MIMO-Based Indoor Localisation With Hybrid Neural Networks: Leveraging Synthetic Images From Tidy Data for Enhanced Deep Learning

Manuel Castillo-Cara, Jesus Martínez-Gómez, Javier Ballesteros-Jerez, Ismael García-Varea,
Raúl García-Castro, and Luis Orozco-Barbosa, *Life Member, IEEE*

***Abstract*—Indoor localization determines an object's position within enclosed spaces, with applications in navigation, asset tracking, robotics, and context-aware computing. Technologies range from WiFi and Bluetooth to advanced systems like Massive Multiple Input-Multiple Output (MIMO). MIMO, initially designed to enhance wireless communication, is now key in indoor positioning due to its spatial diversity and multipath propagation. This study integrates MIMO-based indoor localization with Hybrid Neural Networks (HyNN), converting structured datasets into synthetic images using TINTO. This research marks the first application of HyNNs using synthetic images for MIMO-based indoor localization. Our key contributions include: (i) adapting TINTO for regression problems; (ii) using synthetic images as input data for our model; (iii) designing a novel HyNN with a Convolutional Neural Network branch for synthetic images and an MultiLayer Percetron branch for tidy data; and (iv) demonstrating improved results and metrics compared to prior literature. These advancements highlight the potential of HyNNs in enhancing the accuracy and efficiency of indoor localization systems.**

***Index Terms*—Massive MIMO, deep learning, hybrid neural network, synthetic images, positioning, indoor localisation.**

I. Introduction

INDOOR localization involves determining the position of an object or device within an enclosed area using sensors and various technological methods [1], [2]. This field has significant applications in navigation, asset tracking, robotics, and context-aware computing [3], [4], [5]. Wireless technologies used in localization systems include WiFi, Bluetooth Low Energy (BLE) [1], [6], Zigbee, Radio Frequency Identification Device (RFID) [5], Massive Multiple Input - Multiple Output (MIMO) [7], [8], and Ultra WideBand (UWB) [3], [9]. The choice of technology depends on factors like complexity, accuracy, and environment [4], [10].

MIMO-based indoor localization integrates wireless communication with localization techniques to accurately determine positions in indoor environments [11]. MIMO leverages spatial diversity and multipath propagation to capture detailed environmental data, improving localization accuracy with low complexity and reduced costs by utilizing existing infrastructure [12], [13], [14]. Localization algorithms typically use Channel State Information (CSI) or Received Signal Strength Indicator (RSSI) from MIMO systems to gather details about the propagation environment [8], [14]. Recent studies have applied neural networks to enhance the accuracy of MIMO-based indoor localization [7].

Recent advances in transforming tidy data into synthetic images enable Convolutional Neural Networks (CNNs) to leverage feature sequences, as in speech and imaging applications [15], [16], [17]. However, tidy data typically lacks inherent spatial relationships among features, making it less compatible for CNN modelling [18], [19].

To address these issues, researchers have developed innovative methods to map features to specific pixel positions within images to ensure related features are proximate [19], [20], [21]. Historically focused on CNNs alone –limiting exploration of integrated architectures– these transformations show significant promise for deep learning [22], [23]. Hybrid Neural Networks (HyNNs), combining the strengths of CNNs and MLPs, provide a compelling alternative to overcome these limitations. By leveraging the complementary capabilities of CNNs for synthetic images and MLPs for tidy data, HyNNs offer improved learning and generalization in complex scenarios [17], [22], [23].

This article contributes to advancing the research field by demonstrating the potential of such hybrid architectures applied to indoor localization, offering insights into their advantages and

Received 12 June 2024; revised 10 December 2024 and 13 January 2025; accepted 14 March 2025. Date of publication 31 March 2025; date of current version 19 May 2025. This work was supported in part by the Government of Castilla-La Mancha and "ERDF A way of making Europe" under Project SBPLY/21/180225/000062, in part by CYTED under Grant 520rt0011, in part by the Spanish Government (AEI) under Projects PID2022-137344OB-C33 /AEI/10.13039/501100011033, Project PID2021-123627OB-C52, and Project PID2019–106758GB–C33, in part by Madrid Government (Comunidad de Madrid-Spain) through the Multiannual Agreement with the Universidad Politécnica de Madrid in the Excellence Programme for University Teaching Staff, in the context of the V PRICIT (Regional Programme of Research and Technological Innovation), and in part by the Universidad de Castilla-La Mancha and "ERDF A way of making Europe" under Project 2022-GRIN-34437. The guest editor coordinating the review of this article and approving it for publication was Prof. Zhu Han. *(Corresponding author: Javier Ballesteros-Jerez.)*

Manuel Castillo-Cara is with the Universidad Nacional de Educación a Distancia, 28040 Madrid, Spain.

Jesus Martínez-Gómez, Javier Ballesteros-Jerez, Ismael García-Varea, and Luis Orozco-Barbosa are with the Universidad de Castilla-La Mancha, 02071 Albacete, Spain (e-mail: javier.ballesteros@uclm.es).

Raúl García-Castro is with the Universidad Politécnica de Madrid, 28040 Madrid, Spain.

Digital Object Identifier 10.1109/JSTSP.2025.3555067

showcasing improvements over prior methods in the domain [1], [24]. These hybrid architectures are particularly suitable for addressing the unique challenges posed by MIMO-based indoor localization, as discussed below.

In the context of MIMO-based indoor localization, the high dimensionality and complexity of the data pose unique challenges due to spatial and temporal variability [8], [25]. Traditional methods may not be able to handle this complexity effectively. Transforming tidy data into synthetic images for use in HyNNs addresses these challenges by capturing spatial relationships and variability in the data critical for accurate localization [1], [17]. This approach takes advantage of the strengths of CNNs in processing image data and MLPs in processing structured data, providing a robust solution to the specific requirements of MIMO-based indoor localization.

In this paper, we integrate HyNN architectures with MIMO-based indoor localization by creating synthetic images from tidy data using the TINTO tool [15]. Our approach demonstrates improved metrics and model training generalization. Notably, this is the first application of HyNNs with synthetic images for MIMO-based indoor localization with potential for broader applications. Our key contributions are:

- The adaptation and use of the TINTO tool to transform tidy data into synthetic images tailored for standard regression problems.
- The use of synthetic images generated by TINTO as input data for our learning model, supporting MIMO-based indoor localization.
- The HyNN architecture combines a CNN branch for synthetic images with an MLP branch for tidy data, enhancing learning and generalisation.
- The enhancement of results and metrics from the original paper [7] and classical ML algorithms through the application of HyNN, fed by synthetic images from TINTO.

The remainder of this paper is organised as follows. Section II reviews recent literature of this research. Section III describes the experimental setup. Section IV details the TINTO algorithm and the generation of synthetic images. Section V outlines the learning process, including ML algorithms, HyNN, and complexity. Section VI presents and compares the findings from ML approaches and HyNN. Finally, Section VII provides conclusions and future research directions.

II. Related Work

This section presents a key literature analysis to support the enhancement and comprehension of the proposed solution.

A. MIMO Indoor Localisation

The development of MIMO systems has significantly advanced user localization by leveraging multipath information embedded in CSI measurements. While traditional methods relied on triangulation using distance and angle calculations [26], deep learning now enables direct localization from channel measurements, simplifying the process [26]. Han et al. [13] further enhanced RSS-based localization by introducing Reconfigurable Intelligent Surfaces (RIS) to optimize reflected signals, achieving notable accuracy improvements through phase shift optimization.

MIMO technology holds great promise for indoor localization, capable of achieving high accuracy with relatively low complexity. Various wireless-based localization systems highlight the importance of fingerprinting methods [12]. Our proposal is motivated by the latest advances in deep learning applied to high-precision indoor wireless localization.

In [7], De Bast et al. used a massive MIMO dataset with over 250,000 CSI measurements to develop a 1D-CNN achieving centimeter-level accuracy with a mean error (ME) below 6 cm in a 2.5 m × 2.5 m area using 64 antennas. We consider this method [7] our reference and use the same dataset [25] for all our tests.

Finally, Tian et al. [27] evaluate a deep learning pipeline composed of two parallel Fully Connected Neural Networks (FCNN), achieving millimetre accuracy in tracking a robot's position. Gong et al. [28] study MIMO fingerprint positioning in a non-LoS scenario, showing 93% reliability at one-meter accuracy. Widmaier et al. [14] demonstrate neural network-based indoor localization using raw CSI data, achieving a precision of 23 cm in an 80 m^2 area in LoS conditions.

B. Tidy Data Into Synthetic Image Transformation

Currently, researchers are investigating methods to convert tidy data [23] into synthetic images suitable for feeding two-dimensional CNNs. These approaches include: (i) feature permutation techniques like IGTD [16], [17]; (ii) visual representation methods such as BarGraph, DistanceMatrix, Combination [21], and SuperTML [29], [30]; and (iii) reduced feature representation techniques like DeepInsight [20], TINTO [1], [15], and REFINED [19].

BarGraph, DistanceMatrix, and Combination [21] depict normalized characteristics and their relationships, but lack spatial distribution awareness –a key advantage of CNNs. SuperTML [29], [30], inspired by NLP-based Super Characters, encodes data in 1-channel images but struggles with small value fluctuations and spatial variability.

IGTD [16], [17] optimizes pixel arrangements to minimize discrepancies between feature distances and pixel distances, considering spatial distribution. While effective, its performance depends heavily on hyperparameter tuning and produces 1-channel images. In contrast, REFINED [19] computes the feature distance matrix and applies the MDS algorithm to generate a preliminary feature map, followed by Bayesian fitting to compress the image and assign features to pixels. This produces spatially aware 3-channel images.

Finally, DeepInsight [20] applies dimensionality reduction algorithms (DRA) (e.g., *t*-SNE, PCA) to generate 1-channel synthetic images, while TINTO [1], [15], implemented in Python, extends it with formal specifications and a blurring technique to improve the results.

C. Synthetic Images for MIMO Localization

Recent studies highlight the transformation of tidy data [23] into synthetic images to improve learning via CNNs. This transformation also facilitates the implementation of HyNNs that can handle different data types, since CNNs process images and MLPs manage tidy data. Basic models like SuperTML [29], [30], BarGraph, DistanceMatrix, and Combination [21] do not fully account for the spatial distribution of features. In addition, previous research has largely focused on applying these techniques exclusively with CNNs [22], resulting in incremental advances. In contrast, our study introduces a novel HyNN architecture that significantly improves CNN performance and rivals state-of-the-art models through improved generalization and robustness.

MIMO data's high dimensionality and spatial-temporal variability challenge traditional methods. Transforming tidy data into synthetic images with HyNNs captures spatial relationships critical for accurate localization. More sophisticated models like IGTD [16], [17], REFINED [19], DeepInsight [20], and TINTO [1], [15] capture spatial distribution more effectively. These methodologies are primarily applied to classification problems, with CNNs learning from the generated images. Their potential in regression tasks or with HyNNs remains unexplored.

To address this, we adapted TINTO to create synthetic images tailored for HyNN integration with tidy data. Our HyNN features a CNN branch for images and an MLP branch for tidy data, converging to output a continuous numerical value, paving the way for ML regression tasks. This research marks the first application of HyNNs using synthetic images from tidy data. It demonstrates significant potential in refining model learning and generalization for MIMO-based indoor localization, enhancing the outcomes of the original research [7]. This work does not employ further data preprocessing techniques, achieving robust results from the original data.

III. Background: Tools, Scenario, Metrics and Guidelines

The section outlines the principal tools, dataset, indoor scenario, metrics and guidelines used in the study.

A. Tools: Hardware and Software

For training, we deployed the HyNN using TINTO-generated synthetic images and conducted quality metric assessments on the high-performance supercomputer GALGO at the Albacete Research Institute of Informatics (I^3 A) within the Universidad de Castilla-La Mancha (UCLM). The training process utilised two computing nodes, each equipped with 2 Intel Xeon Platinum 8452Y 2.00 GHz processors, providing a total of 72 cores (36 cores per processor). Each node is also supported by 1.5TB of RAM and 2 NVIDIA L40 graphic cards, each with 48GB of RAM.

The implementation was done in Python. For the HyNN design, we primarily used the PyTorch and TensorFlow and, for the ML algorithms, we used Scikit-Learn. In addition, we used the TINTOlib, which we developed, to transform tidy data into synthetic images [15], [31].

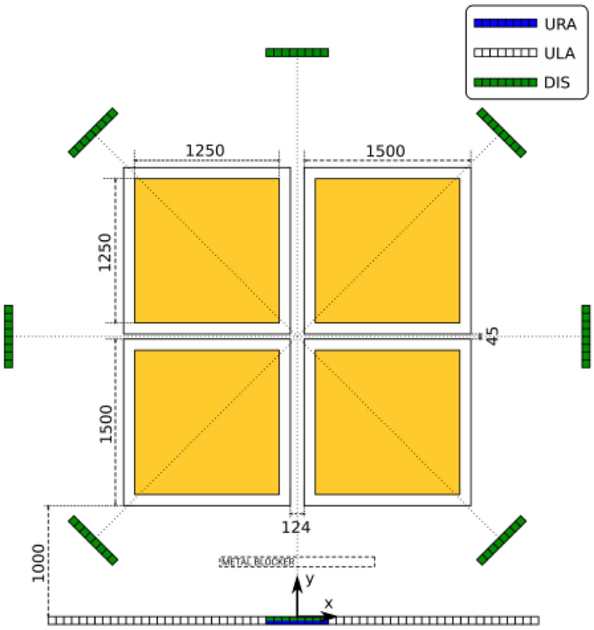


Fig. 1. The three measurement scenarios. The antennas are spaced around the users using a scenario specific topology. The users are positioned inside the green areas. All measurements on the Figure are in mm. (Figure taken from [25]).

B. MIMO Indoor Scenario

The ultra-dense indoor Massive MIMO CSI dataset includes numerous samples of CSI, collected via the 64-antenna KU Leuven Massive MIMO testbed (see Fig. 1) [7], [25]. Additionally, the base station of the testbed was constructed with considerable adaptability in regards to the antenna array's deployment. As a result, three distinct data sets were generated, each featuring a distinctive antenna deployment. Initially, a 8 by 8 Uniform Rectangular Array (URA) was situated utilising a metal blocker in Line-of-Sight (LoS) condition (see URA distribution in Fig. 1). Subsequently, a Uniform Linear Array (ULA) consisting of 64 antennas was installed in a single line (see ULA distribution in Fig. 1). Finally, the antennas were distributed across the room in sets of eight, creating the distributed (DIS) scenario (see DIS distribution in Fig. 1). The ULA scenario is selected for the optimisation of the HyNN architecture and parameters due to their relevance in the conclusions and because they are the most accurate (as referenced in Section V).

The physical features of the environment consist of antennas located at a distance of 1 metre from the XY-tables. In the illustration, yellow rectangles indicate the areas measuring $[1.25 \times 1.25]$ m where users can be moved by the XY-tables. The spacing between the XY-tables is determined by the space required for the motors that power the movements and the cables that connect them to the controllers. The tables were synchronised via Ethernet with the base station to guarantee that the sampled H obtained a precise spatial label, creating a remarkably reliable dataset. Throughout the measurements, the base station was arranged to operate at a centre frequency of 2.61 GHz, which results in a wavelength λ of 114.56 mm. The system incorporated a bandwidth of 20 MHz.

TABLE I
A SAMPLE EXAMPLE OF THE FINAL FORMAT OF THE DATASET IN TIDY DATA

A1C1-m	**A1C1-ϕ**	**A1C2-m**	**A1C2-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
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
0.126	0.588	0.138	0.501	$\cdots$	-892	3994
0.150	-1.999	0.143	-1.961	$\cdots$	-1217	1225

A is the antenna, *C* is the carrier, *m* is the modulus, ϕ is the phase, *PosX* is the *X*-position and *PosY* is the Y-position.

C. MIMO Localisation Dataset

The user's channel is gathered via CNC-tables, thereby generating a dataset with highly precise spatial labelling for all samples [25]. In this process, the user's position is systematically moved over a nine-square metre area, and halted every five millimetres, eventually yielding 252,004 samples for each topology that was measured. The base station is outfitted with 64 antennas, each receiving a predefined pilot signal from all positions. Using pilot signals, the CSI is estimated for 100 carriers evenly spaced in frequency over a 20 MHz bandwidth. Consequently, at one location, the complex number matrix H represents the CSI measured. The matrix comprises N rows and K columns, where N represents the number of base stations antennas and K represents the number of carriers. Thus, the measured CSI can be denoted with the matrix $H \in C^{N \times K}$ [7].

The complete datasets were utilised for this experiment and analysed for antenna quantities of 8, 16, 32, and 64, as demonstrated in [7]. It is imperative to mention that the datasets are extensive, comprising 100 carriers per 64 antennas, alongside 252,004 samples. Consequently, a dataset matrix of [6400 × 252,004] was initially generated. The datasets includes two target values that indicate the X- and Y-position, respectively. Hence, two distinct models; one for predicting the position in X and the other for predicting the position in Y, are formulated. Notably, the targets under consideration are continuous numerical values (regression problem).

D. Dataset Preprocessing

As said before, the experiment used a maximum of 64 antennas with 100 carriers each. Consequently, a numpy array of [64 × 100] was produced for each position, containing the CSI obtained for that particular position. The array consists of values with an imaginary number. To store complex numbers internally, rectangular or cartesian coordinates were used. A complex number z is determined entirely by its real part $z.real$ and its imaginary part $z.imag$, e.g, $(1.36 + 1.02i)$. Thus, polar coordinates use imaginary numbers to transform to radians, specifically modulus and angle.

Therefore, m is saved in its corresponding index within the `Antenna1Carrier1-`m array, while ϕ is saved in the corresponding index of the `Antenna1Carrier1-`ϕ array. To illustrate, Table I displays the final data format in a clear and concise manner, with the final two columns indicating the predicted values, namely, the X- and Y-positions. Note that in addition to the 64 antennas and 100 carriers per antenna discussed earlier, we have m and ϕ for each carrier. For instance, with 64 antennas, there would be 64 (antennas) $\cdot$ 100 (carriers) $\cdot$ 2 (m) and $(\phi) = 12,800$ features. Following this, for 64 antennas, the number of samples would not change, leading to a dataset matrix sized at [12,800 × 252,004]. Note that this work does not employ any further data preprocessing techniques, such as feature selection or data preprocessing, to achieve the robust results obtained from the original data.

E. Evaluation Metrics

The target values of the dataset are given for the X- and Y-position as two independent variables that are continuous numerical, i.e., we have a classical regression problem. Furthermore, we have to create two models, one to predict the X-position and the other to predict the Y-position. Hence, the metrics for any regression problem are used, i.e., R^2, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

Moreover, for comparative purposes with the results reported in [7], we analyse the Mean Error (ME) on the validation and test split as a function of the number of antennas used for the scenarios [7], [25].

Distance ME (in mm) is calculated as follows:

$$ME(mm) = \mathbb{E}\{|p - \hat{p}|\}$$

Wavelength ME (in λ) is calculated as follows:

$$ME(\lambda) = \frac{\mathbb{E}\{|p - \hat{p}|\}}{\lambda}$$

where

- p is the measured position (X, Y) for the user.
- $\hat{p}$ the estimated position (X, Y) for the model.
- λ giving a wavelength of 114.56 mm.

The results are shown in absolute accuracy, using the millimetre as the unit, and in relative accuracy, using one wavelength λ as the unit. This allows one to compare the results independently from the used carrier frequency.

F. Assessment Criteria

Drawing from existing literature, this study explores the use of MIMO-based indoor localization. It is crucial to note that TINTO requires data to be in numerical and tabular format [23] for the conversion of tidy data into synthetic images [1], [15]. The performance of models with 8, 16, 32, and 64 antennas have been analysed, consistent with the original work [7], [25]. In a similar vein, we have acquired a data matrix of [12,800 × 252,004] by considering 100 carriers for each antenna, along with their respective modulus and angles. Furthermore, we have developed two HyNNs to evaluate the X- and Y-positions, respectively. For this purpose, we employed the TINTO method using the TINTOlib python library [31]. TINTO allows the utilisation of HyNNs, which are shown to enhance learning and enable the use of synthetic images in ML regression problems, as supported by this paper.

In this context, Fig. 2 illustrates the methodology applied in this experiment. Firstly, we carry out an extensive analysis of the entire experiment, which includes the MIMO scenario

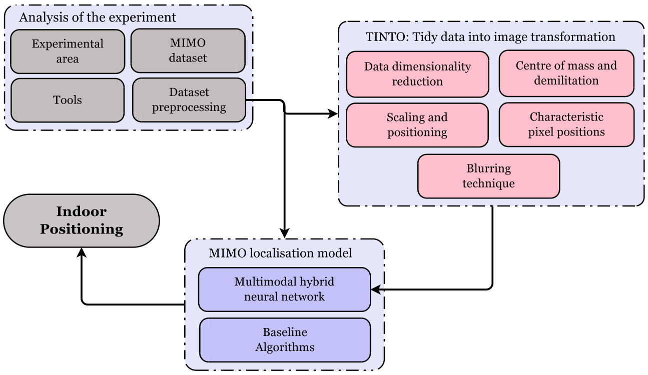

Analysis of the experiment
Experimental area
MIMO dataset
Tools
Dataset preprocessing
TINTO: Tidy data into image transformation
Data dimensionality reduction
Centre of mass and demilitation
Scaling and positioning
Characteristic pixel positions
Blurring technique
Indoor Positioning
MIMO localisation model
Multimodal hybrid neural network
Baseline Algorithms

Fig. 2. Overall schema proposal.

and the dataset (illustrated in blocks labelled “Analysis of the experiment”). The previous preprocessing of the dataset is essential to enable proper handling by the regression algorithms and its conversion into synthetic images using TINTO. The “TINTO: Tidy data into image transformation” block, accepts the preprocessed datasets as input and produces a synthetic image for each sample within the dataset (detailed in Section IV). During the modelling phase, found in the “MIMO localisation model” block, the HyNN, which is the primary addition alongside TINTO, as well as the regression algorithms serving as baseline results, are assessed. The HyNN is provided with two inputs, specifically (i) the dataset in tidy data format [23]; and (ii) the synthetic images generated by TINTO [15]. Nevertheless, the baseline algorithm only receives the dataset in tidy data format as input. Note that this work does not employ any further data preprocessing techniques, such as feature selection or data preprocessing, to achieve the robust results obtained from the original data.

IV. TINTO: Tidy Data Into Synthetic Image Transformation

TINTO methodically transforms datasets in tidy data format into synthetic images using a DRA. TINTO stands out for its unique attributes related to synthetic image creation and the strategic placement of characteristic pixels within these images. Users can customise these features via TINTO’s adjustable parameters [1], [15]. This paper delves into the mathematical and algorithmic foundations of TINTO and showcases results derived from its synthetic images in subsequent sections.

A. TINTO Algorithm

TINTO employs a sophisticated mathematical model to convert tidy data into synthetic images. The transformation process involves mapping tidy data into a two-dimensional space defined by the X and Y axes. This translation enables the creation of synthetic images where characteristic pixels represent each dataset sample, effectively encoding the data into a structured spatial format.

The procedure consists of five primary tasks [1], [15]: (1) DRA: For this task, a DRA was employed, with TINTO exploring the algorithm - PCA. The initial matrix is transposed, where each feature is denoted by a distinct colour for better clarity, despite the resulting image being a 1-channel (B/W); (2) Centre of mass and delimitation: Once a coordinate is obtained, the centroid of the points is determined, followed by defining the boundary of the area; (3) Scaling and pixel positions: The matrix is transposed, scaled and rounded to integer values; (4) Characteristic pixel positions: The values obtained would provide the locations of the characteristic pixels used in generating the synthetic image pattern; and (5) Blurring technique: The proposed methodology aims to utilise a blurring technique on characteristic pixels in order to increase their area size.

TABLE II
TINTO Parameters Available in the TINTOlib Library and the Values Used in This Experiment

Parameter	Type value	Value	Description
algorithm	[PCA, *t*-SNE]	PCA	Dimensionality Reduction Algorithm
pixels	integer	35	Number of pixels in the image [pixels x pixels]
blurring	boolean	True	Use of blurring
amplification	float	ϕ	Blurring amplification radius
distance	integer[0,1]	0.1	Distance of blurring (%)
steps	integer	4	The steps of the blurring
option	[mean, max]	mean	Technique to address pixel overlapping
seed	integer	20	Random numbers
submatrix	boolean	True	Using submatrices to perform efficiently

TINTO can be described as outlined in Algorithm 1. Noted that TINTO also involves an intricate algorithmic component, as demonstrated in [15]. Consequently, TINTO is a potent tool for generating synthetic images in a systematic manner using tidy data in tidy data format. For this paper, we generated synthetic images according to the specifications outlined in Table II. A rigorous mathematical formalisation of TINTO fully investigates these aspects.

B. TINTO Mathematical Formalisation

TINTO is formally described through an intricate mathematical model, comprised of five main numerical algorithmic steps.

1) Apply the DRA to the standardised characteristics. Given the matrix $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{varN}}$, where N is the number of samples (rows) and $varN$ is the number of features (columns).

$$A = \begin{bmatrix} a_{11} & \dots & a_{1varN} \\ a_{21} & \dots & a_{2varN} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NvarN} \end{bmatrix}$$

Transpose the matrix A to obtain $A^t \in \mathbb{R}^{\mathrm{varN}\times\mathrm{N}}$:

$$A^t = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{N1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1varN} & a_{2varN} & \dots & a_{NvarN} \end{bmatrix}$$

Algorithm 1: TINTO.

```
Data: algorithm, pixels, blur, amplification, distance,
      steps, option, seed, times
Result: Gray-scale images
#Normalise features
$X = norm01(X)$;
#Apply RDA
$B = RDA(X^t, 2)$;
#Compute the characteristic points
$P = \{p_1, \ldots, p_{varN}\}$ where $p_i = (b_{i1}, b_{i2}) \forall i \in varN$;
#Scale and define pixel position
$C = G(X)$;
#Calculate characteristic pixel
$Q = F_2(F_1(p_i))$;
#Apply burring
$Q = Blurring(Q))$;
for $i = 0 : N$ do
    $img = H(C, Q)$;
    #Save image
    $save(img)$;
end
```

Apply the DRA with PCA, to the transposed matrix A^t. The resulting matrix reduces N to 2 columns. The resulting matrix is $B \in \mathbb{R}^{\text{varN} \times 2}$

$$DRA(A^t, 2) = B = \begin{bmatrix} b_{11} & b_{12} \\ \vdots & \vdots \\ b_{varN1} & b_{varN2} \end{bmatrix}$$

2) Calculate the centre of mass and delimit the representation plane. Define a point p_i (where $i \in [1, varN]$) for each row of B, where the coordinates (X, Y) are provided by the first and second column, correspondingly.

$$p_i = (b_{i1}, b_{i2}) \to P = \{p_1, \ldots, p_{varN}\}$$

Find the centre of mass p_{MC} of the points p_i.

$$p_{MC} = \left(\frac{\sum_{i=1}^{varN} b_{i1}}{varN}, \frac{\sum_{i=1}^{varN} b_{i2}}{varN} \right)$$

Calculate the distance from the farthest point to the centre of mass $d_{\max}$.

$$d_{\max} = \lceil \max_{\{i \in [1, varN]\}} \|p_i^{\to} - p_{MC}^{\to}\| \rceil$$

where $\|\|$ is the norm function, $\lceil\rceil$ is the ceil function, and $p^{\to}$ is a vector from the coordinate origin to the point p. Once $d_{\max}$ is computed, the Cartesian plane S containing the points p_i is calculated.

$$\begin{aligned} S = \{(x, y) : x, y \in \mathbb{R}/ \\ x \in [p_{MCx} - d_{\max}, p_{MCx} + d_{\max}], \\ y \in [p_{MCy} - d_{\max}, p_{MCy} + d_{\max}]\} \end{aligned}$$

Perform a linear transformation $F_1 : S \to T$

$$F_1(x, y) = (x - p_{MCx} + d_{\max}, y - p_{MCy} - d_{\max}); \quad \forall (x, y) \in S$$

When using $d_{\max}$, the square bounding the feature image space has a size of $2d_{\max} \times 2d_{\max}$. T is defined as:

$$T = \{(x, y) : x, y \in \mathbb{R}/x \in [0, 2 \times d_{\max}], \quad y \in [-2 \times d_{\max}, 0]\}$$

3) Scale and define the position of the pixels. In this step, the T matrix is scaled to obtain the pixels of the image. This is accomplished through the application of transformation $F_2 : T \to U$.

$$F_2(x, y) = \left(\left\lfloor x \times \frac{pixels - 1}{2 \times d_{\max}} \right\rceil, \left\lfloor |y \times \frac{pixels - 1}{2 \times d_{\max}}| \right\rceil \right); \quad \forall (x, y) \in T$$

where $pixels$ is the number of pixels of the resulting image and $||$ and $\lfloor\rceil$ are the absolute value and rounding functions (solely to integer value positions), respectively.
In turn, the set U is defined as:

$$U = \{(x, y) : x, y \in \mathbb{N}/x \in [0, pixels - 1], \quad y \in [0, pixels - 1]\}$$

At the same time it is necessary to transform the characteristic pixels p_i positions of the characteristic pixels q_i.

$$q_i = F_2(F_1(p_i)) \to Q = \{q_1, \ldots, q_n\}$$

The values must then be scaled to obtain the gray range [0,255] using $G : R^{N \times varN} \to R^{N \times varN}$.

$$\begin{aligned} G(A) &= [\lfloor 255 \times a_{ij} \rceil] \in R^{N \times varN} \\ &= [c_{ij}] \in R^{N \times varN} = C \end{aligned}$$

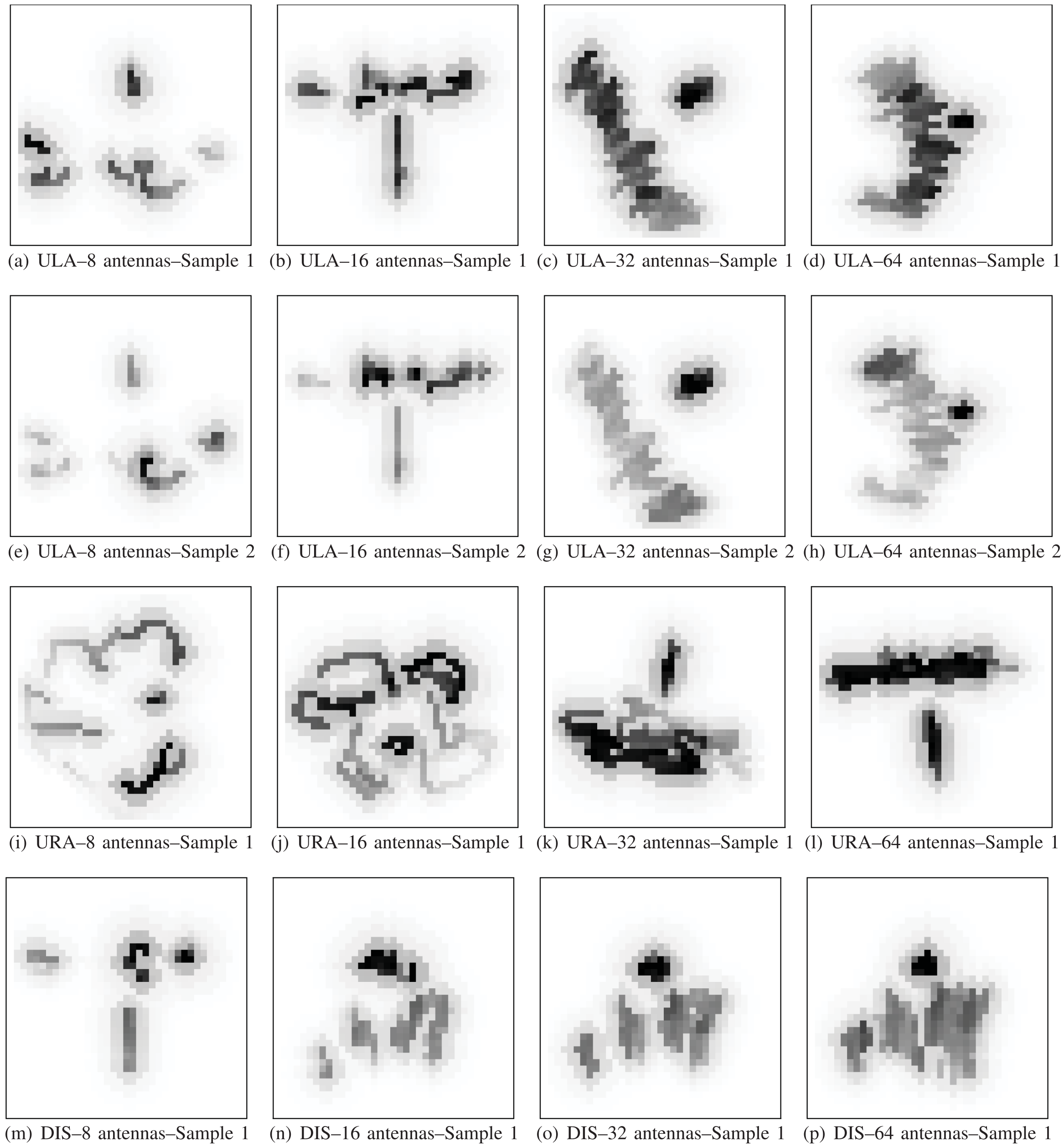
where $C \in R^{N \times varN}$ is the matrix containing the final values after normalisation and scaling to [0,1], to be placed at the corresponding positions in the characteristic pixels.

4) Create synthetic images using the positions of the characteristic pixels Q and the matrix of values C with a linear transformation $H : R^{N \times varN} \times R^{varN \times 2} \to R^{N \times pixel \times pixel}$.

$$\begin{aligned} & H(C, Q) \\ & = \left[w_{ijk} = \begin{cases} c_{i\lambda}, & \text{if } (j, k) = q_\lambda \in Q, \lambda \in [1, n] \\ 0, & \text{otherwise} \end{cases} \right] \\ & \in R^{N \times pixels \times pixels} \end{aligned}$$

where, $C \in R^{N \times varN}$ and $Q \in R^{varN \times 2}$.

5) Finally, blurring technique enables expansion and blurring of the characteristic pixel value to its neighbouring pixels, thus enriching image information. When two or more blurring pixels overlap, the average value of the overlapping


(a) ULA–8 antennas–Sample 1 (b) ULA–16 antennas–Sample 1 (c) ULA–32 antennas–Sample 1 (d) ULA–64 antennas–Sample 1

(e) ULA–8 antennas–Sample 2 (f) ULA–16 antennas–Sample 2 (g) ULA–32 antennas–Sample 2 (h) ULA–64 antennas–Sample 2

(i) URA–8 antennas–Sample 1 (j) URA–16 antennas–Sample 1 (k) URA–32 antennas–Sample 1 (l) URA–64 antennas–Sample 1

(m) DIS–8 antennas–Sample 1 (n) DIS–16 antennas–Sample 1 (o) DIS–32 antennas–Sample 1 (p) DIS–64 antennas–Sample 1

Fig. 3. Synthetic image samples generated by TINTO for different samples in the ULA, URA and DIS scenarios.

pixels is calculated according to the following formula:

$$intensity_i = amplification \times \frac{intensity_0}{\pi \times (steps \times distance)^2}$$

where $intensity_0$ is the value of the characteristic pixel; $amplification$ is the blurring amplification, $steps$ are the steps of the blurring; and $distance$ is the distance of blurring (in %).

C. TINTO Synthetic Images

As discussed, TINTO enables the conversion of a dataset in tidy data format into synthetic images. This conversion allows the use not only of CNNs but also HyNNs such as the one designed in this experiment. The network consists of a CNN branch that learns from generated synthetic images and an MLP that learns from the original tidy data. The conversion process is straightforward, with every dataset sample generating a image. Note that, as described in Sections III-C and III-F, we have developed two CNN+MLP models: (i) one to learn the X-position and (ii) another to learn the Y-position. The study involved comparing the outcomes by using metrics with 8, 16, 32, and 64 antennas.

As a result, Fig. 3 presents the TINTO synthetic images of the aforementioned antennas. As a result, synthetic images of the antennas were generated using TINTO. Synthetic images for the ULA (see Fig. 3(a), (b), (c), and (d)) are created using the blurring technique, which adds more contextual and organised information to the image, resulting in better numerical metrics due to the clearly defined lines and borders, as well as a more detailed spatial representation of the features [1], [15]. On the other hand, Fig. 3(e), (f), (g) and (h) illustrate the appearance of the generated synthetic images produced by TINTO for a distinct sample from the previous ones. As depicted in Fig. 3, comparing the synthetic images of the first and second row reveals identical clusters, albeit with varying pixel intensity. This pixel intensity is significant in feature extraction and spatial distribution for CNNs.

Finally, the spatial representation of the synthetic images generated by TINTO for the URA (see Fig. 3(i), (j), (k) and (l), respectively) and DIS (see Fig. 3(m), (n), (o) and (p), respectively) scenarios is illustrated in the last two rows of Fig 3. These images correspond to the same sample presented in the first two rows for ULA scenario. Notably, there is completely different between each scenario concerning the spatial pattern drawn by characteristic pixels and the blurring technique. This is a coherent argument because even with the same number of antennas, the numerical values in every contact sheet differ significantly. It is worth noting that HyNN optimisation is performed on ULA images. It is reasonable to suggest that, given the disparity among ULA, URA and DIS TINTO synthetic images, adjusting the HyNN can enhance the results of these scenarios (as referenced in Section V-D).

V. Learning Process Modelling

This section examines the various evaluated models and the training process carried out in this work. First, it presents the classical ML algorithms employed. Second, the specifications of the HyNN, which comprises a CNN+MLP, is presented. Additionally, the HyNN architecture is outlined. Finally, the training phase specification is presented.

A. Regression Algorithms

For initial baseline algorithms, Supervised Learning Algorithms (SLAs), particularly regression models, are advantageous in addressing indoor MIMO-based localization. Hence, for SLAs, we possess knowledge of the input parameters and output, which are iteratively trained on the dataset. Upon completing the model training, we can make predictions and then evaluate the model performance via comparison of predicted values with ground truth data. As outlined in Section III-A, several regression algorithms available in Scikit-Learn were used in this experiment. For the purpose of comprehension and visualisation, a subset of algorithms that obtained the most favourable RMSE in an initial survey with ULA scenario with 8 antennas: `LinearRegression` (LiR), `DesionTreeRegressor` (DT), `KNeighborsRegressor` (KNN), `BaggingRegressor` (BR), `AdaBoostRegressor` (AB), `ExtraTreesRegressor` (ET), `GradientBoostingRegressor` (GBM), `HistGradientBoostingRegressor` (HGBM) and `XGBRegressor` (XGB).

In addition, for the purposes of this study, the classical ML algorithms were only evaluated on the ULA scenario, which provided the best quantitative results, and therefore a comparison has been made with the HyNN. Consequently, other algorithms with worse results were discarded.

B. Hybrid Neural Network Architecture

The HyNN architecture consists of two main branches (see Fig. 4). The first branch analyses synthetic images generated by TINTO through a CNN, while the second branch analyses tidy data through an MLP.

The CNN in the first branch has two convolutional sub-branches with [16, 32, 64, 64] filters, respectively. The first sub-branch uses a $[3 \times 3]$ kernel and max pooling of $[2 \times 2]$, while the second sub-branch uses a $[5 \times 5]$ kernel and average pooling of $[2 \times 2]$. The two sub-branches are then flattened and concatenated with a five-layer FCNN with [512,256,128,64,32] neurons, respectively. A dropout of 0.3 is applied in the five layers.

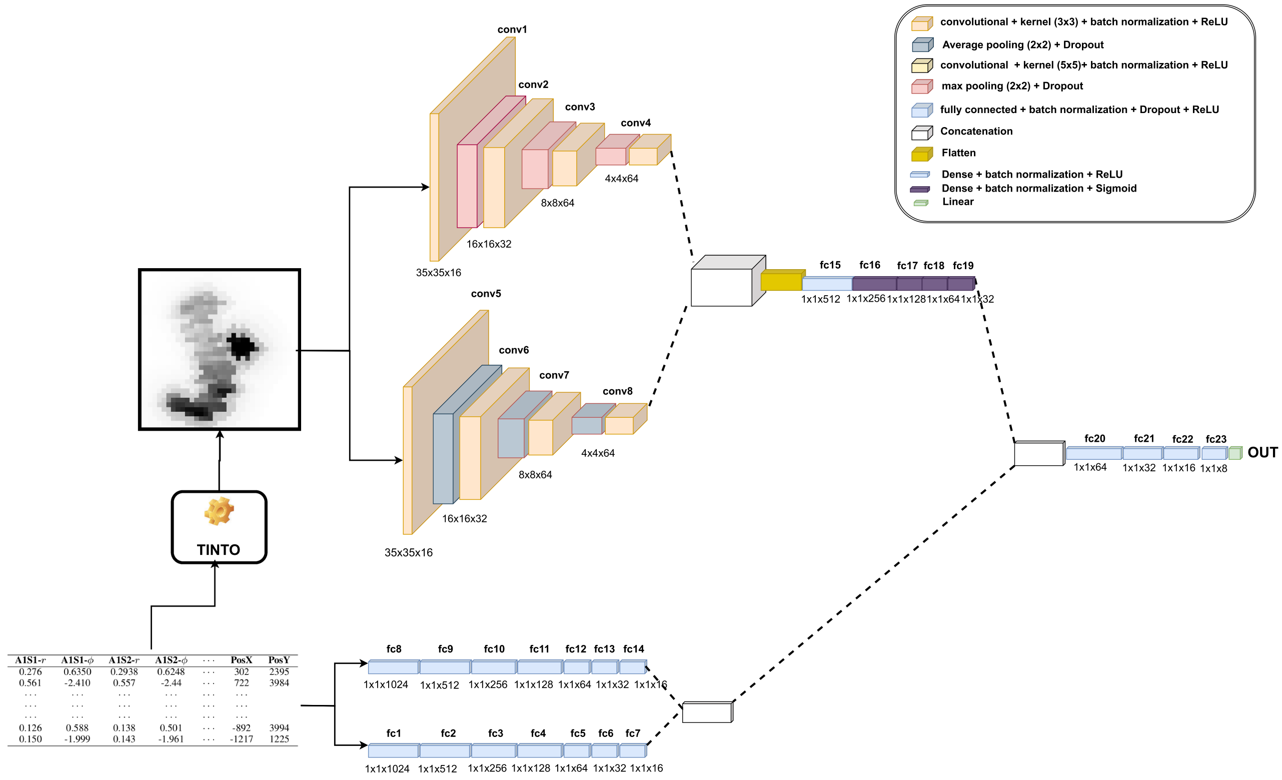
The MLP in the second branch has also two sub-branches with seven hidden layers each, with [1024,512,256,128,64,32,16] neurons, respectively. A ReLU activation layer, a batch normalisation layer, and a dropout of 0.3 are applied after each layer. The outputs of the two branches are then concatenated.

The results of the CNN and MLP are then merged, followed by four hidden layers with [64,32,16,8] neurons, respectively. A ReLU activation layer, a batch normalisation layer, and a dropout of 0.3 are applied after each layer. Finally, it is followed by the output layer with one neuron and a linear activation function to give a continuous number value for the regression problem.

To address the two target variables in the dataset, X- and Y-positions, two independent models were created, one for each variable. Both models have a similar structure, which includes two main branches: a CNN branch that learns from synthetic images generated by TINTO, and an MLP branch that learns from the original dataset. Each branch has two sub-branches. The two main branches are then merged to provide stability and generalisation to the model's learning.

Among the various fusion techniques evaluated, we tested partial concatenation strategies (within the CNN or MLP branches) and final concatenation strategies (between the CNN and MLP branches). The Transformer-Encoder-based concatenation consistently outperformed other methods, including Early Fusion, Weighted Fusion, and Attention-Based Mechanisms, demonstrating significant improvements in both accuracy and generalization. The Transformer-Encoder's superior performance lies in its ability to model complex interactions between features extracted by the CNN (spatial information) and the MLP (structured data), employing the Multihead-Attention mechanisms. This enables the model to effectively capture both local and global patterns, which is particularly beneficial for the MIMO dataset.

Additionally, the hierarchical feature integration provided by the Transformer-Encoder enhances the model's capacity to handle multi-scale dependencies and cross-modal relationships. This approach led to a reduction of RMSE by an average of 15% compared to Early Fusion, the next best-performing method, highlighting its effectiveness for datasets requiring a combination of spatial and structured data. Given these results,



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

Fig. 4. Hybrid Neural Network architecture comprises a CNN branch for the generated synthetic images from TINTO tool and an MLP branch for the tidy data.

we recommend Transformer-Encoder-based concatenation as a primary strategy for designing HyNNs. Despite its additional computational overhead, the significant improvements in quantitative results and interpretability make it an ideal choice for a wide range of applications. This technique enables HyNN models to achieve superior performance, particularly in complex datasets like MIMO.

Note that the parameters for the Transformer-Encoder, CNN and MLP were set through empirical configuration until meaningful results were achieved. The focus of this paper is to demonstrate the effectiveness of the HyNN for the MIMO dataset, rather than exploring the optimal search for these parameters.

C. *Single-Branch Architectures: CNN & MLP*

In addition to the HyNN architecture described above, we evaluated the performance of individual models using a branch CNN and a branch MLP. These models utilize the same architectures as the corresponding branches within the HyNN, as illustrated in Fig. 4, with the CNN processing only synthetic images generated by TINTO and the MLP exclusively handling the original tidy data. The results of these CNN and MLP are presented alongside the HyNN results to highlight the impact of combining synthetic image-based feature extraction with tidy data processing.

D. *Training Phase Specifications*

The two independent models for X- and Y-position outputs are trained following a consistent procedure. TINTO generates identical synthetic images for each run, and data is partitioned with 80% for training, 15% for validation, and 5% for testing, using the same partitions and seed as in the original article [7]. The Adam optimizer is set with a learning rate of 1e-3 for X-position and 1e-4 for Y-position, and MSE served as the loss function with a batch size of 32. Early stopping is implemented to restore weights from the epoch with the least loss if no improvement occurred over 20 epochs. Training was capped at 200 epochs, though models typically converged within 150 epochs. For the training of ML algorithms, the split dataset used is the same as for the HyNN. The training is computationally and time-consuming, with over 50 hours required in ML algorithms and over 15 hours in HyNN, for the ULA scenario with 64 antennas.

Finally, potential improvements to the HyNN could involve refining internal parameters, such as weight initialization, layer configurations, or incorporating additional branches. In this study, further optimization was not pursued since the baseline configuration yielded satisfactory results. While the HyNN is configured to perform optimally in the ULA scenario with TINTO synthetic images, additional architectural and hyperparameter adjustments could lead to even more significant outcomes in URA and DIS scenarios.

E. *Computational Complexity Analysis*

We analyse the complexity Big O of TINTO and the models which n_var is the number of variables, N is the total of instances in the dataset and $pixels$ in height and width (see

TABLE III
ALGORITHMIC COMPLEXITIES FOR TINTO AND DEEP NEURAL NETWORKS

Method	Complexity
TINTO (PCA)	$O(N \times n\_var^2 + n\_var^3)$
TINTO (*t*-SNE)	$O(N \times n\_var^2)$
MLP	$O(n\_var)$
CNN	$O(pixels^2)$
HyNN	$O(n\_var + pixels^2)$

n_var is the number of variables, *N* is the total of instances in the dataset and *pixels* in height and width.

Table III). The TINTO method, used for transforming tidy data into synthetic images, has a complexity that varies with the DRA. PCA, while more complex than *t*-SNE, was chosen for its better performance and stability (less sensitive to the seed [1]) in our experiments.

On the contrary, the complexity of each model is as follows: (i) MLP: $O(n\_var)$, because the MLP complexity scales linearly with the number of input features n_var, and subsequent layers are of fixed size; (ii) CNN: $O(pixels^2)$ since the complexity is dominated by the convolutional operations, which primarily depend on the spatial dimensions $[pixels \times pixels]$ of the input image; and (iii) HyNN: $O(n\_var + pixels^2)$, since it combines an MLP ($O(n\_var)$) and a CNN ($O(pixels^2)$), and the Transformer-Encoder fusion adds only a constant factor ($O(1)$), the overall complexity is $O(n\_var + pixels^2)$.

VI. Experimental Results & Evaluation

This section aims to assess the efficacy of our proposed method using the scenarios outlined in Section III-C. We will begin by confirming that our method achieves promising results and investigate the risk of overfitting. Upon successful validation, we will proceed to compare our proposal against state-of-the-art solutions. The section will conclude with an examination of a new research proposal, such as the one introduced in [7], comparing its results directly with those achieved by our approach.

A. Experiment 1.- Proposal Validation

The first experiment is designed to find out the suitability of the HyNN for regression problems, such as those found in MIMO-based indoor localization. We conducted this by training the HyNN across three different scenarios within our selected dataset, using ULA, URA and DIS configurations with 8, 16, 32, and 64 antennas. In each scenario, we trained separate models for the X- and Y-positions. The data was divided into training, validation, and test sets with ratios of 85%, 10%, and 5%, respectively, aligning with the methodology used by the benchmark [7] (see Section V-D). We chose RMSE as the primary metric for reporting results, which are detailed in Table IV, for both X- and Y-positions.

The analysis of these results yields several significant insights. Initially, it is clear that the obtained results are promising, which is further validated in subsequent experiments. This outcome is important as it confirms our model's effectiveness not only with ULA –the configuration it was originally designed to work with– but also with URA and DIS scenarios, underscoring HyNN's ability to generalise across different scenarios. Additionally, the model scales positively with the increased complexity introduced by a larger number of antennas, suggesting that HyNN effectively integrates more data to enhance performance. Crucially, the results suggest that HyNN mitigates the risk of overfitting, with the best performance observed in the test split. This phenomenon is graphically analysed in Fig. 5, where RMSE metric evolution is presented for training, validation, and test splits.

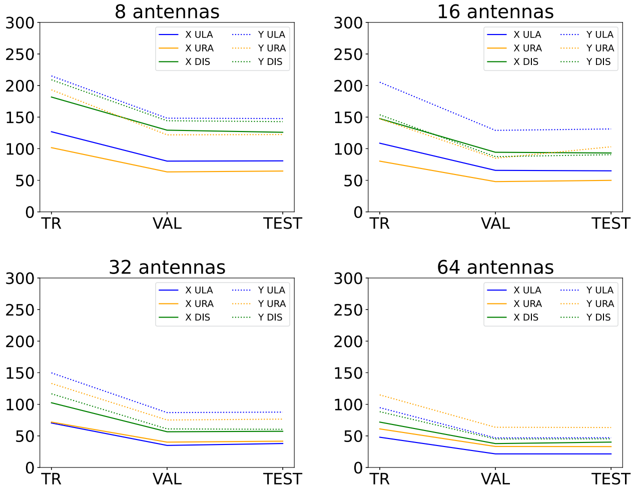


Fig. 5. Evolution of training (TR), validation (VAL), and test (TEST) RMSE metrics for HyNN for the ULA, URA and DIS scenarios in X- and Y-positions.

B. Experiment 2.- Comparison With State-of-The-Art Solutions

Having established the promising attributes of HyNN, the second experiment is designed to compare it against standard solutions meeting two requirements: (i) they are extensively used in the literature; and (ii) there are canonical implementations for easy integration and fair comparisons. We selected a comprehensive list of methods –namely, AB, BR, DT, ET, GMB, HGMB, KNN, LiR, and XGB– all of which are detailed in the Scikit-Learn Python library (as referenced in Section V-A). Additionally, an ablation study was conducted to assess whether HyNN outperforms individual MLP or CNN models, providing insights into its superior performance (as referenced in Section V-C).

Consistent with the first experiment, we maintained the same distribution for training, validation, and testing splits (85%, 10%, and 5%), and the same metric. However, to streamline the data presentation, we limit our results to the testing split. We also remove the 64-antennas configuration to conserve on presentation space and computational resources, given the large training times associated with it. This reduction is warranted to facilitate a broad comparison across a significant, yet manageable, number of methods and scenarios –specifically, 18 scenarios combining ULA/URA/DIS configurations with 8/16/32 antennas for X/Y positioning are deemed sufficient for drawing meaningful insights.

TABLE IV
RMSE Results (In mm) for the HyNN in train/validation/test for the ULA, URA and DIS Scenarios With 8, 16, 32 Y 64 Antennas

Antennas	Scenario	X-position			Y-position		
		Train	Validation	Test	Train	Validation	Test
8	ULA	126.86	80.39	80.75	215.23	148.32	147.71
	URA	101.65	63.24	**64.62**	193.19	121.92	**122.43**
	DIS	181.79	129.32	125.99	209.10	144.26	142.84
16	ULA	108.71	65.69	64.96	205.30	129.02	131.26
	URA	80.29	47.87	**49.86**	147.22	84.89	103.04
	DIS	147.63	94.34	93.20	153.70	87.43	**90.53**
32	ULA	70.38	34.87	**37.90**	149.80	86.89	87.61
	URA	71.83	40.09	41.59	133.02	75.14	76.50
	DIS	102.45	56.51	57.24	116.65	61.02	**60.43**
64	ULA	47.86	21.50	**21.46**	94.72	46.85	47.07
	URA	61.05	33.23	33.02	114.92	63.63	63.22
	DIS	71.83	37.76	40.14	88.32	45.11	**45.35**

Best results, in test split, for each number of antennas are shown in bold.

TABLE V
RMSE (In mm) in Test Split for ULA, URA and DIS Scenarios for 8, 16, 32 and 64 Antennas Comparing Classical Regression Algorithms With the HyNN

		ULA				URA				DIS			
Model	Position	8	16	32	64	8	16	32	64	8	16	32	64
AB	X	754.72	653.67	429.89	253.30	699.48	655.68	578.71	440.10	538.16	494.65	386.98	297.55
	Y	762.68	693.21	548.65	428.67	797.15	785.88	720.26	665.38	599.54	569.81	470.37	348.98
BR	X	355.99	286.90	150.62	82.67	268.72	278.96	235.17	160.81	256.66	206.67	163.26	116.99
	Y	380.36	297.54	180.47	134.84	324.12	314.13	265.79	224.38	278.41	228.54	173.11	123.13
DT	X	555.26	459.19	254.01	153.09	443.43	436.40	400.94	268.58	384.11	339.87	283.65	212.98
	Y	588.38	474.82	316.07	253.81	524.01	503.94	454.84	392.54	434.52	364.54	300.03	226.30
ET	X	195.16	176.22	90.75	40.72	146.04	155.39	135.06	94.04	169.01	123.47	90.49	55.94
	Y	248.77	191.82	104.76	67.52	187.08	173.27	155.06	129.58	182.46	135.15	96.33	60.44
GBM	X	602.01	497.48	296.22	166.22	498.71	469.22	400.94	308.34	401.94	348.88	293.80	227.00
	Y	618.06	513.65	369.73	307.75	551.24	428.57	443.23	433.01	454.56	401.42	348.01	266.92
HGBM	X	336.09	301.52	197.24	114.52	299.46	304.88	244.05	195.04	313.33	260.74	216.34	158.56
	Y	418.18	352.71	259.87	203.43	320.65	297.25	293.48	257.69	357.46	285.04	239.04	176.97
KNN	X	**65.10**	**32.45**	**9.61**	**4.60**	**54.81**	**26.28**	**10.93**	**5.79**	**102.96**	**29.13**	**4.28**	**2.42**
	Y	151.63	**80.99**	**22.95**	**6.68**	141.47	**69.12**	**25.12**	**8.01**	**125.89**	**33.53**	**4.88**	**2.09**
LiR	X	530.69	421.35	305.07	147.53	492.87	421.21	352.66	259.79	382.54	318.64	261.44	186.10
	Y	418.97	347.85	271.11	199.51	495.06	424.30	336.04	263.69	431.91	362.85	289.47	201.11
XGB	X	237.09	219.08	149.80	99.02	202.18	204.77	184.80	152.68	237.92	197.47	168.47	140.63
	Y	277.84	253.37	208.69	174.07	260.87	234.04	227.18	207.67	275.29	213.67	184.19	149.79
MLP	X	95.66	80.36	47.05	34.79	98.06	68.91	50.37	55.38	202.00	96.63	75.22	65.54
	Y	230.38	205.11	107.77	120.36	178.81	143.40	167.89	100.90	165.48	99.02	58.76	123.00
CNN	X	228.10	257.50	359.51	187.90	153.34	163.65	161.92	147.24	638.74	361.75	294.29	229.54
	Y	441.17	403.72	401.65	357.78	384.52	326.18	250.85	363.60	642.13	458.47	531.85	431.53
HyNN	X	80.75	64.96	37.90	21.46	64.62	49.86	41.59	33.02	125.99	93.20	57.24	40.14
	Y	**147.71**	131.26	87.61	47.07	**122.43**	103.04	76.50	63.22	142.84	90.53	60.43	45.35

Best results are shown in bold.

TABLE VI
Ranking of Methods Using KNN as Reference

Method	Rank	p-value	Win	Tie	Loss
KNN	1.08	-	-	-	-
HyNN	1.96	**4.0053e-01**	22	0	2
MLP	3.25	**7.4746e-02**	24	0	0
ET	3.71	3.5005e-02	24	0	0
XGB	5.50	8.8065e-05	24	0	0
BR	6.12	6.3665e-06	24	0	0
HGBM	7.25	1.8767e-08	24	0	0
LiR	8.42	1.2923e-11	24	0	0
CNN	9.00	2.2601e-13	24	0	0
DT	9.54	3.9766e-15	24	0	0
GBM	10.33	6.2717e-18	24	0	0
AB	11.83	5.7751e-24	24	0	0

Results with a p-value > 0.05 are shown in bold.

The results are presented in Table V, with the top-performing scenario for each method highlighted in bold. We can elucidate from the results that KNN is the method achieving the best results, but this lazy method is not suitable for real-world applications due to its substantial memory requirements (as it needs to store each training instance). Among all the other methods, HyNN appears to be the most promising method, as it is the only one that outperforms KNN in two scenarios –URA and ULA with 8 antennas for Y-positioning– and reports the second-best results in all other scenarios. However, we deemed necessary to conduct a statistically significant test to back up our findings. To this end, we run a Friedman test with a 0.05 confidence level, using the null hypothesis that all methods are equivalent. This hypothesis is rejected with a p-value of $4.3148e-46$, and we used a post-hoc statistical analysis using the Holm procedure to compare the performance of all classifiers against the one reporting the best results.

The results are presented in Table VI, where we confirm that HyNN and MLP are not significantly different from KNN (with p-values of 0.40053 and 0.074746, respectively), while all other methods show significant differences compared to KNN.

TABLE VII
COMPARISON, IN ME (mm AND λ), OF THE ULA, URA AND DIS SCENARIOS FOR THE HYNN AND THE RESULTS REPORTED IN [7] IN THE TEST PARTITION RESULTS

Antennas	Model	ULA		URA		DIS	
		ME (mm)	ME (λ)	ME (mm)	ME (λ)	ME (mm)	ME (λ)
8	[7]	244.80	2.137	230.08	2.008	197.62	1.725
	HyNN	144.37	1.260	115.80	1.011	162.58	1.419
16	[7]	132.87	1.160	120.73	1.053	152.37	1.330
	HyNN	125.25	1.093	97.69	0.853	113.80	0.993
32	[7]	77.87	0.680	85.19	0.744	130.64	1.140
	HyNN	80.93	0.706	74.33	0.649	73.20	0.639
64	[7]	59.05	0.515	55.35	0.483	82.30	0.718
	HyNN	**43.75**	**0.382**	61.16	0.534	53.36	0.466

Best result is shown in bold.

Although KNN achieves the best results overall, it is discarded for real-world applications due to its substantial memory requirements, as previously explained. This positions both HyNN and MLP as practical and effective solutions for MIMO-based indoor localization tasks, with HyNN offering additional advantages due to its ability to leverage complementary data modalities.

Our ablation study further reveals that HyNN architecture significantly outperforms individual CNN models, highlighting the limitations of relying solely on synthetic image data –a common practice in the literature [22]. Specifically, HyNN combines the strengths of CNNs for spatial feature extraction and MLPs for handling tidy data, resulting in a more robust and comprehensive integration of multimodal features. While the difference between HyNN and MLP is not statistically significant, HyNN demonstrates better performance than MLP in all but one result (see Table V), underscoring its superior capacity to handle multimodal data effectively. Moreover, HyNN consistently achieves improved performance across diverse scenarios, developing a more robust and generalizable model (see Figure 5).

In summary, HyNN's dual-branch architecture captures both numerical and spatial relationships, enabling consistent improvements across scenarios. This versatility and robustness position HyNN as a significant advancement in neural network architectures and a reliable choice for MIMO-based localization tasks.

C. Experiment 3.- Comparison With Outstanding Proposals

The last experiment seeks to compare HyNN with outstanding proposals from the research community. In this case, we have selected the work presented in [7] due to its great results, but also because there are some similarities between both proposals, including the internal use of neural networks.

The comparison is carried out following the procedure presented in [7], and we replicate all the internal parameters during training. Namely, we use exactly the same train, validation and test split by using the same random seeds. We also select the same metrics for being presented, and we report the results using values for ME, in mm and wavelength (λ), as detailed in Section III-E. Differing from previous experiments that assessed X- and Y-positions independently, here we calculated the Euclidean distance between estimated and actual positions, integrating both X and Y components.

Therefore, Table VII shows the obtained results, and we can see how our proposal appears as a promising competitor. The best overall result is obtained for ULA in combination with 64 antennas, and it is obtained by HyNN. The best result gets 43.75 mm as the ME, which allows us to position an agent with an average error of nearly 4 cm.

To ensure a fair comparison between the two solutions, we applied a Wilcoxon signed-rank test. The test yielded a p-value of 0.0012, falling below the 0.05 threshold, leading us to reject the null hypothesis of method equivalence. Consequently, this allows us to expose HyNN as the best method in the overall experimentation. Indeed, a closer look at the results reveals that HyNN improves in 10 out of 12 results (see Table VII). Even in instances where our model did not outperform, the margin of difference was minimal, further affirming the robustness of our proposal. It is worth noting that the model was configured to achieve robust performance in the ULA 8-antenna scenario. This approach ensures generalizability of our findings across configurations, but further refinement specific to each scenario could yield additional improvements (see Section V-D).

VII. CONCLUSION & OPEN CHALLENGES

This article explored the transformation of tidy data into synthetic images to process MIMO signals, aiming to develop advanced wireless indoor localization mechanisms. Our results demonstrate that converting tidy data into synthetic images significantly enhances the model's learning and generalization capabilities. We show that HyNNs, which integrate a CNN branch for analysing synthetic images generated by TINTO and an MLP branch for tidy data, offer substantial improvements in model performance and stability, surpassing traditional CNN-based approaches. Additionally, our results indicate improvements not only over traditional ML algorithms but also compared to the results presented in [7]. Furthermore, our findings confirm that increasing the number of antennas enhances the generalization of the HyNN, with the 64-antenna configuration achieving a significative precision.

Some open challenges remain to be addressed. One key area for future development is the optimization of the blurring technique to follow signal degradation more accurately, rather than relying on empirical methods as done in this study. Utilizing channel propagation models to develop the blurring of characteristic pixels in the image, aligning it with the signal's degradation, could lead to a hybrid communications-based blurring approach.

Moreover, the novel HyNN architecture presented in this paper, based on synthetic image generation, can be further enhanced by integrating advanced models such as Vision Transformers or Kolmogorov-Arnold Networks to replace either the CNN branch or the MLP branch, respectively. An interesting avenue for future work is exploring Transformer Encoder-based concatenation methods to analyse which features from tidy data or images are most informative. These approaches could enhance feature selection and attention mechanisms, enabling a more dynamic fusion of features between the CNN and MLP branches, and offering greater flexibility in adapting to different data types and environments. Another significant challenge is evaluating the model's performance in larger and more complex environments, including multi-level buildings, to understand the impact of precision in such scenarios.

VII. Data availability

This scientific experiment contains the extended data and source code available in Zenodo [32]:

References

[1] R. Talla-Chumpitaz, M. Castillo-Cara, L. Orozco-Barbosa, and R. García-Castro, "A novel deep learning approach using blurring image techniques for bluetooth-based indoor localisation," *Inf. Fusion*, vol. 91, pp. 173–186, 2023.

[2] B. Teng, X. Yuan, R. Wang, and S. Jin, "Bayesian user localization and tracking for reconfigurable intelligent surface aided MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 5, pp. 1040–1054, Aug. 2022.

[3] X. Lin, J. Gan, C. Jiang, S. Xue, and Y. Liang, "Wi-Fi-Based indoor localization and navigation: A robot-aided hybrid deep learning approach," *Sensors*, vol. 23, no. 14, 2023, Art. no. 6320.

[4] A. Singh, M. Emam, and Y. Al Mtawa, "Comparative analysis of indoor localization across various wireless technologies," *Eng*, vol. 4, no. 3, pp. 2293–2308, 2023.

[5] S. J. Hayward, K. van Lopik, C. Hinde, and A. A. West, "A survey of indoor location technologies, techniques and applications in industry," *Internet Things*, vol. 20, 2022, Art. no. 100608.

[6] J. Lovon-Melgarejo, M. Castillo-Cara, L. Orozco-Barbosa, and I. García-Varea, "Supervised learning algorithms for indoor localization fingerprinting using BLE4.0 beacons," in *Proc. IEEE Latin Amer. Conf. Computacional Intell.*, 2017, pp. 1–6.

[7] S. D. Bast, A. P. Guevara, and S. Pollin, "CSI-based positioning in massive MIMO systems using convolutional neural networks," in*Proc. IEEE 91st Veh. Technol. Conf.*, Antwerp, Belgium, 2020, pp. 1–5.

[8] C. Li, S. De Bast, E. Tanghe, S. Pollin, and W. Joseph, "Toward fine-grained indoor localization based on massive MIMO-OFDM system: Experiment and analysis," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5318–5328, Mar. 2022.

[9] S. Sadowski, P. Spachos, and K. N. Plataniotis, "Memoryless techniques and wireless technologies for indoor localization with the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10996–11005, Nov. 2020.

[10] H. Obeidat, W. Shuaieb, O. Obeidat, and R. Abd-Alhameed, "A review of indoor localization techniques and wireless technologies," *Wireless Pers. Commun.*, vol. 119, pp. 289–327, 2021.

[11] Y. Han, S. Jin, C.-K. Wen, and T. Q. S. Quek, "Localization and channel reconstruction for extra large RIS-assisted massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 5, pp. 1011–1025, Aug. 2022.

[12] O. Alamu, B. Iyaomolere, and A. Abdulrahman, "An overview of massive MIMO localization techniques in wireless cellular networks: Recent advances and outlook," *Ad Hoc Networks*, vol. 111, 2021, Art. no. 102353.

[13] H. Zhang, H. Zhang, B. Di, K. Bian, Z. Han, and L. Song, "MetaLocalization: Reconfigurable intelligent surface aided multi-user wireless indoor localization," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7743–7757, Dec. 2021.

[14] M. Widmaier, M. Arnold, S. Dorner, S. Cammerer, and S. ten Brink, "Towards practical indoor positioning based on massive MIMO systems," in *Proc. IEEE 90th Veh. Technol. Conf.*, 2019, pp. 1–6.

[15] M. Castillo-Cara, R. Talla-Chumpitaz, Raúl García-Castro, and L. Orozco-Barbosa, "TINTO: Converting tidy data into image for classification with 2-dimensional convolutional neural networks," *SoftwareX*, vol. 22, 2023, Art. no. 101391.

[16] Y. Zhu et al., "Converting tabular data into images for deep learning with convolutional neural networks," *Sci. Reports*, vol. 11, 2021, Art. no. 11325.

[17] V. Gómez-Martínez et al., "LM-IGDT: A 2D image generator for low-dimensional and mixed-type tabular data to leverage the potential of convolutional neural networks," 2024, *arXiv:2406.14566*.

[18] A. Sharma, Y. López, S. Jia, A. Lysenko, K. A. Boroevich, and T. Tsunoda, "Multi-representation DeepInsight: An improvement on tabular data analysis," in *bioRxiv*, 2023.

[19] O. Bazgir, R. Zhang, S. R. Dhruba, R. Rahman, S. Ghosh, and R. Pal, "Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks," *Nature Commun.*, vol. 11, 2020, Art. no. 4391.

[20] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Sci. Reports*, vol. 9, 2019, Art. no. 11399.

[21] A. Sharma and D. Kumar, "Classification with 2-D convolutional neural networks for breast cancer diagnosis," *Sci. Reports*, vol. 12, 2022, Art. no. 21857.

[22] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 35, no. 6, pp. 7499–7519, Jun. 2024.

[23] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Inf. Fusion*, vol. 81, pp. 84–90, 2022.

[24] M. Mallik and C. Chowdhury, "Characteristic analysis of fingerprint datasets from a pragmatic view of indoor localization using machine learning approaches," *J. Supercomputing*, vol. 79, pp. 18507–18546, 2023.

[25] S. De Bast and S. Pollin, "Ultra dense indoor MaMIMO CSI dataset," *IEEE Dataport*, 2021, doi: 10.21227/nr6k-8r78.

[26] F. Wen, H. Wymeersch, B. Peng, W. P. Tay, H. C. So, and D. Yang, "A survey on 5G massive MIMO localization," *Digit. Signal Process.*, vol. 94, pp. 21–28, 2019.

[27] G. Tian, I. Yaman, M. Sandra, X. Cai, L. Liu, and F. Tufvesson, "High-precision machine-learning based indoor localization with massive MIMO system," in *Proc.IEEE Int. Conf. Commun.*, 2023, pp. 3690–3695.

[28] X. Gong, X. Yu, X. Liu, and X. Gao, "Machine learning-based fingerprint positioning for massive MIMO systems," *IEEE Access*, vol. 10, pp. 89320–89330, 2022.

[29] B. Sun et al., "SuperTML: Two-dimensional word embedding for the precognition on structured tabular data," in *Proc. 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 2973–2981.

[30] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong, and C. Young, "Super characters: A conversion from sentiment classification to image classification," in *Proc. 9th Workshop Comput. Approaches Subjectivity, Sentiment Social Media Anal.*, 2018, pp. 309–315.

[31] Manuel Castillo-Cara and Raúl García-Castro, TINTOlib, 2023. Accessed: Dec. 18, 2024. [Online]. Available: https://tintolib.readthedocs.io/en/latest/

[32] M. Castillo-Cara et al., "MIMO indoor localization-based a hybrid neural network approach transforming tidy data into synthetic images," Dec., 2024, doi: 10.5281/zenodo.14289235.