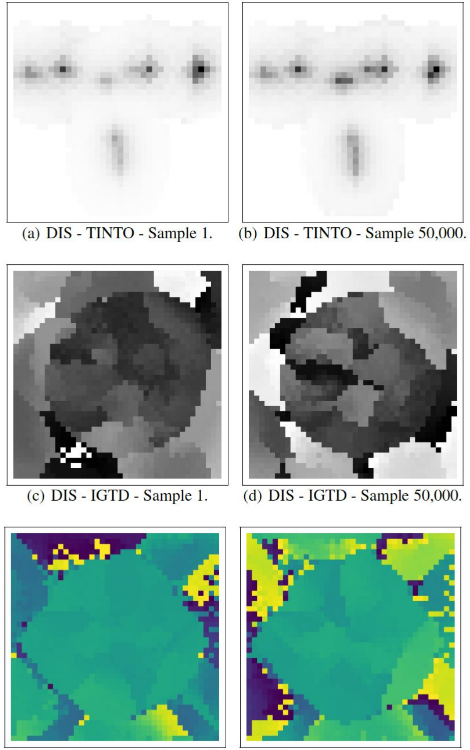
Figure taken from the paper: S. D. Bast, A. P. Guevara and S. Pollin, "CSI-based Positioning in Massive MIMO systems using Convolutional Neural Networks," 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 2020, pp. 1-5

Example use case

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15

Example use case

Be07	Be08	Be09	Be10	Be11	Sector
−65	−61	−74	−73	−67	1
−60	−57	−83	−62	−69	2
−66	−70	−78	−63	−73	3
...	...	...	...	...	...
−58	−66	−71	−73	−69	14
−60	−62	−73	−69	−57	15



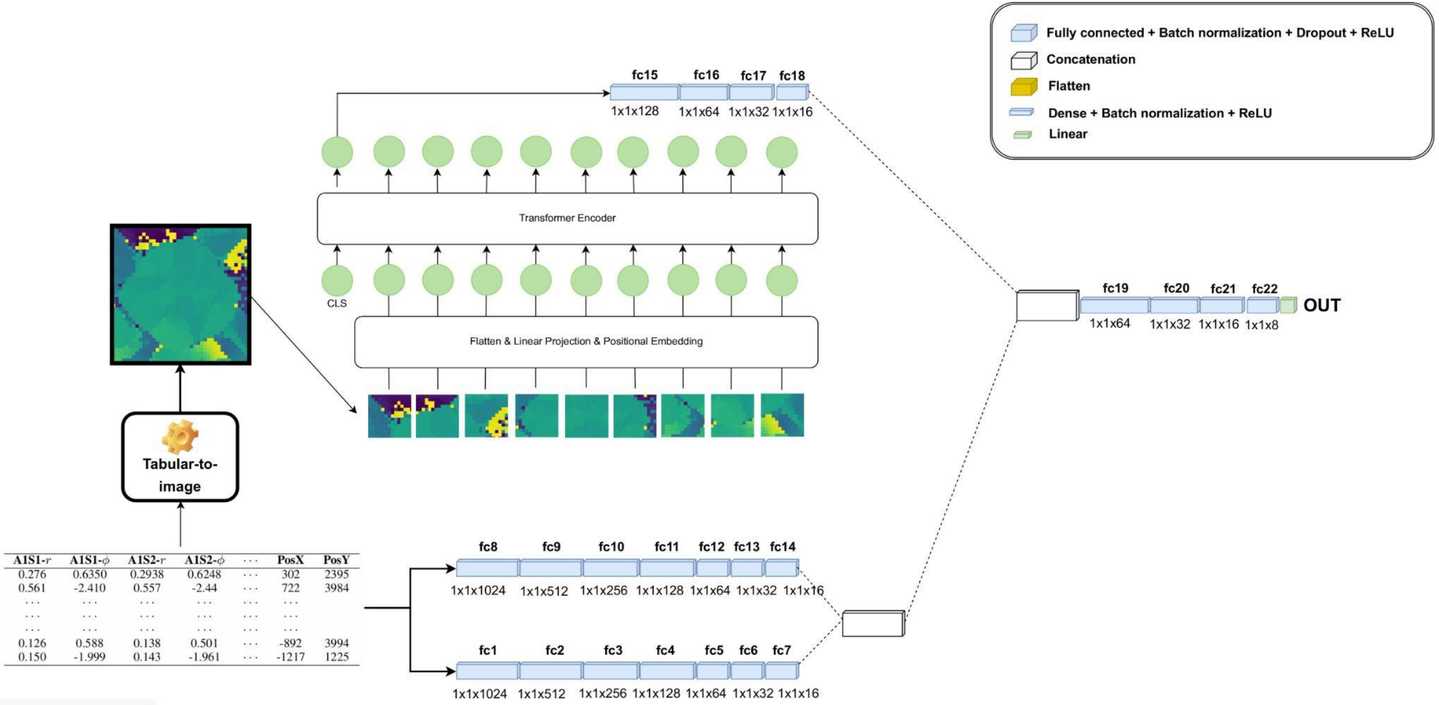
(a) DIS - TINTO - Sample 1. (b) DIS - TINTO - Sample 50,000.

(c) DIS - IGTD - Sample 1. (d) DIS - IGTD - Sample 50,000.

(e) DIS - REFINED - Sample 1. (f) DIS - REFINED - Sample 50,000.

Figure 3. Synthetic image samples generated by TINTOlib for different samples in 8 antennas DIS scenario.

Hybrid Model



A1S1-r	A1S1-ϕ	A1S2-r	A1S2-ϕ	···	PosX	PosY
0.276	0.6350	0.2938	0.6248	···	302	2395
0.561	-2.410	0.557	-2.44	···	722	3984
···	···	···	···	···	···	
···	···	···	···	···	···	
···	···	···	···	···	···	
0.126	0.588	0.138	0.501	···	-892	3994
0.150	-1.999	0.143	-1.961	···	-1217	1225

Example use case

Table 2. RMSE (in mm) in validation (Val) and test split. Best results are shown in bold.

Algorithm	PosX		PosY	
	Val	Test	Val	Test
BR	226.05	225.00	251.43	255.54
ET	163.15	161.65	180.00	185.70
HGB	194.10	194.97	236.55	236.46
KNN	**110.50**	**110.54**	**133.70**	**140.16**
LiR	383.05	386.95	432.83	439.10
MLP	179.80	178.82	326.11	334.76
RF	226.09	225.18	251.37	255.62
RCV	383.04	386.94	432.80	439.06
XGB	178.41	180.03	202.45	201.66
LGB	194.14	194.15	231.19	232.89

Example use case

Table 2. RMSE (in mm) in validation (Val) and test split. Best results are shown in bold.

Algorithm	PosX		PosY	
	Val	Test	Val	Test
BR	226.05	225.00	251.43	255.54
ET	163.15	161.65	180.00	185.70
HGB	194.10	194.97	236.55	236.46
KNN	**110.50**	**110.54**	**133.70**	**140.16**
LiR	383.05	386.95	432.83	439.10
MLP	179.80	178.82	326.11	334.76
RF	226.09	225.18	251.37	255.62
RCV	383.04	386.94	432.80	439.06
XGB	178.41	180.03	202.45	201.66
LGB	194.14	194.15	231.19	232.89

Table 3. RMSE (in mm) for the different HyNNs architectures and HyViT in Validation (Val) and test. Best results are shown in bold.

Position	Model	TINTO		IGTD		REFINED	
		Val	Test	Val	Test	Val	Test
PosX	HyCNN	187.10	188.10	92.8	92.21	105.69	105.38
	HyTNN	178.28	179.25	119.59	119.62	115.90	114.98
	HyTTNN	181.96	184.19	179.01	180.05	193.56	196.09
	HyGTNN	176.71	176.43	173.42	174.20	173.38	174.02
	HyViT	**103.27**	**104.17**	**46.57**	**45.77**	**41.38**	**41.84**
PosY	HyCNN	152.19	151.94	101.01	99.45	115.40	114.69
	HyTNN	143.10	143.29	95.95	95.83	112.27	112.02
	HyTTNN	151.35	151.97	155.35	154.12	147.22	146.01
	HyGTNN	155.06	153.40	154.68	154.50	157.10	155.39
	HyViT	**121.77**	**123.90**	**70.84**	**68.93**	**90.11**	**90.56**

Example use case

Table 2. RMSE (in mm) in validation (Val) and test split. Best results are shown in bold.

Algorithm	PosX		PosY	
	Val	Test	Val	Test
BR	226.05	225.00	251.43	255.54
ET	163.15	161.65	180.00	185.70
HGB	194.10	194.97	236.55	236.46
KNN	**110.50**	**110.54**	**133.70**	**140.16**
LiR	383.05	386.95	432.83	439.10
MLP	179.80	178.82	326.11	334.76
RF	226.09	225.18	251.37	255.62
RCV	383.04	386.94	432.80	439.06
XGB	178.41	180.03	202.45	201.66
LGB	194.14	194.15	231.19	232.89

Table 3. RMSE (in mm) for the different HyNNs architectures and HyViT in Validation (Val) and test. Best results are shown in bold.

Position	Model	TINTO		IGTD		REFINED	
		Val	Test	Val	Test	Val	Test
PosX	HyCNN	187.10	188.10	92.8	92.21	105.69	105.38
	HyTNN	178.28	179.25	119.59	119.62	115.90	114.98
	HyTTNN	181.96	184.19	179.01	180.05	193.56	196.09
	HyGTNN	176.71	176.43	173.42	174.20	173.38	174.02
	HyViT	**103.27**	**104.17**	**46.57**	**45.77**	**41.38**	**41.84**
PosY	HyCNN	152.19	151.94	101.01	99.45	115.40	114.69
	HyTNN	143.10	143.29	95.95	95.83	112.27	112.02
	HyTTNN	151.35	151.97	155.35	154.12	147.22	146.01
	HyGTNN	155.06	153.40	154.68	154.50	157.10	155.39
	HyViT	**121.77**	**123.90**	**70.84**	**68.93**	**90.11**	**90.56**

Index

1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

TINTOlib

BarGraph

Parameters	Description	Default value	Valid values
`problem`	Defines the problem type for grouping images.	'supervised'	['supervised', 'unsupervised', 'regression']
`pixel_width`	The width (in pixels) for each column.	1	integer
`gap`	The separation (in pixels) between each column.	0	integer
`zoom`	Factor to scale the saved image size	1	int
`verbose`	Show in terminal the execution.	False	[True, False]

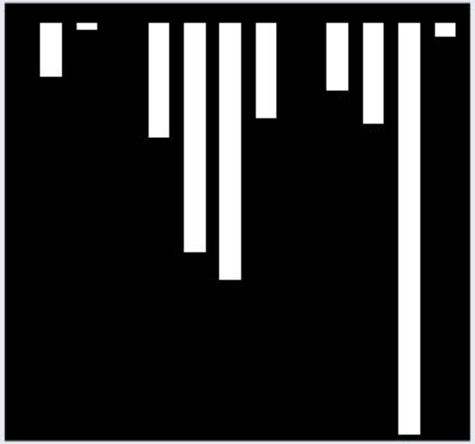
BarGraph

BarGraph

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



Distance Matrix

Parameter	Description	Default	Valid Values
`zoom`	Scale factor for saved image size	1	int
`random_seed`	Seed for reproducibility	1	int
`verbose`	Show execution details	False	[True, False]

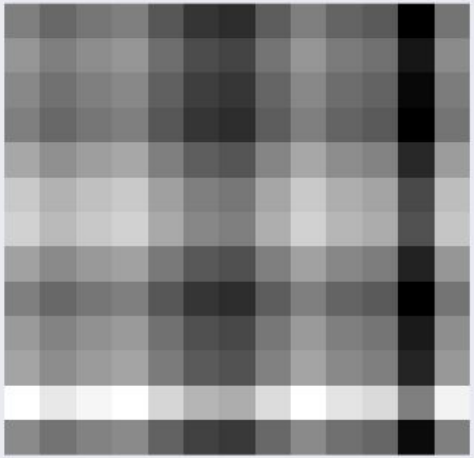
Distance Matrix

DistanceMatrix

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



Combination

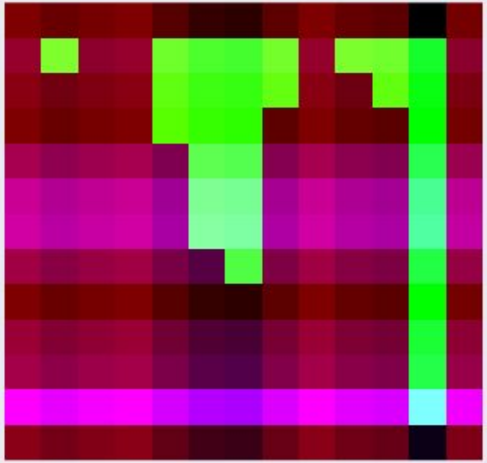
Parameter	Description	Default	Valid Values
`zoom`	Scale factor for saved image size	1	int
`random_seed`	Seed for reproducibility	1	int
`verbose`	Show execution details	False	[True, False]

Combination

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



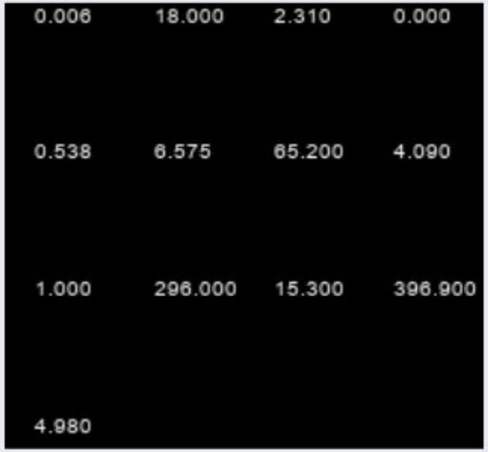
Parameters	Description	Default value	Valid values
`problem`	Defines how images are grouped	‘supervised’	[‘supervised’, ‘unsupervised’, ‘regression’]
`image_pixels`	Number of pixels per side (total pixels = pixels × pixels)	224	integer
`feature_importance`	If False, uses equal font sizes (SuperTML-EF); if True, font size is proportional to feature importance (SuperTML-VF)	False	[True, False]
`font_size`	Font size used to render text on images	10	integer
`random_seed`	Seed for reproducibility	1	integer
`verbose`	Show execution details	False	[True, False]

SuperTML

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



FeatureWrap

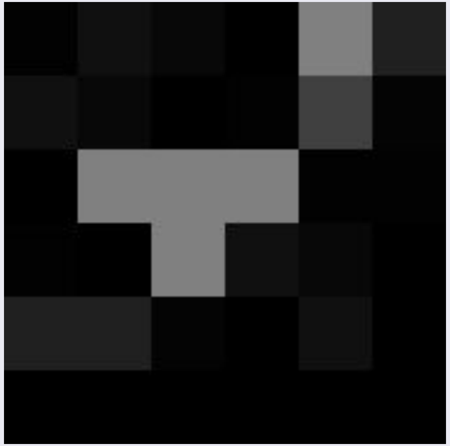
Parameters	Description	Default value	Valid values
`problem`	Defines how images are grouped	‘supervised’	[‘supervised’, ‘unsupervised’, ‘regression’]
`size`	Image dimensions in pixels (rows × columns)	[8,8]	[int, int]
`bins`	Number of bins for grouping numeric data	10	int
`zoom`	Factor to scale the saved image size	1	int
`verbose`	Show execution details	False	[True, False]

FeatureWrap

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



Binary Image Encoding (BIE)

Parameters	Description	Default value	Valid values
`problem`	The type of problem, this will define how the images are grouped.	'supervised'	['supervised', 'unsupervised', 'regression']
`precision`	Number of bits used to represent each feature.	32	[32, 64]
`zoom`	Factor to scale the saved image size	1	Positive integer
`verbose`	Show in terminal the execution.	False	[True, False]

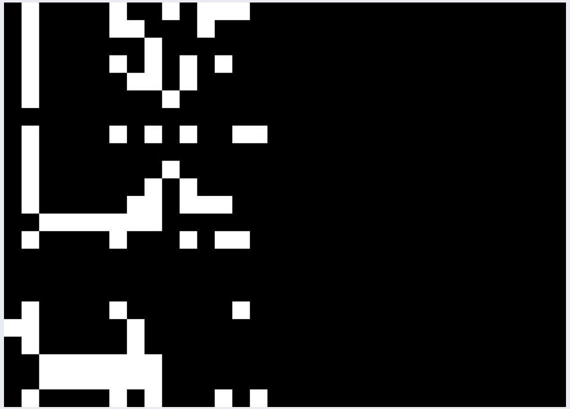
Binary Image Encoding (BIE)

BIE

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



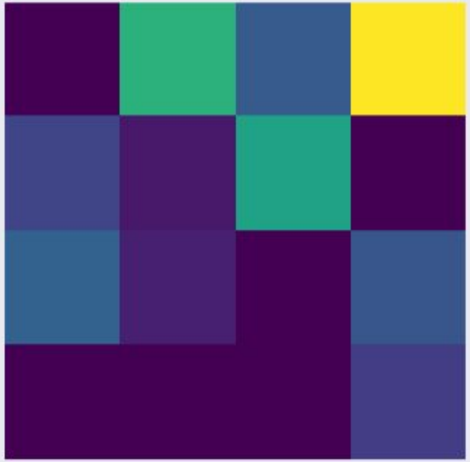
Parameter	Description	Default	Valid Values
`problem`	Defines the problem type for grouping images	'supervised'	['supervised', 'unsupervised', 'regression']
`n_processors`	Number of processors to use (must be ≥ 2)	8	int (≥ 2)
`hcIterations`	Iterations of the hill climbing algorithm	5	int
`zoom`	Factor to scale the saved image size	1	int
`random_seed`	Seed for reproducibility	1	int
`verbose`	Show execution details in terminal	False	[True, False]

REFINED

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



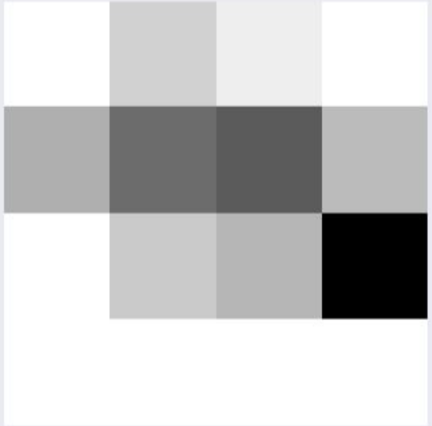
Parameter	Description	Default	Valid Values
`problem`	Defines the problem type for grouping images	'supervised'	['supervised', 'unsupervised', 'regression']
`scale`	Image size (rows × columns); must be ≥ number of features	[6, 6]	[int, int]
`fea_dist_method`	Method to evaluate feature similarity	'Pearson'	['Pearson', 'Spearman', 'set', 'Euclidean']
`image_dist_method`	Method to calculate distances	'Euclidean'	['Euclidean', 'Manhattan']
`max_step`	Max steps before algorithm stops if not converged	1000	int
`val_step`	Steps between convergence checks	50	int
`error`	Function to evaluate the difference between feature distance ranking and pixel distance ranking	'squared'	['squared', 'abs']
`switch_t`	Threshold to decide when to switch elements	0	int
`min_gain`	Minimum improvement needed to continue; else, algorithm stops	0.00001	float
`zoom`	Scale factor for saved image size	1	int
`random_seed`	Seed for reproducibility	1	int
`verbose`	Show execution details	False	[True, False]

IGTD

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



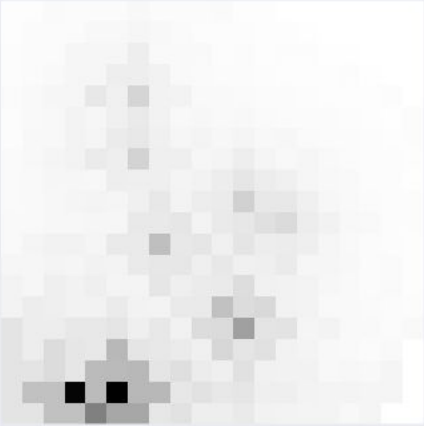
Parameters	Description	Default value	Valid values
`problem`	Defines the problem type for grouping images	‘supervised’	[‘supervised’, ‘unsupervised’, ‘regression’]
`algorithm`	Chooses the dimensionality reduction algorithm	PCA	[PCA, t-SNE]
`pixels`	Sets the image size by specifying pixels per side (total pixels = pixels × pixels)	20	integer
`submatrix`	Determines if a submatrix is used for blurring	True	[True, False]
`blur`	Enables or disables blurring	False	[True, False]
`amplification`	Only with blur=true, blurring amplification	`np.pi`	float
`distance`	Only with blur=true, blurring distance in pixels	2	integer
`steps`	Only with blur=true, blurring steps	4	integer
`option`	Only with blur=true, method for handling overlapping pixels	mean	[mean, maximum]
`times`	Only with algorithm=t-SNE, times replication in t-SNE	4	integer
`zoom`	Factor to scale the saved image size	1	int
`random_seed`	Seed for reproducibility	1	integer
`verbose`	Show in terminal the execution	False	[True, False]

TINTO

INPUT

0.00632	18.00	2.31	0	0.538	6.575	65.20	4.09	1	296.0	15.3	396	4.98

OUPUT



Index

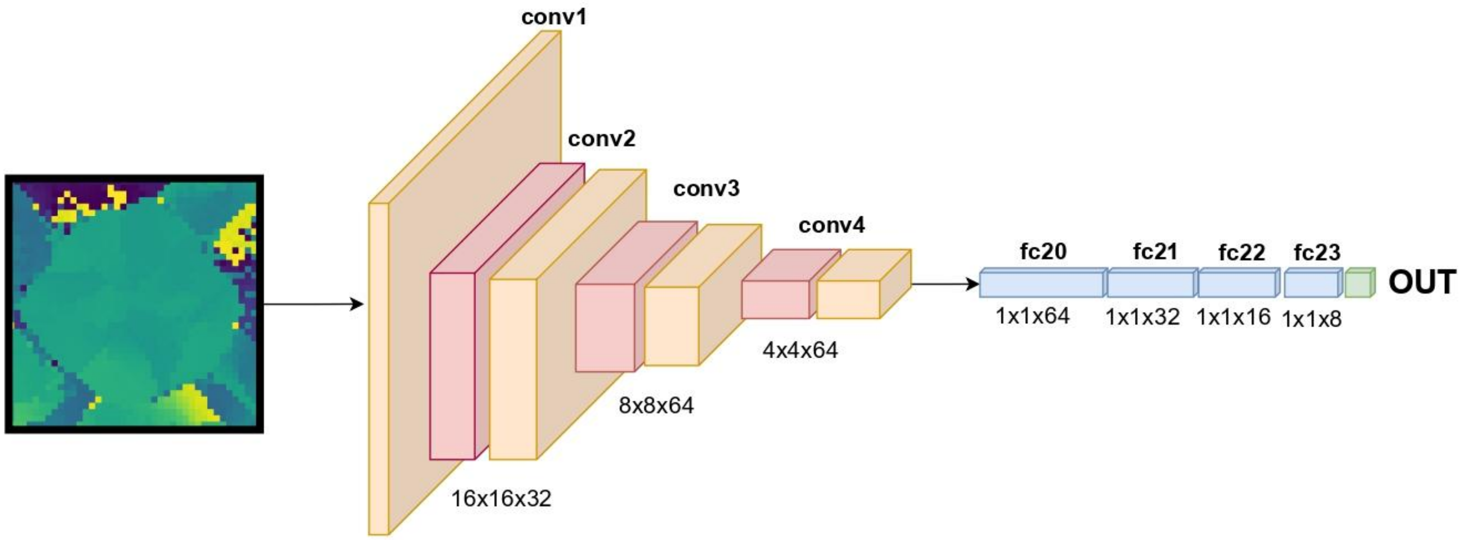
1. The challenges associated with tabular data in deep learning
2. Deep learning in tabular data
3. Methods for transforming tabular data into synthetic images
4. Leveraging vision models on these images
5. Example use case
6. TINTOlib library
7. Practical session

Dataset

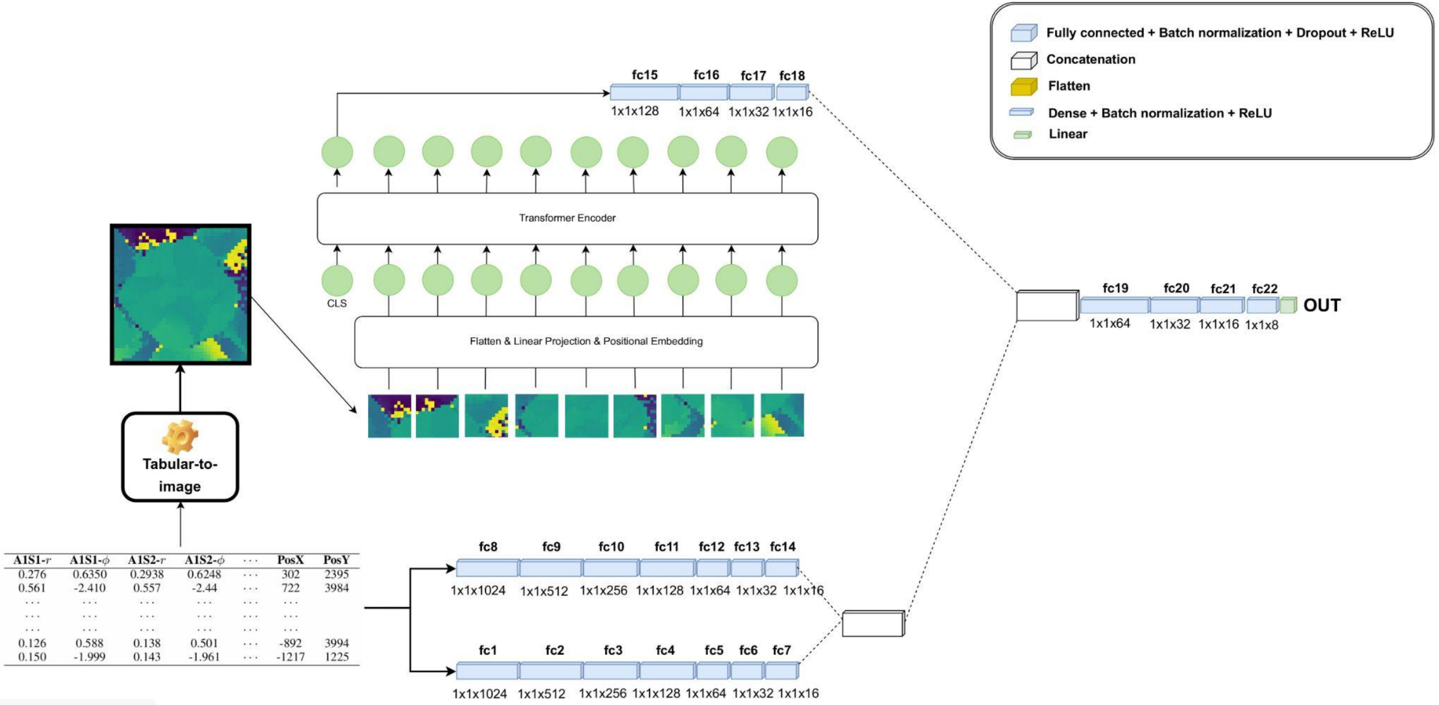
Boston housing dataset

- Regression task
- 506 samples
- 13 features

Simple model



Hybrid Model



Results

Model	MSE	RMSE	MAE	R^2
GradientBoostingRegressor	10.28	3.21	2.18	0.90
HistGradientBoostingRegressor	11.98	3.46	2.35	0.89
LGBMRegressor	12.12	3.48	2.30	0.89
XGBRegressor	12.19	3.49	2.31	0.89
ExtraTreesRegressor	13.64	3.69	2.23	0.87
AdaBoostRegressor	14.17	3.76	2.68	0.87
BaggingRegressor	15.17	3.90	2.72	0.86
RandomForestRegressor	15.51	3.94	2.54	0.85
DecisionTreeRegressor	21.42	4.63	3.17	0.80
ExtraTreeRegressor	23.20	4.82	3.15	0.78

Results

Model	MSE	RMSE	MAE	R^2
GradientBoostingRegressor	10.28	3.21	2.18	0.90
HistGradientBoostingRegressor	11.98	3.46	2.35	0.89
LGBMRegressor	12.12	3.48	2.30	0.89
XGBRegressor	12.19	3.49	2.31	0.89
ExtraTreesRegressor	13.64	3.69	2.23	0.87
AdaBoostRegressor	14.17	3.76	2.68	0.87
BaggingRegressor	15.17	3.90	2.72	0.86
RandomForestRegressor	15.51	3.94	2.54	0.85
DecisionTreeRegressor	21.42	4.63	3.17	0.80
ExtraTreeRegressor	23.20	4.82	3.15	0.78

More Info

- **Official Documentation**: [TINTOlib Documentation]
- **PyPI Library**: [TINTOlib on PyPI]
- **GitHub Repository (TINTOlib)**: [TINOlib GitHubT]
- **GitHub Repository (TINTO)**: [TINTO GitHub]
- **Research Paper on TINTO and Indoor Localization**: [Article with Formal Mathematical Definition]
- **Research Paper on TINTO in Python**: [Article with Python Definition]