

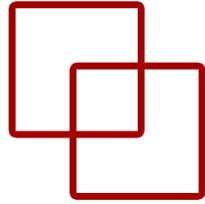
Improving Deep Learning by Exploiting Synthetic Images



Dr. Manuel Castillo-Cara
www.manuelcastillo.eu

Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)

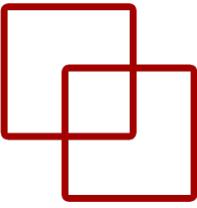
Preliminar



- Improving Deep Learning by Exploiting Synthetic Images © 2024 by Manuel Castillo-Cara is licensed under Attribution-NonCommercial 4.0 International



Índice



- Background
- Arquitecturas unitarias
- Assessment Criteria
- Conversion methods
- TINTOlib
- Hybrid Architectures
- Use cases

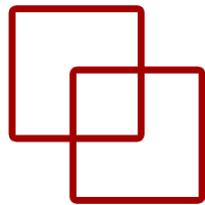
Background

ETS de
Ingeniería
Informática



UNED

Regression vs. Classification



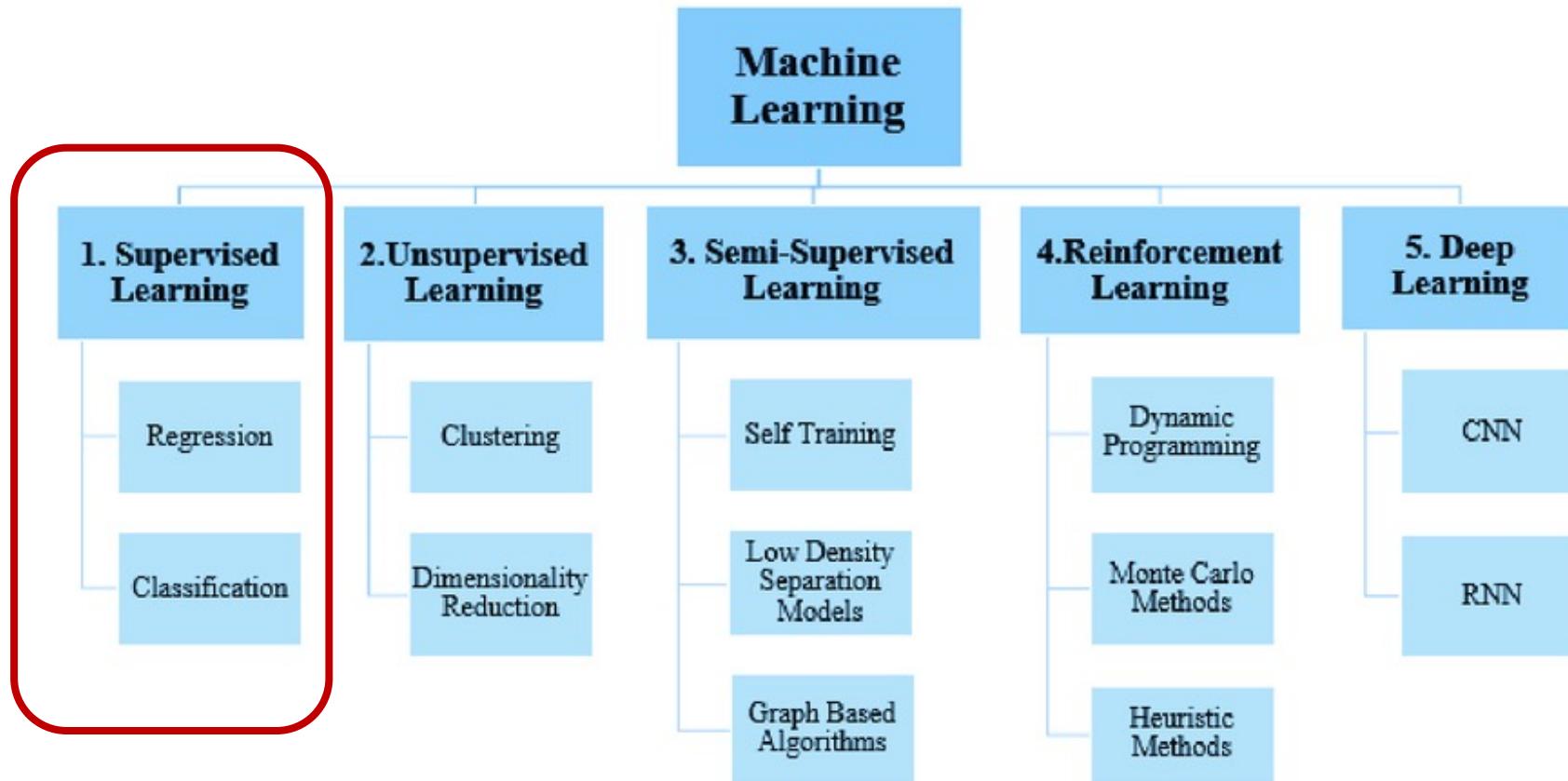
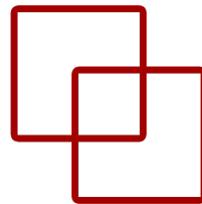
| | A | B | C | D | E | F | G | H | I | |
|---|------|------|------|------|------|------|------|-----|-----------------|--|
| 1 | preg | plas | pres | skin | insu | mass | pedi | age | class | |
| 2 | 1 | 85 | 66 | 29 | 0 | 26.6 | 351 | 31 | tested_negative | |
| 3 | 5 | 116 | 74 | 0 | 0 | 25.6 | 201 | 30 | tested_negative | |
| 4 | 10 | 115 | 0 | 0 | 0 | 35.3 | 134 | 29 | tested_negative | |
| 5 | 4 | 110 | 92 | 0 | 0 | 37.6 | 191 | 30 | tested_negative | |
| 6 | 10 | 139 | 80 | 0 | 0 | 27.1 | 1441 | 57 | tested_negative | |
| 7 | 8 | 99 | 84 | 0 | 0 | 35.4 | 388 | 50 | tested_negative | |
| 8 | 5 | 117 | 92 | 0 | 0 | 34.1 | 337 | 38 | tested_negative | |
| 9 | 5 | 109 | 75 | 26 | 0 | 36 | 546 | 60 | tested_negative | |

Classification

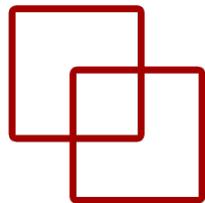
Regression

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---------|--------|-------|------|-----|------|------|--------|-----|-----|---------|--------|-------|-------|
| 1 | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | class |
| 2 | 0.00632 | 18.231 | | 0 | 538 | 6575 | 65.2 | 4.09 | 1 | 296 | 15.3 | 396.9 | 4.98 | 24 |
| 3 | 0.02731 | 0.707 | | 0 | 469 | 6421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.9 | 9.14 | 21.6 |
| 4 | 0.02729 | 0.707 | | 0 | 469 | 7185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| 5 | 0.03237 | 0.218 | | 0 | 458 | 6998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| 6 | 0.06905 | 0.218 | | 0 | 458 | 7147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.9 | 5.33 | 36.2 |
| 7 | 0.02985 | 0.218 | | 0 | 458 | 6.43 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 394.12 | 5.21 | 28.7 |

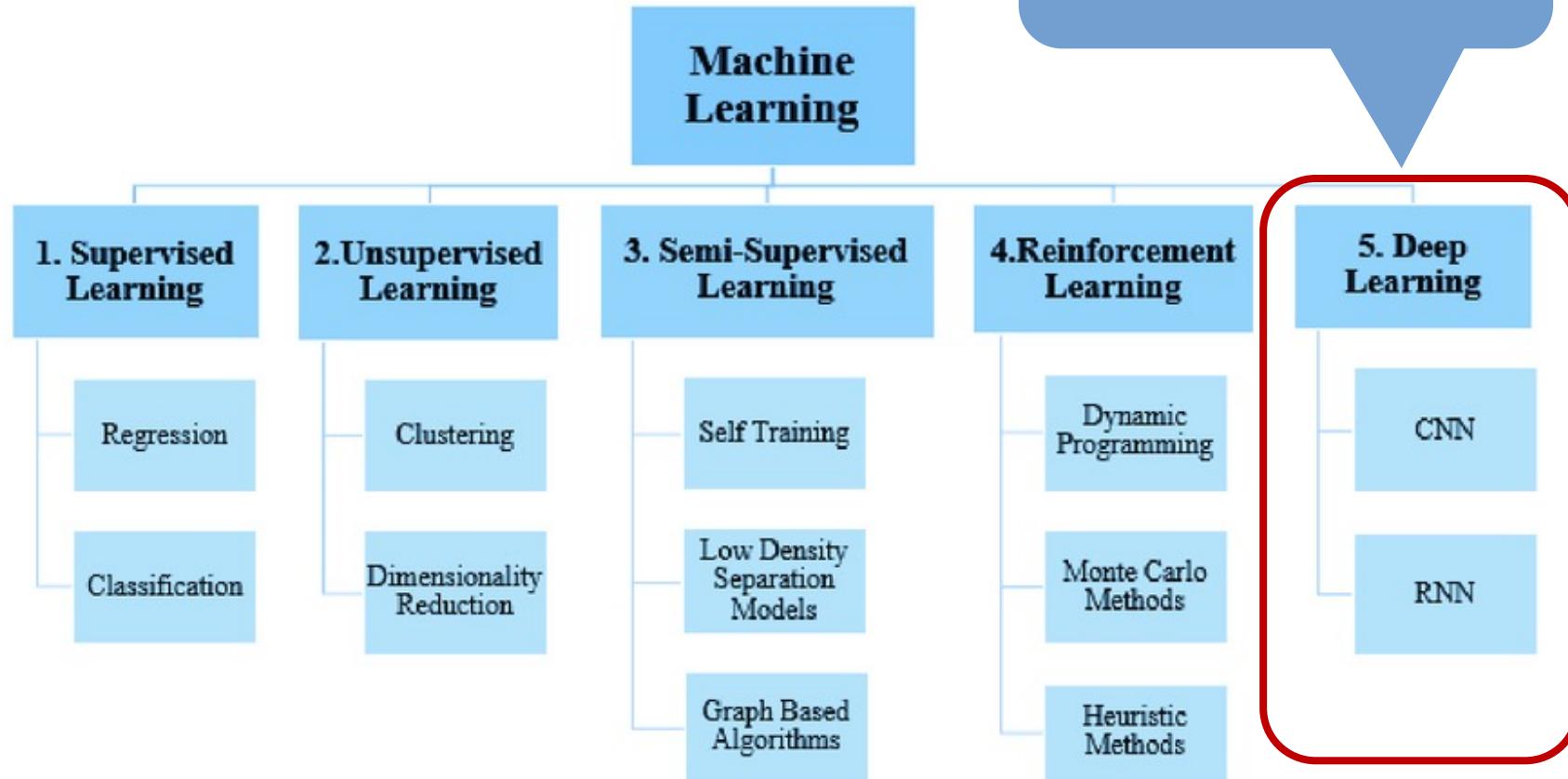
Algoritmos



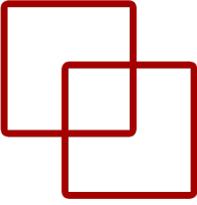
Algoritmos



Nos interesa
estos

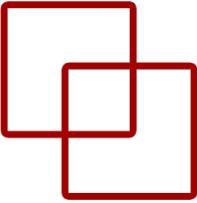


Problema de investigación abierto



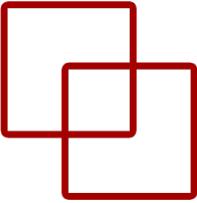
- Kadra et al. nombró datasets como “*last unconquered castle*” para modelos basados en Deep Neural Networks (DNN).
- La adaptación de las DNN a Datos Tabulares (TD) para tareas de inferencia o generación de datos sigue siendo un **gran desafío**.

Problema de investigación abierto



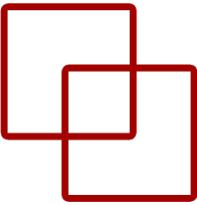
- Kadra et al. nombró datasets como “***last unconquered castle***” para modelos basados en Deep Neural Networks (DNN).
- La adaptación de las DNN a Datos Tabulares (TD) para tareas de inferencia o generación de datos sigue siendo un **gran desafío**.
- Vadim Borísov et al. hacen un benchmark de diferentes métodos/datasets entre ensembles vs. DNN y **ganar los ensembles**. De hecho, afirman:
 - “El progreso de la investigación sobre modelos competitivos de aprendizaje profundo para datos tabulares **se está estancando**”.
 - “Es un **área de investigación abierta**”.

Problema de investigación abierto



- Kadra et al. nombró datasets como “***last unconquered castle***” para modelos basados en Deep Neural Networks (DNN).
- La adaptación de las DNN a Datos Tabulares (TD) para tareas de inferencia o generación de datos sigue siendo un **gran desafío**.
- Vadim Borísov et al. hacen un benchmark de diferentes métodos/datasets entre ensembles vs. DNN y **ganar los ensembles**. De hecho, afirman:
 - “El progreso de la investigación sobre modelos competitivos de aprendizaje profundo para datos tabulares **se está estancando**”.
 - “Es un **área de investigación abierta**”.
- ShwartzZiv y Armon en su artículo “*Tabular data: Deep learning is not all you need*”, compararon enfoques de DNN vs. árbol de decisión de aumento de gradiente (GBDT). **Los GBDT cuestionaron los DNN**, lo que concluyeron que el modelado de datos tabulares utilizando DNN sigue siendo un **problema de investigación abierto**.

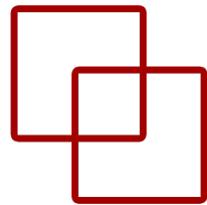
Problema de investigación abierto



- Kadra et al. nombró datasets como “***last unconquered castle***” para modelos basados en Deep Neural Networks (DNN).
- La adaptación de las DNN a Datos Tabulares (TD) para tareas de inferencia o generación de datos sigue siendo un **gran desafío**.
- Vadim Borísov et al. hacen un benchmark de diferentes métodos/datasets entre ensembles vs. DNN y **ganar los ensembles**. De hecho, afirman:
 - “El progreso de la investigación sobre modelos competitivos de aprendizaje profundo para datos tabulares **se está estancando**”.
 - “Es un **área de investigación abierta**”.
- ShwartzZiv y Armon en su artículo “*Tabular data: Deep learning is not all you need*”, compararon enfoques de DNN vs. árbol de decisión de aumento de gradiente (GBDT). **Los GBDT cuestionaron los DNN**, lo que concluyeron que el modelado de datos tabulares utilizando DNN sigue siendo un **problema de investigación abierto**.
- Las implementaciones exitosas de aplicaciones basadas en datos requieren resolver varias tareas, entre las cuales identificamos **3 desafíos centrales**: (1) Inferencia; (2) Generación de datos; y (3) Interpretabilidad.
 - **La tarea más crucial es la inferencia**, que se ocupa de hacer predicciones basadas en observaciones pasadas.

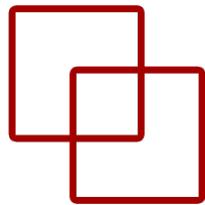
¿Por qué no se mejora con DNN?

Cuatro posibles razones

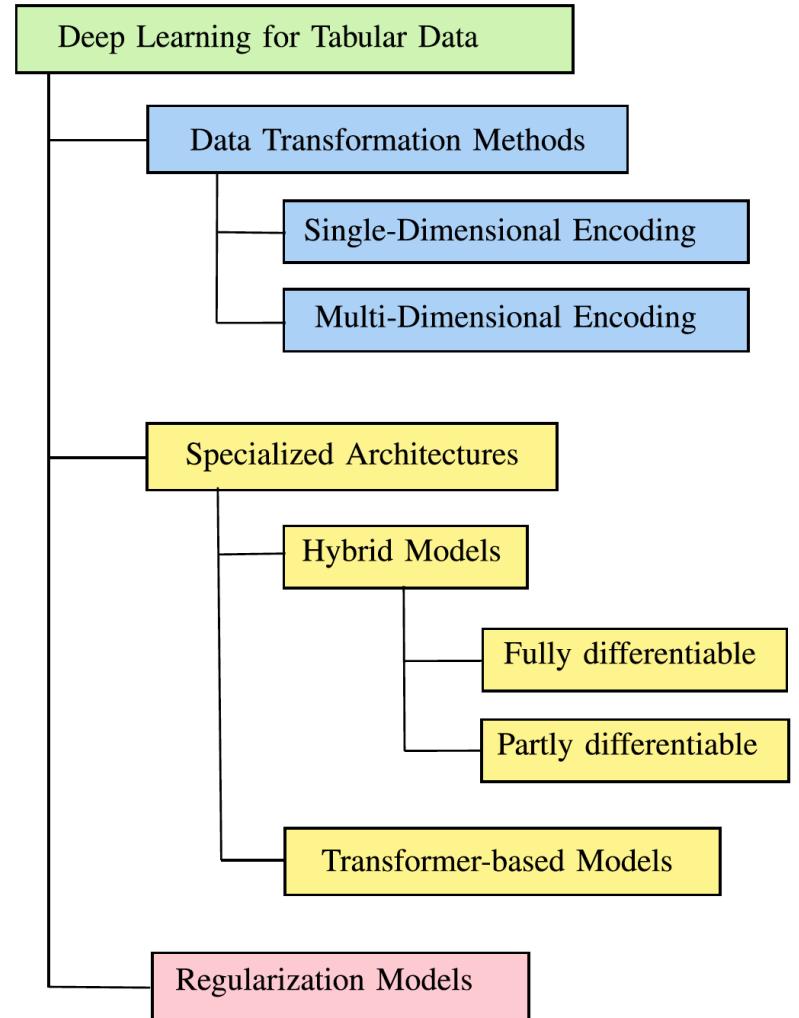


- **Datos de entrenamiento de baja calidad:** valores faltantes, valores atípicos, corruptos y datos erróneos o inconsistentes, desequilibrio de clases. Esto afecta a todos los algoritmos; sin embargo, la mayoría de los algoritmos basados en árboles de decisión pueden manejar valores faltantes o rangos de variables diferentes/extremos internamente buscando aproximaciones apropiadas y valores divididos.
- **Dependencias espaciales irregulares faltantes o complejas:** a menudo no existe correlación espacial entre las variables en conjuntos de datos tabulares o las dependencias entre características son bastante complejas e irregulares. Cuando se trabaja con datos tabulares, la estructura y las relaciones entre sus características deben aprenderse desde cero. Por lo tanto, los sesgos inductivos utilizados en modelos populares para datos homogéneos, como las CNN, no son adecuados para modelar este tipo de datos.
- **Dependencia del preprocessamiento:** una ventaja clave del aprendizaje profundo sobre datos homogéneos es que incluye un paso de aprendizaje de representación implícito, por lo que solo se requiere una cantidad mínima de preprocessamiento o construcción explícita de características. Sin embargo, para datos tabulares y DNN, el rendimiento puede depender en gran medida de la estrategia de preprocessamiento seleccionada. Finalmente, los métodos de preprocessamiento para redes neuronales profundas pueden conducir a la pérdida de información, lo que lleva a una reducción en el rendimiento predictivo.
- **Importancia de las características individuales:** si bien normalmente cambiar la clase de una imagen requiere un cambio coordinado en muchas características, es decir, píxeles, el cambio más pequeño posible de una característica categórica (o binaria) puede invertir por completo una predicción en datos tabulares. A diferencia de las DNN, los algoritmos de árbol de decisiones pueden manejar excepcionalmente bien la importancia. [Shavitt y Segal](#) han argumentado que la regularización del peso individual puede mitigar este desafío y han motivado más trabajo en esta dirección.

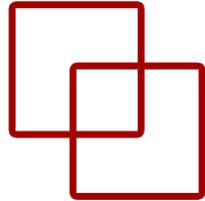
Taxonomía en los enfoques



- **Métodos de transformación de datos:** Transforman datos categóricos y numéricos. Esto generalmente se hace para permitir que los modelos DNN extraigan mejor la señal de información. Los métodos de este grupo no requieren nuevas arquitecturas ni adaptaciones del proceso de procesamiento de datos existente. Podemos subdividir aún más esta área en codificaciones unidimensionales (para transformar cada característica de forma independiente) y codificaciones multidimensionales (asignan un registro completo a otra representación).
- **Arquitecturas especializadas:** Se requiere una arquitectura de DNN diferente para los datos tabulares. Dos tipos de arquitecturas son de particular importancia: los modelos híbridos fusionan enfoques clásicos de aprendizaje automático (por ejemplo, árboles de decisión) con redes neuronales, mientras que los modelos basados en Transformers se basan en mecanismos de atención a la pérdida de información.
- **Modelos de regularización:** Afirma que una de las principales razones del rendimiento moderado de los modelos de aprendizaje profundo en datos tabulares es su extrema no linealidad y complejidad del modelo. Por lo tanto, se proponen como solución fuertes esquemas de regularización.

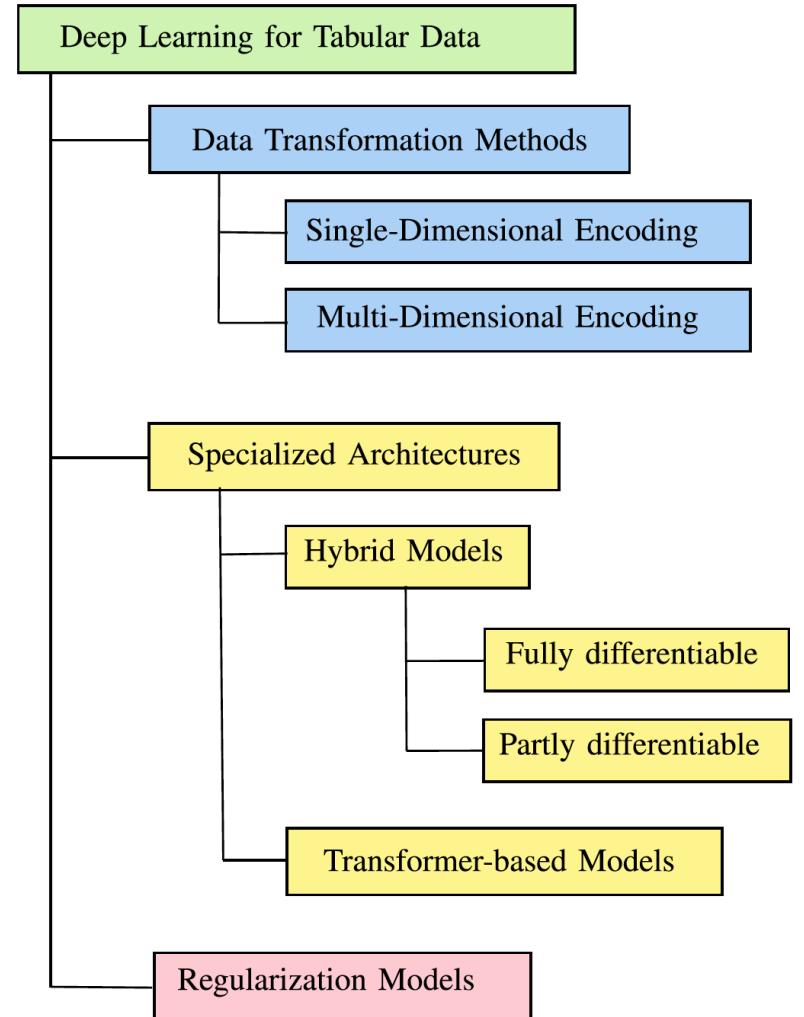


Taxonomía en los enfoques

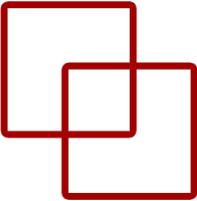


- **Métodos de transformación de datos:** Transforman datos categóricos y numéricos. Esto generalmente se hace para permitir que los modelos DNN extraigan mejor la señal de información. Los métodos de este grupo no requieren nuevas arquitecturas ni adaptaciones del proceso de procesamiento de datos existente. Podemos subdividir aún más esta área en codificaciones unidimensionales (para transformar cada característica de forma independiente) y codificaciones multidimensionales (asignan un registro completo a otra representación).
- **Arquitecturas especializadas:** Se refiere a las arquitecturas que están diseñadas específicamente para los datos tabulares. Dos tipos de arquitecturas especializadas son particularmente importantes: los modelos híbridos fusionados y los modelos basados en transformadores. Los modelos híbridos fusionados combinan el aprendizaje automático (por ejemplo, árboles de decisión) con el aprendizaje profundo. Los modelos basados en transformadores se basan en mecanismos de atención a la pérdida de información.
- **Modelos de regularización:** Afirma que una de las principales razones del rendimiento moderado de los modelos de aprendizaje profundo en datos tabulares es su extrema no linealidad y complejidad del modelo. Por lo tanto, se proponen como solución fuertes esquemas de regularización.

Desechamos esta
para nuestros
fines

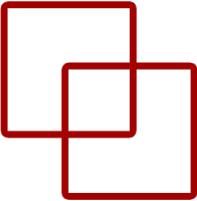


Taxonomía en los enfoques



| | Method | Interpretability | Key Characteristics |
|----------------------------|-----------------------------|------------------|--|
| Encoding | SuperTML [78] | | Transform tabular data into images for CNNs |
| | VIME [79] | | Self-supervised learning and contextual embedding |
| | IGTD [72] | | Transform tabular data into images for CNNs |
| | SCARF [80] | | Self-supervised contrastive learning |
| Architectures, Hybrid | Wide&Deep [81] | | Embedding layer for categorical features |
| | DeepFM [15] | | Factorization machine for categorical data |
| | SDT [82] | ✓ | Distill neural network into interpretable decision tree |
| | xDeepFM [83] | | Compressed interaction network |
| | TabNN [84] | | DNNs based on feature groups distilled from GBDT |
| | DeepGBM [62] | | Two DNNs, distill knowledge from decision tree |
| | NODE [7] | | Differentiable oblivious decision trees ensemble |
| | NAM [85] | ✓ | Separate neural networks for each input variable |
| | NON [86] | | Network-on-network model |
| | DNN2LR [87] | | Calculate cross feature yields with DNNs for LR |
| | Net-DNF [50] | | Structure based on disjunctive normal form |
| | Boost-GNN [88] | | GNN on top decision trees from the GBDT algorithm |
| | SDTR [89] | | Hierarchical differentiable neural regression model |
| | TabNet [6] | ✓ | Sequential attention structure |
| Architectures, Transformer | TabTransformer [90] | ✓ | Transformer network for categorical data |
| | SAINT [9] | ✓ | Attention over both rows and columns |
| | ARM-Net [91] | | Adaptive relational modeling with multi-headgated attention network |
| | Non-Param. Transformer [92] | | Process the entire data set at once, use attention between data points |
| | RLN [63] | ✓ | Hyperparameters regularization scheme |
| Regul. | STG [93] | | Stochastic gate regularization |
| | Regularized DNNs [10] | | A "cocktail" of regularization techniques |

Taxonomía en los enfoques

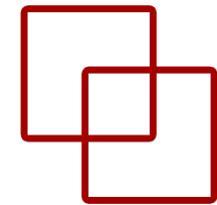


| | Method | Interpretability | Key Characteristics |
|----------------------------|-----------------------------|------------------|--|
| Encoding | SuperTML [78] | | Transform tabular data into images for CNNs |
| | VIME [79] | | Self-supervised learning and contextual embedding |
| | IGTD [72] | | Transform tabular data into images for CNNs |
| | SCARF [80] | | Self-supervised contrastive learning |
| Architectures, Hybrid | Wide&Deep [81] | | Embedding layer for categorical features |
| | DeepFM [15] | | Factorization machine for categorical data |
| | SDT [82] | ✓ | Distill neural network into interpretable decision tree |
| | xDeepFM [83] | | Compressed interaction network |
| | TabNN [84] | | DNNs based on feature groups distilled from GBDT |
| | DeepGBM [62] | | Two DNNs, distill knowledge from decision tree |
| | NODE [7] | | Differentiable oblivious decision trees ensemble |
| | NAM [85] | ✓ | Separate neural networks for each input variable |
| | NON [86] | | Network-on-network model |
| | DNN2LR [87] | | Calculate cross feature yields with DNNs for LR |
| | Net-DNF [50] | | Structure based on disjunctive normal form |
| | Boost-GNN [88] | | GNN on top decision trees from the GBDT algorithm |
| | SDTR [89] | | Hierarchical differentiable neural regression model |
| | | | |
| Architectures, Transformer | TabNet [6] | ✓ | Sequential attention structure |
| | TabTransformer [90] | ✓ | Transformer network for categorical data |
| | SAINT [9] | ✓ | Attention over both rows and columns |
| | ARM-Net [91] | | Adaptive relational modeling with multi-headgated attention network |
| | Non-Param. Transformer [92] | | Process the entire data set at once, use attention between data points |
| Regul. | RLN [63] | ✓ | Hyperparameters regularization scheme |
| | STG [93] | | Stochastic gate regularization |
| | Regularized DNNs [10] | | A "cocktail" of regularization techniques |

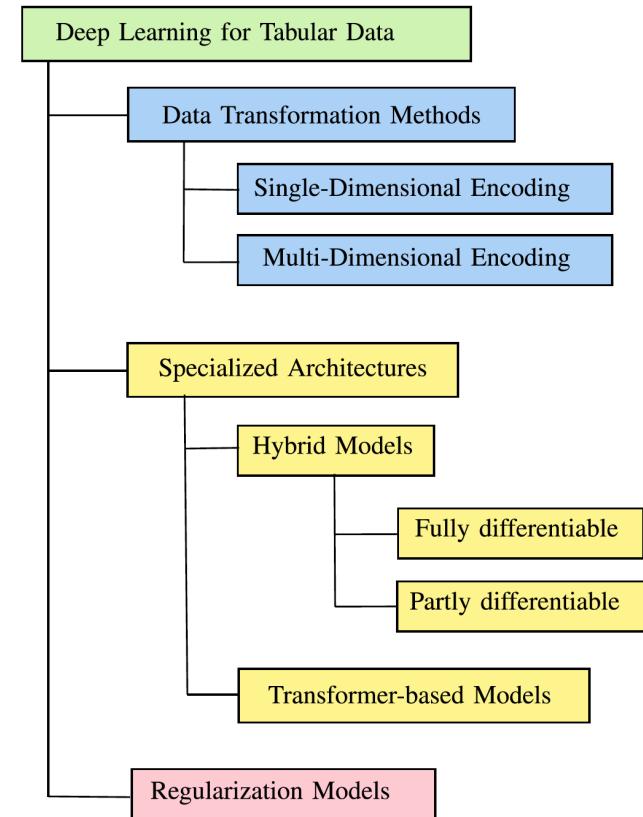
Nos centramos en estos

Taxonomías

Arquitecturas especializadas

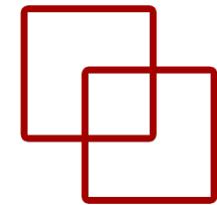


- Se centra en el desarrollo de nuevas arquitecturas DNN diseñadas específicamente para datos tabulares heterogéneos.
- Dos grupos: (i) : Modelos híbridos; y (ii) Modelos basados en Transformers.
- **Modelos híbridos:** Integran diferentes tipos de algoritmos y DNN, e.g., árboles de decisión para variables categóricas y MLP para numéricas que son finalmente unidas dando una predicción final.
- **Modelos basados en Transformers:** Utilizan mecanismos de atención para datos tabulares heterogéneos.
 - En [TabNet](#) este mecanismo selecciona los atributos más relevantes en cada paso del proceso de modelado, mejorando la métrica y agregando **explicabilidad** en las predicciones.
 - Pero hay más, [Tab-Transformers](#), [ARM-NET](#)...

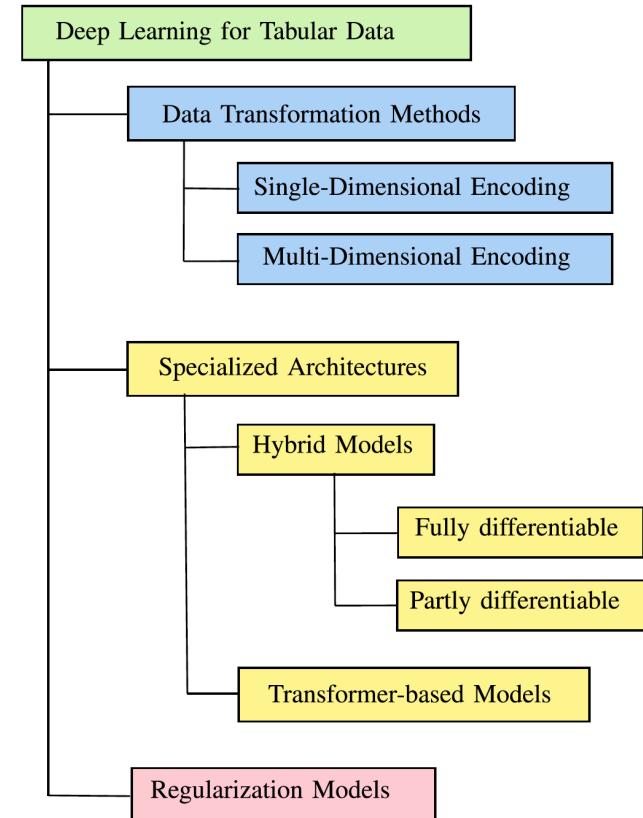


Taxonomías

Métodos de transformación de datos

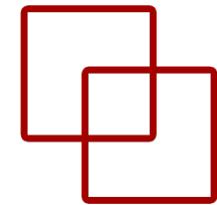


- **Codificación unidimensional:** uno de los obstáculos críticos para el aprendizaje profundo con datos tabulares son las variables categóricas.
 - Las DNN solo aceptan vectores de números reales, se deben transformar antes de usarlos, e.g., {Apple, Banana} = {0, 1} o con OHE.
- **Codificación multidimensional:** Varias aproximaciones
 - Red *Value Imputation and Mask Estimation* (VIME) que entrena un codificador que transforma las características categóricas y numéricas en una nueva representación homogénea e informativa.



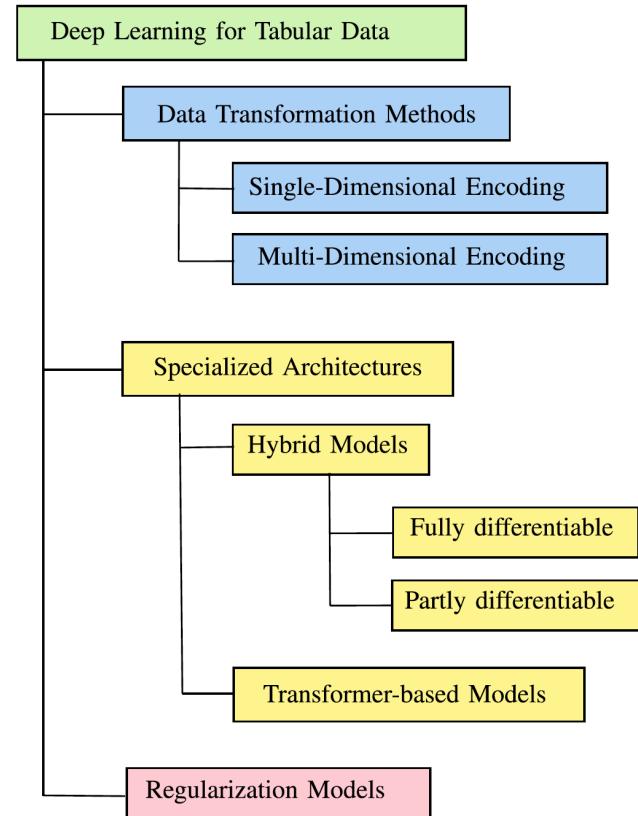
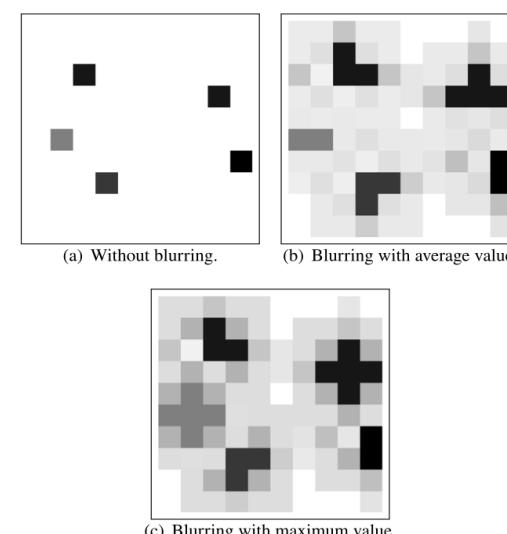
Taxonomías

Métodos de transformación de datos



- **Codificación unidimensional:** uno de los obstáculos críticos para el aprendizaje profundo con datos tabulares son las variables categóricas.
 - Las DNN solo aceptan vectores de números reales, se deben transformar antes de usarlos, e.g., {Apple, Banana} = {0, 1} o con OHE.
- **Codificación multidimensional:** Varias aproximaciones
 - Red *Value Imputation and Mask Estimation* (VIME) que entrena un codificador que transforma las características categóricas y numéricas en una nueva representación homogénea e informativa.
 - **Transformar datos tabulares a un formato más homogéneo, i.e., imágenes sintéticas**, i.e., TINTO, IGTD, REFIND, etc.

| 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 |



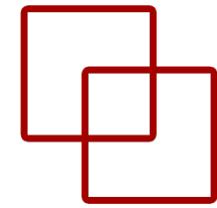
\Arquitecturas unitarias

ETS de
Ingeniería
Informática

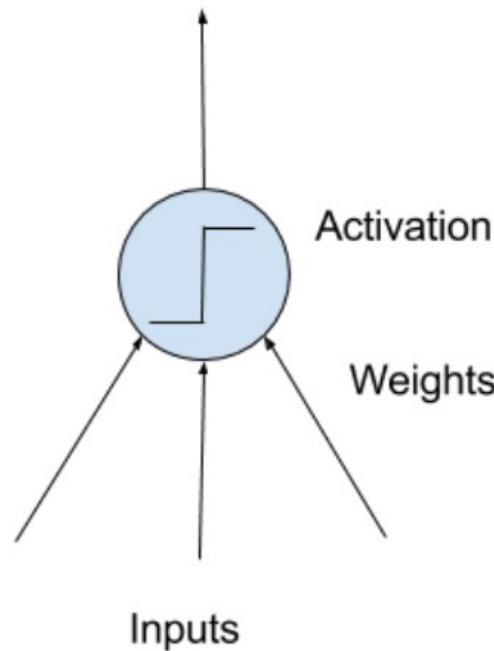


UNED

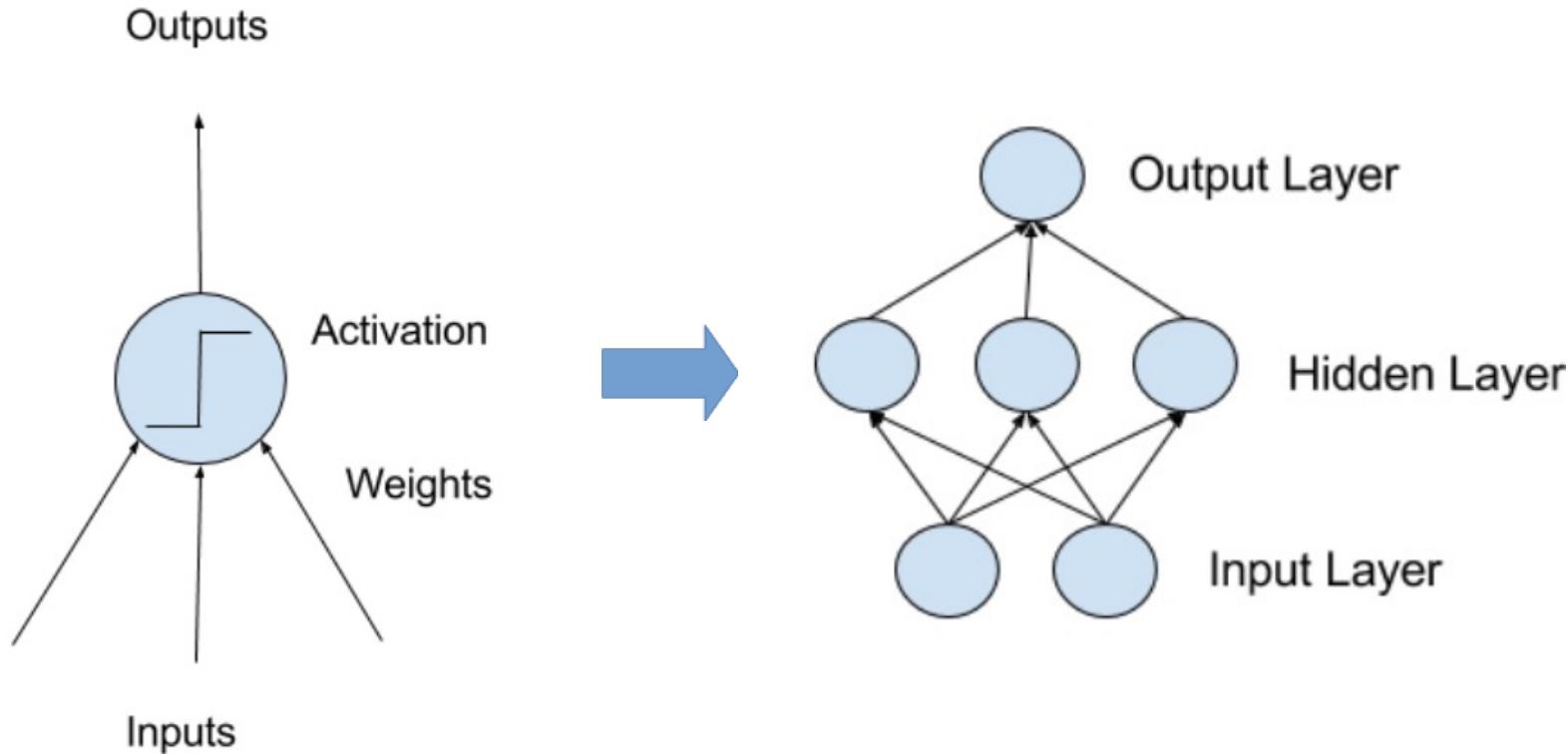
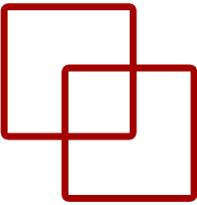
Multilayer Perceptrón



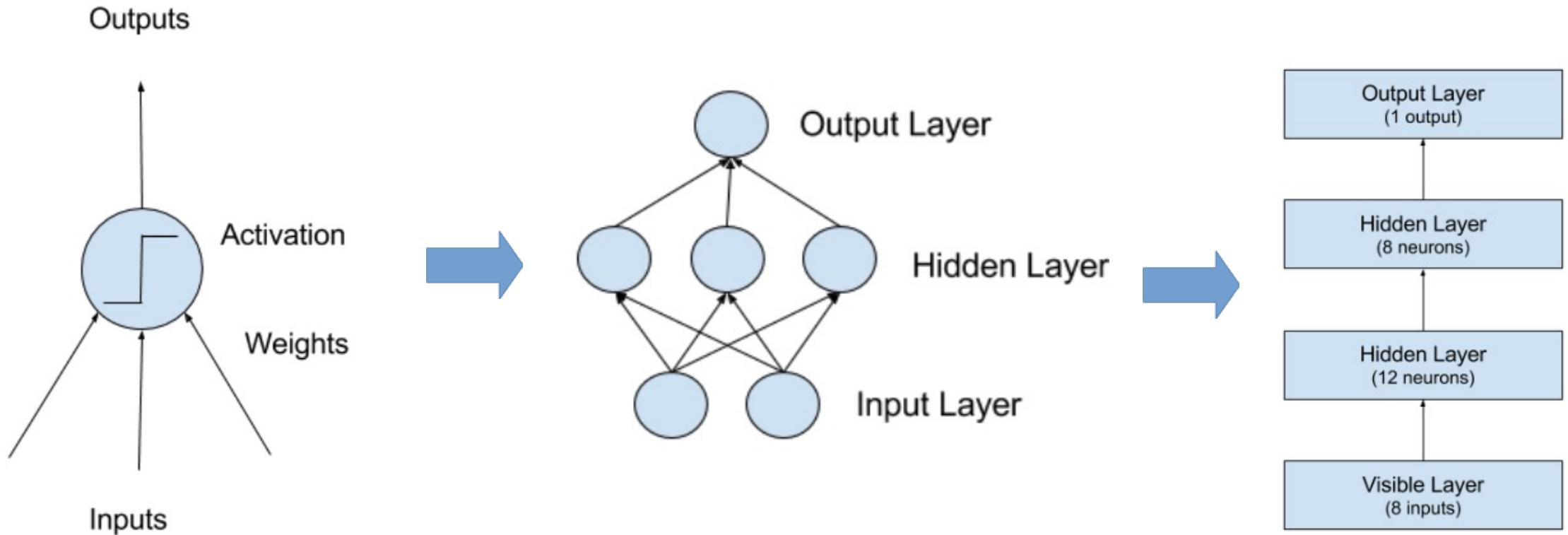
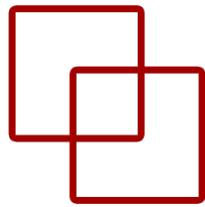
Outputs



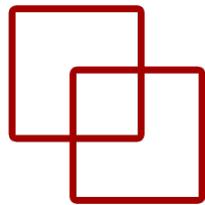
Multilayer Perceptrón



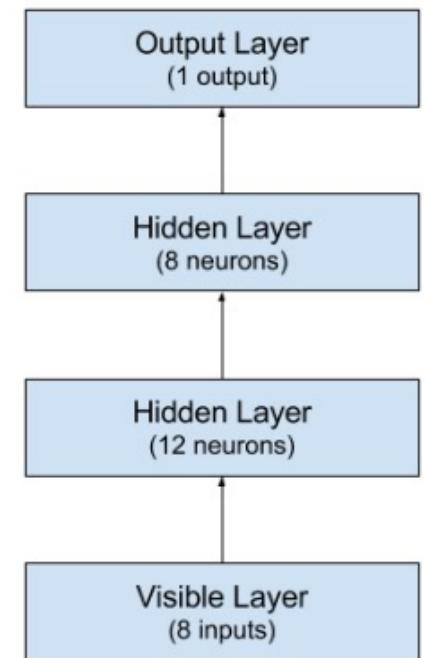
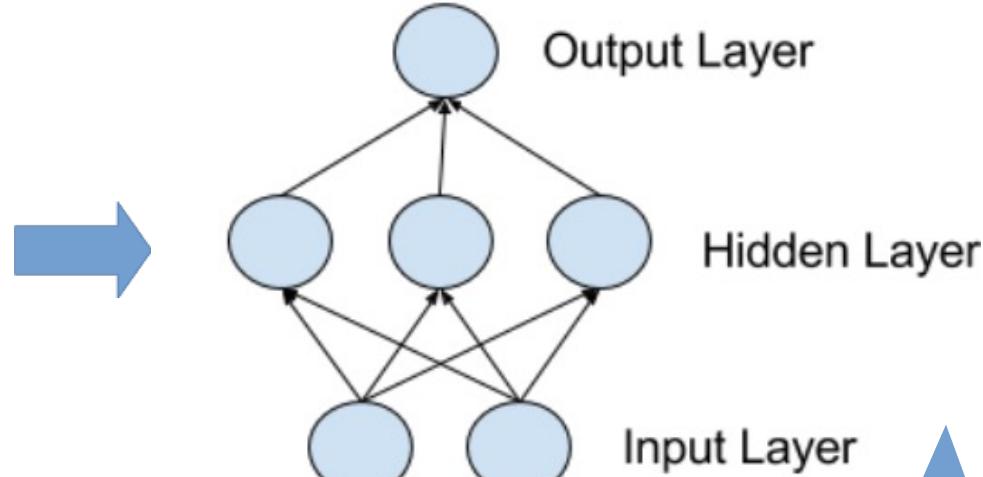
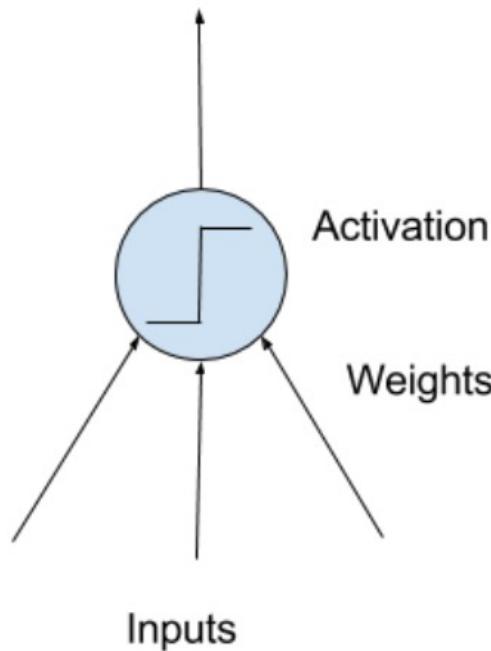
Multilayer Perceptrón



Multilayer Perceptrón

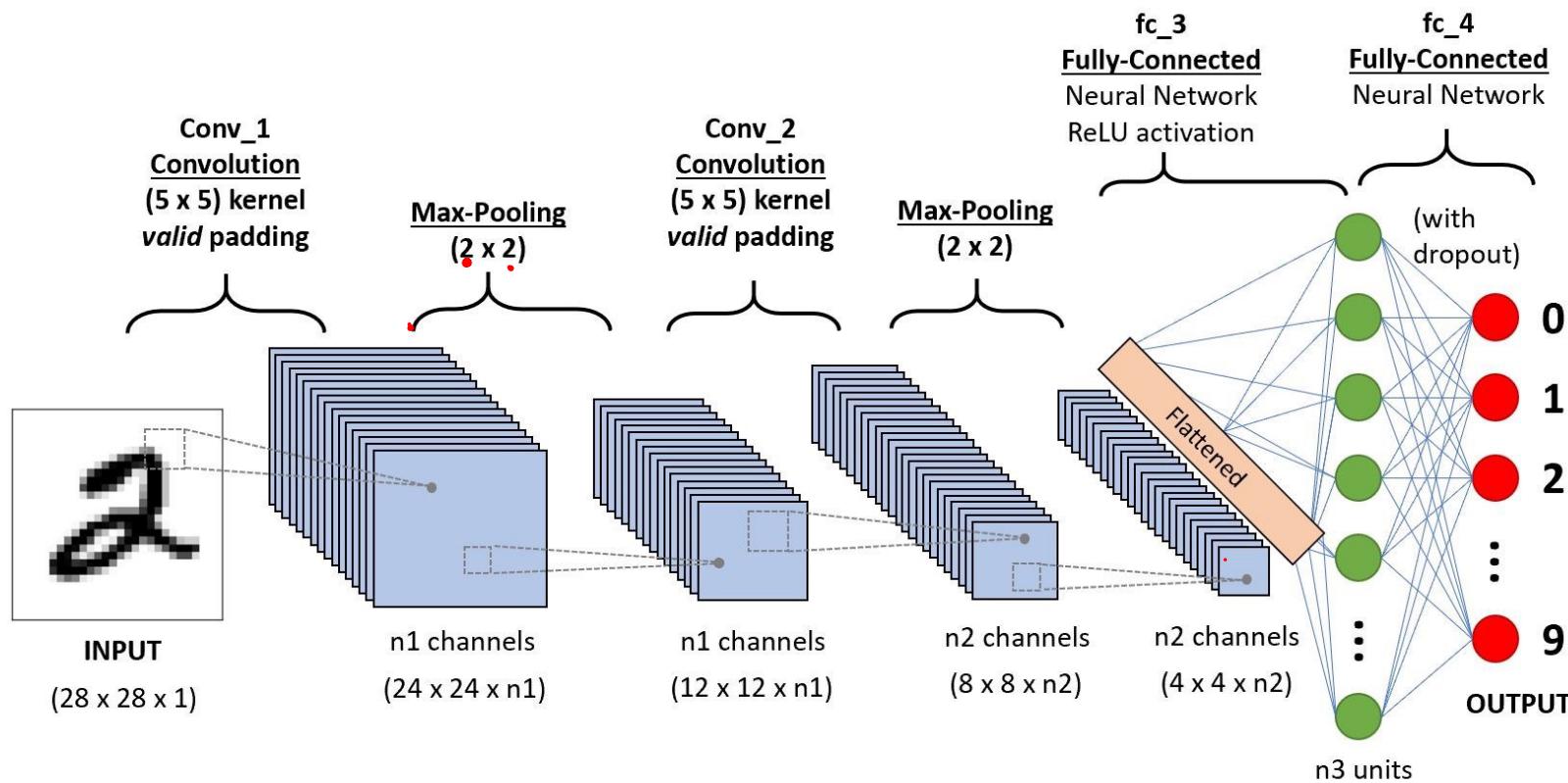
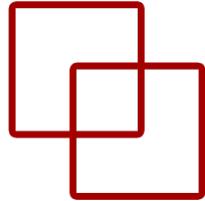


Outputs

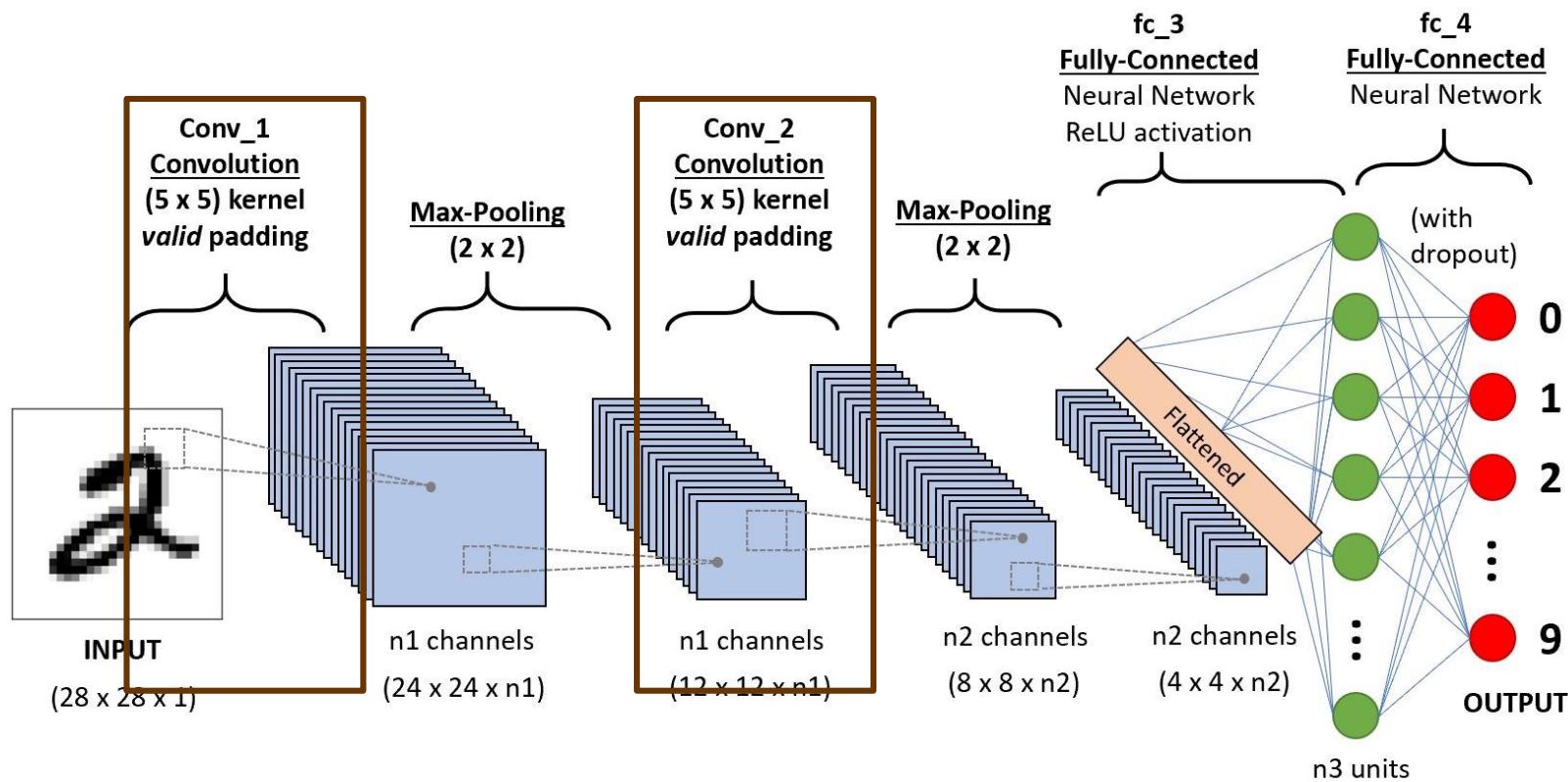
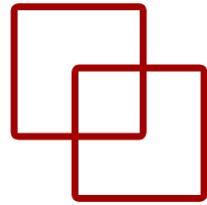


¡Para datos tabulares!

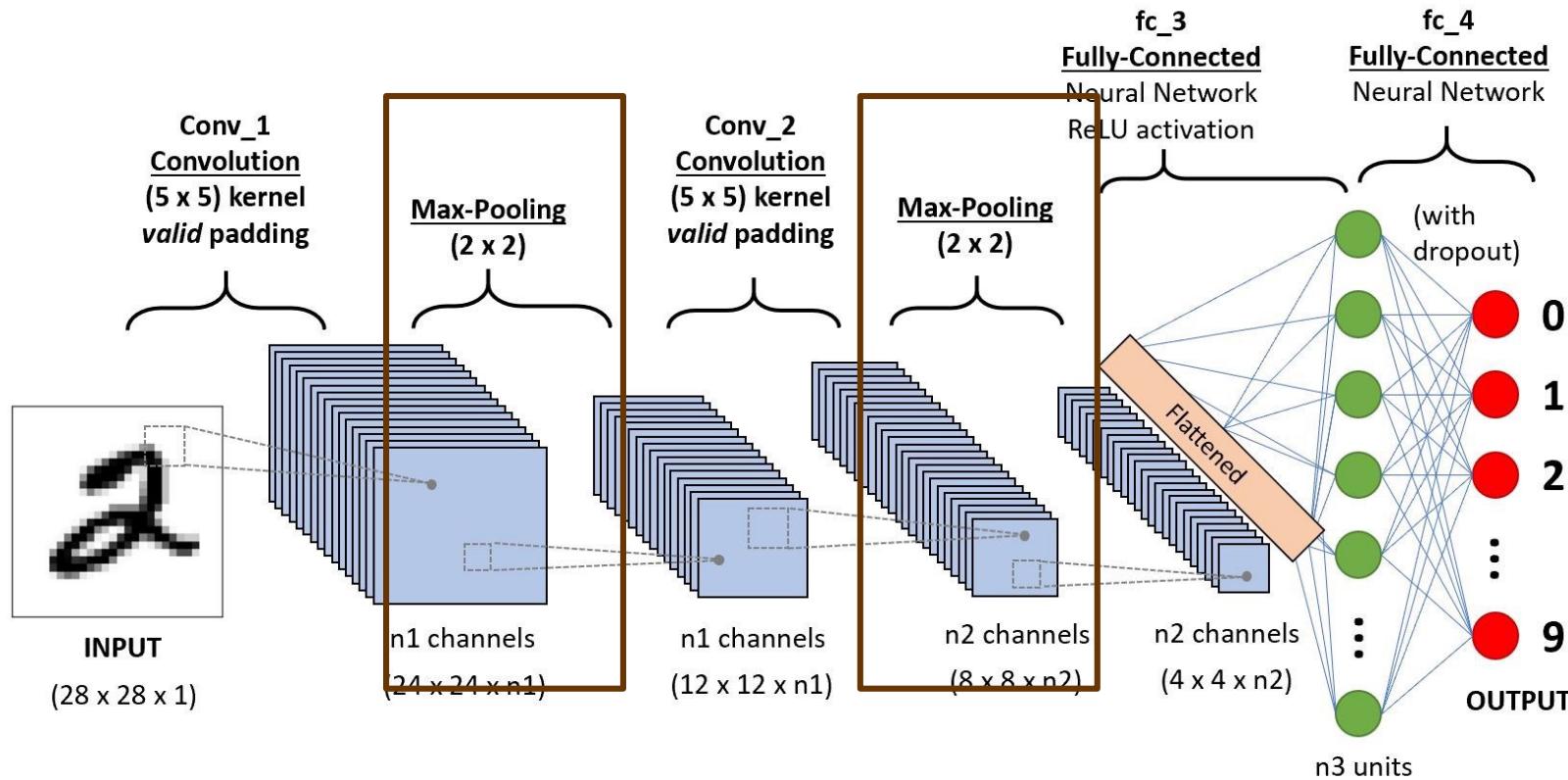
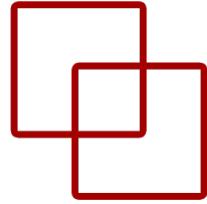
Redes Neuronales Convolucionales



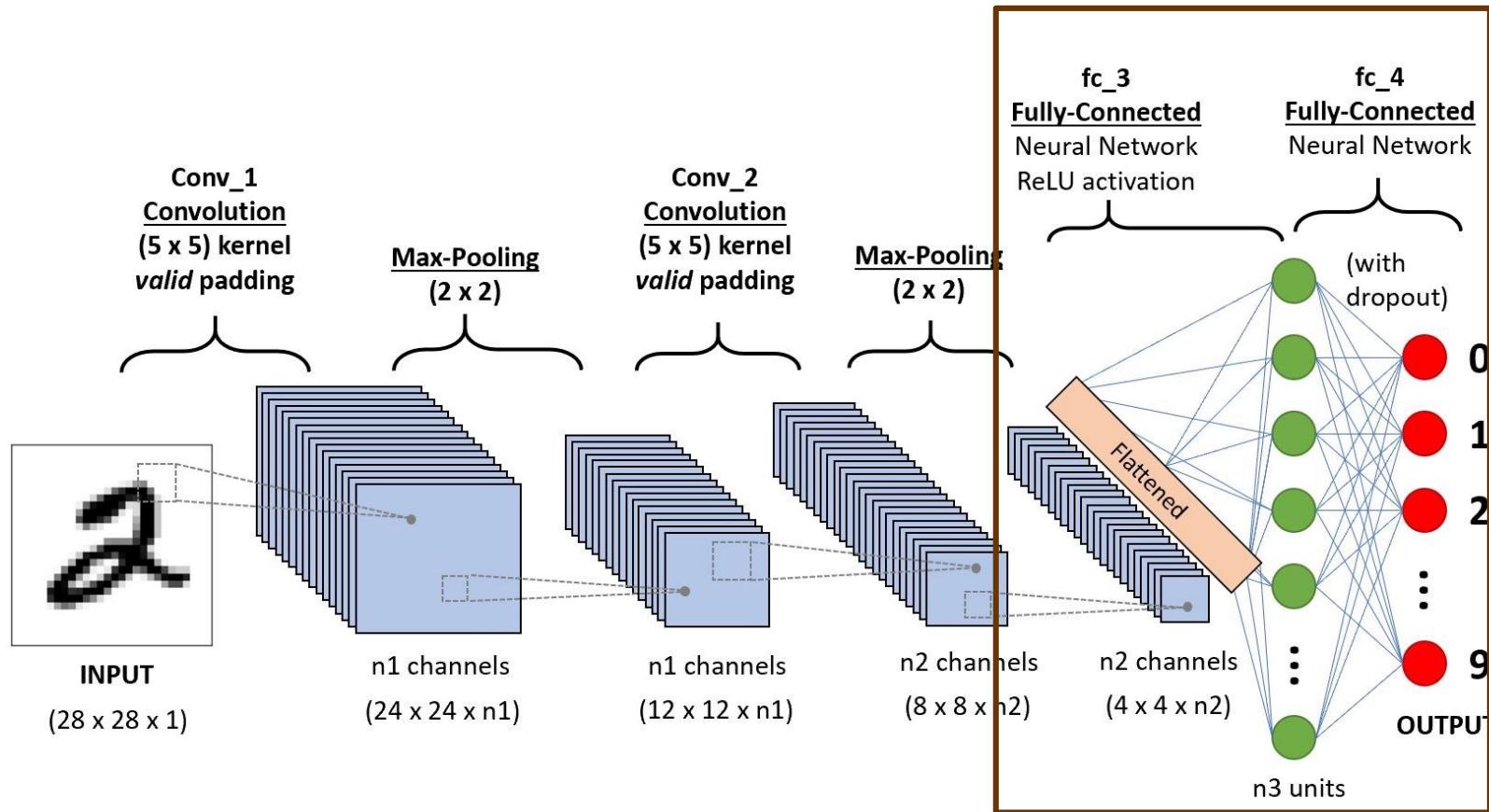
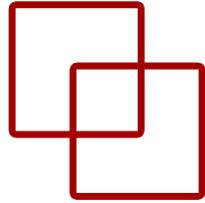
Capas convolucionales



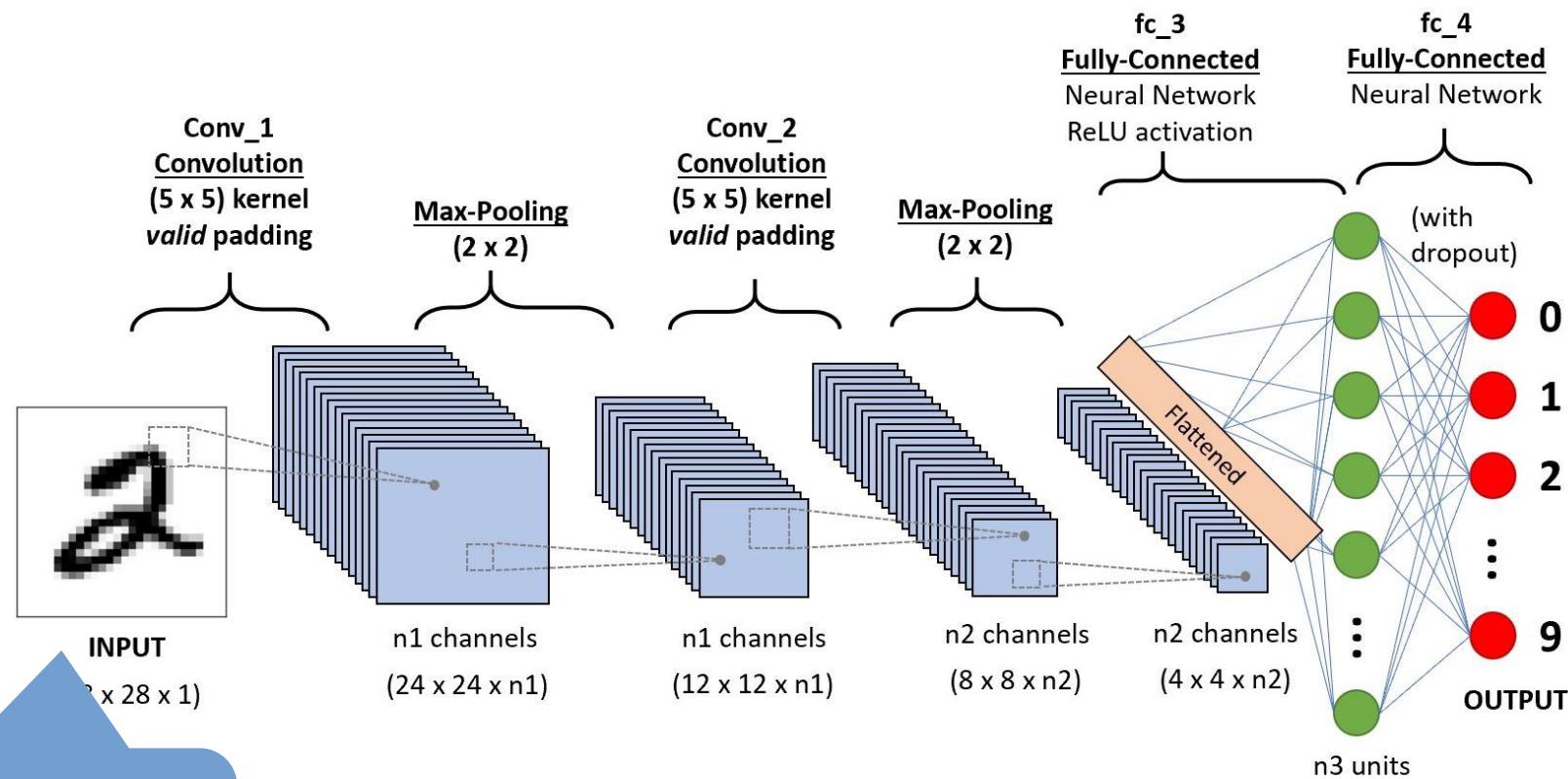
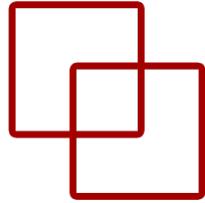
Capas de agrupación (*pooling*)



Capas completamente conectadas



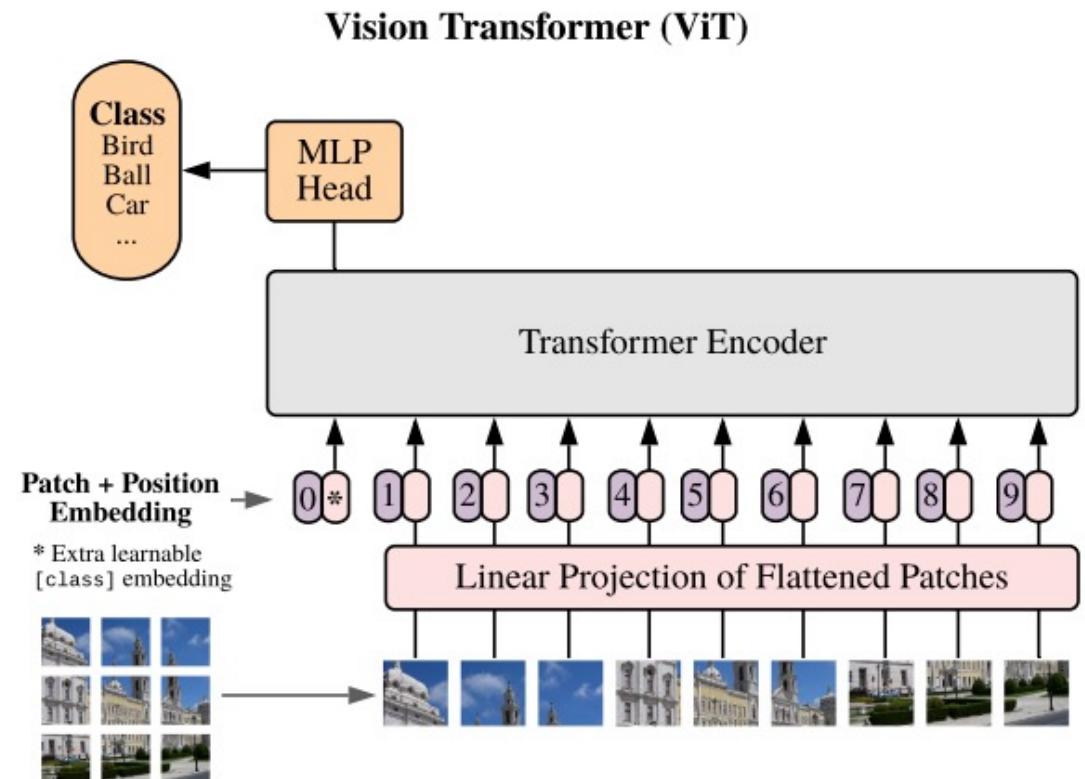
Capas completamente conectadas



¡Para imágenes!

Vision Transformer (ViT)

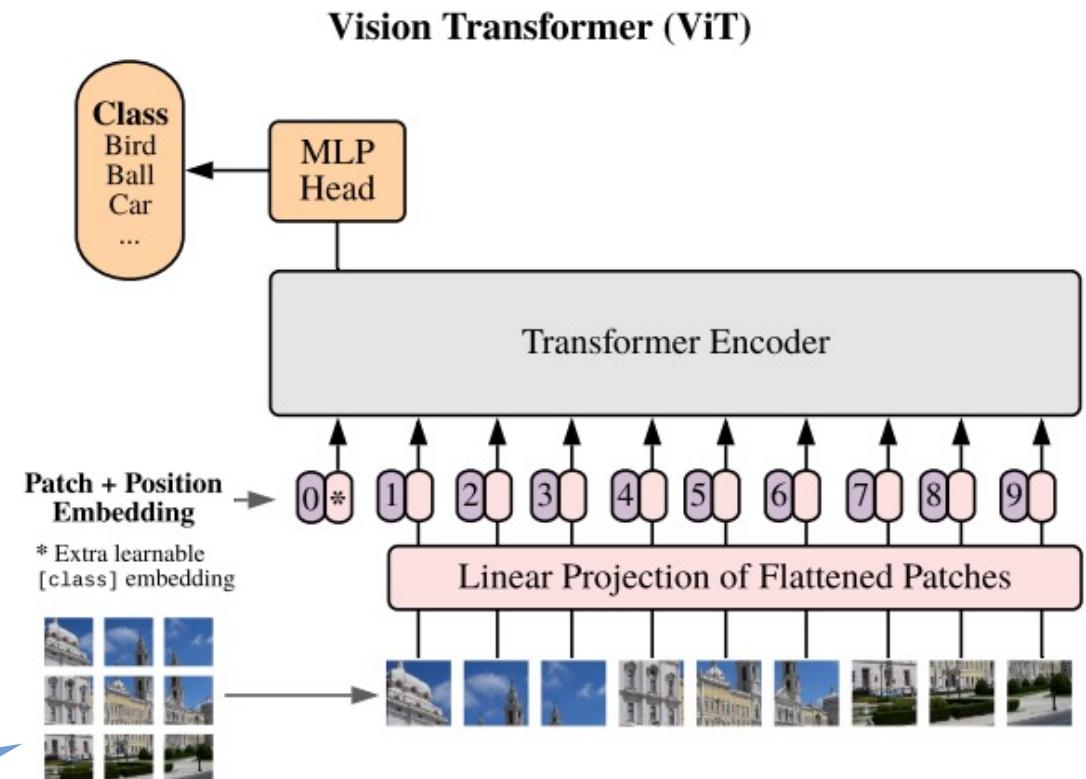
- Los datos de entrada al modelo del transformador se proporcionan en forma de imágenes bidimensionales.
- La imagen de entrada, de alto H , ancho W y canales C , se corta en patches bidimensionales más pequeños. Esto da como resultado un número $N=HW/P^2$ de patches, donde cada patch tiene una resolución de $P\times P$ píxeles.



Vision Transformer (ViT)

- Los datos de entrada al modelo del transformador se proporcionan en forma de imágenes bidimensionales.
- La imagen de entrada, de alto H , ancho W y canales C , se corta en patchs bidimensionales más pequeños. Esto da como resultado un número $N=HW/P^2$ de patchs, donde cada patch tiene una píxeles.

¡Para imágenes!



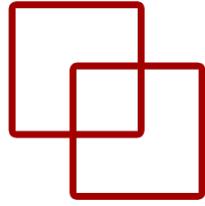
Assesment Criteria

ETS de
Ingeniería
Informática



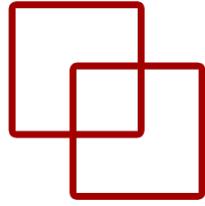
UNED

Transformación de datos

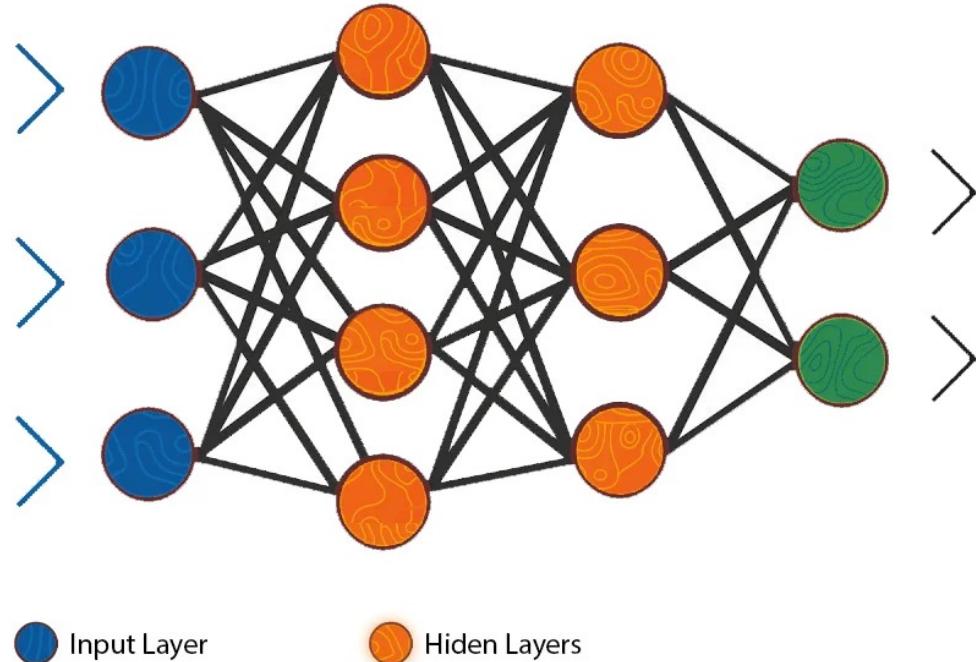
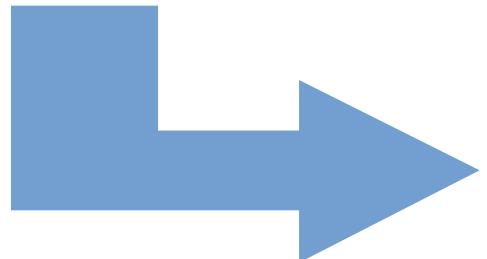


| Be07 | Be08 | Be09 | Be10 | Be11 | Sector |
|------|------|------|------|------|--------|
| -65 | -61 | -74 | -73 | -67 | 1 |
| -60 | -57 | -83 | -62 | -69 | 2 |
| -66 | -70 | -78 | -63 | -73 | 3 |
| ... | ... | ... | ... | ... | ... |
| -58 | -66 | -71 | -73 | -69 | 14 |
| -60 | -62 | -73 | -69 |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

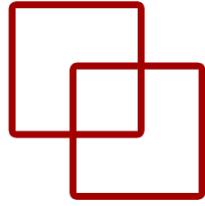
Transformación de datos



| 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 |

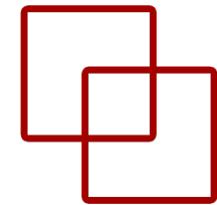


Transformación de datos

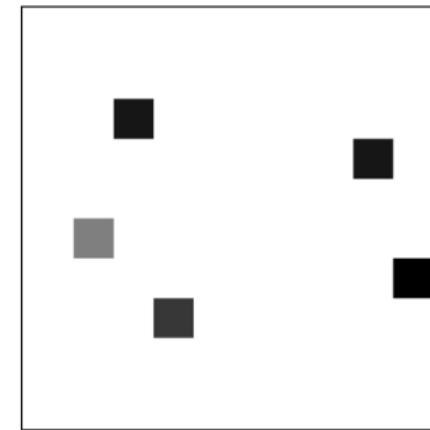
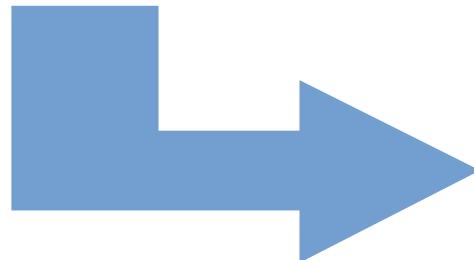


| Be07 | Be08 | Be09 | Be10 | Be11 | Sector |
|------|------|------|------|------|--------|
| -65 | -61 | -74 | -73 | -67 | 1 |
| -60 | -57 | -83 | -62 | -69 | 2 |
| -66 | -70 | -78 | -63 | -73 | 3 |
| ... | ... | ... | ... | ... | ... |
| -58 | -66 | -71 | -73 | -69 | 14 |
| -60 | -62 | -73 | -69 |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

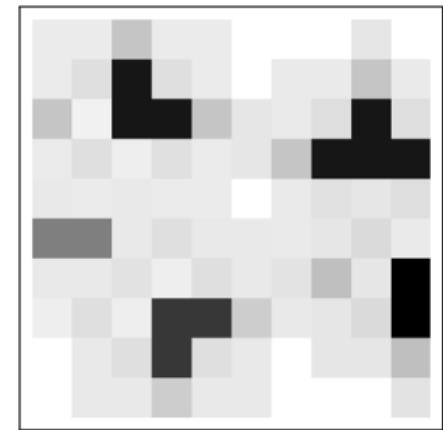
Transformación de datos



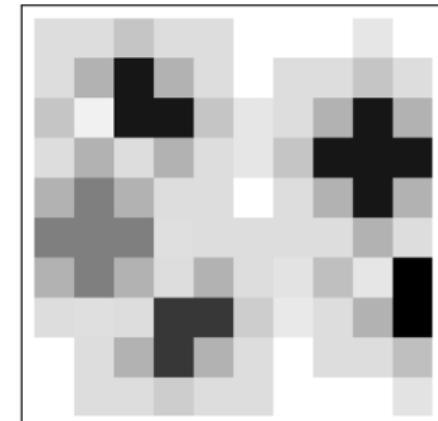
| Be07 | Be08 | Be09 | Be10 | Be11 | Sector |
|------|------|------|------|------|--------|
| -65 | -61 | -74 | -73 | -67 | 1 |
| -60 | -57 | -83 | -62 | -69 | 2 |
| -66 | -70 | -78 | -63 | -73 | 3 |
| ... | ... | ... | ... | ... | ... |
| -58 | -66 | -71 | -73 | -69 | 14 |
| -60 | -62 | -73 | -69 |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



(a) Without blurring.

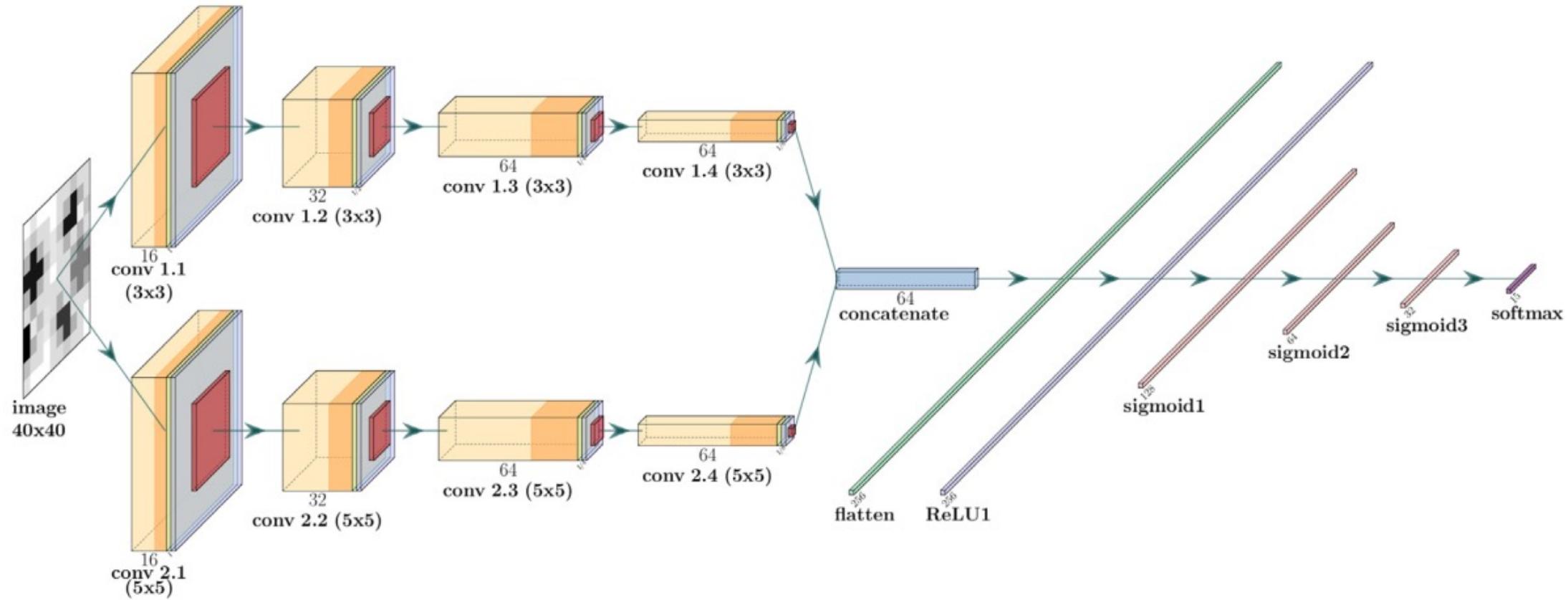
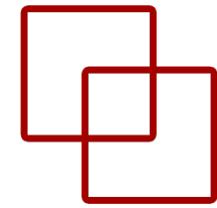


(b) Blurring with average value.

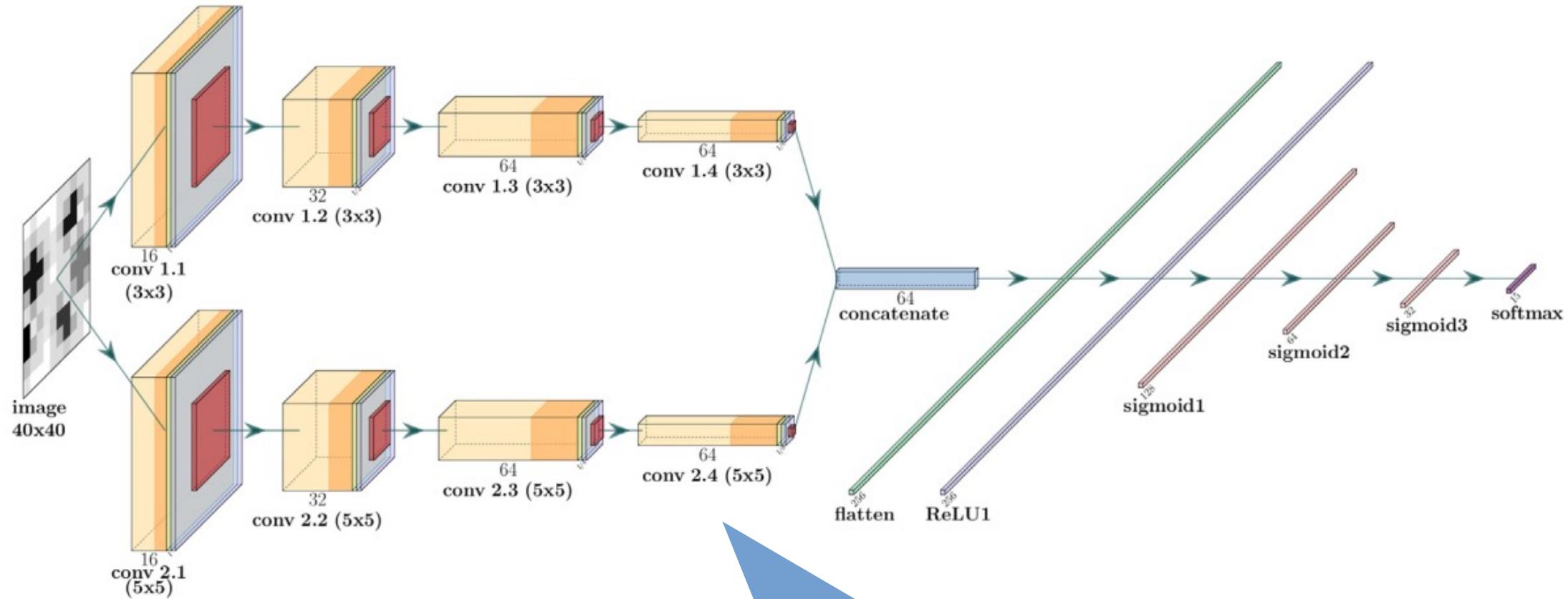
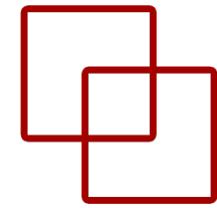


(c) Blurring with maximum value.

CNN Puras

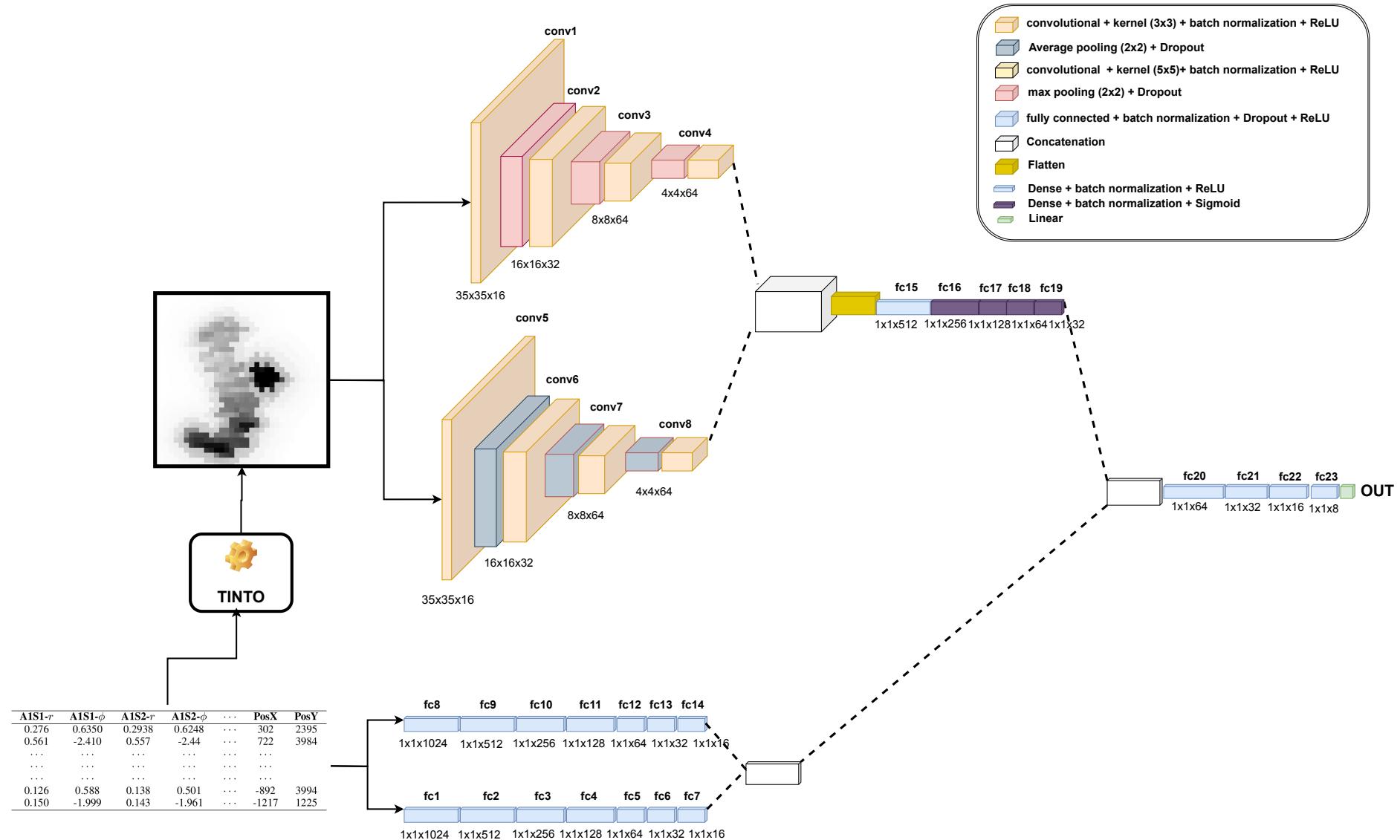
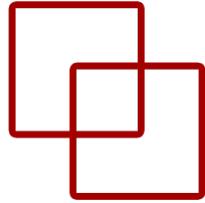


CNN Puras

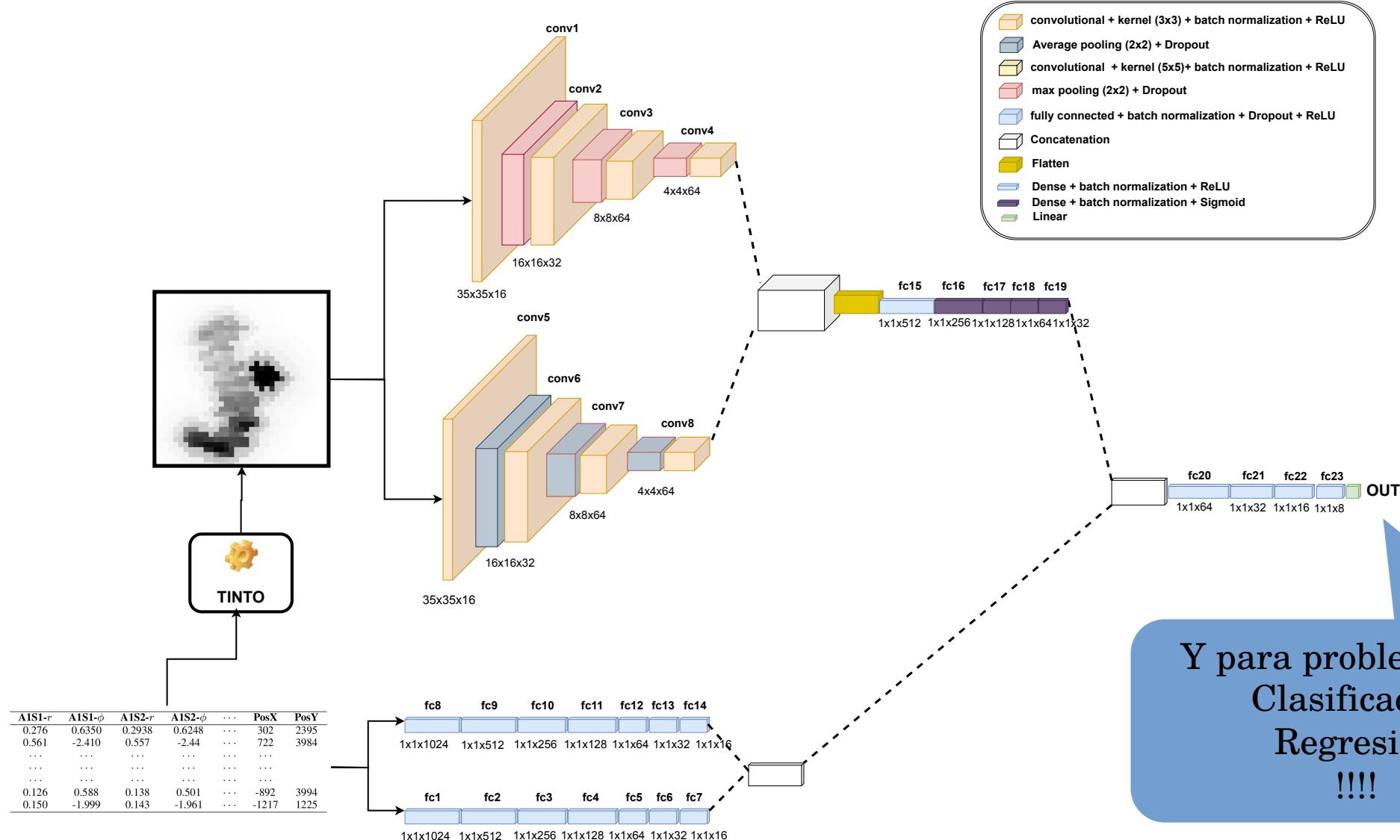
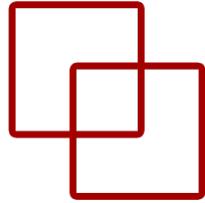


Pero no sólo
CNNs

Hybrid Neural Network



Hybrid Neural Network



Y para problemas de:
Clasificación
Regresión
!!!!

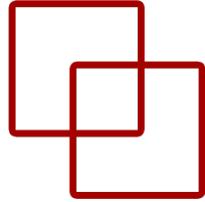
Métodos de conversión

ETS de
Ingeniería
Informática



UNED

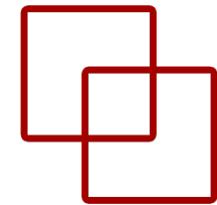
TINTOlib



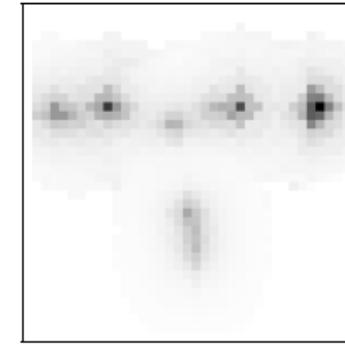
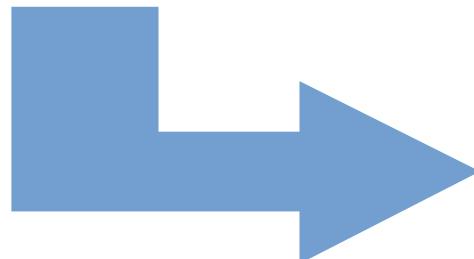
POLITÉCNICA



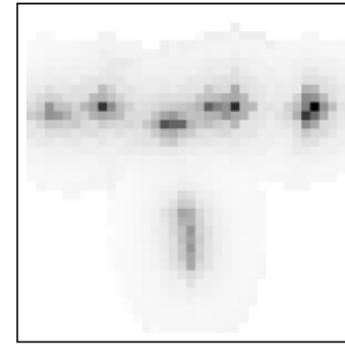
Transformación de datos



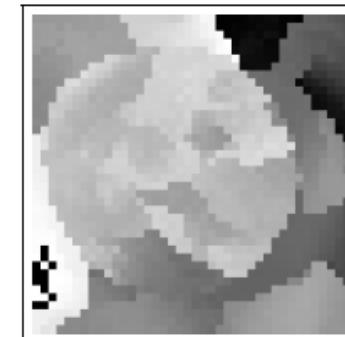
| Be07 | Be08 | Be09 | Be10 | Be11 | Sector |
|------|------|------|------|------|--------|
| -65 | -61 | -74 | -73 | -67 | 1 |
| -60 | -57 | -83 | -62 | -69 | 2 |
| -66 | -70 | -78 | -63 | -73 | 3 |
| ... | ... | ... | ... | ... | ... |
| -58 | -66 | -71 | -73 | -69 | 14 |
| -60 | -62 | -73 | -69 |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



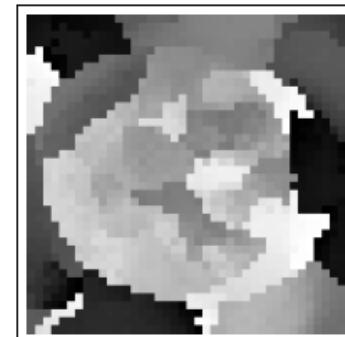
(a) TINTO - Sample 1.



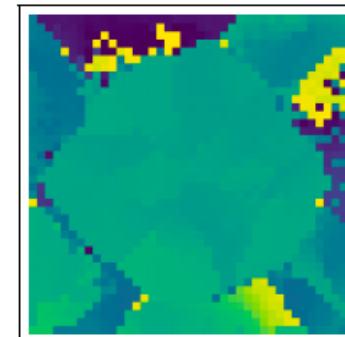
(b) TINTO - Sample 50,000.



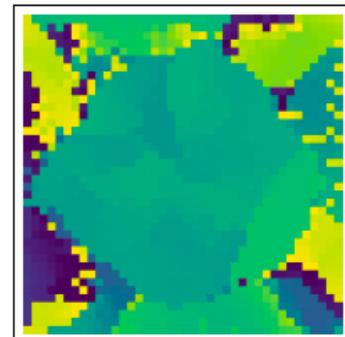
(c) IGTD - Sample 1.



(d) IGTD - Sample 50,000.

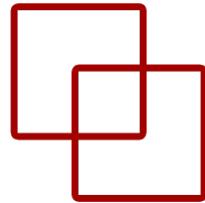


(e) REFINED - Sample 1.



(f) REFINED - Sample 50,000.

Métodos de transformación



| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or t -SNE + blurring |
| DeepInsight [15] | General | PCA or t -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |

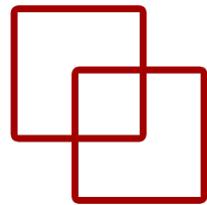
BarGraph,
DistanceMatrix y
Combination

ETS de
Ingeniería
Informática

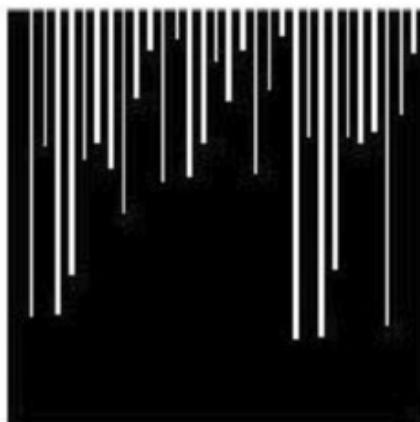


UNED

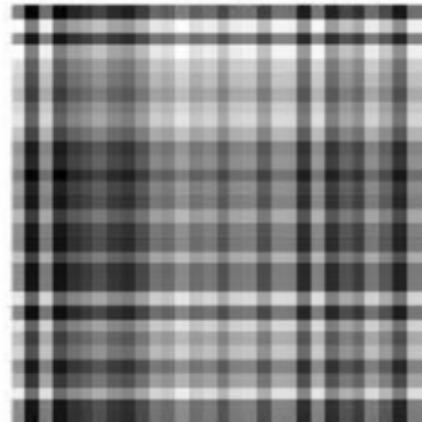
BarGraph, DistanceMatrix y Combination



| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or <i>t</i> -SNE + blurring |
| DeepInsight [15] | General | PCA or <i>t</i> -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |



(a) BarGraph



(b) DistanceMatrix

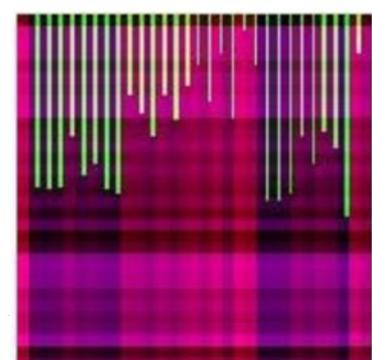
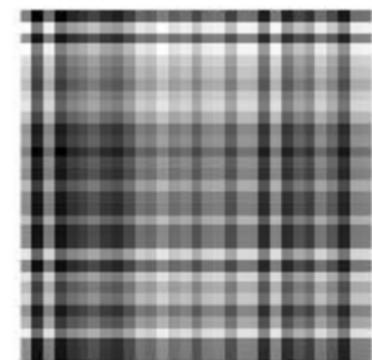
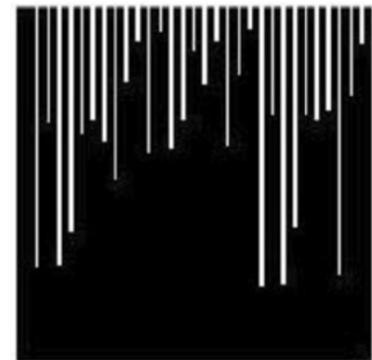


(c) Combination

BarGraph, DistanceMatrix y Combination

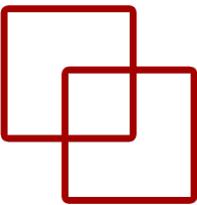
Metodología

- Representan las características normalizadas y las relaciones entre sí:
- **Equidistant Bar Graph:**
 - Se representan los valores de las variables normalizadas entre [0,1] en un **gráfico de barras**.
 - Sus hiperparámetros permiten ajustar el grosor de las barras y la distancia entre las mismas.
 - Las imágenes resultantes están dadas en 1 canal (W/B).
- **Normalized Distance Matrix:**
 - Representa una **matriz de distancias** de todas las variables normalizadas entre [0,1].
 - Los valores se muestran en escala de grises.
 - Las imágenes resultantes están dadas en 1 canal (W/B).
- **Combination of option:**
 - Este método es una **combinación de los dos anteriores**.
 - Las imágenes resultantes están dadas en 3 canales (RGB).
- Realizan cálculos muy **sencillos** con coste **computacional muy bajo**.
- Sin embargo, **no** tiene en cuenta la **distribución espacial** de las variables.



BarGraph

Algoritmo



- El tamaño de las imágenes viene dado por la multiplicación entre $varN$ y la suma de las variables $pixel_width$ y gap .

$$H = (pixel_width + gap) * varN$$

- Cada barra del gráfico de barras representa una característica de X_i donde $i \in [1, N]$.
- Para calcular el tamaño de cada barra de cada característica hay que multiplicar la altura de la imagen con el valor normalizado de $X_i(j)$ donde $j \in [1, varN]$.
- Cada instancia viene definida por un vector B_i que contiene la altura de cada característica.

$$B_i = [H * X_i]$$

- Una vez calculadas las alturas de todas las instancias de X se crean las imágenes dejando un espacio entre las barras igual a gap .

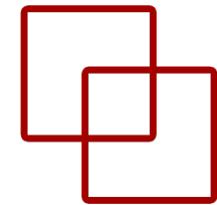
Algorithm 4: BarGraph

Data: pixel_width, gap
Result: Monochrome images with bar graphs

Define image height/width:
 $H = (pixel_width + gap) * varN;$
for $i = 0 : N$ **do**
 Initialize an empty image M_i of size $(H \times H)$;
 Compute the bar heights of the image:
 $B_i = [H * X_i];$
 Initialize Bar counter: $k = 0$;
 for $j = 0 : H$ with increment $gap+pixel_width$ **do**
 Incorporate the bar into the image:
 $M_i[B_i(k), j : (j + pixel_width)] = 1;$
 Increment counter: $k + 1$;
 if $k < varN$ **then**
 | break;
 end
 end
end

DistanceMatrix

Algoritmo



1. Crea las imágenes tras calcular la matriz de distancias normalizadas de las características de la instancia.

2. Tras normalizar las instancias X a $[0,1]$, se calcula la matriz de distancias entre las características para todo X_k donde $k \in [1, N]$.

3. El cálculo de la matriz de distancias se realiza mediante la siguiente ecuación:

$$M_k = [X_k(i) - X_k(j)]; \forall (i, j) \in varN, \forall k \in N$$

4. Una vez calculadas todas las matrices de distancias de X , se debe aplicar una normalización para establecer una escala en blanco/negro $[0, 255]$.

$$norm_grey(M_k) = [| 255 \times m_{kij} |]$$

Algorithm 5: DistanceMatrix

Data: scale

Result: B/W images with

Normalise features: $X = norm01(X)$;

for $k = 0 : N$ **do**

 Create an empty 2D matrix M_k of size
 $(varN \times varN)$;

for $i = 0 : varN$ **do**

for $j = 0 : varN$ **do**

 Calculate distance between variables:
 $M_k(i, j) = X_k(i) - X_k(j)$;

end

end

end

Normalize M_k : $M_k = norm_grey(M_k)$;

Expand pixels according to scale:

$M_k(i, j) = M_k(i, j) * scale$;

Save M_k as a grayscale image

end

Combination Algoritmo

1. Combina BarGraph y DistanceMatrix en un solo método algorítmico.
2. Para este caso, la imagen resultante se compone de 3 capas de color:
 - Capa de DistanceMatrix
 - Capa de BarGraph
 - Capa que contiene una matriz en la que cada fila contiene el valor normalizado de cada característica de X . La fila 1 contendrá el valor normalizado de la característica 1, la fila 2 el valor 2, etc.
3. Para calcular la Capa 3, de color, se utiliza la siguiente Ecuación, donde $k \in X$ e $i, j \in [1, varN]$.

$$M_k(i, j) = X_k(i)$$

Algorithm 6: Combination

Data: pixel_width, gap
Result: B/W images

Define bar height/width:
 $H = (\text{pixel_width} + \text{gap}) * \text{varN};$
Normalise features: $X = \text{norm01}(X);$

for $i = 0 : N$ **do**

\\STEP 1: DistanceMatrix;
Create a zero matrix $M1_i$ of size $(\text{varN} \times \text{varN});$
for $z = 0 : \text{varN}$ **do**
 for $w = 0 : \text{varN}$ **do**
 | Calculate distance between variables:
 | $M1_k(z, w) = X_k(z) - X_k(w);$
 end
end
Normalise $M1_k$: $M1_k = \text{norm_grey}(M1_k);$
Expand pixels:
 $M1_k(i, j) = M1_k(i, j) * (\text{pixel_width} + \text{gap});$

\\STEP 2: BarGraph;
Create a zero matrix $M2_i$ of size $(H \times H);$
Create the height of the image bars: $B_i = [H * X_i];$
Bar counter: $k = 0;$
for $j = 0 : H$ with increment $\text{gap} + \text{pixel_width}$ **do**
 Add the bar to the image:
 $M2_i[B_i(k), j : (j + \text{pixel_width})] = 1;$
 $k + 1;$
 if $k < \text{varN}$ **then**
 | **break**;
 end
end

\\STEP 3: Feature by row;
Create a zero matrix $M3_i$ of size $(\text{varN} \times \text{varN});$
for $m = 0 : \text{varN}$ **do**
 for $n = 0 : \text{varN}$ **do**
 | $M3_i(m, n) = X_i(m)$
 end
end
Normalise: $M3_i$: $M3_i = \text{norm01}(M3_i);$
Expand pixels:
 $M3_i(m, n) = M3_i(m, n) * (\text{pixel_width} + \text{gap});$

\\STEP 4: Create image;
Join matrices and save in RGB ($M1_i, M2_i, M3_i$);

end



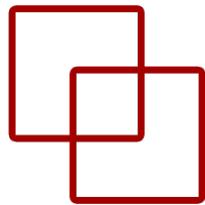
SuperTML



ETS de
Ingeniería
Informática



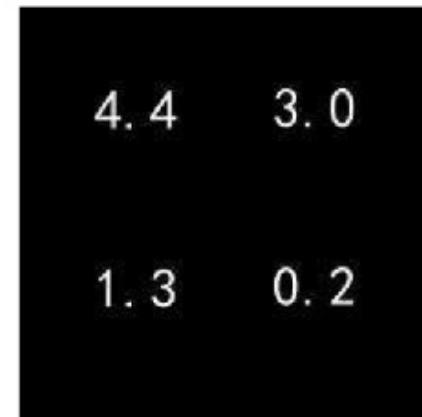
SuperTML



| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or t -SNE + blurring |
| DeepInsight [15] | General | PCA or t -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |



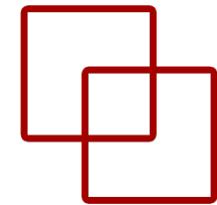
(a) SuperTML_VF



(b) SuperTML_EF

SuperTML

Metodología



- Basado en el método de NLP *Super Characters*, en el cual se dibujan los valores de los datos directamente en la imagen de 1 canal (W/B).
- Se le asigna una **región de la imagen** y un tamaño para representar cada característica en formato de texto o número escrito.
- Los autores proponen 2 enfoques:
 - *SuperTML_VF*: Cada característica dispone de una región y fuente de diferente tamaño en función de su **relevancia**. Cuanta **más importancia** tenga la característica, **más grande** será su tamaño de fuente y región.
 - *SuperTML_EF*: En este enfoque cada característica dispone de una región y fuente del **mismo tamaño**.
- El método **no tiene en cuenta la distribución espacial** de las variables, más allá de la variación de tamaño y fuente en *SuperTML_VF*.
- Otro de los **inconvenientes** puede ser las pequeñas variaciones en las variables.
 - Por ejemplo, tiene dificultades en entender que 3,9999 y 4,0 son valores casi idénticos numéricamente.



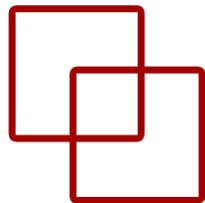
(a) SuperTML_VF



(b) SuperTML_EF

SuperTML

Algoritmo



1. Para definir las secciones de la imagen en las que se escribirán las características que debemos tener en cuenta los parámetros *columns* (número de columnas en las que se escribirán las características) e *image_size* (píxeles de anchura y altura).
2. La anchura de las secciones viene definida por: $\text{column_size} = \frac{\text{image_size}}{\text{columns}}$
3. Para calcular la posición x_j e y_j , donde $j \in [1, \text{varN}]$, en la que se escribirá cada característica, hay que realizar los siguientes cálculos.
 - Para calcular x_j primero hay que definir la columna a la que pertenece la instancia x_j , dado por la Ecuación donde se aplica el módulo de *columns* sobre la posición j de la variable X_j .
$$\text{column}_j = j \mod \text{columns}$$
4. Una vez calculada column_j se puede obtener x_j por la Ec. donde p es un margen izquierdo.
$$x_j = p + \text{column}_j \times \text{column_size}$$
5. Para obtener y_j hay que calcular previamente el número de filas que contiene la imagen. Se realiza mediante la división redondeada hacia arriba del número de variables entre el número de columnas.
$$\text{rows} = \left\lceil \frac{\text{varN}}{\text{columns}} \right\rceil$$
6. Se calcula el tamaño de las filas mediante $\text{row_size} = \frac{\text{image_size}}{\text{rows}}$
7. Se calcula la fila correspondiente de cada característica mediante $\text{row}_j = j \mod \text{rows}$
8. Se calcula y_j aplicando un margen de $\text{row_size}/2$ para centrar el texto en el medio de la fila
$$y_j = \frac{\text{row_size}}{2} + \text{row}_j \times \text{row_size}$$

Algorithm 7: SuperTML

Data: columns, image_size, font_size
Result: Imágenes con los valores de las instancias

Calcular el tamaño de cada columna:
 $\text{column_size} = \frac{\text{image_size}}{\text{columns}}$;

Calcular el tamaño de cada fila:
 $\text{row_size} = \frac{\text{image_size}}{\text{rows}}$;

```
for i = 0 : N do
    Crear una imagen vacía de tamaño image_size;
    for j = 0 : varN do
        x =  $\frac{\text{column\_size}}{10} + (j \mod \text{columns}) \times \text{column\_size}$ ;
        y =  $\frac{\text{row\_size}}{2} + (j \mod \text{rows}) \times \text{row\_size}$ ;
        Dibujar el texto en la posición (x, y) con el valor  $X_i(j)$  ;
    end
end
```

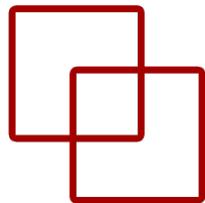
IGTD

ETS de
Ingeniería
Informática

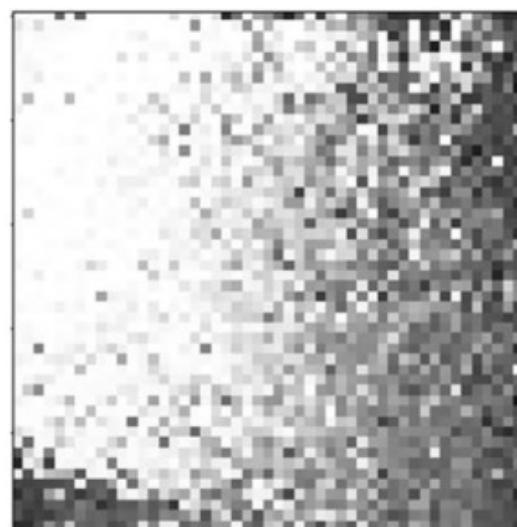


UNED

Image Generator for Tabular Data (IGTD)

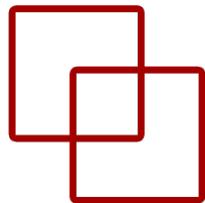


| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or <i>t</i> -SNE + blurring |
| DeepInsight [15] | General | PCA or <i>t</i> -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |

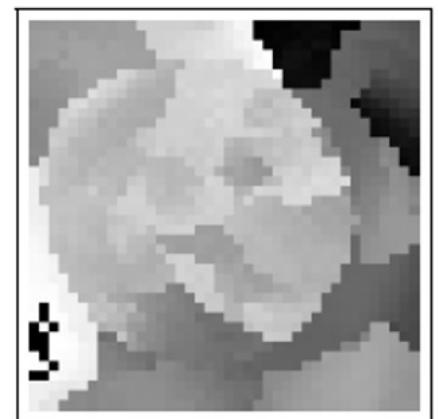


IGTD

Metodología

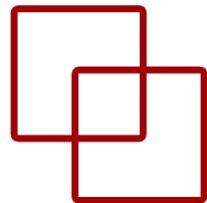


- Asigna **una variable por píxel** y realiza **permutaciones** en la distribución de los píxeles hasta encontrar una **distribución óptima**.
- Las distancias se calculan en función de sus **coordenadas** en la imagen.
- En cada iteración se permuta el píxel que más tiempo lleve sin ser permutado con el píxel que **minimiza** la diferencia entre los rankings.
- El algoritmo **terminará** cuando alcance en **número máximo de pasos** predefinidos como hiperparámetro.
- Las imágenes resultantes están dadas en 1 canal (W/B).
- Tiene en cuenta la **distribución espacial** de las características.
- Gracias a la permutación de características se pueden obtener imágenes que **agrupen las características similares entre sí y separen las dispares**.
- La selección de los **hiperparámetros** de IGTd tiene un papel fundamental y su efectividad dependerá de ello.



IGTD

Metodología



- Asigna una variable por píxel y realiza permutaciones en la distribución de los píxeles hasta encontrar una distribución óptima.
- Las permutaciones se realizan con el objetivo de minimizar la diferencia entre el ranking de distancias por pares de las características y la clasificación de distancias por pares entre los píxeles asignados.
- Las distancias se calculan en función de sus coordenadas en la imagen.
- En cada iteración se permuta el píxel que más tiempo lleve sin ser permutado con el píxel que minimiza la diferencia entre los rankings.
- El algoritmo terminará cuando alcance en número máximo de pasos predefinidos como hiperparámetro.
- Las imágenes resultantes están dadas en 1 canal (W/B).
- Tiene en cuenta la distribución espacial de las características.
- Gracias a la permutación de características se pueden obtener imágenes que agrupen las características similares entre sí y separen las dispares.
- La selección de los hiperparámetros de IGTD tiene un papel fundamental y su efectividad dependerá de ello.

IGTD Algoritmo

1. R a matrix of the ranking of pairwise distances of the normalised features, where $r_{i,j}$ is the ranking value of the distance between feature X_i and feature X_j .
 - The distance function could be Pearson, Spearman, Euclidean or set.
2. Q a matrix of the ranking of pairwise distances of the feature positions in the image, where $q_{i,j}$ is the ranking value of the distance between feature X_i and feature X_j .
 - The distance function is given by the parameter could be Euclidean or Manhattan.
3. The size of R and Q is given by the variable scale ($M \times N$) that defines the number of rows and columns of the ranking.
 - In case the total pixels ($M * N$) is greater than the number of $varN$, padding p will be added so that $M * N + p = varN$ is met.
 - In case the total pixels ($M * N$) is less than the number of $varN$, w variables will be eliminated so that $M * N + p = varN$ is met.
 - No criterion is defined to select the features, so the last w features will be eliminated.
4. Rankings are ascending, small distances are assigned a small ranking value and viceversa.
 - The main idea is to achieve that the most similar variables (ranking R) are also the closest ones (ranking Q).
 - For this, an error function is defined that must be minimised

$$\text{err}(R, Q) = \sum_{i=2}^N \sum_{j=1}^{i-1} \text{diff}(r_{i,j}, q_{i,j})$$

where diff is the function to evaluate the difference between $(r_{i,j}, q_{i,j})$ and is given by the variable error (square or absolute error).

Algorithm 2: IGT

Data: scale, fea_dost_method, image_dist_method, save_image_size, max_step, val_step, error, switch_t, min_gain

Result: Grayscale images with one pixel / feature

Calculate initial R by fea_dist_method ;

Calculate initial Q by image_dist_method;

while $s = 0 < max_step$ **do**

\\Select feature with the longest time without being evaluated;

$n^* = \underset{n \in \{1, 2, \dots, N\}}{\operatorname{argmin}} h_n$;

\\Select feature that by permuting reduces error;

$l^* = \underset{l \in \{1, \dots, n^*-1, n^*+1, \dots, N\}}{\operatorname{argmax}} (\text{err}(R, Q) - \text{err}(R_{n^* \sim l}, Q))$;

\\If $\text{err} > switch_t \Rightarrow$ perform permutation ;

If $\frac{\text{err}(R, Q) - \text{err}(R_{n^* \sim l}, Q)}{\text{err}(R, Q)} > switch_t$ **then**

$k_s = k_{s-1}$;

Permute n^* and l^* in k_s ;

$e_s = \text{err}(R_{n^* \sim l}, Q)$;

$h_{n^*}, h_{l^*} = s$;

$R = R_{n^* \sim l}$;

else

$h_{n^*} = s$;

$e_s = e_{s-1}$;

$k_s = k_{s-1}$;

end

If $s < val_step$ **then**

\\Evaluate gain once the val_step iterations are reached;

$gain = \frac{e_s - val_step - e_u}{e_s - val_step} \forall u \in \{s - val_step + 1, \dots, s\}$;

\\If min gain is not reached the algorithm is stopped. ;

If $gain < min_gain$ **then**

| break

end

end

Find the iteration with the least error:

$v^* = \underset{v \in \{1, \dots, s\}}{\operatorname{argmin}} e_v$;

Transform the images using the distribution k_{v^*} ;

Save the scaled images in a save_image_size

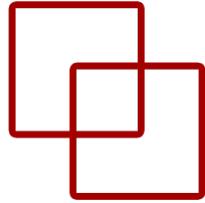
REFINED

ETS de
Ingeniería
Informática

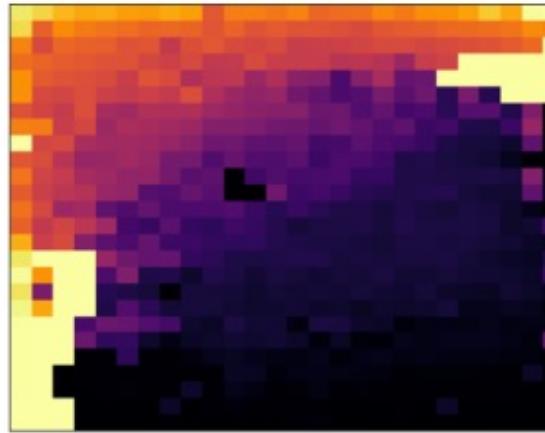


UNED

REFINED

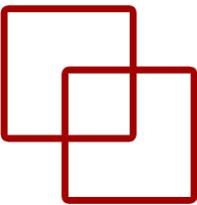


| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or t -SNE + blurring |
| DeepInsight [15] | General | PCA or t -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |

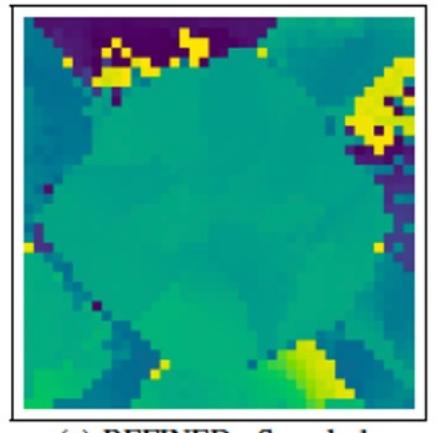


REFINED

Metodología

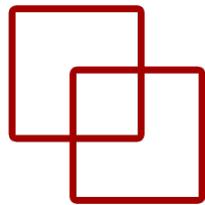


- *REpresentation of Features as Images with NEighborhood Dependencie* (REFINED) está basado en el algoritmo de reducción de dimensionalidad **MDS**.
- Esta técnica **calcula la matriz de distancias** de las características.
- Después se aplica MDS para crear un **mapa de características inicial**.
- Aplicando MDS no se puede garantizar que cada punto mapeado esté vinculado a un píxel característico y pueden generarse imágenes dispersas.
 - Es por ello, que el siguiente paso es aplicar una variación Bayesiana del MDS (BMDS) que permite **compactar la imagen** y asignar **una característica a cada píxel**.
- Esta variación hace uso de un algoritmo *hill climbing* con una función de coste que mide la diferencia entre:
 - La **distancia euclídea** de las nuevas localizaciones de las características y las distancias reales estimadas.
- Se realizan permutaciones iterativamente entre un píxel central y su vecino que **minimice esa función de coste**.
- Las imágenes resultantes están dadas en 3 canales (RGB).
- Tiene cuenta **la distribución espacial** de las variables.



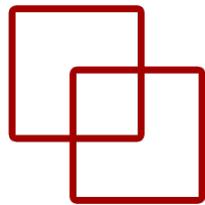
REFINED

Metodología



- REpresentation of Features as Images with NEighborhood Dependencie (REFINED) es un método basado en el algoritmo de reducción de dimensionalidad MDS.
- Esta técnica calcula la matriz de distancias de las características.
- Después se aplica MDS para crear un mapa de características inicial.
- Aplicando MDS no se puede garantizar que cada punto mapeado esté vinculado a un píxel característico y pueden generarse imágenes dispersas.
 - Es por ello, que el siguiente paso es aplicar una variación Bayesiana del MDS (BMDS) que permite compactar la imagen y asignar una característica a cada píxel.
- Esta variación hace uso de un algoritmo *hill climbing* con una función de coste que mide la diferencia entre:
 - La distancia euclídea de las nuevas localizaciones de las características y las distancias reales estimadas.
- Se realizan permutaciones iterativamente entre un píxel central y su vecino que minimice esa función de coste.
- Las imágenes resultantes están dadas en 3 canales (RGB).
- Tiene cuenta la distribución espacial de las variables.

REFINED Algoritmo



1. Crear una matriz D de distancias euclídeas de las características donde $D \in IR^{varN \times varN}$.

$$D = \begin{bmatrix} d_{11} & \dots & d_{1varN} \\ d_{21} & \dots & d_{2varN} \\ \vdots & \ddots & \vdots \\ d_{N1} & \dots & d_{varNvarN} \end{bmatrix}$$

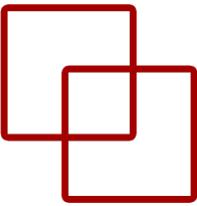
donde, $d_{ij} = x_i - x_j$ y x_i representa el valor de la característica i de X y x_j representa el valor de la característica j de X .

2. Crear un mapa de características inicial que preserve las distancias de las características en un espacio 2D mediante el método MDS.
 - Se crea la matriz $S \in IR^{varN \times 2}$ donde el valor b_{i1} es la coordenada en el eje x de la variable i y b_{i2} es la coordenada en el eje y de la variable i , donde $i \in [1, varN]$.

$$MDS(D, 2) = S = \begin{bmatrix} b_{11} & b_{12} \\ \vdots & \vdots \\ b_{varN1} & b_{varN2} \end{bmatrix}$$

3. Aplicar el método BMDS propuesto por los autores para estimar la localización de las características, con la condición de que exista una característica por píxel.
 - Para la localización de los píxeles característicos se define un proceso homogéneo de Poisson (HPP) con una $\lambda = varN / [0, 1]^2$.
 - Dada la matriz de instancias D se define d_{jk} distancias observadas entre la variable j y k , donde $j, k \in [1, varN]$.

REFINED Algoritmo



3. (continuación) Aplicar el método BMDS propuesto por los autores para estimar la localización de las características, con la condición de que exista una característica por píxel.

- Las distancias reales no observadas son definidas como:

$$\delta_{jk} = \sqrt{\sum_l (s_{j,l} - s_{k,l})^2}$$

- donde s es un sistema de coordenadas 2D dado S que denota la localización real de las variables predictoras j y k , donde $l \in [1,2]$.
- El modelo queda descrito por la siguiente ecuación

$$f(d-s, \sigma^2) \propto (\sigma^2)^{-\frac{q}{2}} \exp[-\frac{1}{2\sigma^2} \sum_{j>k} (d_{jk} - \delta_{jk})^2 - \sum_{i>k} \log \Phi(\frac{\delta_{jk}}{\sigma})]$$

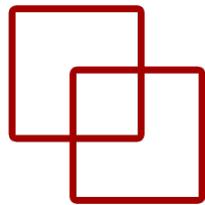
- donde $q = \binom{varN}{2}$ es el número total de distancias y $\Phi(\cdot)$ es la función de distribución acumulada (CDF) de la distribución normal estándar. La distribución a priori está dada por una distribución Gamma inversa $\sigma^2 \sim IG(a,b)$ con $a > 2$ y $b > 0$. La distribución a posteriori queda de la siguiente manera:

$$[s, \sigma^2 | d] \propto (\sigma^2)^{-\frac{q}{2} + a + 1} \exp[-\frac{1}{2\sigma^2} \sum_{j>k} (d_{jk} - \delta_{jk})^2 - \sum_{i>k} \log \Phi(\frac{\delta_{jk}}{\sigma}) - \frac{b}{\sigma^2}]$$

- Finalmente, hace uso del muestreador *Metropolis-in-Gibbs* para calcular las localizaciones a posteriori de las características. En este punto se obtiene la matriz M que contiene la distribución inicial de las características en una imagen 2D.

REFINED

Algoritmo



4. Aplicar el algoritmo *hill climbing* para reducir la diferencia entre la matriz de distancias D y el mapa de características inicial M .
 - Para ello se permuta un centroide (píxel característico que se está evaluando en esa iteración) con el vecino que más disminuya la función de coste.
 - Se evalúan todos los píxeles característicos en cada paso del algoritmo *hill climbing*.
 - El resultado de este paso es la matriz C que contiene la posición final de cada característica en la imagen.
5. Una vez optimizada la posición de cada píxel característico, se realiza la transformación de las instancias X y se guardan las imágenes.

Algorithm 3: REFINED

Data: hcIterations

Result: Images with instance values

Create a distance matrix:

$D = \text{euclidean_distances}(X);$

Reduce the dimensionality of X with MDS:

$M = MDS(X);$

Calculate non-overlapping coordinates:

$B = BMDS(D, M);$

Hill Climbing with hcIterations iterations:

$C = HC(D, B);$

Apply transformation C to X ;

for $i = 0 : N$ **do**

$img_i = \text{transform}(C, X_i);$

 Save image img_i ;

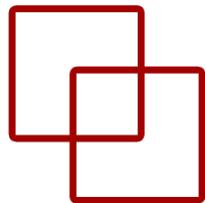
end

TINTO

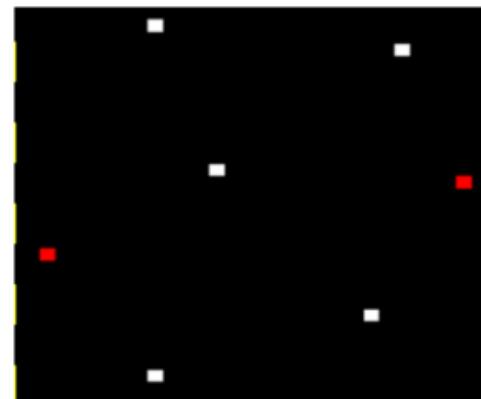
ETS de
Ingeniería
Informática

UNED

DeepInsight – TINTO



| Technique | Field | Basis |
|------------------------|----------------|---|
| TINTO [1], [11] | General | PCA or t -SNE + blurring |
| DeepInsight [15] | General | PCA or t -SNE |
| IGTD [12] | General | Permutations between pixels |
| REFINED [9] | General | MDS + permutations between pixels |
| SuperTML [6] | General | Draw the values in the image |
| BarGraph [13] | General | Bar representation of the features |
| DistanceMatrix [13] | General | Matrix of distances between features |
| Combination [13] | General | BarGraph and DistanceMatrix combination |
| OmicsMapNet [10] | Omics | Treemaps using hierarchical data |
| Signal Fingerprint [2] | Fingerprinting | Use of eigenvector as image |



(a) DeepInsight.

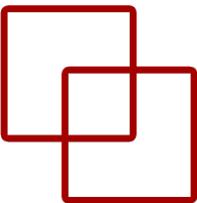


(b) TINTO.

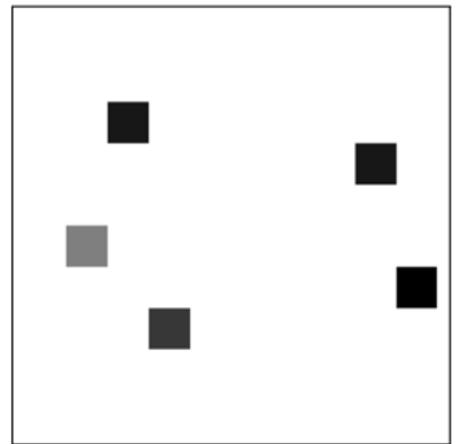
1. Talla-Chumpitaz, Ret al. (2023). A novel deep learning approach using blurring image techniques for Bluetooth-based indoor localisation. *Information Fusion*, 91, 173-186.
2. Castillo-Cara, Manuel, et al (2023). TINTO: Converting Tidy Data into image for classification with 2-Dimensional Convolutional 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.

DeepInsight – TINTO

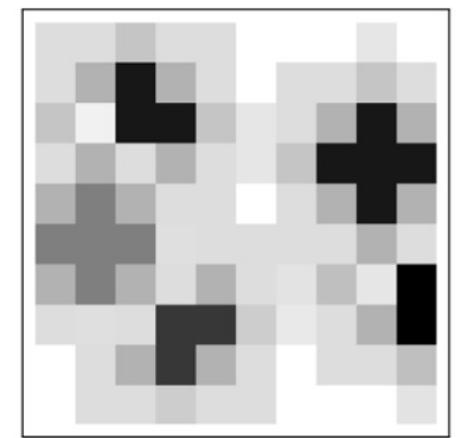
Metodología



- DeepInsight se basa en algoritmos *t-SNE* o PCA.
 - Tras aplicar el algoritmo se obtiene una **representación 2D** de la distribución de las características en función de su similitud.
 - Al realizar la transformación puede existir **solapamiento** entre píxeles característicos y se aborda realizando la media de los valores.
 - El siguiente paso es aplicar el algoritmo *hull convexo*, que se encarga de **agrupar las características** en el menor rectángulo posible.
 - El último paso es realizar una rotación de la imagen resultante.
-
- TINTO es una ampliación de DeepInsight pero escrito en Python.
 - Este método proporciona la **especificación formal** de los pasos para obtener la posición de los píxeles característicos y **soluciona el problema de solapamiento** que existe en DeepInsight.
 - Además, se propone el uso de un método de **Blurring** o difuminado en los píxeles característicos que simula el desvanecimiento de la señal.
 - Las imágenes resultantes **están dadas en 1 canal** (B/W).



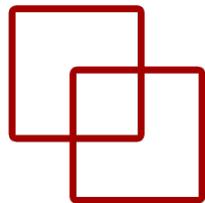
(a) Without blurring.



(c) Blurring with maximum value.

DeepInsight – TINTO

Metodología

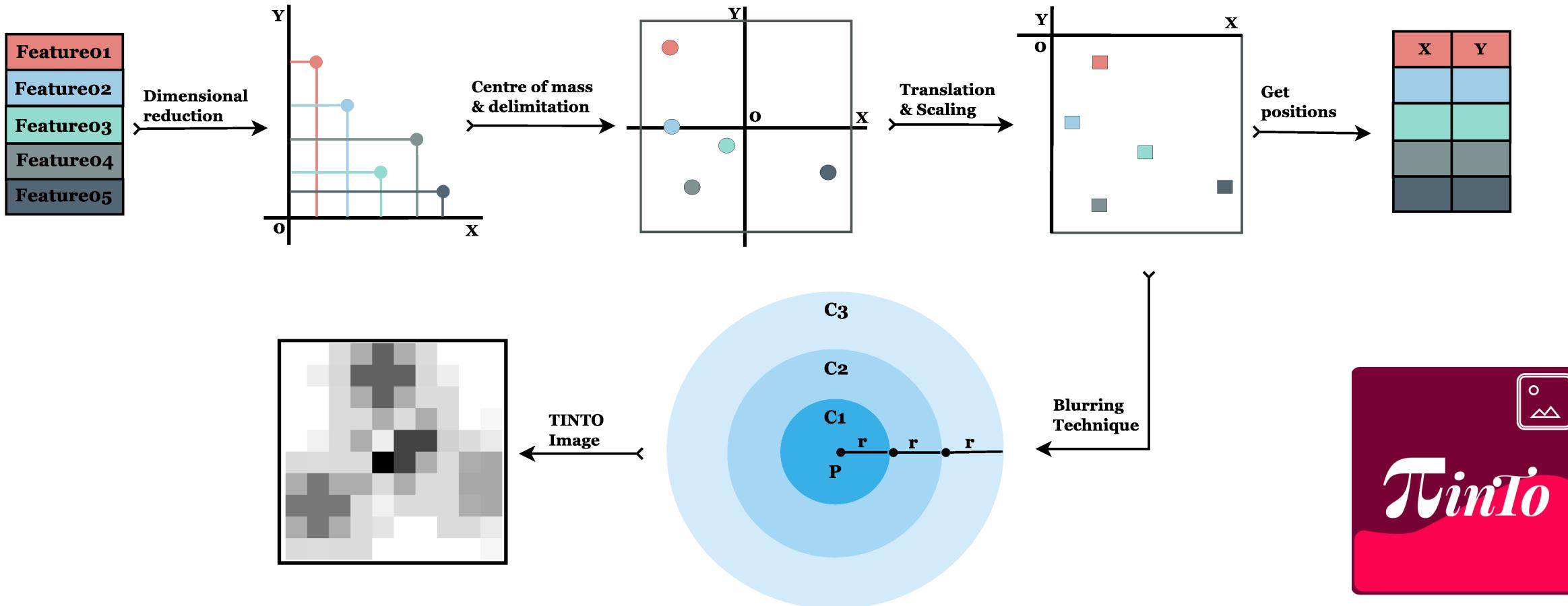
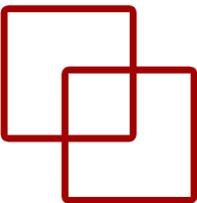


- DeepInsight es un método que se basa en algoritmos de reducción de dimensionalidad como *t-SNE* o PCA.
- Tras aplicar el algoritmo de reducción de dimensionalidad se obtendrá una representación en 2D de la distribución de las características en función de su similitud.
- Al realizar la transformación puede existir solapamiento entre píxeles característicos y se aborda realizando la media de los valores.
- El siguiente paso es aplicar el algoritmo *hull convexo*, que se encarga de agrupar las características en el menor rectángulo posible.
- El último paso es realizar una rotación de la imagen resultante.

- TINTO es una ampliación del método DeepInsight.
- Este método proporciona la especificación formal de los pasos para obtener la posición de los píxeles característicos y soluciona el problema de solapamiento que existe en DeepInsight.
- Además, se propone el uso de un método de Blurring o difuminado en los píxeles característicos que simula el desvanecimiento de la señal.
- Las imágenes resultantes están dadas en 1 canal (B/W).

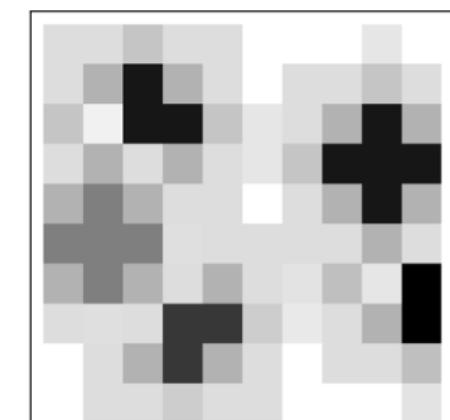
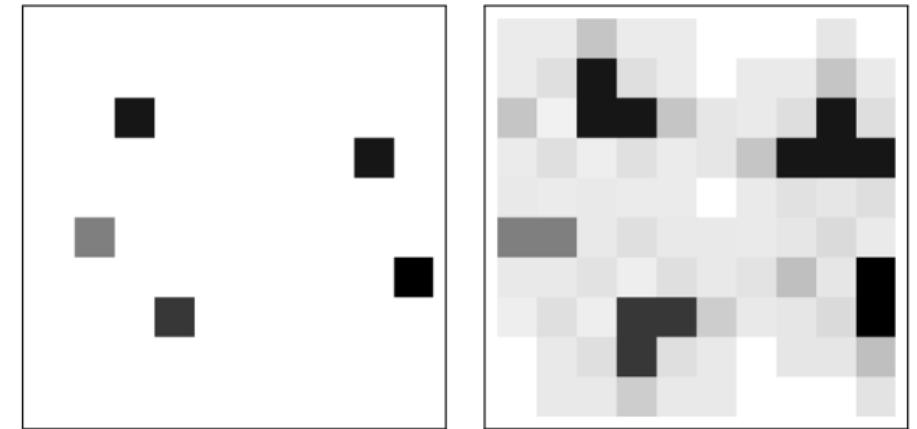
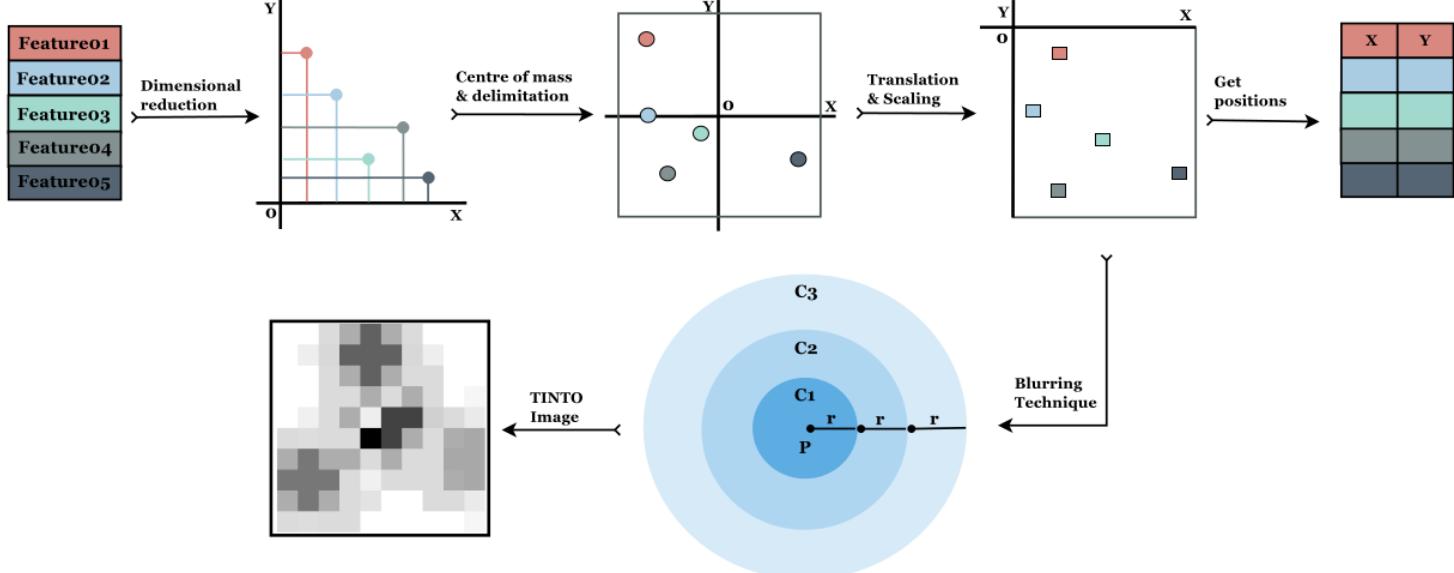
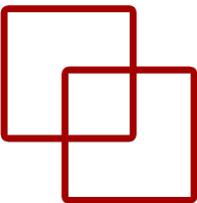
TINTO

Metodología



TINTO

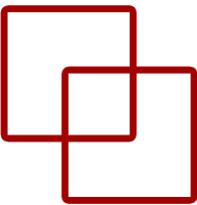
Metodología



(c) Blurring with maximum value.

TINTO

Algoritmo



1. Reducción de la dimensionalidad de los datos: La matriz inicial se transpone. Obsérvese que cada característica se representa con un color diferente a efectos de presentación, aunque la figura resultante constará de dos canales, es decir, blanco y negro. Esta tarea utiliza un algoritmo de reducción de la dimensionalidad. En este caso, TINTO exploró dos de estos algoritmos: PCAy t-SNE.
2. Centro de masa y delimitación: Una vez obtenidas las coordenadas, se determina el centro de gravedad de los puntos y, a continuación, se delimita la zona.
3. Escalado y posición de los píxeles. La matriz se transpone, se escala y los valores se redondean a valores enteros.
4. Posiciones características de los píxeles: Los valores obtenidos serían las posiciones de los píxeles característicos para la creación del patrón de la imagen.
5. Por último, la técnica de difuminado permite expandir y difuminar el valor característico del píxel a sus píxeles vecinos, enriqueciendo así la información de la imagen. Puede hacer un blurring max o mean si se superponen dos píxeles.

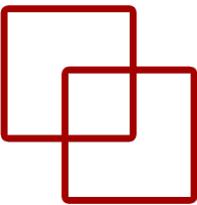
Algorithm 1: TINTO

Data: algorithm, pixels, blur, amplification, distance, steps, option, seed, times
Result: Grayscale images

Normalize features: $X = \text{norm01}(X)$;
Apply dimensionality reduction: $B = \text{DRA}(X^t, 2)$;
Calculate characteristic points: $P = \{p_1, \dots, p_{varN}\}$ where $p_i = (b_{i1}, b_{i2}) \forall i \in varN$;
Scale and define pixel positions: $C = G(X)$;
Calculate characteristic points: $Q = F_2(F_1(p_i))$;
for $i = 0 : N$ **do**
 $img = H(C, Q)$;
 Save image img ;
end

TINTO

Algoritmo



1. Reducción de la dimensionalidad de los datos: La matriz inicial se transpone. Obsérvese que cada característica se representa con un color diferente a efectos de presentación, aunque la figura resultante constará de un solo color.
2. Centro de masa y desviación estándar: se determina el centro de masa y se delimita la zona de interés.
3. Escalado y posición: se escala la imagen y los valores se redondean.
4. Posiciones características: serían las posiciones de los píxeles del patrón de la imagen.
5. Por último, la técnica de fusionamiento de información (Information Fusion) aplica el valor característico del píxel a sus píxeles vecinos, enriqueciendo así la información de la imagen. Puede hacer un blurring max o mean si se superponen dos píxeles.

Código y formulización matemática muy compleja. Podéis verlo en estos dos artículos:

Information Fusion
SoftwareX

blur, amplification, distance, times

norm01(X);
function: $B = DRA(X^t, 2)$;
points: $P =$
 $p_i = (b_{i1}, b_{i2}) \forall i \in varN$;
positions: $C = G(X)$;
points: $Q = F_2(F_1(p_i))$;



TINTOlib

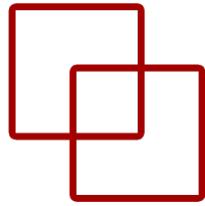


**ETS de
Ingeniería
Informática**



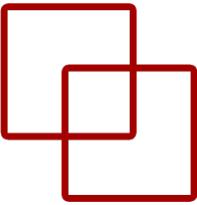
UNED

Métodos TINTOlib



| Método | Importar | Declarar |
|----------------|--|------------------------------|
| TINTO | from TINTOlib.tinto import TINTO | TINTO(*parameters*) |
| REFINED | from TINTOlib.refined import Refined | REFINED(*parameters*) |
| IGTD | from TINTOlib.igtd import IGTd | IGTD(*parameters*) |
| SuperTML | from TINTOlib.supertml import SuperTML | SuperTML(*parameters*) |
| DistanceMatrix | from TINTOlib.distanceMatrix import DistanceMatrix | DistanceMatrix(*parameters*) |
| BarGraph | from TINTOlib.barGraph import BarGraph | BarGraph(*parameters*) |
| Combination | from TINTOlib.combination import Combination | Combination(*parameters*) |

Arquitectura TINTOlib



| TINTO |
|-------------------------------|
| problem |
| verbose |
| algorithm |
| blur |
| amplification |
| distance |
| steps |
| option |
| seed |
| times |
| submatrix |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |

| REFINED |
|-------------------------------|
| problem |
| verbose |
| hIterations |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |
| SuperTML |
| problem |
| verbose |
| Postal Code |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |

| IGTD |
|-------------------------------|
| problem |
| verbose |
| scale |
| scale |
| fea_dist_method |
| image_dist_method |
| save_image_size |
| max_step |
| val_step |
| error |
| switch_t |
| min_gain |
| seed |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |

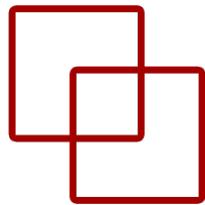
| BarGraph |
|-------------------------------|
| problem |
| verbose |
| pixel_width |
| gap |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |

| DistanceMatrix |
|-------------------------------|
| problem |
| verbose |
| scale |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |

| Combination |
|-------------------------------|
| problem |
| verbose |
| pixel_width |
| gap |
| saveHyperparameters(filename) |
| loadHyperparameters(filename) |
| generateImages(data,folder) |



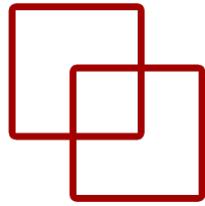
Clases y Parámetros comunes



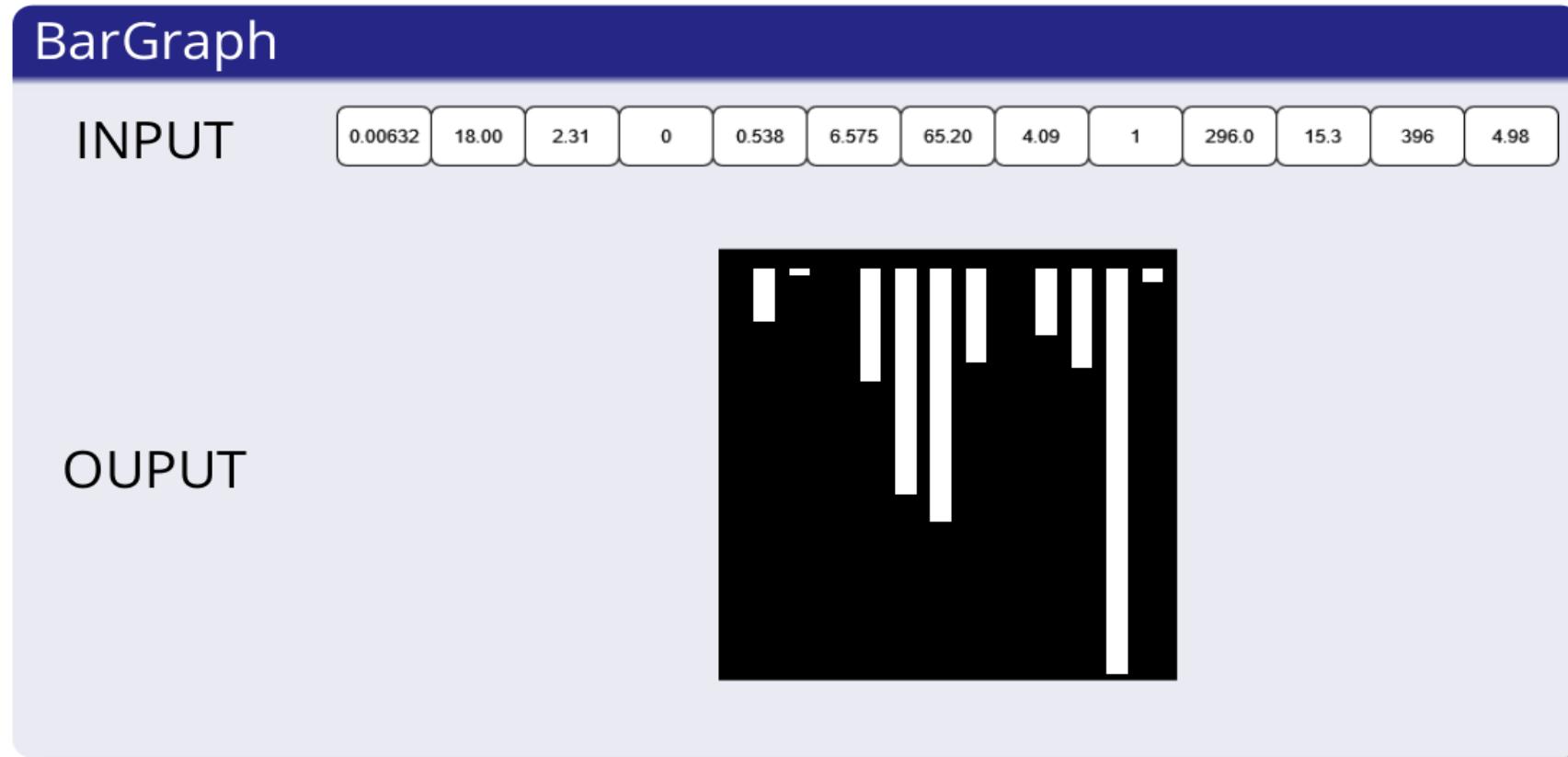
| Función | Descripción | Output |
|-------------------------------|--|--|
| saveHyperparameters(filename) | Guarda los hiperparámetros usados | Archivo .pkl con la configuración usada |
| loadHyperparameters(filename) | Carga los hiperparámetros de un archivo .pkl Genera las imágenes y las agrupa en carpetas. | - |
| generateImages(data, folder) | ■ data: dataframe pandas o ruta al csv de datos ■ folder: ruta de la carpeta donde guardar las imágenes | Carpetas con las imágenes y un csv con las rutas de las imágenes |

| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|-----------|--------------------------|-------------------|--|
| problem | [supervised, regression] | supervised | Sirve para definir el tipo de problema que se quiere abordar. Afectará en la distribución por carpetas de las imágenes. En los problemas de clasificación se agruparán las clases por carpetas, mientras que en regresión se agruparan todas las imágenes en la misma carpeta. |
| verbose | boolean | False | Muestra información en la terminal sobre la ejecución de los métodos. |

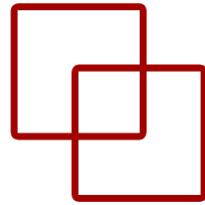
BarGraph



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|-------------|------------------|-------------------|---------------------------------------|
| pixel_width | integer | 3 | Anchura de las barras en píxeles |
| gap | integer | 2 | Píxeles de distancia entre cada barra |



DistanceMatrix



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|-----------|------------------|-------------------|--|
| scale | integer | 1 | La escala de la imagen. Con escala 1 cada píxel representa un valor de la matriz de distancias |

DistanceMatrix

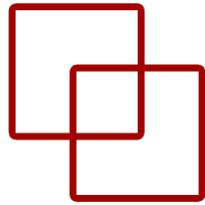
INPUT

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

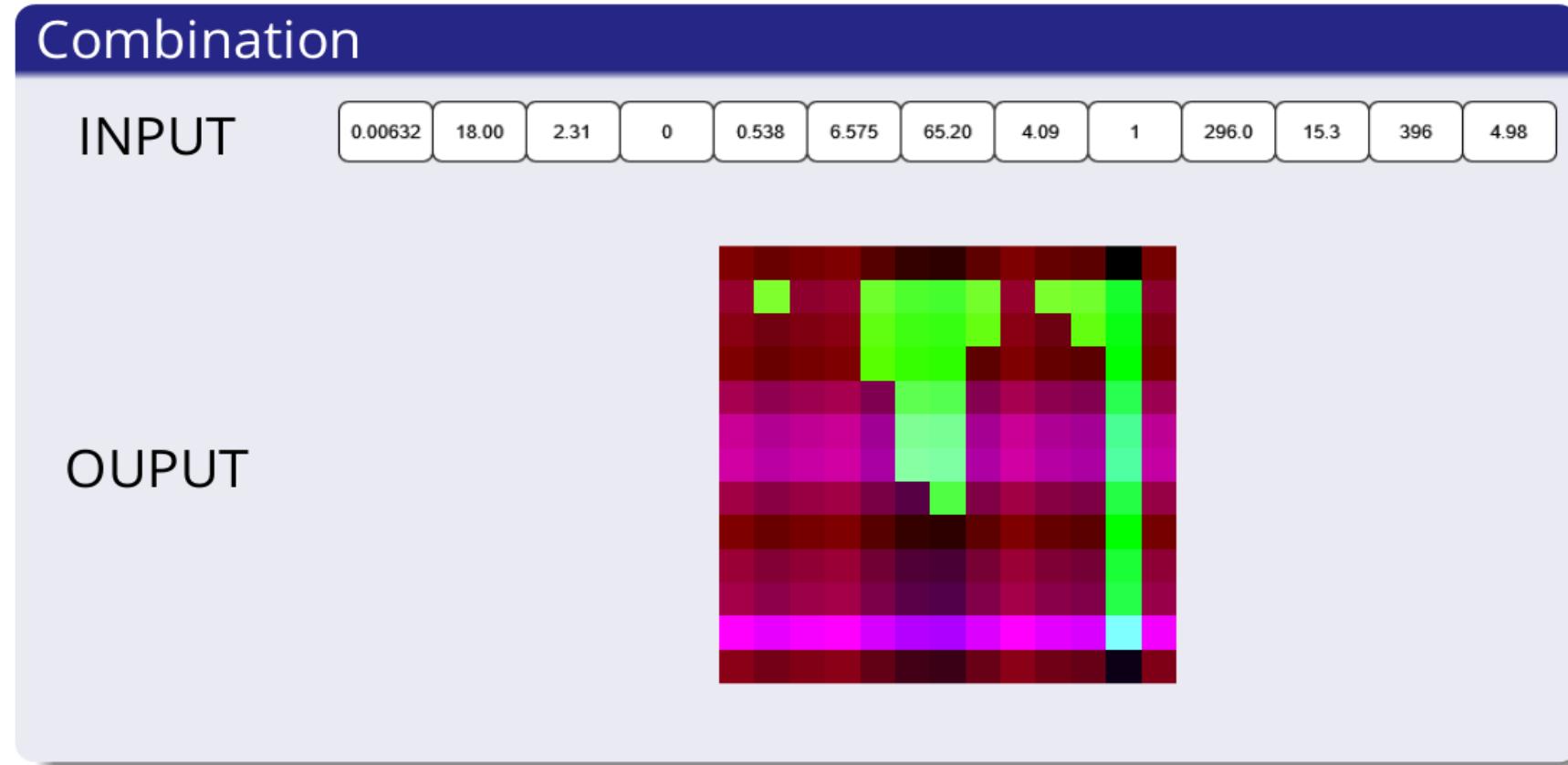
OUTPUT

A 12x12 pixel grayscale heatmap. The pixels transition from white at the bottom to black at the top. A single vertical column of black pixels runs along the right edge of the grid.

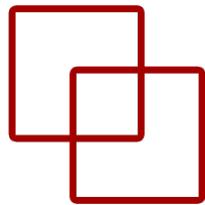
Combination



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|-------------|------------------|-------------------|---------------------------------------|
| pixel_width | integer | 3 | Anchura de las barras en píxeles |
| gap | integer | 2 | Píxeles de distancia entre cada barra |



SuperTML



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|------------|------------------|-------------------|---|
| columns | integer | 4 | Número de columnas en las que se dividirá la imagen |
| image_size | integer | 224 | Tamaño de la imagen |
| font_size | integer | 10 | Tamaño de fuente de la imagen |

SuperTML

INPUT

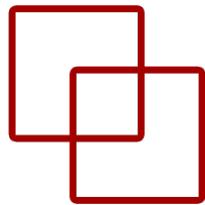
0.00632 18.00 2.31 0 0.538 6.575 65.20 4.09 1 296.0 15.3 396 4.98

OUTPUT

The output is a 4x4 grid of numbers, each representing one of the 16 elements from the input list. The numbers are arranged in a 4x4 pattern:

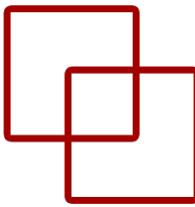
| | | | |
|-------|---------|--------|---------|
| 0.006 | 18.000 | 2.310 | 0.000 |
| 0.538 | 6.575 | 65.200 | 4.090 |
| 1.000 | 296.000 | 15.300 | 396.900 |
| 4.980 | | | |

TINTO



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|---------------|------------------|-------------------|--|
| algorithm | [PCA, t-SNE] | PCA | Algoritmo de reducción de dimensionalidad a utilizar |
| pixels | integer | 20 | El numero de píxeles de cada lado de la imagen. Píxeles totales = píxeles x píxeles |
| blur | boolean | False | Uso del método de blurring o no |
| amplification | float | np.pi | En caso de usar blurring, radio de amplificación del blurring |
| distance | integer[0,1] | 0.1 | En caso de usar blurring, distancia en los que tendrá efecto el blurring (en porcentaje) |
| steps | integer | 4 | En caso de usar blurring, pasos del método de blurring |
| option | [mean, maximum] | mean | En caso de usar blurring, técnica para abordar la superposición de píxeles |
| seed | integer | 20 | Semilla utilizada para los números aleatorios |
| times | integer | 4 | En caso de utilizar t-SNE como algoritmo, indica las veces que se replicara el t-SNE |
| submatrix | boolean | True | Utilizar submatrices para realizar los cálculos de manera más eficiente |

TINTO



TINTO

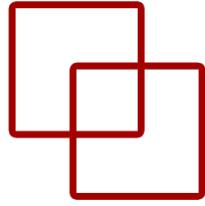
INPUT

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

OUTPUT

A 28x28 pixel grayscale image showing a highly pixelated version of the number '4'. The image is rendered in a blocky, low-resolution style with varying shades of gray.

REFINED



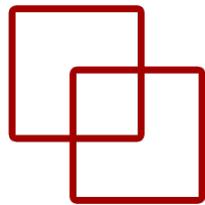
| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|--------------|------------------|-------------------|---|
| hcIterations | integer | 100 | Número de iteraciones máximas del algoritmo hill climbing |

REFINED

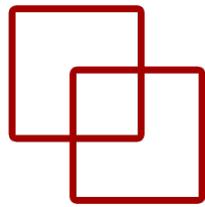
INPUT

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

OUTPUT



| Parámetro | Tipo de variable | Valor por defecto | Descripción |
|-------------------|------------------------------------|-------------------|--|
| scale | [integer,integer] | [2,2] | Píxeles característicos de la imagen final (fila x columna) |
| fea_dist_method | [Pearson, Spearman, Euclidean set] | Pearson | Método utilizado para evaluar la distancia entre pares de las características |
| image_dist_method | [Euclidean, Manhattan] | Euclidean | Método utilizado para calcular las distancias de los píxeles asignados en la imagen |
| save_image_size | integer | 20 | Número de píxeles de la imagen (n x n) |
| max_step | integer | 1000 | El número máximo de pasos en caso de que el algoritmo no converja |
| val_step | integer | 50 | Número de pasos para verificar la ganancia en la función objetivo para determinar la convergencia |
| error | [squared, abs] | squared | Función para evaluar la diferencia entre la clasificación de distancia de características y la clasificación de distancia de píxeles |
| switch_t | integer | 0 | El umbral para determinar si la permutación de píxeles debe realizarse |
| min_gain | float | 0.00001 | Si la función objetivo no se mejora más que 'min_gain' en los pasos 'val_step', el algoritmo termina |
| seed | integer | 1 | Semilla utilizada para los números aleatorios |



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 |
|---------|-------|------|---|-------|-------|-------|------|---|-------|------|-----|------|

OUTPUT

A 4x4 grid of colored squares. The colors transition from white to light gray, medium gray, dark gray, and black. The pattern is a checkerboard-like gradient. A single black square is located in the bottom-right position of the grid.

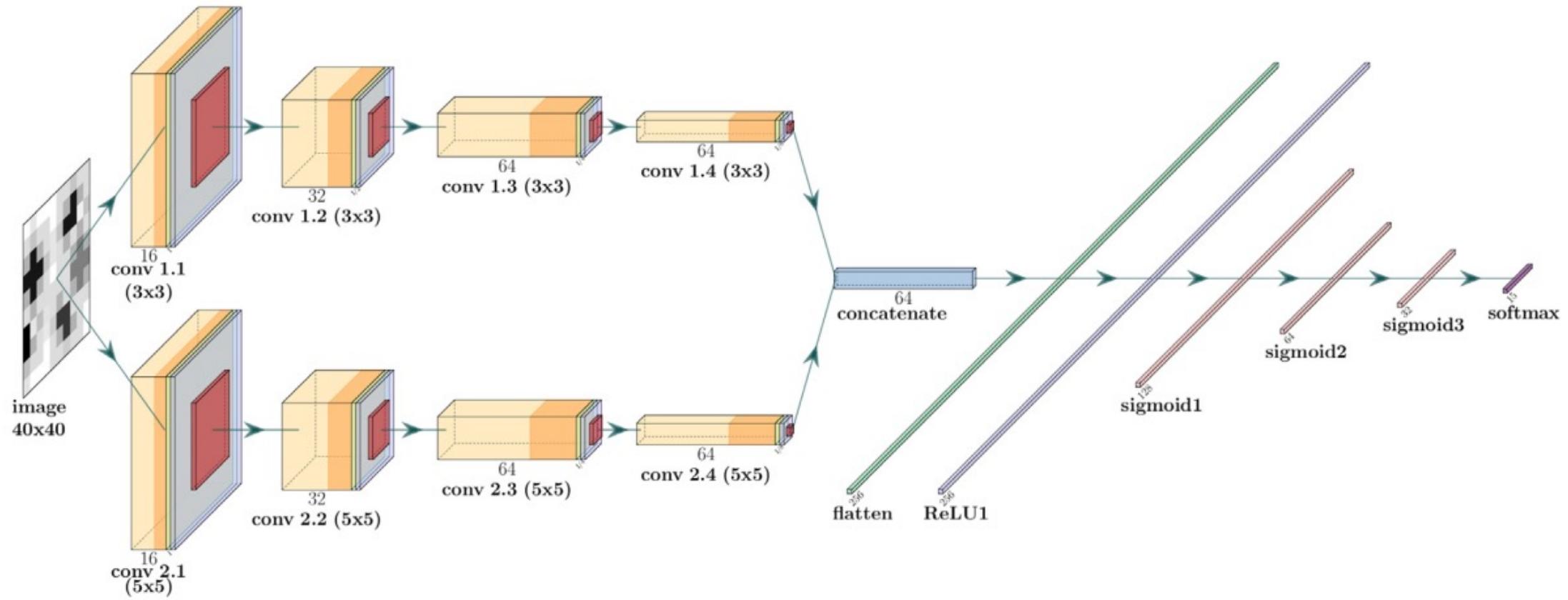
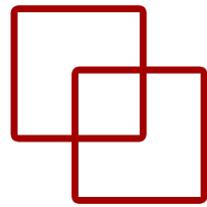
\Arquitecturas Híbridas

ETS de
Ingeniería
Informática

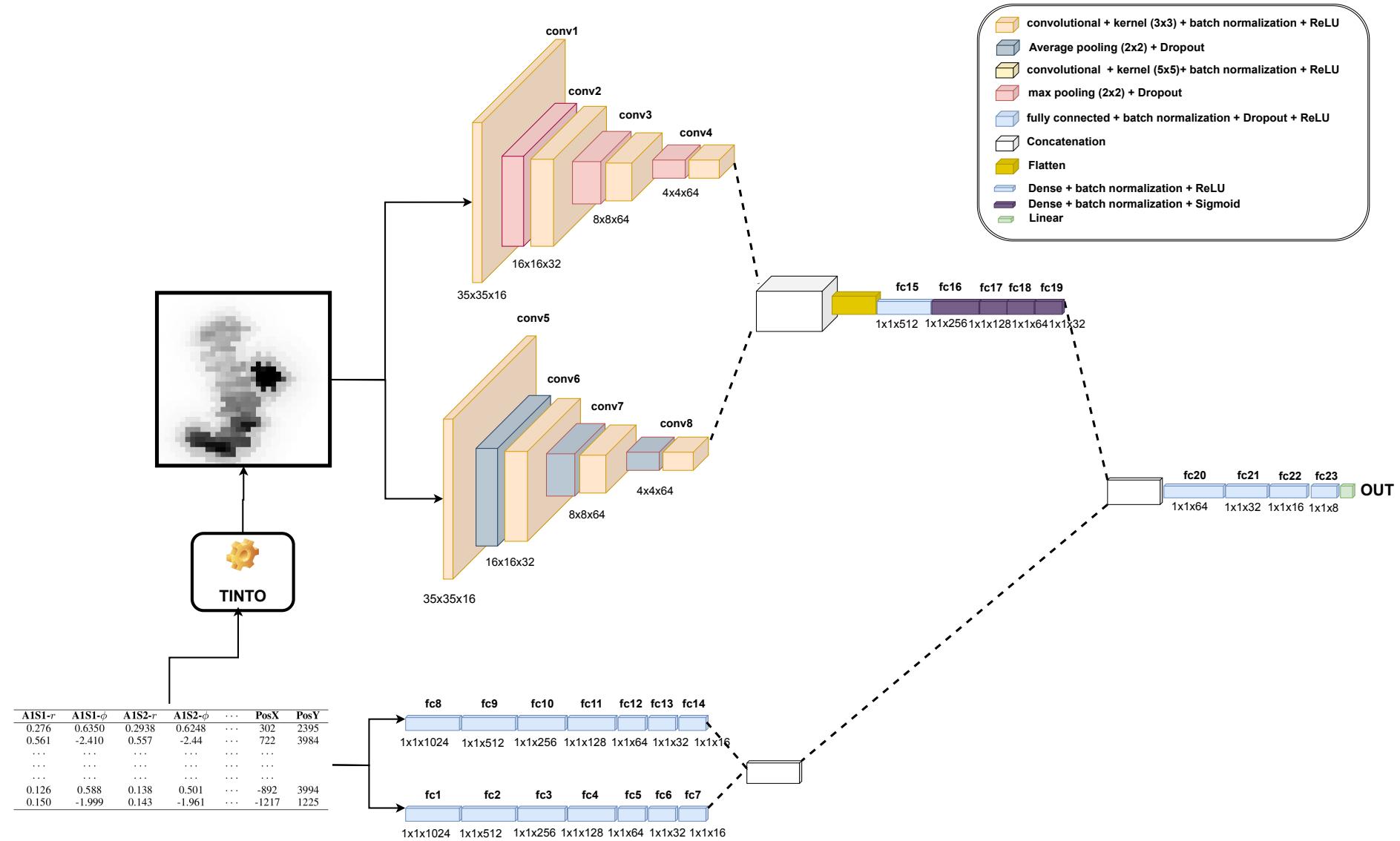
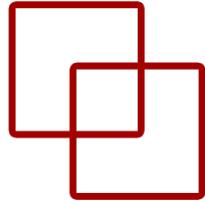


UNED

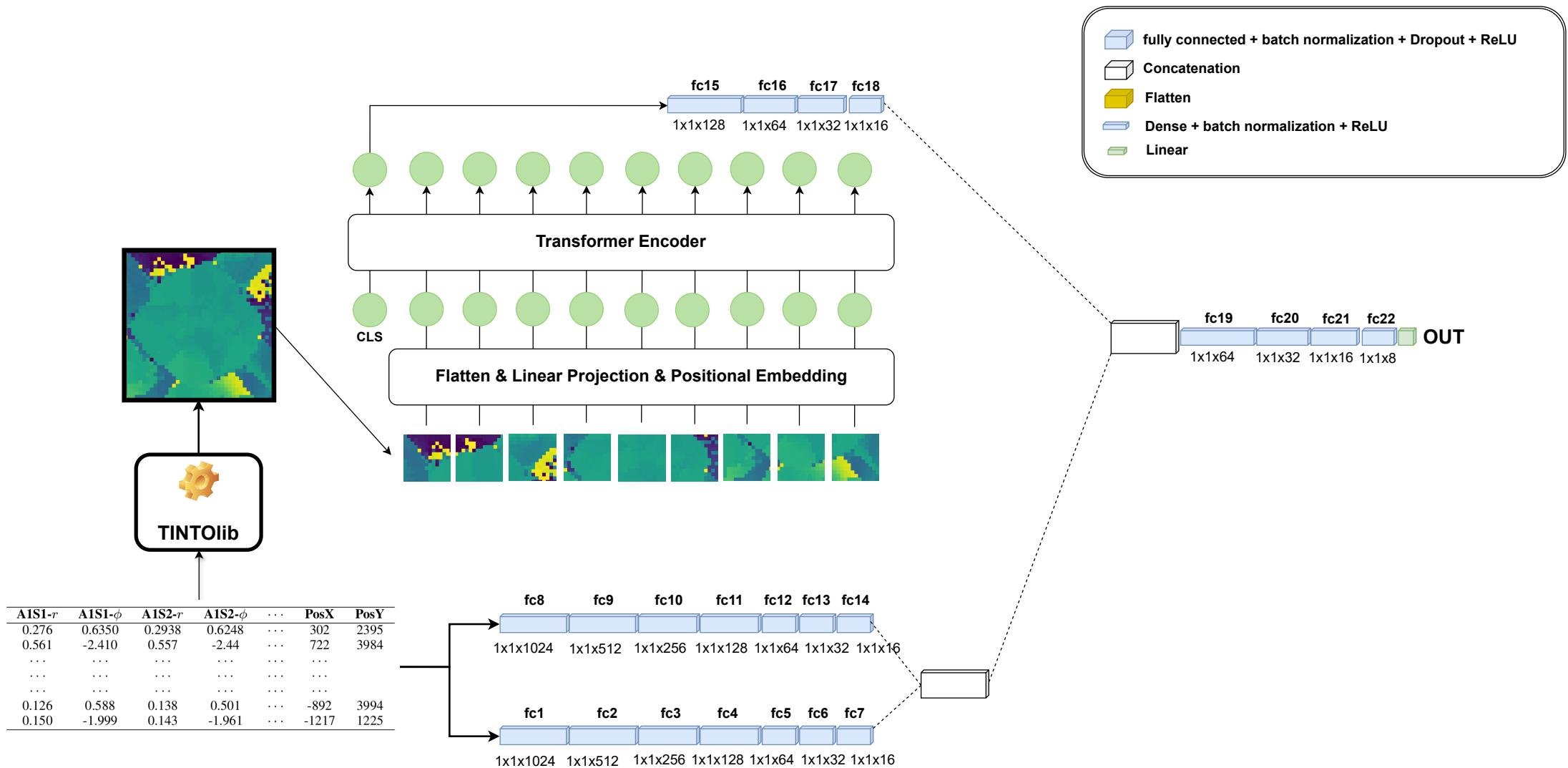
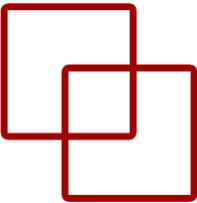
CNN



HyNN



HyViT



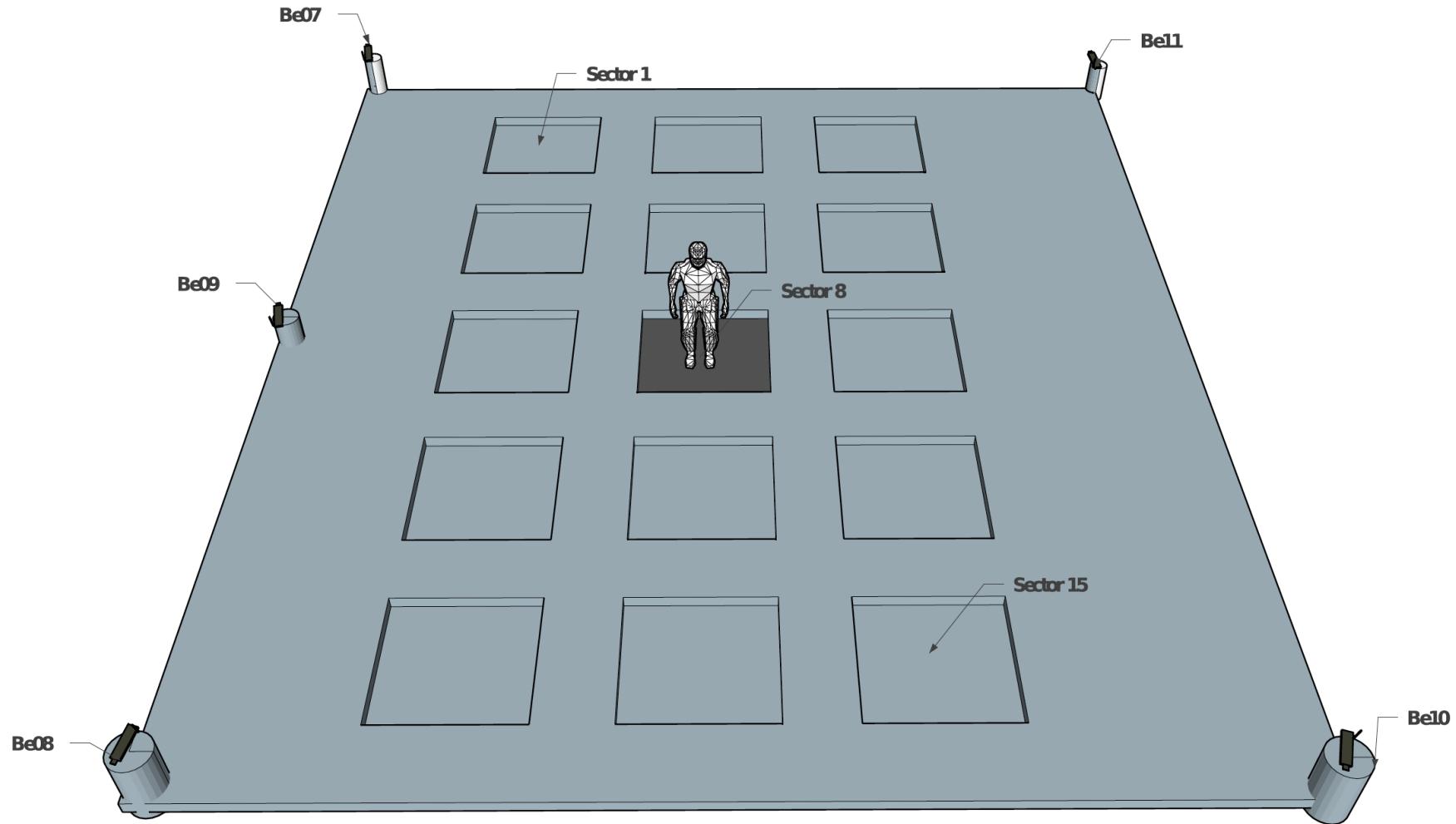
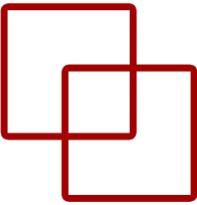
‘Caso de uso 1: Localización en interiores con BLE

ETS de
Ingeniería
Informática

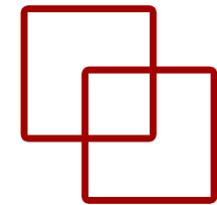


UNED

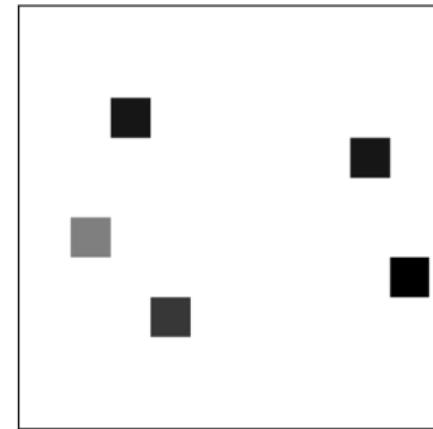
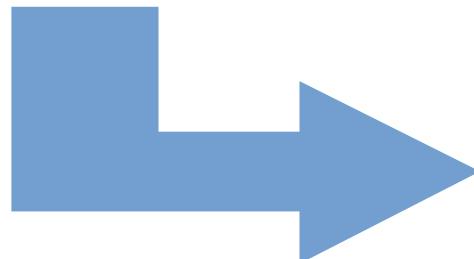
Escenario



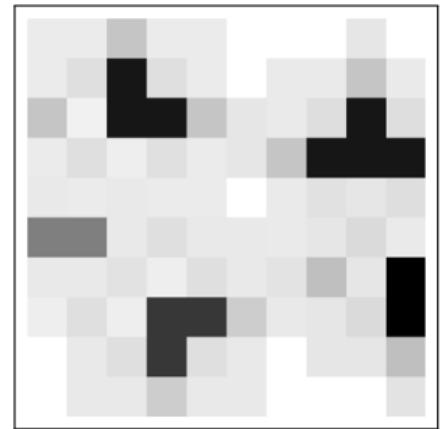
Transformación de datos



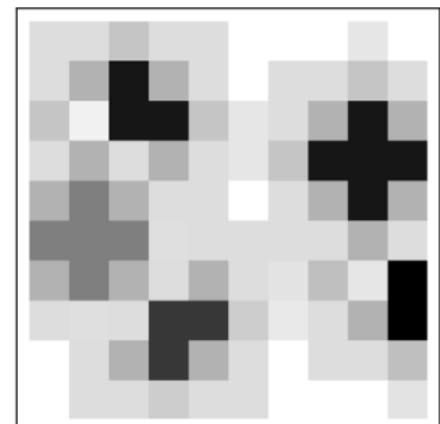
| Be07 | Be08 | Be09 | Be10 | Be11 | Sector |
|------|------|------|------|------|--------|
| -65 | -61 | -74 | -73 | -67 | 1 |
| -60 | -57 | -83 | -62 | -69 | 2 |
| -66 | -70 | -78 | -63 | -73 | 3 |
| ... | ... | ... | ... | ... | ... |
| -58 | -66 | -71 | -73 | -69 | 14 |
| -60 | -62 | -73 | -69 |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



(a) Without blurring.

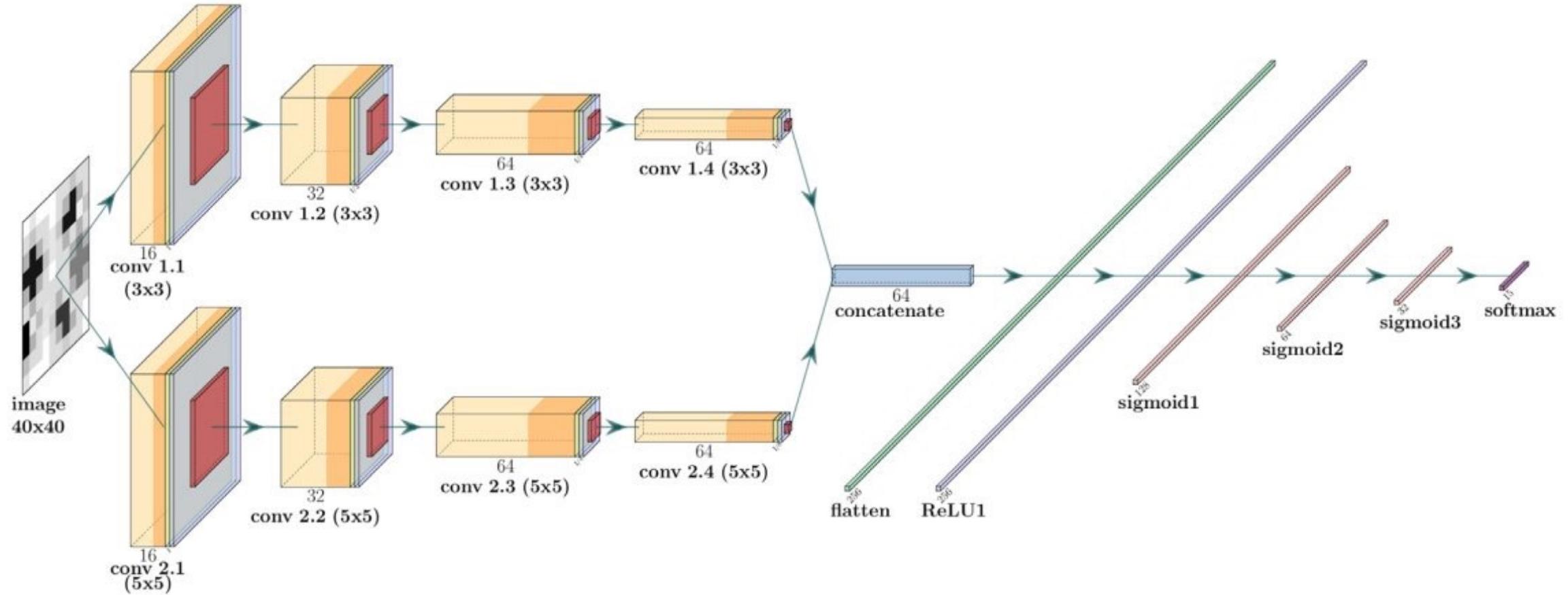
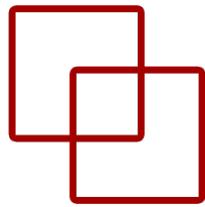


(b) Blurring with average value.

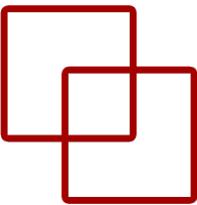


(c) Blurring with maximum value.

Escenario



Resultados TINTO



Con modelos clásicos

| Model | TxPower Setting | Acc | ME | RT |
|--------------|-----------------|--------------|--------------|--------|
| LoR | [4-1-2-3-1] | 76.71 | 0.487 | 10.001 |
| LDA | [6-1-3-3-5] | 77.60 | 0.481 | 0.031 |
| GNB | [4-1-2-6-1] | 86.01 | 0.289 | 0.020 |
| MLP | [4-1-4-6-1] | 72.21 | 0.613 | 3.660 |
| SVM | [6-6-3-3-5] | 83.00 | 0.512 | 2.881 |
| DT | [6-1-3-3-5] | 85.80 | 0.326 | 0.020 |
| <i>k</i> -NN | [6-1-3-6-1] | 87.87 | 0.249 | 0.031 |
| RF | [6-1-3-3-5] | 90.56 | 0.201 | 0.082 |
| ET | [6-1-3-3-5] | 91.03 | 0.209 | 0.050 |
| GBM | [6-1-3-3-5] | 91.87 | 0.180 | 3.269 |
| AB | [6-1-3-3-1] | 89.24 | 0.242 | 3.907 |
| VC | [6-1-3-3-5] | 92.08 | 0.181 | 3.365 |

Con CNN

| TxPower | Alg. | Blurring | Acc | Loss | ME (m) |
|-------------|---------------|----------|--------------|--------------|--------------|
| [3-6-6-3-1] | PCA | Max. | 93.94 | 0.254 | 0.148 |
| [3-6-1-3-5] | <i>t</i> -SNE | Max. | 93.35 | 0.283 | 0.157 |
| [3-6-1-3-2] | PCA | Aver. | 92.91 | 0.364 | 0.161 |
| [3-4-1-2-2] | <i>t</i> -SNE | Aver. | 92.54 | 0.373 | 0.173 |

‘Caso de uso 2: Localización en interiores con MIMO

ETS de
Ingeniería
Informática



UNED

Escenario

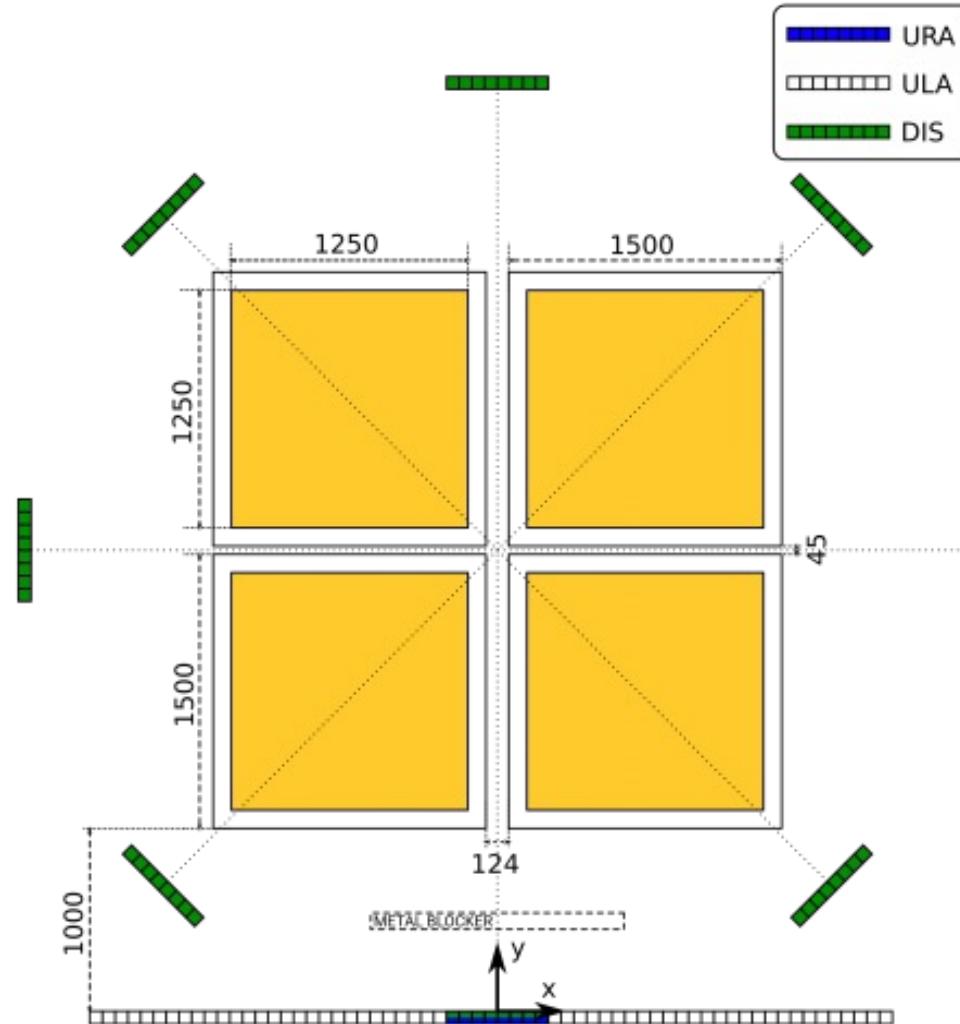
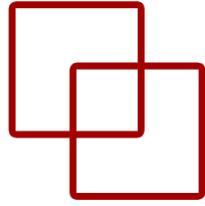
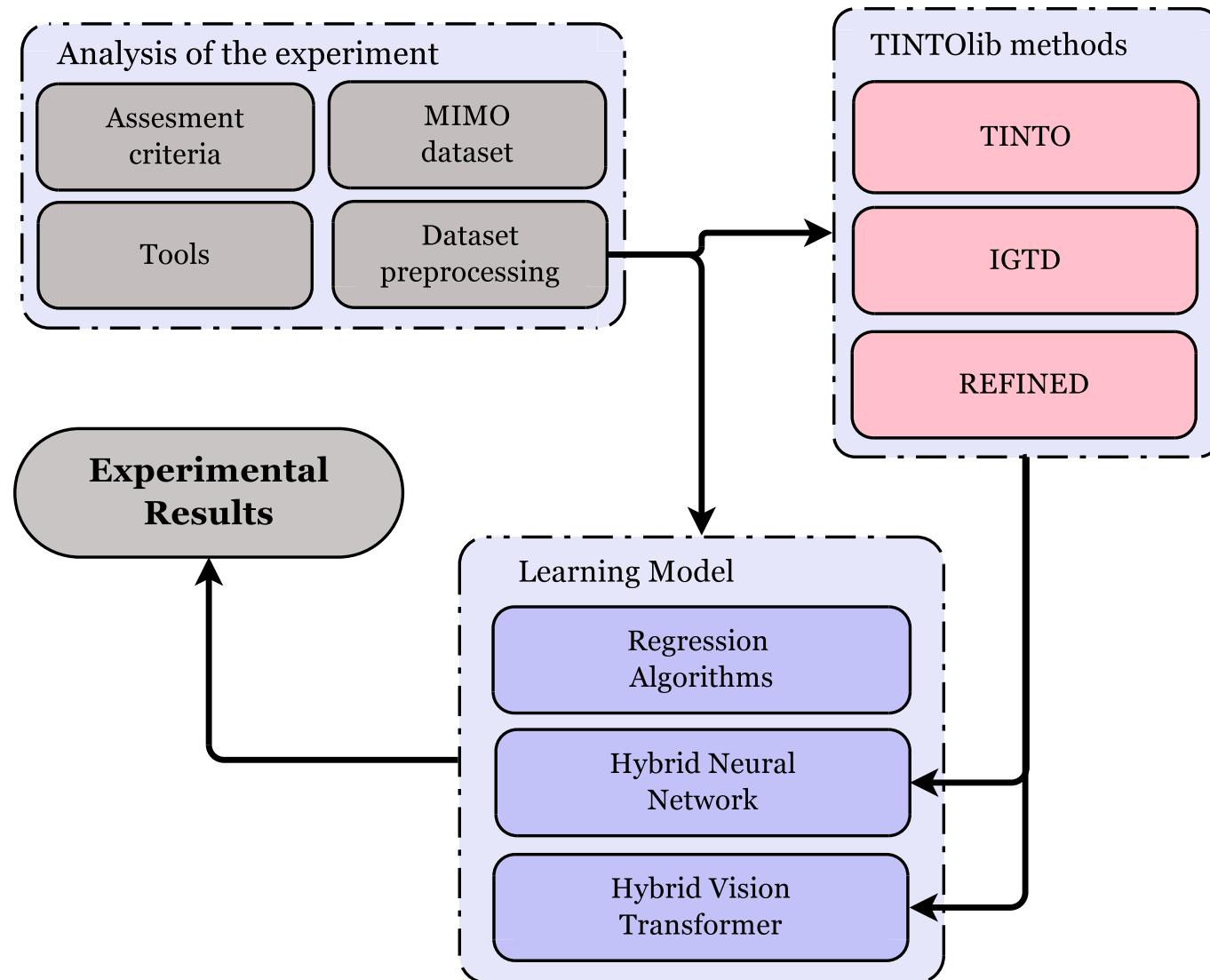
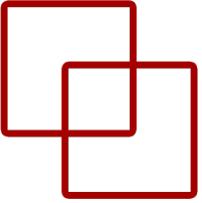
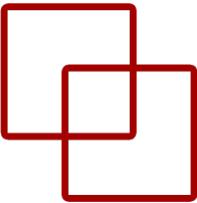


Figure taken from the paper: S. D. Bast, A. P. Guevara and S. Pollin, "CSI-based Positioning in Massive MIMO systems using Convolutional Neural Networks," 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 2020, pp. 1-5

Metodología



Imágenes sintéticas



| A1C1- <i>m</i> | A1C1- ϕ | A1C2- <i>m</i> | A1C2- 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 |

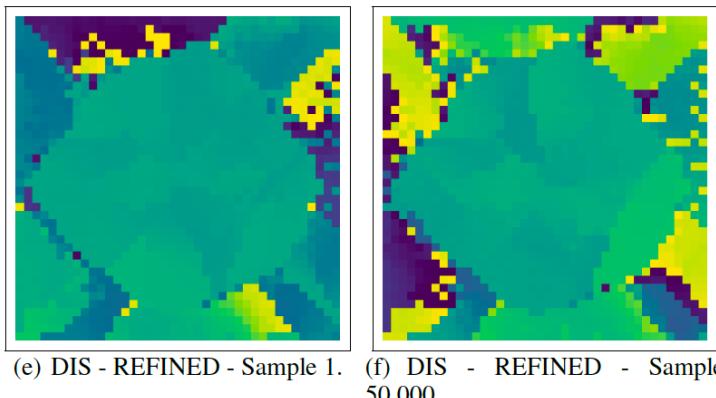
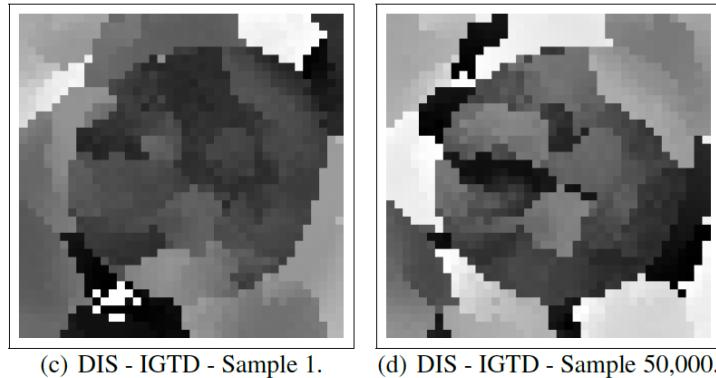
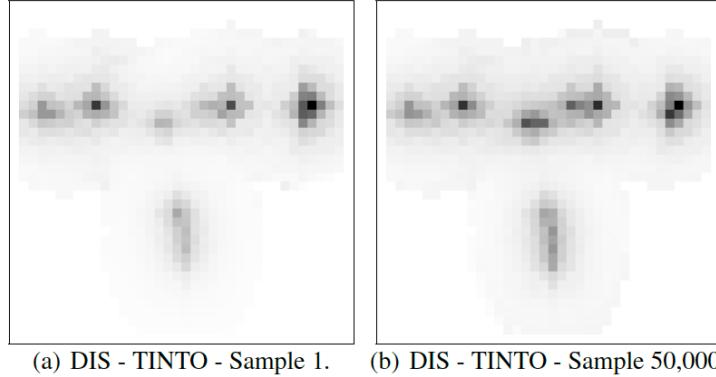
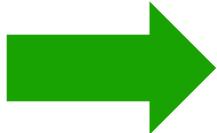
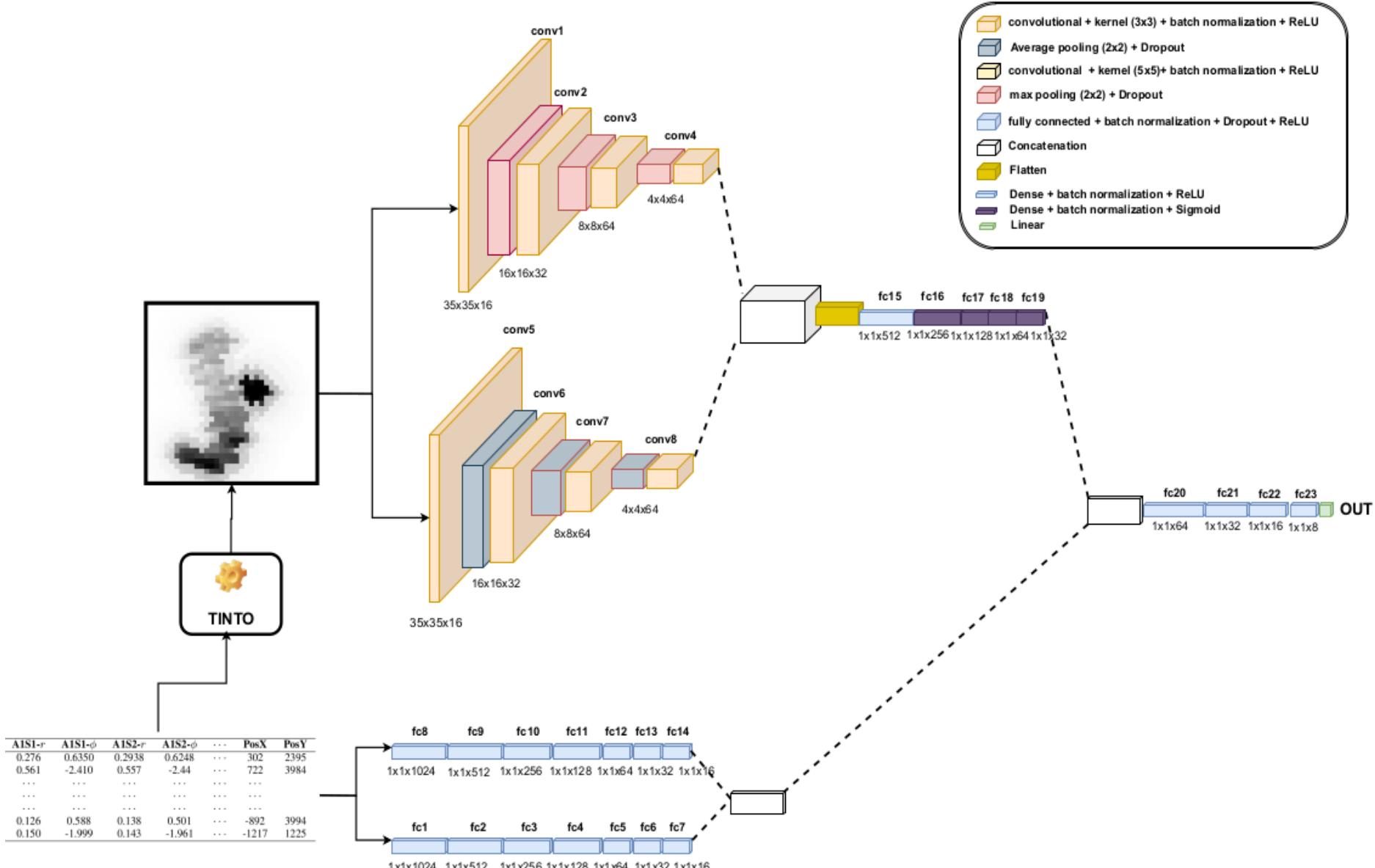
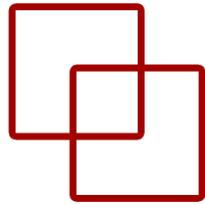
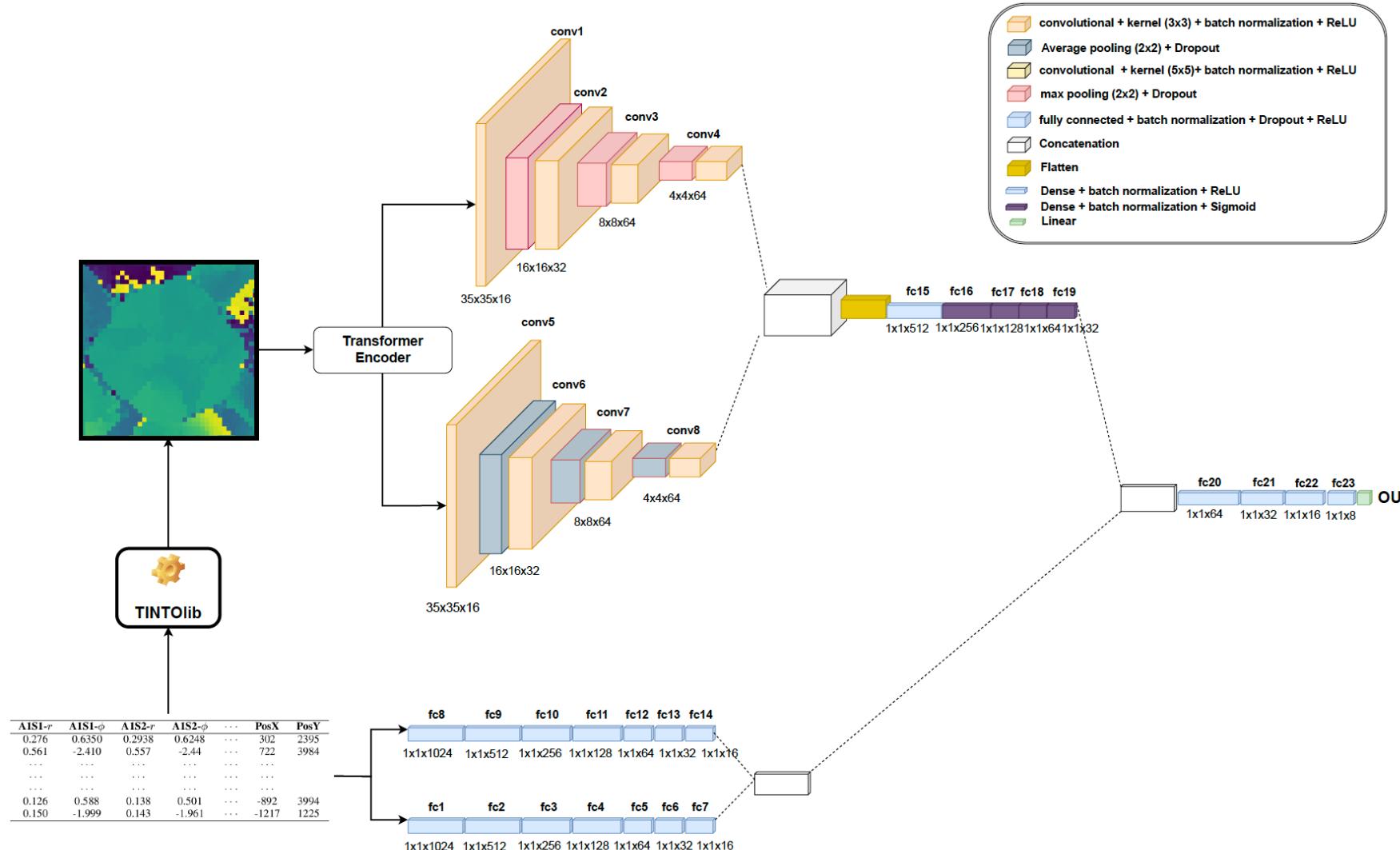
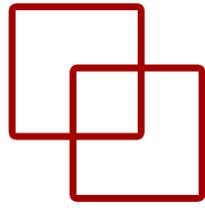


Figure 3. Synthetic image samples generated by TINTOlib for different samples in 8 antennas DIS scenario.

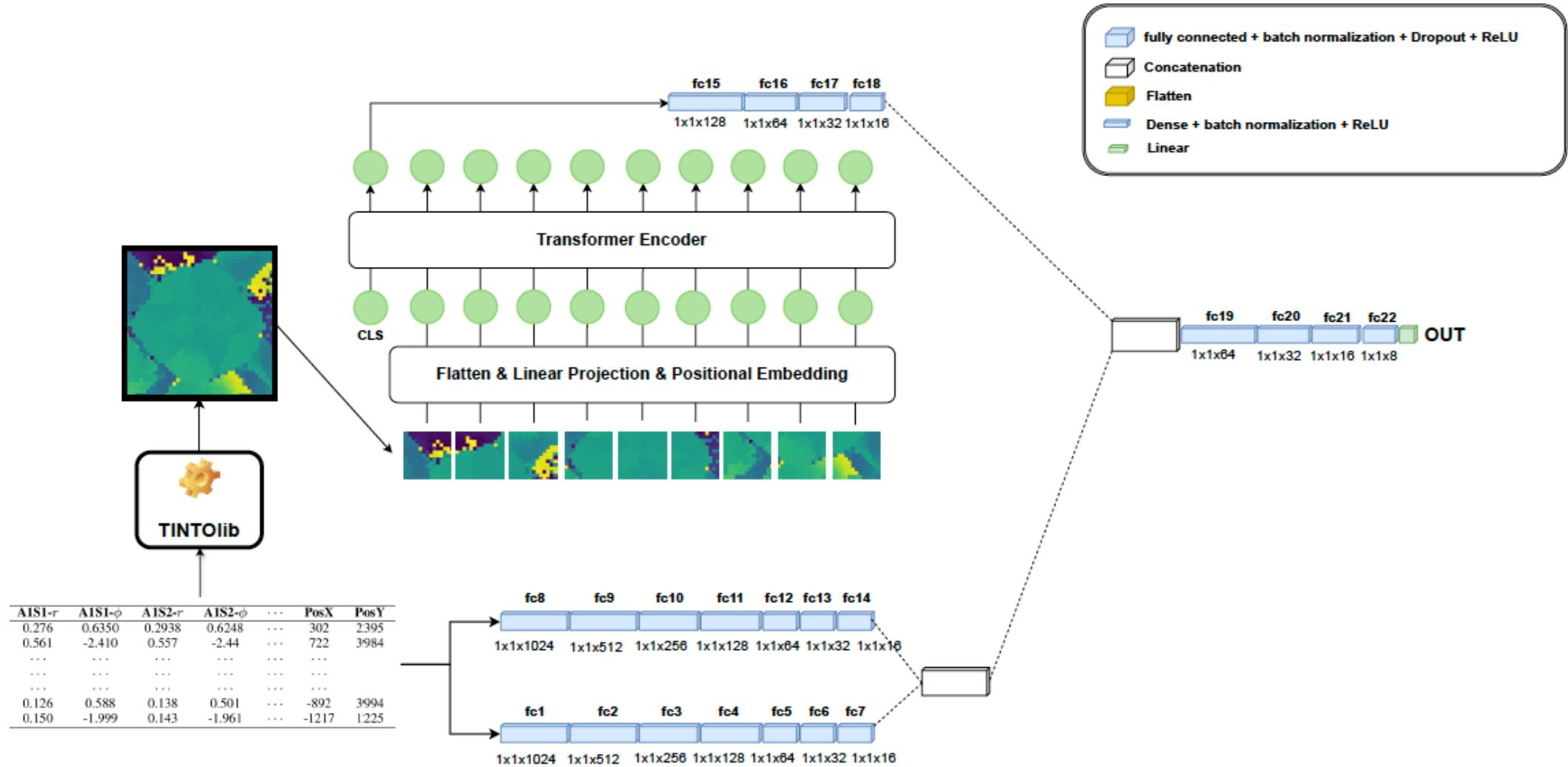
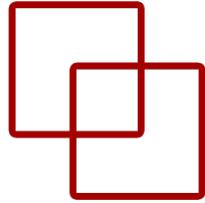
HyNN → CNN+MLP



HyNN → Transformer+CNN+MLP



HyViT → ViT+MLP



Baseline Results

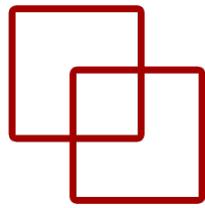


Table 2. RMSE (in mm) in validation (Val) and test split. Best results are shown in bold.

| 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 |

Hybrid Neural Networks Results

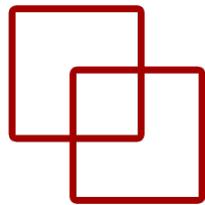


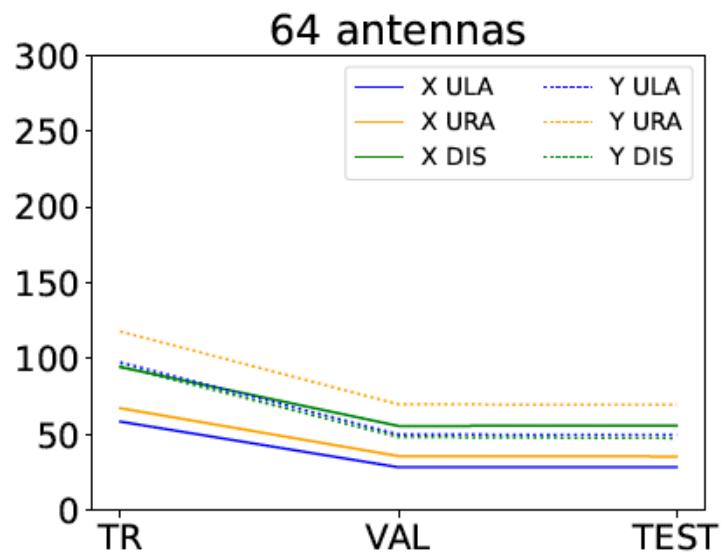
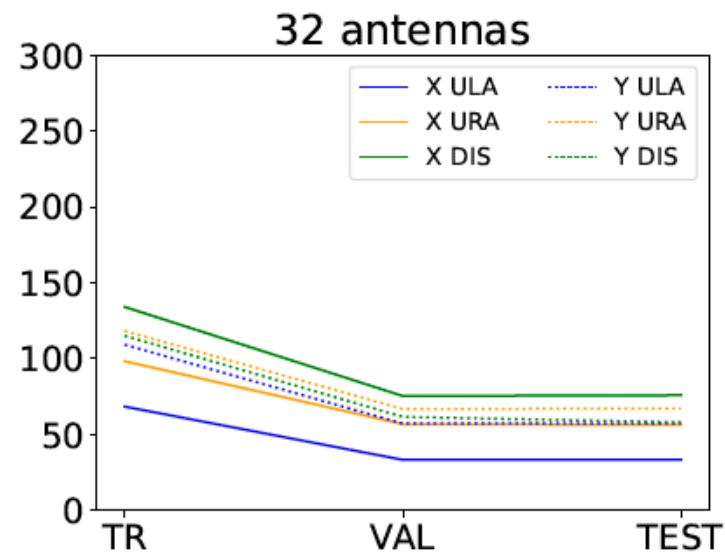
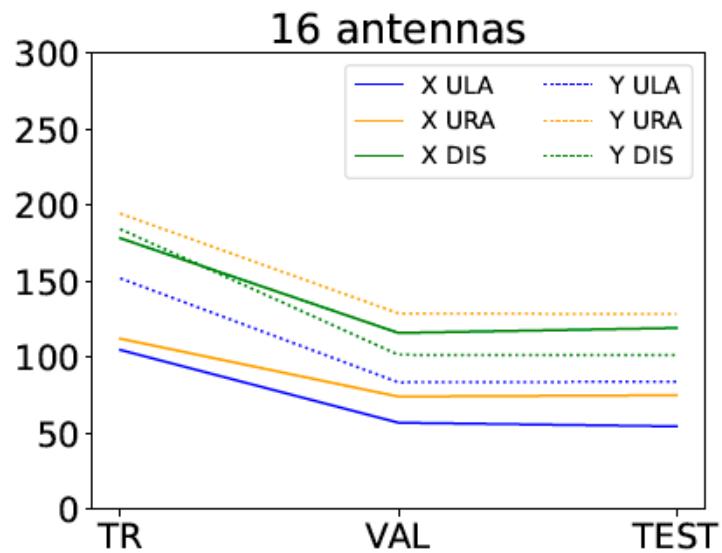
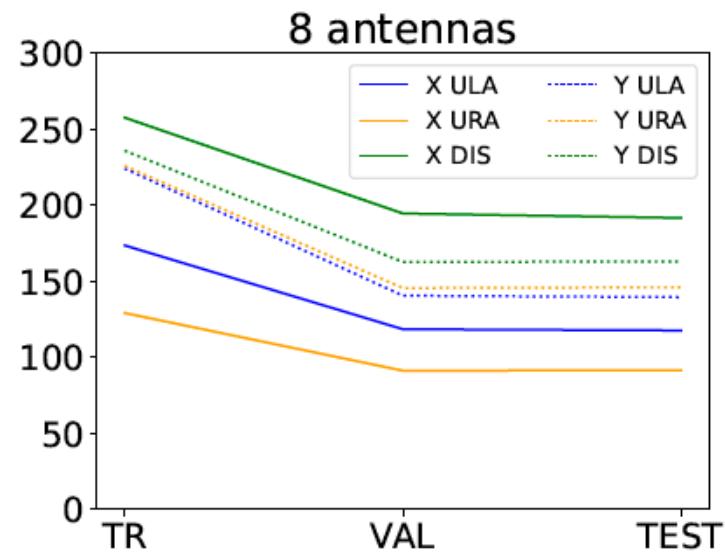
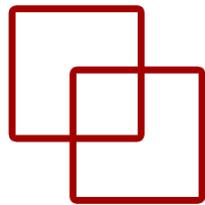
Table 2. RMSE (in mm) in validation (Val) and test split. Best results are shown in bold.

| 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 |

Inferencia



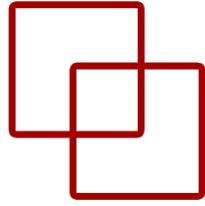
Más
información

ETS de
Ingeniería
Informática



UNED

Más información



- Documentación oficial de TINTOlib:
<https://tintolib.readthedocs.io/en/latest/>
- Librería TINTOlib en PyPI: <https://pypi.org/project/TINTOlib/>
- GitHub con el código de TINTOlib: <https://github.com/oeg-upm/TINTOlib>
- GitHub con el código de TINTO: <https://github.com/oeg-upm/TINTO>
- Artículo sobre TINTO y su aplicación en indoor localization. Incluye la definición formal matemática: <https://doi.org/10.1016/j.inffus.2022.10.011>
- Artículo sobre TINTO y su definición formal en Python:
<https://doi.org/10.1016/j.softx.2023.101391>



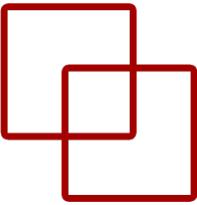
Challenge

ETS de
Ingeniería
Informática



UNED

Datasets



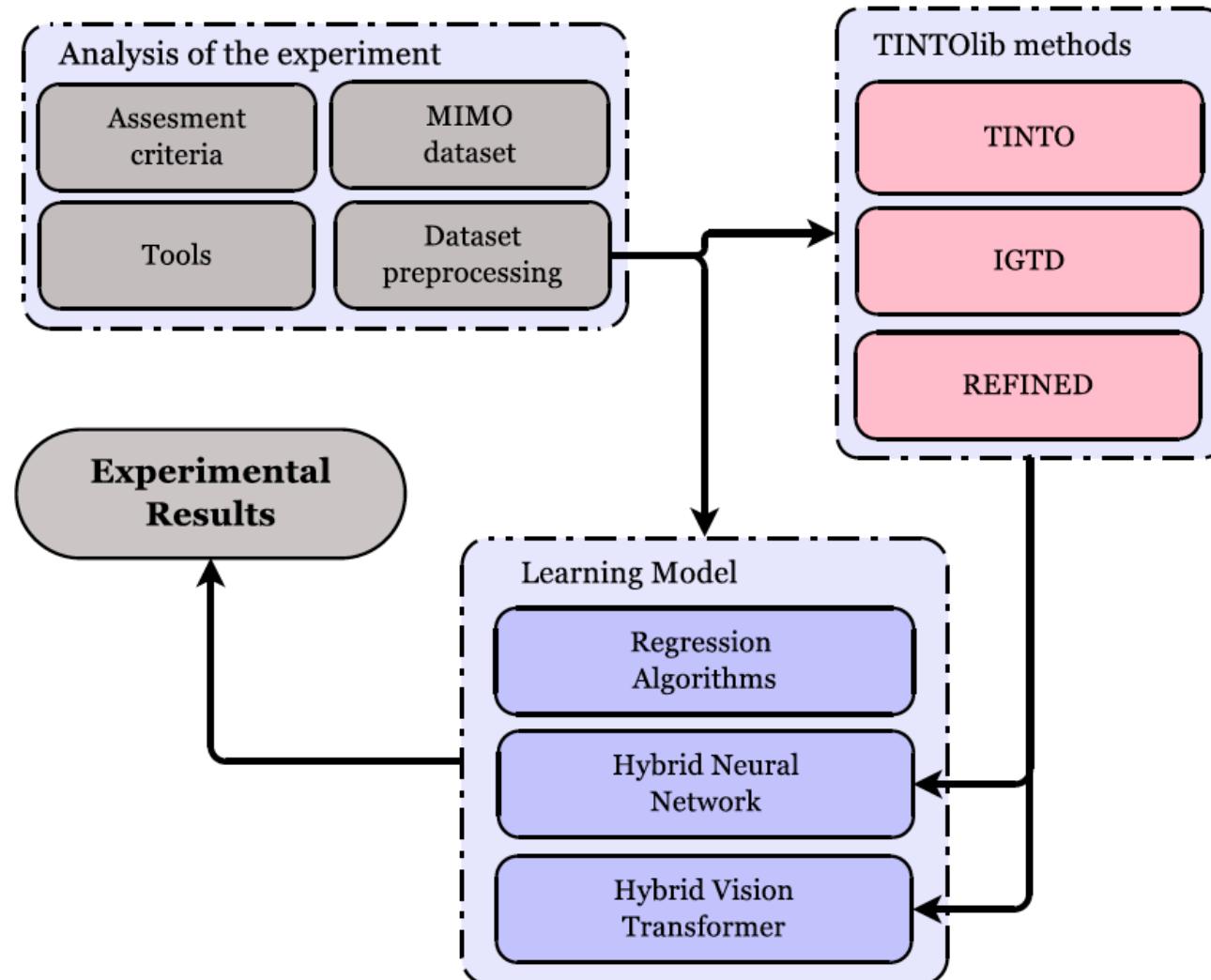
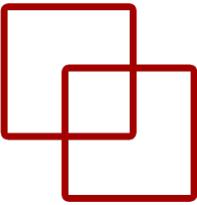
| | HELOC | Adult Income | HIGGS | Covertype | California Housing |
|----------------|--------|--------------|--------|-------------|--------------------|
| #Samples | 9.871 | 32.561 | 11 M. | 581.012 | 20.640 |
| #Num. features | 21 | 6 | 27 | 52 | 8 |
| #Cat. features | 2 | 8 | 1 | 2 | 0 |
| Task | Binary | Binary | Binary | Multi-Class | Regression |
| #Classes | 2 | 2 | 2 | 7 | - |

V. Borisov *et al.* (2023), "Deep Neural Networks and Tabular Data: A Survey," in IEEE Transactions on Neural Networks and Learning Systems

| Data set | Classes | Total | Features |
|--------------------------|--------------|-------|----------|
| Arboviruses/Inconclusive | Arboviruses | 11448 | 27 |
| | Inconclusive | | |
| Dengue/Chikungunya | Dengue | 11448 | |
| | Chikungunya | | |

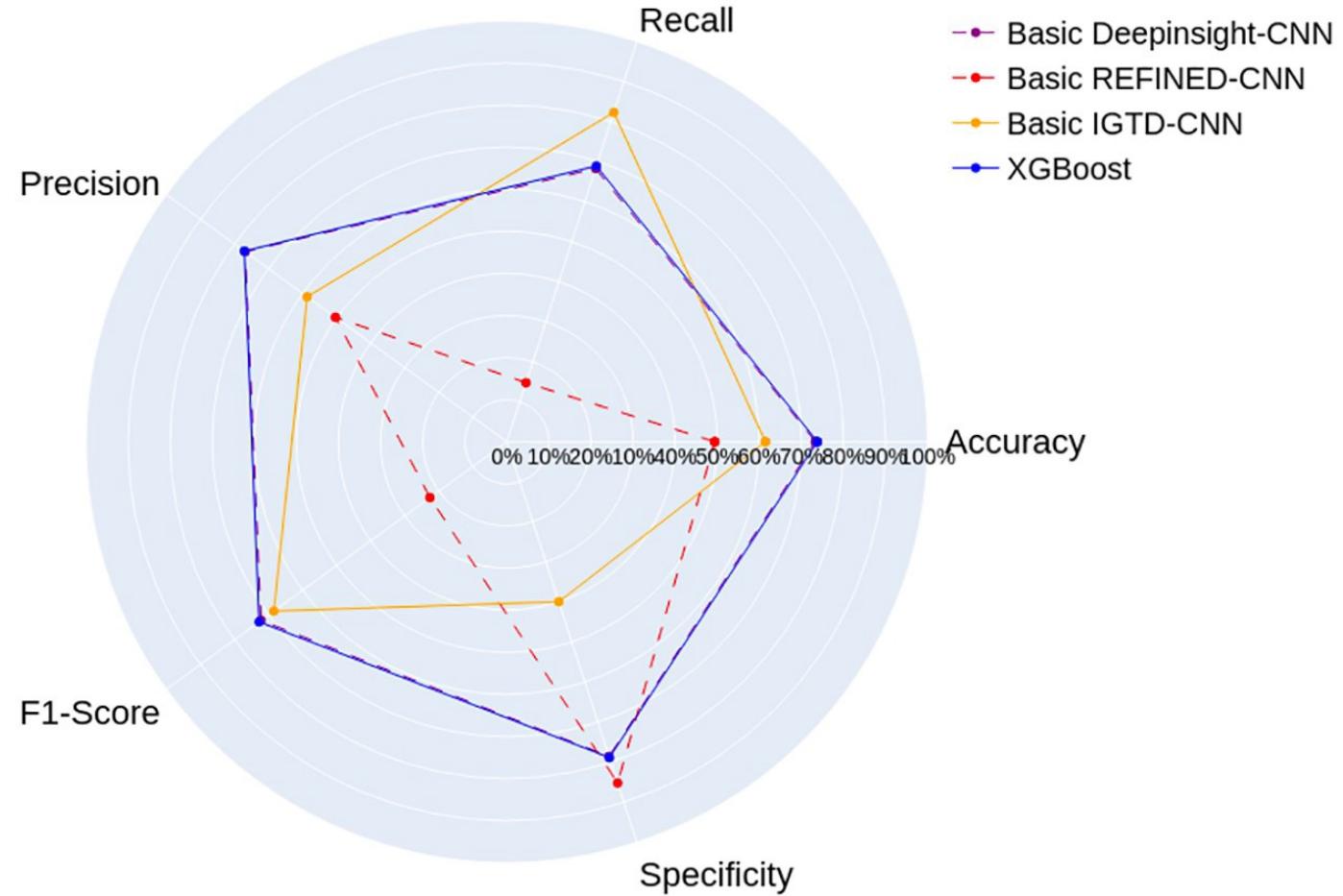
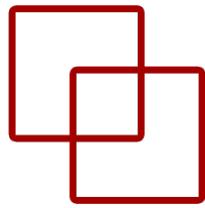
Medeiros Neto, L., et al. (2023). A comparative analysis of converters of tabular data into image for the classification of Arboviruses using Convolutional Neural Networks. Plos one, 18(12),

Challenge



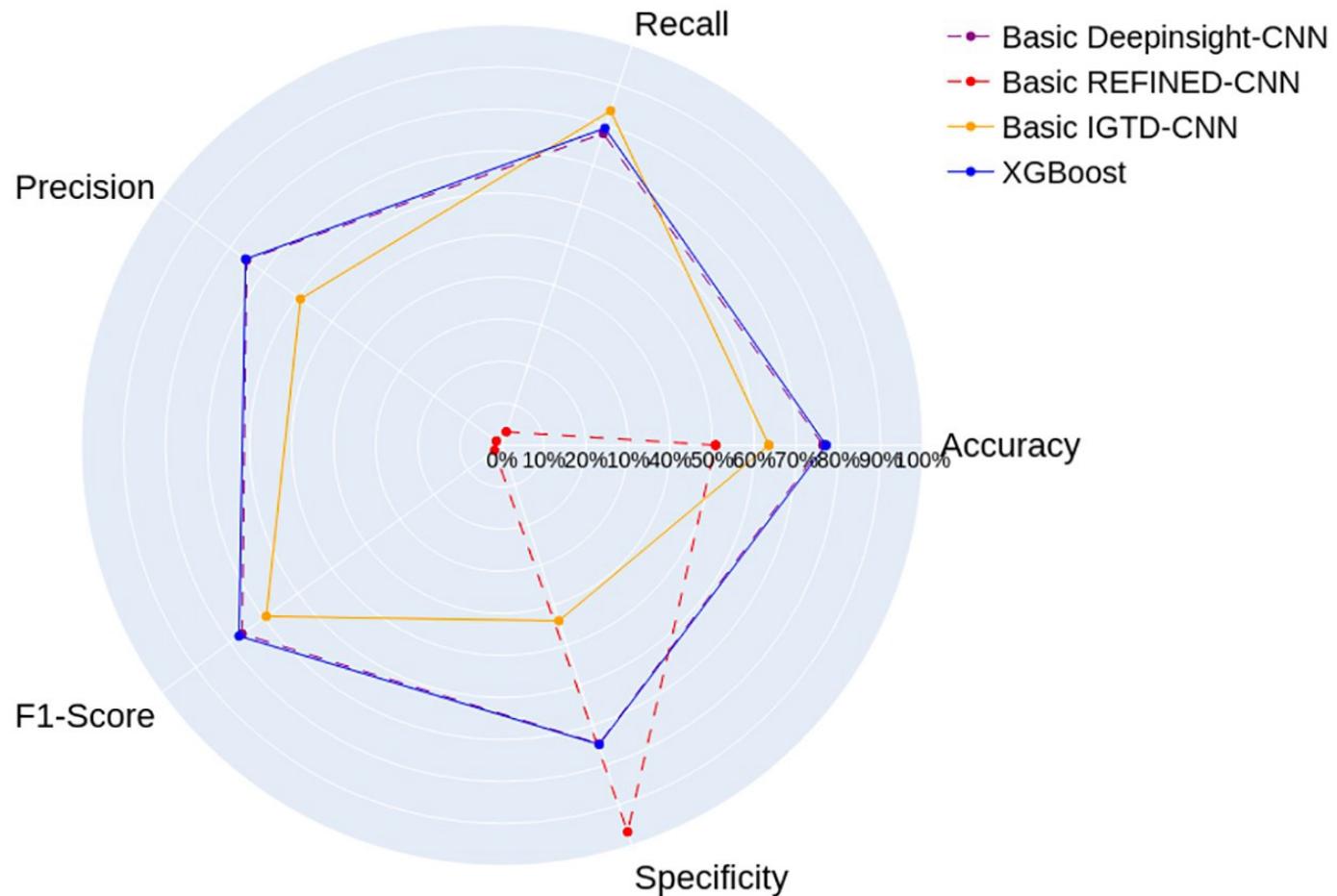
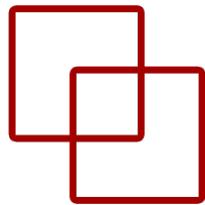
Results

Arboviruses/Inconclusive data set

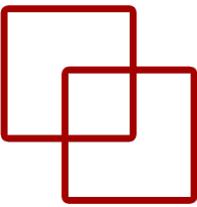


Results

Dengue/Chikungunya data set



Results



| | Method | HELOC | | Adult | | HIGGS | | Covertype | | Cal. Housing |
|------------------|----------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|--------------------|
| | | Acc ↑ | AUC ↑ | MSE ↓ |
| Machine Learning | Linear Model | 73.0±0.0 | 80.1±0.1 | 82.5±0.2 | 85.4±0.2 | 64.1±0.0 | 68.4±0.0 | 72.4±0.0 | 92.8±0.0 | 0.528±0.008 |
| | KNN [58] | 72.2±0.0 | 79.0±0.1 | 83.2±0.2 | 87.5±0.2 | 62.3±0.1 | 67.1±0.0 | 70.2±0.1 | 90.1±0.2 | 0.421±0.009 |
| | Decision Trees [195] | 80.3±0.0 | 89.3±0.1 | 85.3±0.2 | 89.8±0.1 | 71.3±0.0 | 78.7±0.0 | 79.1±0.0 | 95.0±0.0 | 0.404±0.007 |
| | Random Forest [196] | 82.1±0.2 | 90.0±0.2 | 86.1±0.2 | 91.7±0.2 | 71.9±0.0 | 79.7±0.0 | 78.1±0.1 | 96.1±0.0 | 0.272±0.006 |
| | XGBoost [46] | <u>83.5±0.2</u> | 92.2±0.0 | <u>87.3±0.2</u> | <u>92.8±0.1</u> | <u>77.6±0.0</u> | <u>85.9±0.0</u> | 97.3±0.0 | 99.9±0.0 | 0.206±0.005 |
| | LightGBM [70] | <u>83.5±0.1</u> | <u>92.3±0.0</u> | 87.4±0.2 | 92.9±0.1 | 77.1±0.0 | 85.5±0.0 | 93.5±0.0 | 99.7±0.0 | 0.195±0.005 |
| | CatBoost [71] | 83.6±0.3 | 92.4±0.1 | 87.2±0.2 | <u>92.8±0.1</u> | 77.5±0.0 | 85.8±0.0 | <u>96.4±0.0</u> | <u>99.8±0.0</u> | 0.196±0.004 |
| | Model Trees [197] | 82.6±0.2 | 91.5±0.0 | 85.0±0.2 | 90.4±0.1 | 69.8±0.0 | 76.7±0.0 | - | - | 0.385±0.019 |
| Deep Learning | MLP [198] | 73.2±0.3 | 80.3±0.1 | 84.8±0.1 | 90.3±0.2 | 77.1±0.0 | 85.6±0.0 | 91.0±0.4 | 76.1±3.0 | 0.263±0.008 |
| | VIME [79] | 72.7±0.0 | 79.2±0.0 | 84.8±0.2 | 90.5±0.2 | 76.9±0.2 | 85.5±0.1 | 90.9±0.1 | 82.9±0.7 | 0.275±0.007 |
| | DeepFM [15] | 73.6±0.2 | 80.4±0.1 | 86.1±0.2 | 91.7±0.1 | 76.9±0.0 | 83.4±0.0 | - | - | 0.260±0.006 |
| | DeepGBM [62] | 78.0±0.4 | 84.1±0.1 | 84.6±0.3 | 90.8±0.1 | 74.5±0.0 | 83.0±0.0 | - | - | 0.856±0.065 |
| | NODE [7] | 79.8±0.2 | 87.5±0.2 | 85.6±0.3 | 91.1±0.2 | 76.9±0.1 | 85.4±0.1 | 89.9±0.1 | 98.7±0.0 | 0.276±0.005 |
| | NAM [85] | 73.3±0.1 | 80.7±0.3 | 83.4±0.1 | 86.6±0.1 | 53.9±0.6 | 55.0±1.2 | - | - | 0.725±0.022 |
| | Net-DNF [50] | 82.6±0.4 | 91.5±0.2 | 85.7±0.2 | 91.3±0.1 | 76.6±0.1 | 85.1±0.1 | 94.2±0.1 | 99.1±0.0 | - |
| | TabNet [6] | 81.0±0.1 | 90.0±0.1 | 85.4±0.2 | 91.1±0.1 | 76.5±1.3 | 84.9±1.4 | 93.1±0.2 | 99.4±0.0 | 0.346±0.007 |
| | TabTransformer [90] | 73.3±0.1 | 80.1±0.2 | 85.2±0.2 | 90.6±0.2 | 73.8±0.0 | 81.9±0.0 | 76.5±0.3 | 72.9±2.3 | 0.451±0.014 |
| | SAINT [9] | 82.1±0.3 | 90.7±0.2 | 86.1±0.3 | 91.6±0.2 | 79.8±0.0 | 88.3±0.0 | 96.3±0.1 | <u>99.8±0.0</u> | 0.226±0.004 |
| | RLN [63] | 73.2±0.4 | 80.1±0.4 | 81.0±1.6 | 75.9±8.2 | 71.8±0.2 | 79.4±0.2 | 77.2±1.5 | 92.0±0.9 | 0.348±0.013 |
| | STG [93] | 73.1±0.1 | 80.0±0.1 | 85.4±0.1 | 90.9±0.1 | 73.9±0.1 | 81.9±0.1 | 81.8±0.3 | 96.2±0.0 | 0.285±0.006 |

¡Gracias!



Dr. Manuel Castillo-Cara
www.manuelcastillo.eu

Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)