METADATA MANIPULATION APPLIED TO SEMANTIC SEGMENTATION OVER IMAGERY STREAM

(Thesis Seminar II)

Author: Luis Vasquez

Advisor: PhD. Manuel Castillo Cara

December, 2019

Opening concepts

- Evolution in *Artificial Intelligence*
- Processing tools progress
- Fully Convolutional Networks
- Semantic segmentation

Introduction

- Motivation
  - Learning from context
  - Potential in information industry
- Objectives
  - Research the functionality of Deep Learning architectures.
  - Implement a convolutional neural network of U-Net architecture able to handle the needed semantic segmentation process.
  - Improve the expected results applying a metadata preprocessing over the training data.
  - Show the results, contrasting them with the expected.
  - Evaluate the performance of the trained model on systems with low specifications.

STATE OF ART

State of art - Biological analogy

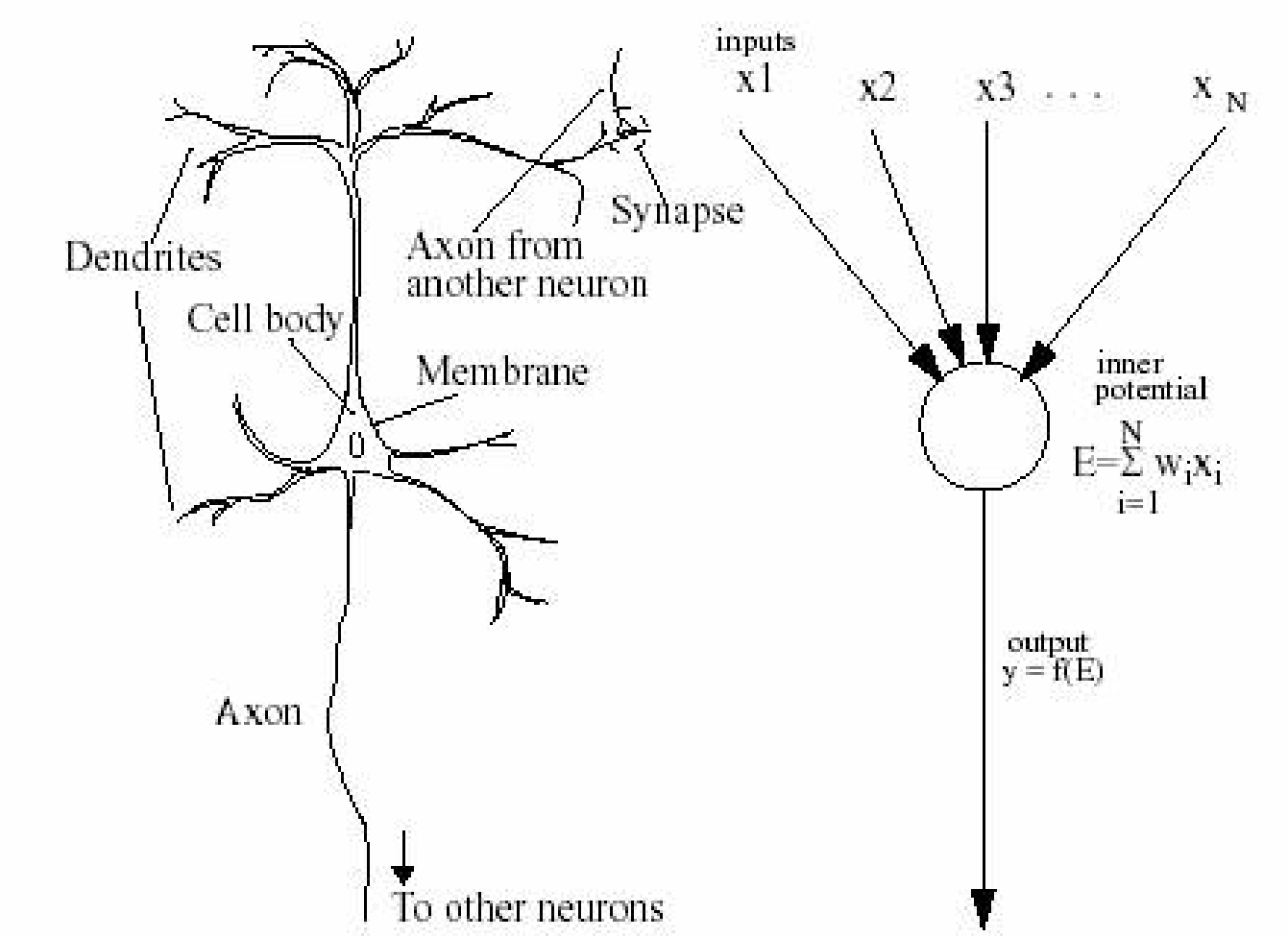

Figure: Comparative scheme between biological and artificial neuron. Source: http://www.hemming.se

State of art - Activation functions

An activation function *f* determines whether the value entered as an argument is sufficient to consider that the neuron has a considerable *participation* in the system, translating this idea to the evaluation of the value based on a limit (*threshold*).

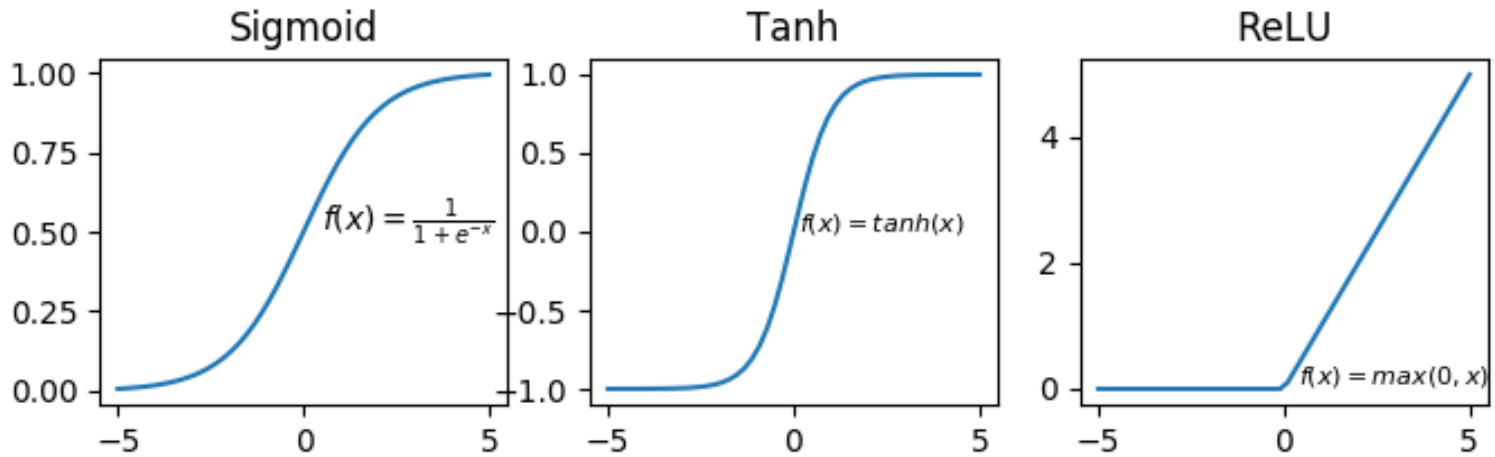

Figure: Activation function. Source: Original elaboration

State of art - Artificial Neural Network

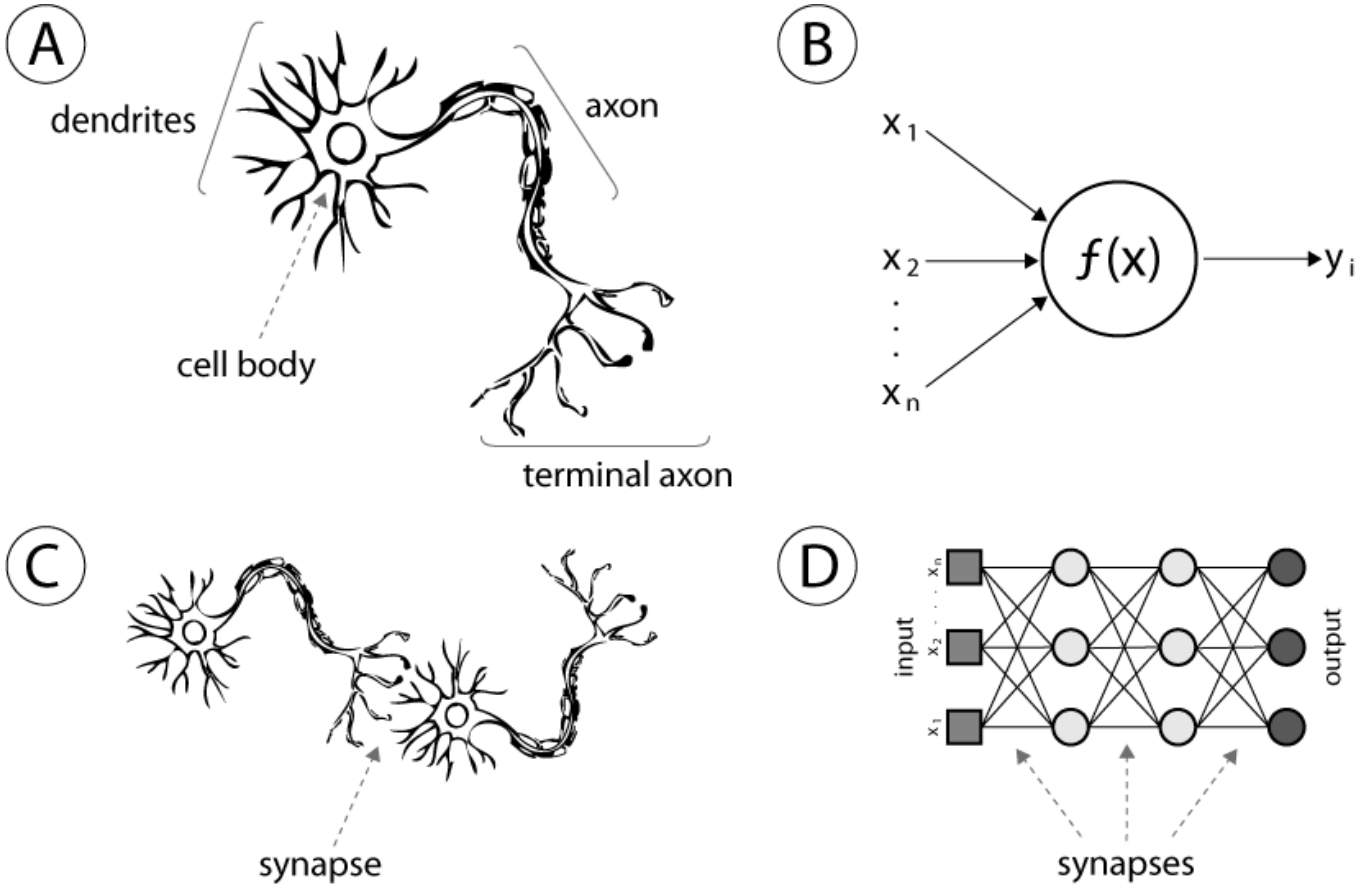

Figure: Comparative scheme between biological and artificial neural structures. Source: https://medium.com

Convolution

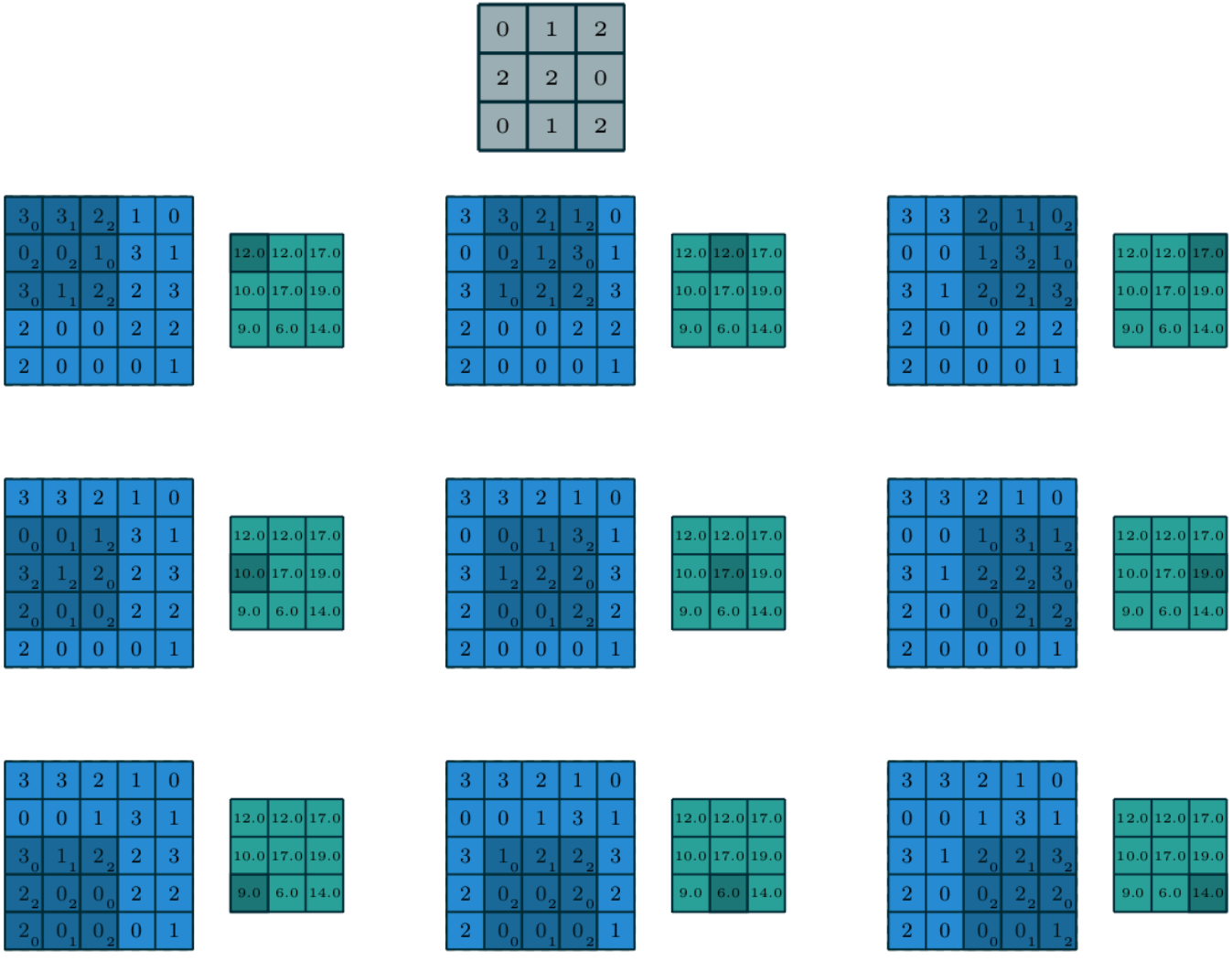

Figure: Convolution process examples. Source: Dumoulin, Visin - *A guide to convolution arithmetic for deep learning* [1]

Convolutional Neural Network

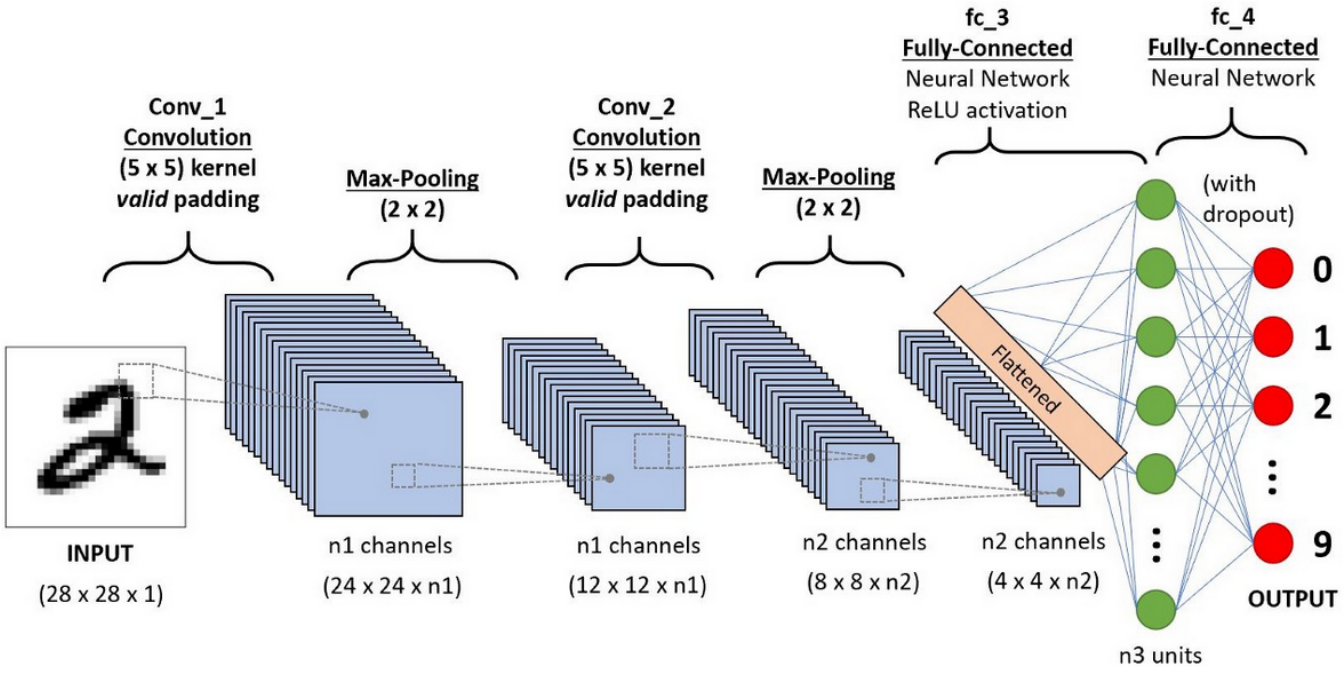

Figure: Convolutional Neural Network example. Source: https://towardsdatascience.com/

- Usage of contextual information
- Signal processing without information loss

Fully Convolutional Networks (FCN)

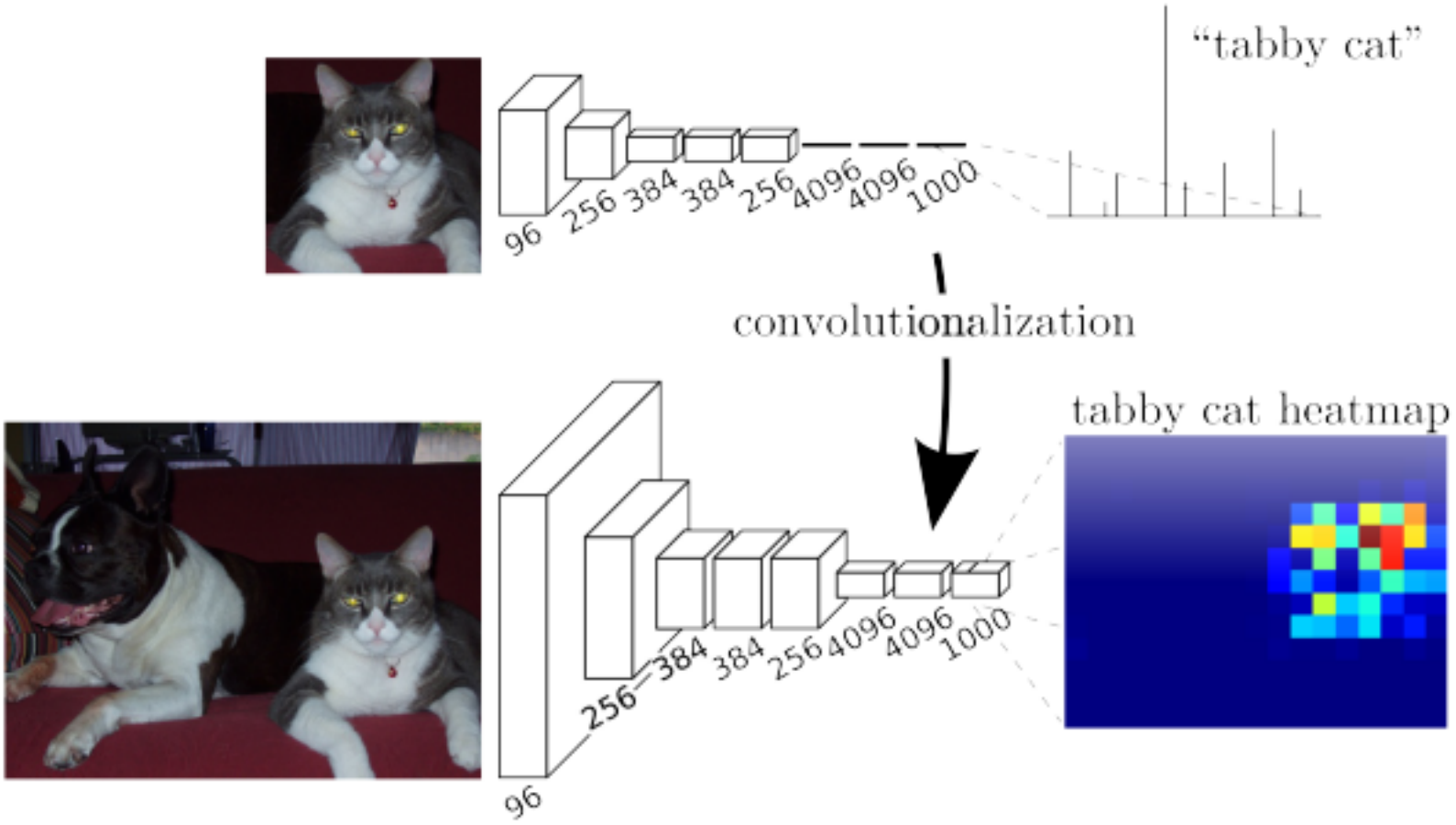

Figure: Comparative analogy between CNN and FCN. Source: Long et al - *Fully convolutional networks for semantic segmentation* [2]

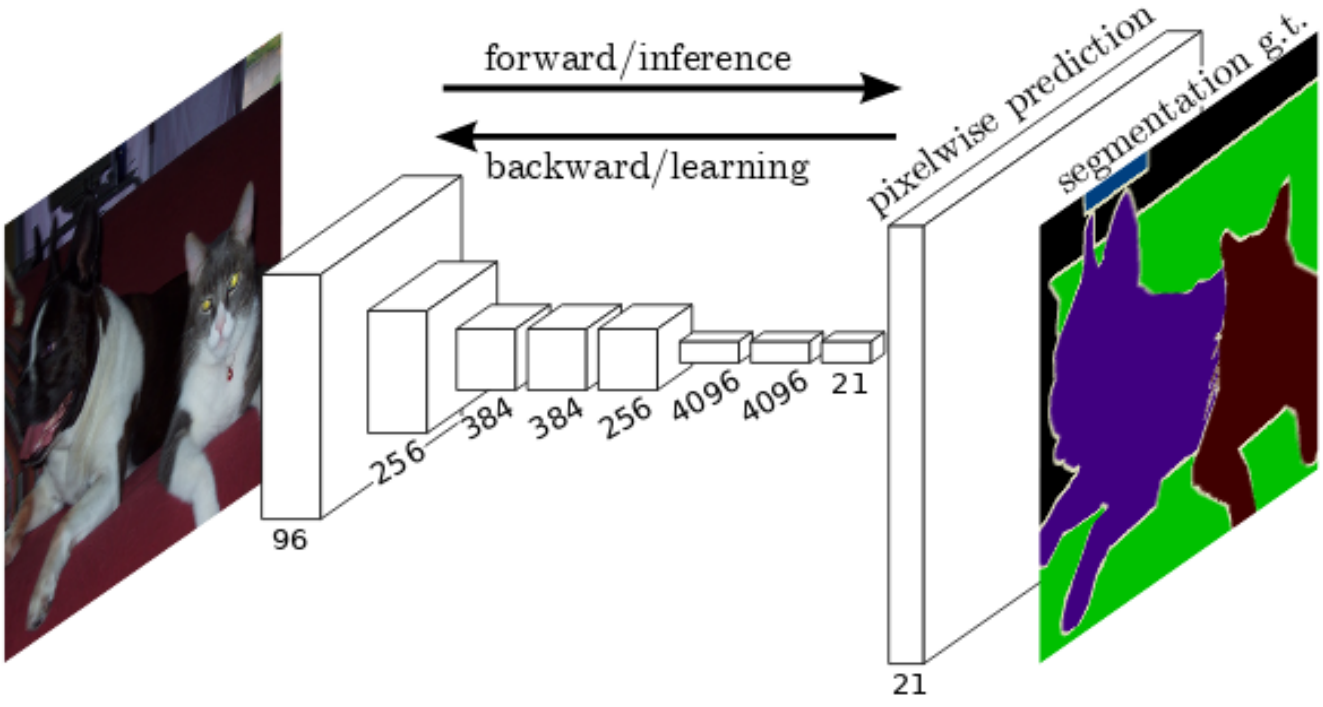

Figure: Semantic segmentation example. Source: Long et al - *Fully convolutional networks for semantic segmentation* [2]

U-Net: FCN for Image Segmentation

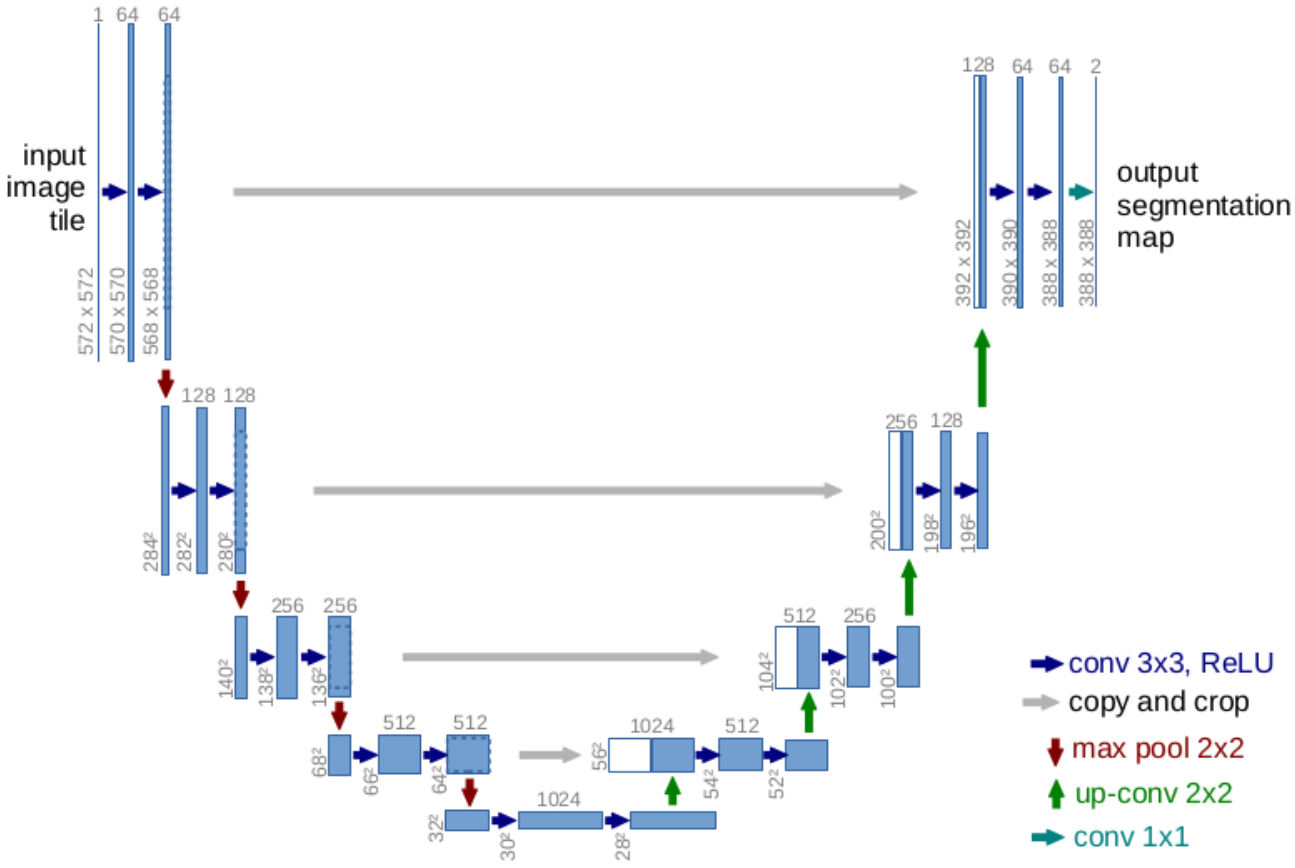

Figure: U-Net architecture. Fuente: O. Ronneberger, P.Fischer y T. Brox. - *U-Net: Convolutional Networks for Biomedical Image Segmentation* [3]

Binary Cross Entropy:

$$BCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{1} y_j log(\hat{y}_j) + (1 - y_j) log(1 - \hat{y}_j) \qquad (1)$$

Where:

- N: Number of data entries on training dataset
- y_j: Expected values
- $\hat{y}_j$: Predicted value

Dice:

$$Dice = \frac{2h}{a+b} \tag{2}$$

Where:

- h: Cardinal intersection of the sets.
- a: Cardinal of the first set.
- b: Cardinal of the second set.

RESOURCES

Resources (1)

- Hardware
  - Lenovo ideapad 100-15IBD
  - Lenovo ideapad 510S-14ISK
  - UDOO x86
- Software
  - Python
  - PyTorch
  - OpenCV

Dataset: LSUN

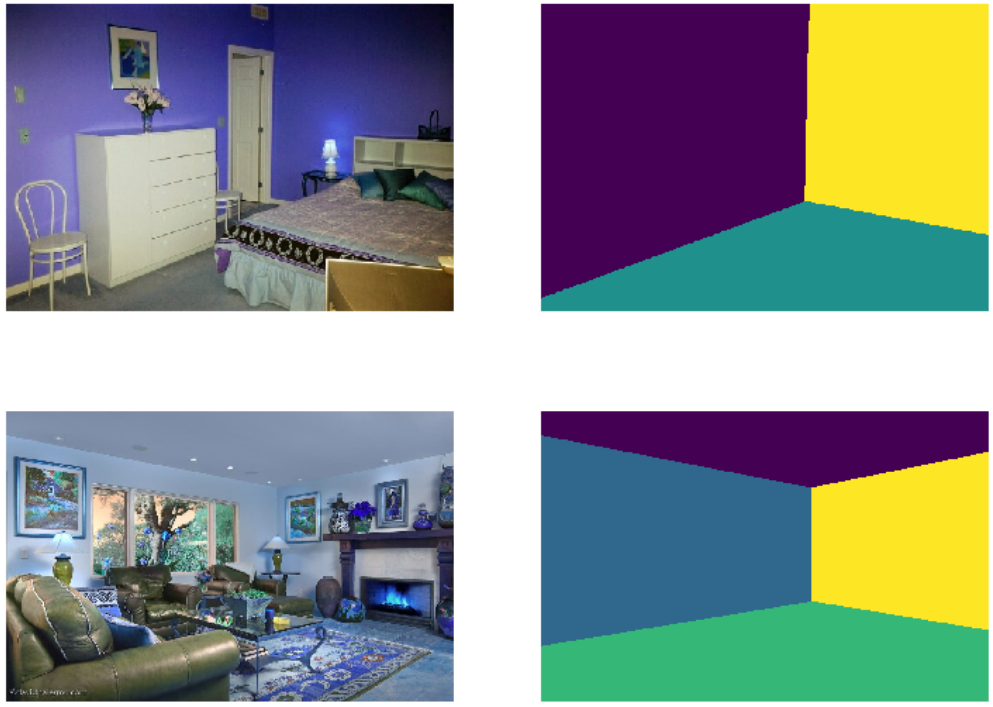

Figure: Pair of data components of LSUN Room Layout Estimation.
Source: Original elaboration

STRUCTURING AND METHOD

Structuring and method: Introduction

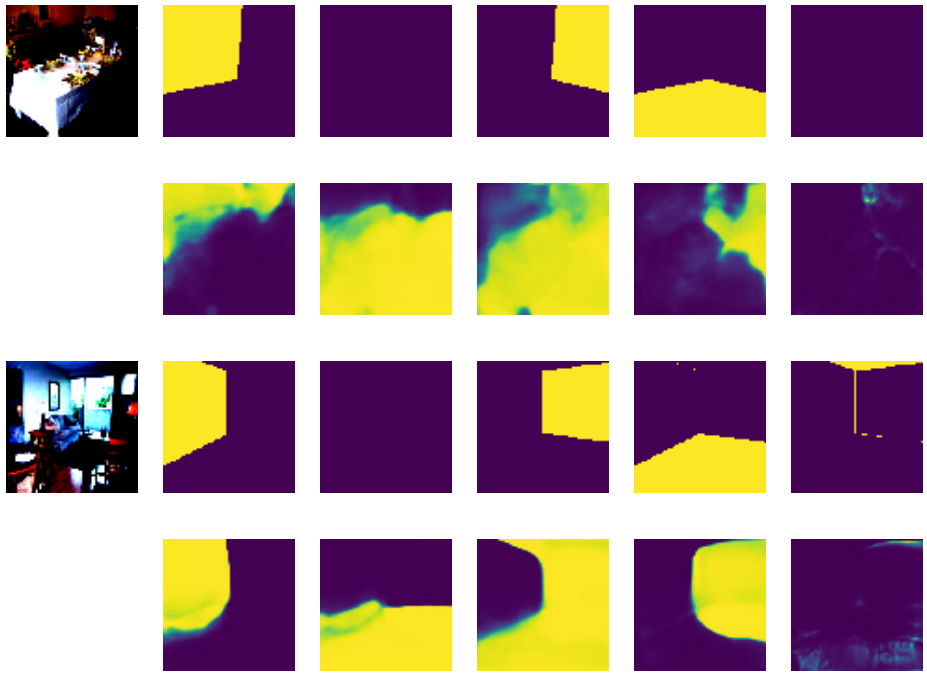

Figure: Example results for the non-standarized model

- Messy training and lack of homogeneity

Structuring and method: Preprocessing (1)

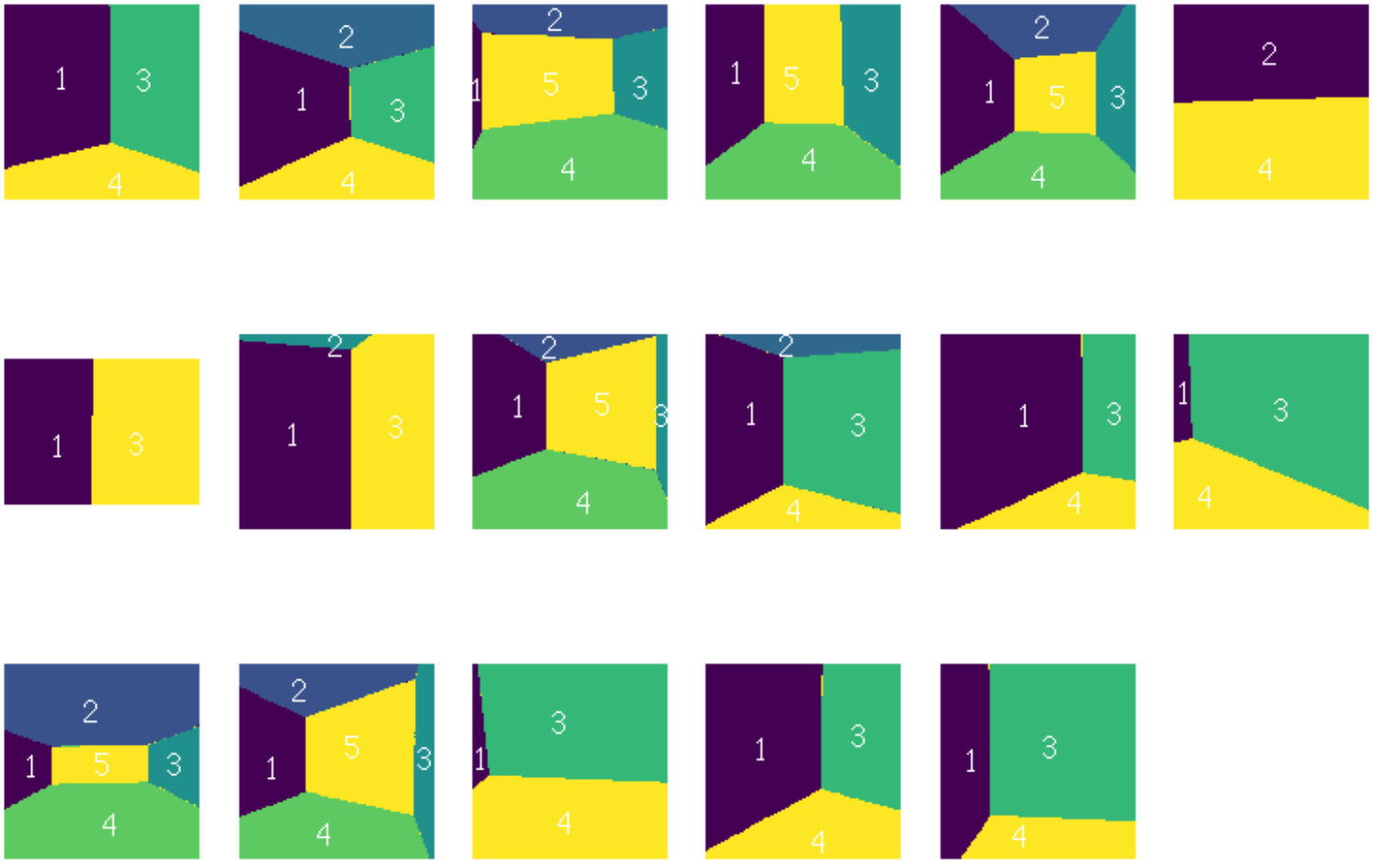

Figure: Mask standardization. Source: Original elaboration

Preprocessing (2)

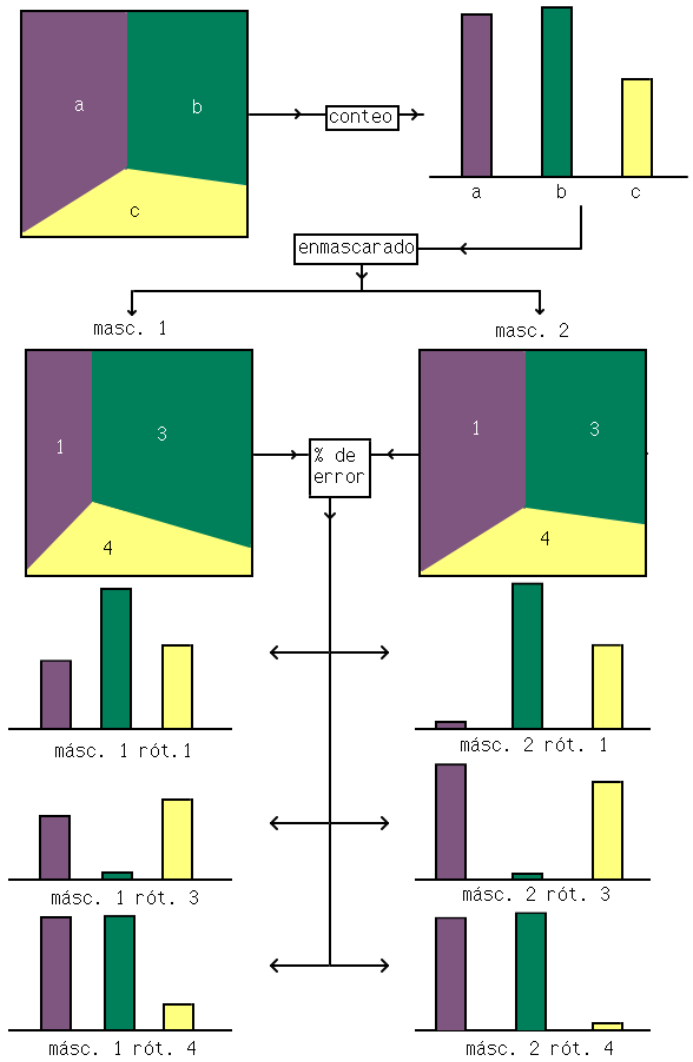

Figure: Masking process for 2 masks. Source: Original elaboration

Preprocessing (3)

$T \leftarrow Target, M \leftarrow Mascara$;
$Regla \leftarrow \{\}$ `// diccionario vacío`
for *rótulo* $r_T \in T$ **do**
    $r_{min} \leftarrow r_M \in M / error(r_M, r_T)$ sea mínimo;
    $error_{min} \leftarrow error(r_{min}, r_T)$;
    $Regla$.añadir($\{r_{min} : (r_T, error_{min})\}$);
end
return *Regla*

Algorithm 1: Masking rule generation

$T \leftarrow Target, M \leftarrow Mascara$;
$R \leftarrow GenerarRegla(T, M)$;
$acc \leftarrow 0$;
for *item* $r \in R$ **do**
    $error \leftarrow r$.valor[1];
    $acc \leftarrow acc + error$;
end
return *acc*

Algorithm 2: Mask score calculation

Preprocessing (4)

$T \leftarrow$ Target;
$\bar{M} \leftarrow$ Lista de mascaras disponibles;
$\bar{R} \leftarrow$ *GenerarReglas*$(T, \bar{M})$;
$R \leftarrow$ *MejorRegla*$(\bar{R})$;
for *pixel* $p \in T$ **do**
    $r_T \leftarrow R$.llave[0];
    $p \leftarrow r_T$;
end
return T

Algorithm 3: Mask application on target

Preprocessing (5)

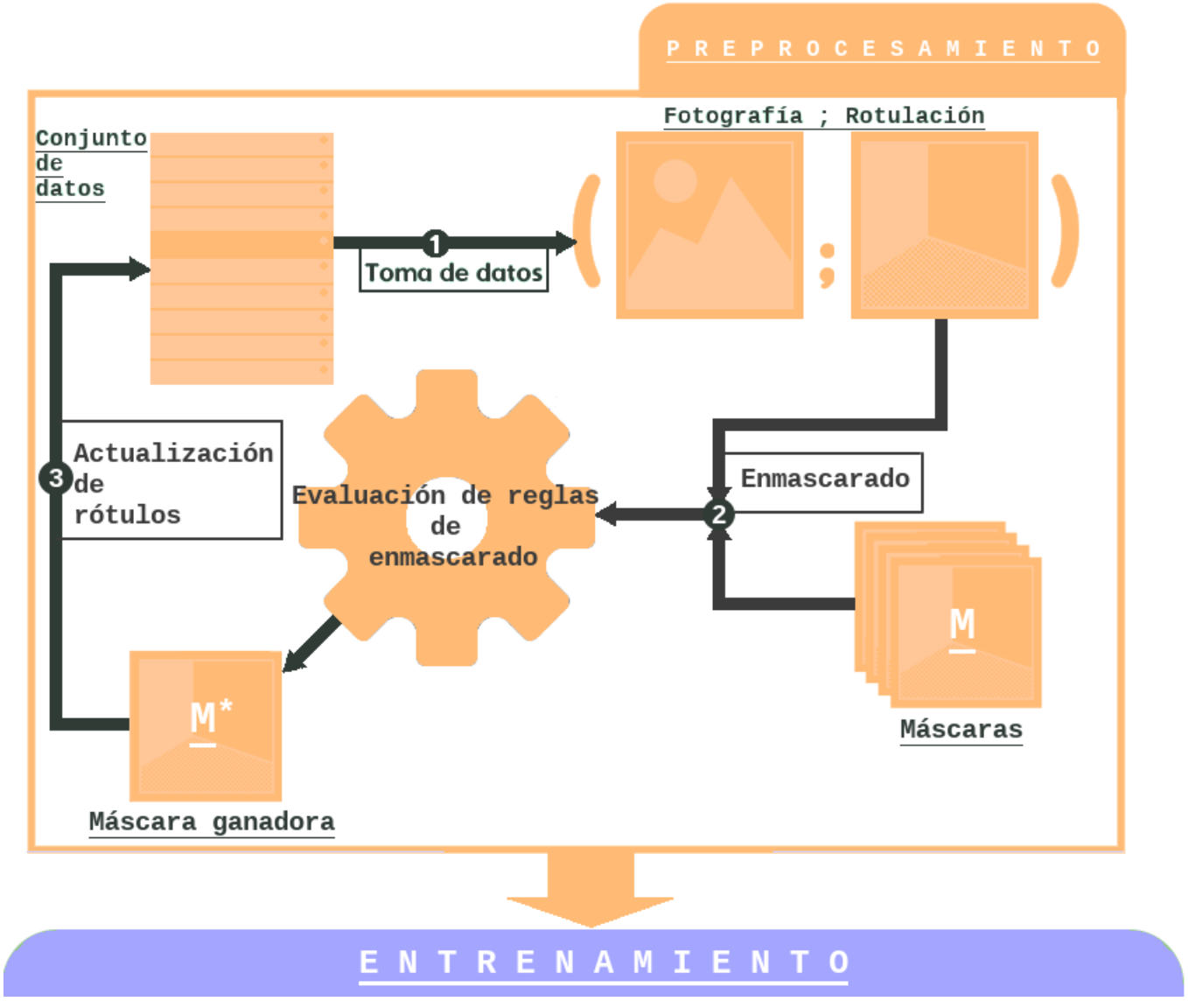

Figure: Preprocessing general scheme. Source: Original elaboration

Training (1)

Table: Training hyperparameters

Hyperparameter	Value
Epoch	50
Batch size	1
Gamma	0.1
Learning rate	0.0001
Stepsize	30

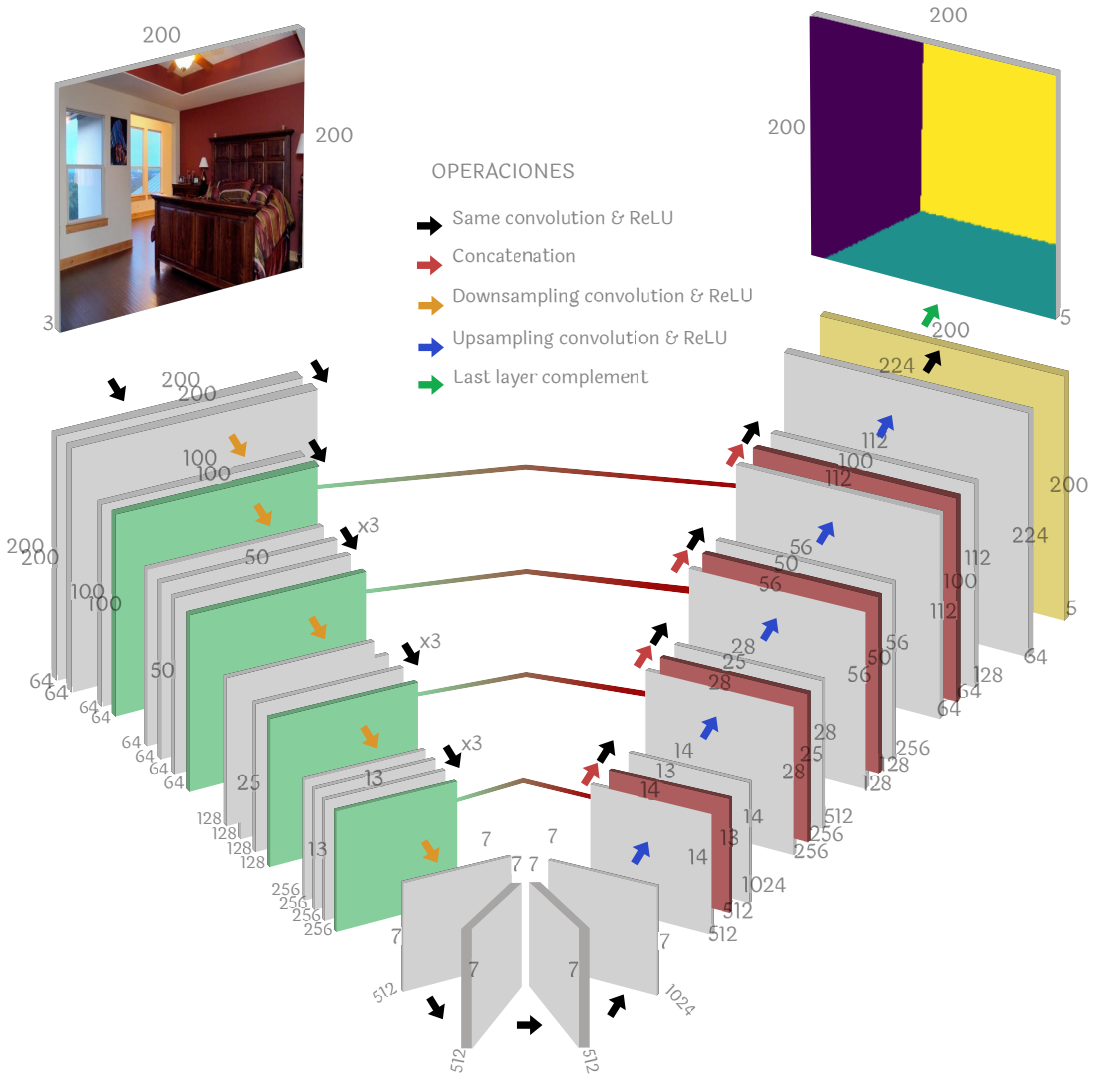

Figure: Custom U-Net architecture diagram. Source: Original elaboration

Training (2)

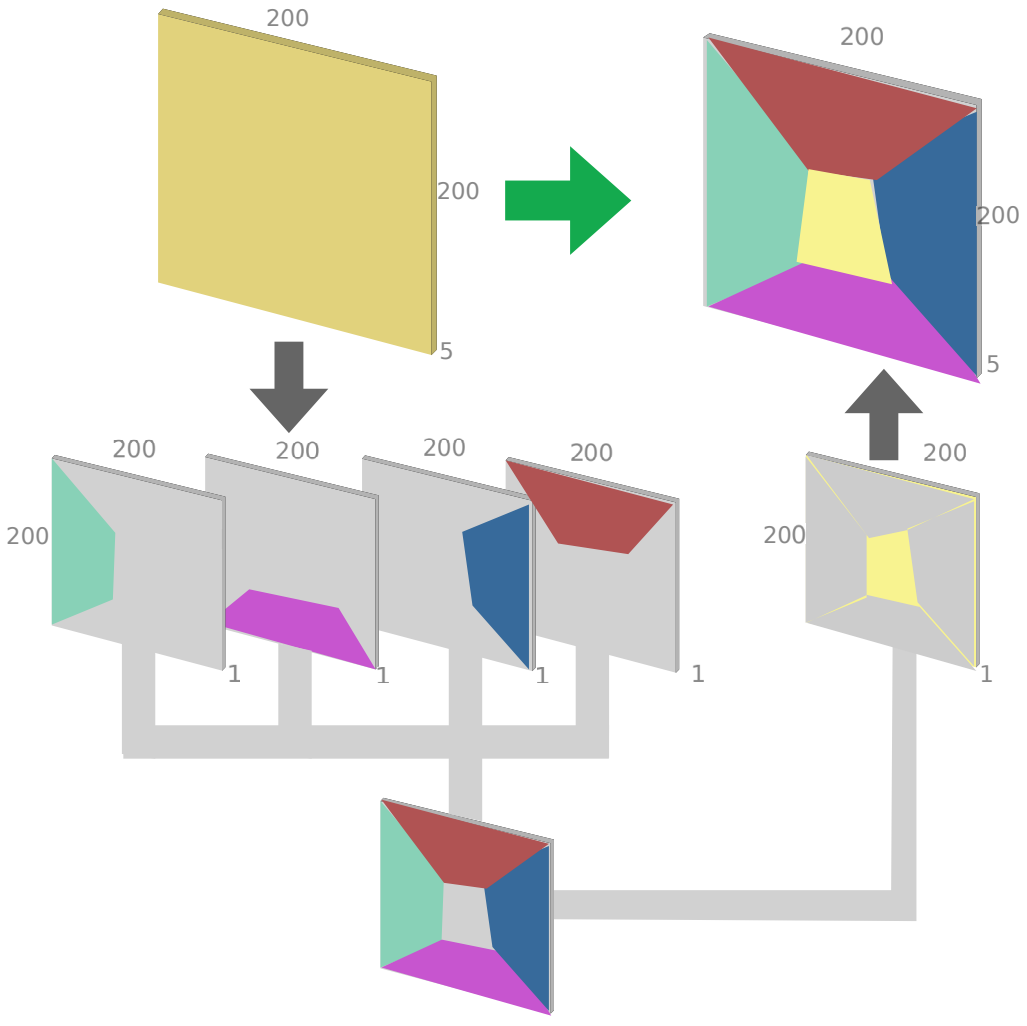

Figure: Refinement layer by channel complement. Source: Original elaboration

Training (3)

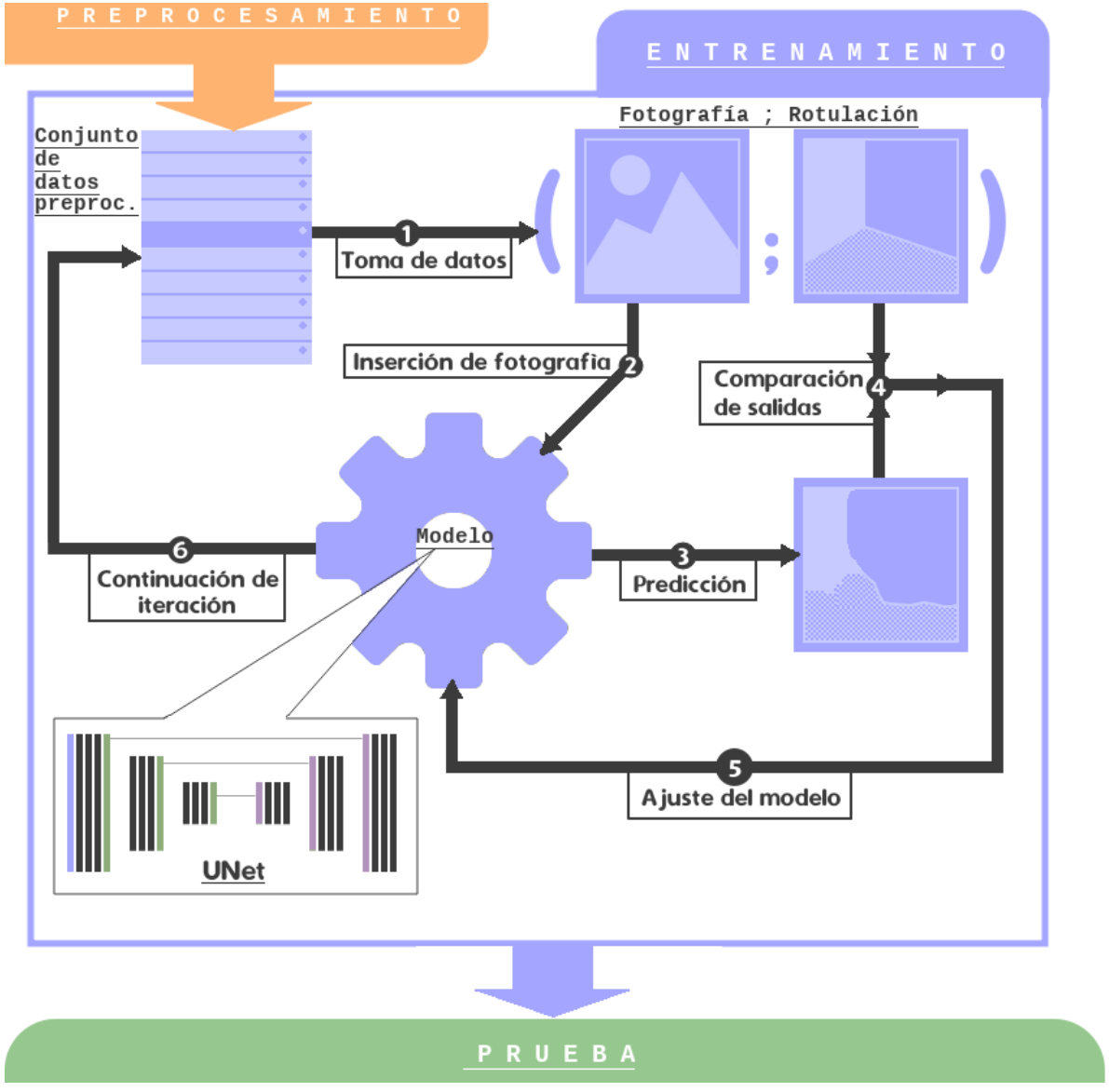

Figure: General training scheme. Source: Original elaboration

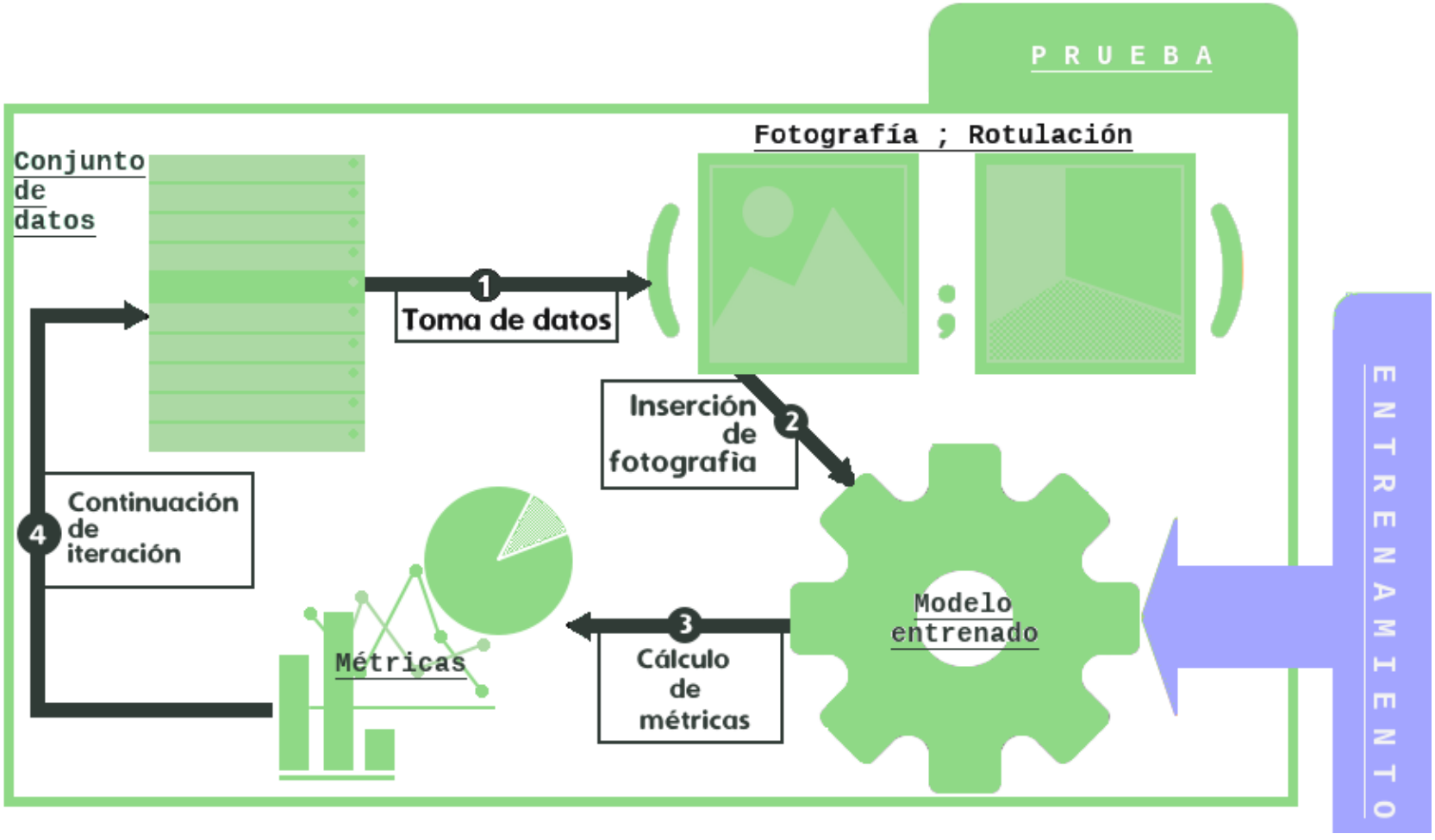

Figure: General testing. Source: Original elaboration

RESULTS

Results: Performance (1)

Table: Execution times for stress tests

	Training			Validation		
	Max. (ms.)	Min. (ms.)	Average (ms.)	Max. (ms.)	Min. (ms.)	Average (ms.)
Lenovo ideapad 100-15IBD	635.77	510.01	620.12	618.29	532.11	580.88
Lenovo ideapad 510S-14ISK	653.98	533.67	614.54	637.32	527.71	567.22
UDOO x86	4765.37	2090.24	2119.40	2339.67	2088.39	2123.66

Results: Performance (2)

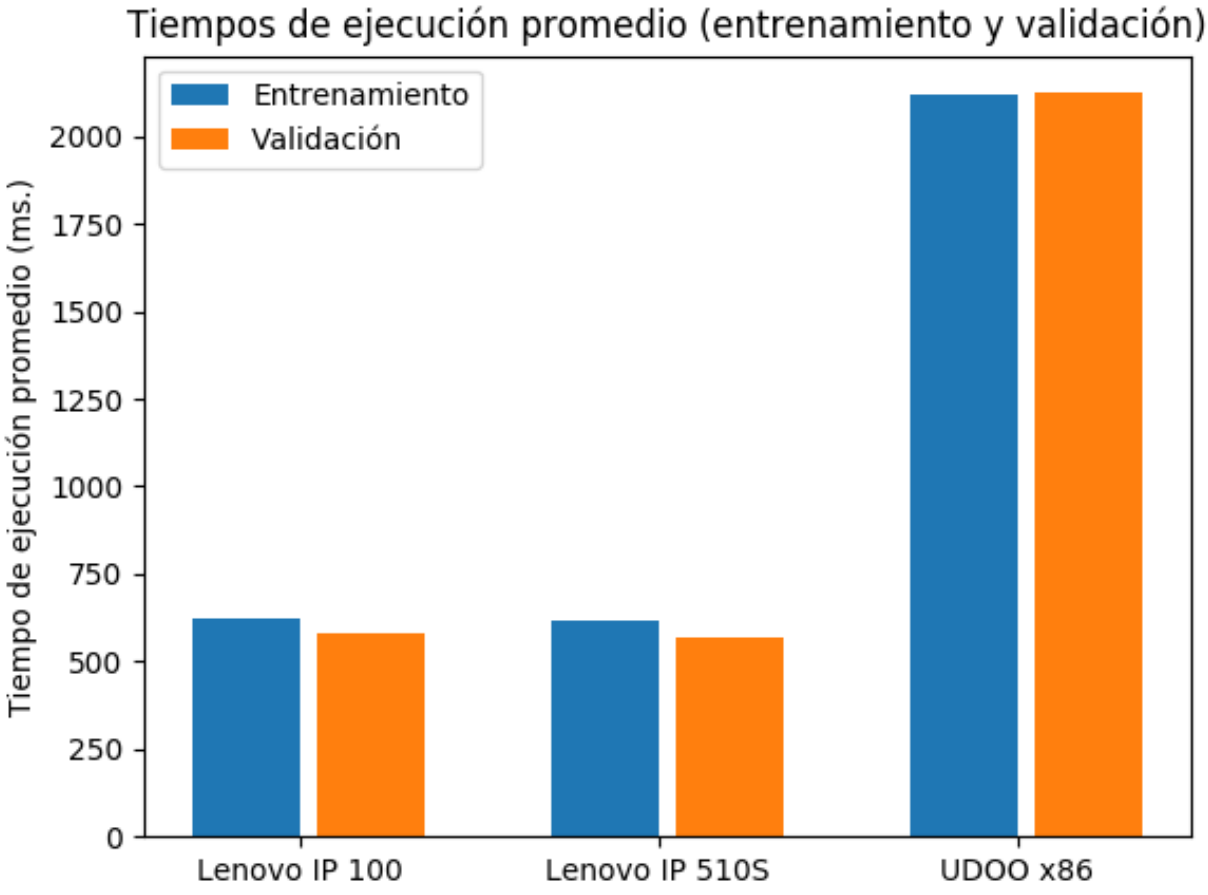

Figure: Comparative plot on average execution times. Source: Original elaboration

Results: Metrics (1.1)

Table: Learning metrics without preprocessing

	Training			Validation		
	Máximo	Mínimo	Promedio	Máximo	Mínimo	Promedio
BCE	0.13	0.0001	0.02	0.85	0.07	0.35
Dice	0.43	0.03	0.10	0.43	0.12	0.20
Loss	0.31	0.01	0.08	0.51	0.20	0.29

Table: Learning metrics applying preprocessing

	Training			Validation		
	Máximo	Mínimo	Promedio	Máximo	Mínimo	Promedio
BCE	0.07	0.0006	0.01	0.36	0.03	0.23
Dice	0.37	0.08	0.15	0.34	0.16	0.21
Loss	0.27	0.05	0.10	0.31	0.22	0.26

Results: Metrics (1.2)

Table: Learning metrics applying preprocessing and refinement

	Training			Validation		
	Máximo	Mínimo	Promedio	Máximo	Mínimo	Promedio
BCE	0.09	0.0006	0.06	0.22	0.19	0.20
Dice	0.15	0.10	0.107	0.331	0.324	0.327
Loss	0.109	0.004	0.021	0.273	0.258	0.264

Results: Metrics (2)

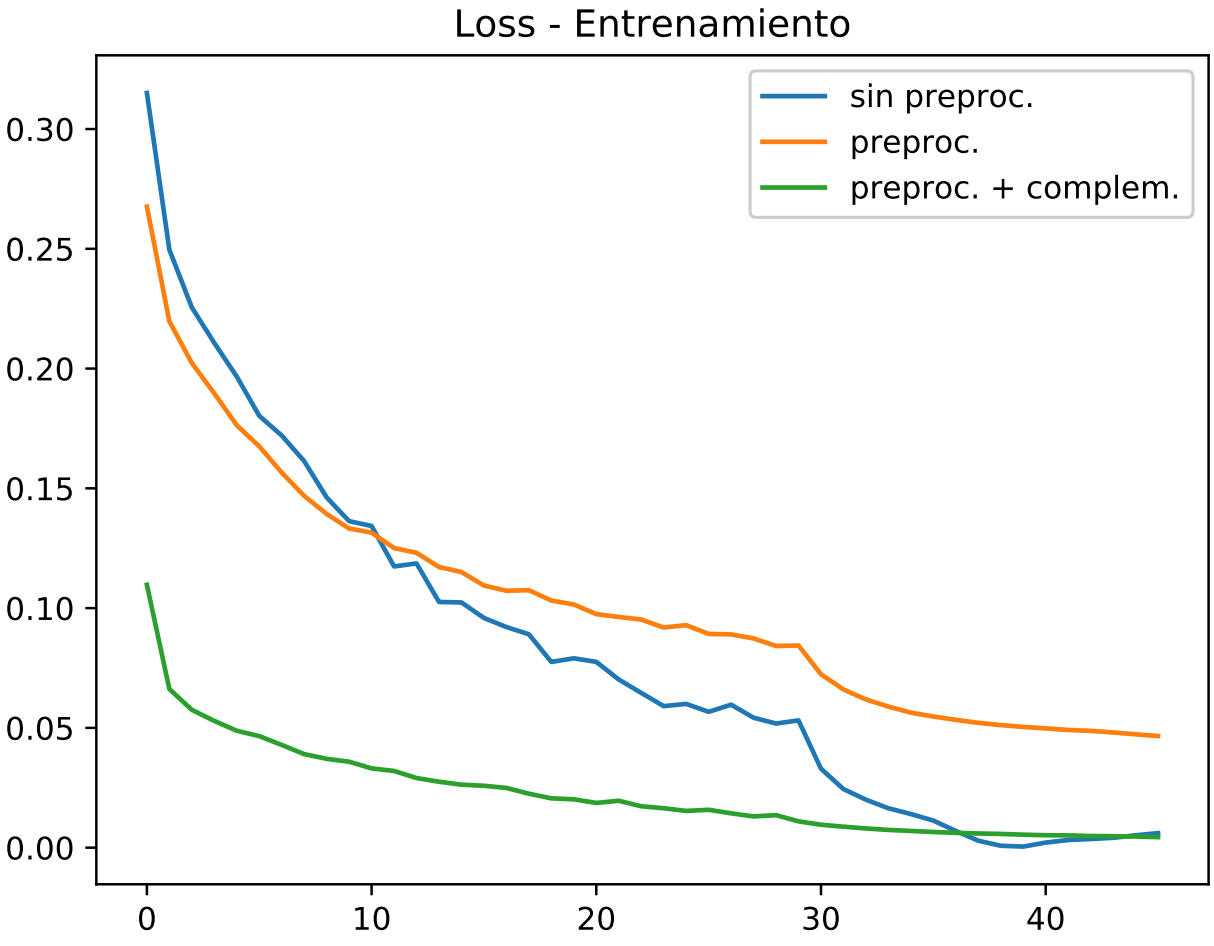

Figure: *Loss* metric behavior for training data. Source: Original elaboration

Results: Metrics (3)

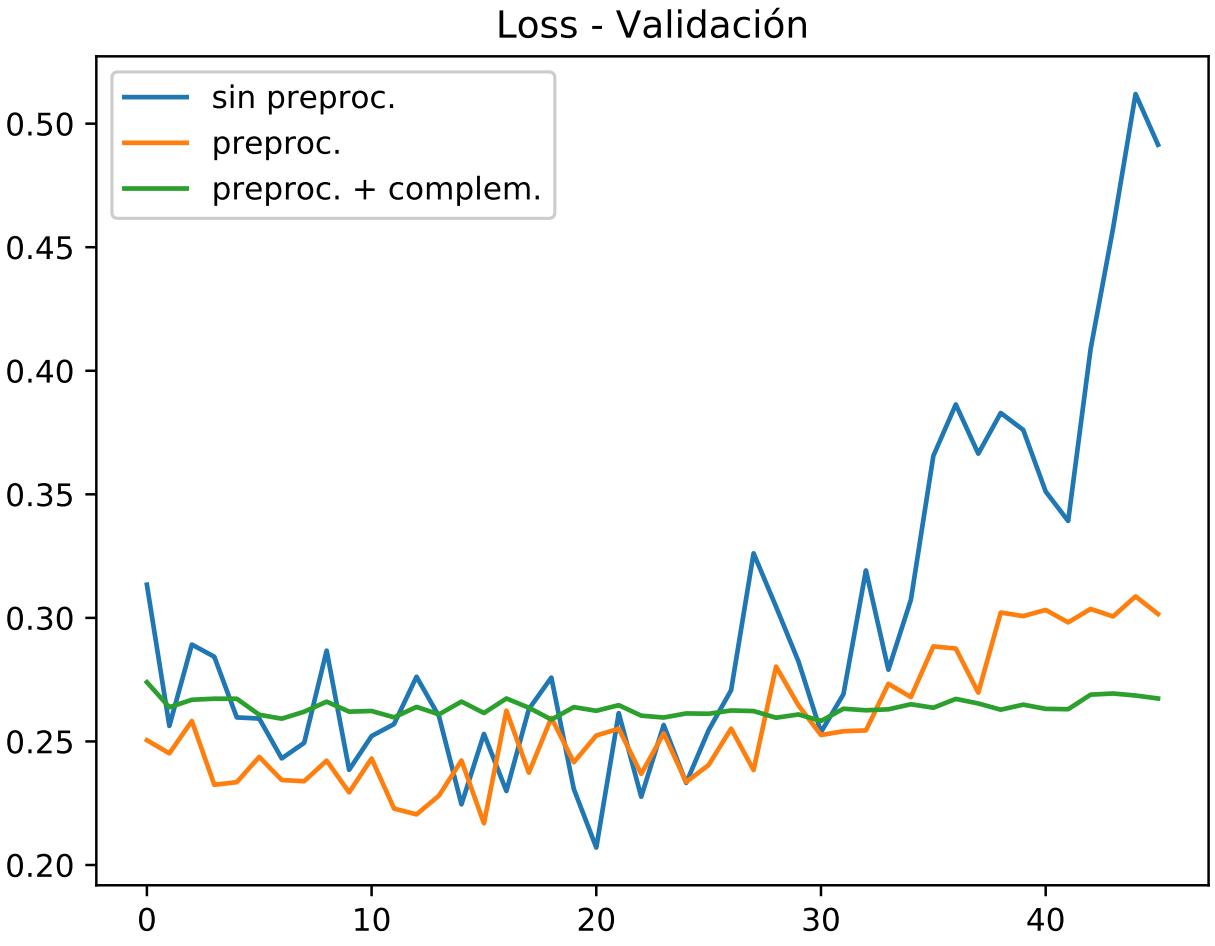

Figure: *Loss* metric behavior for validation data. Source: Original elaboration

Results: Visualization (1)

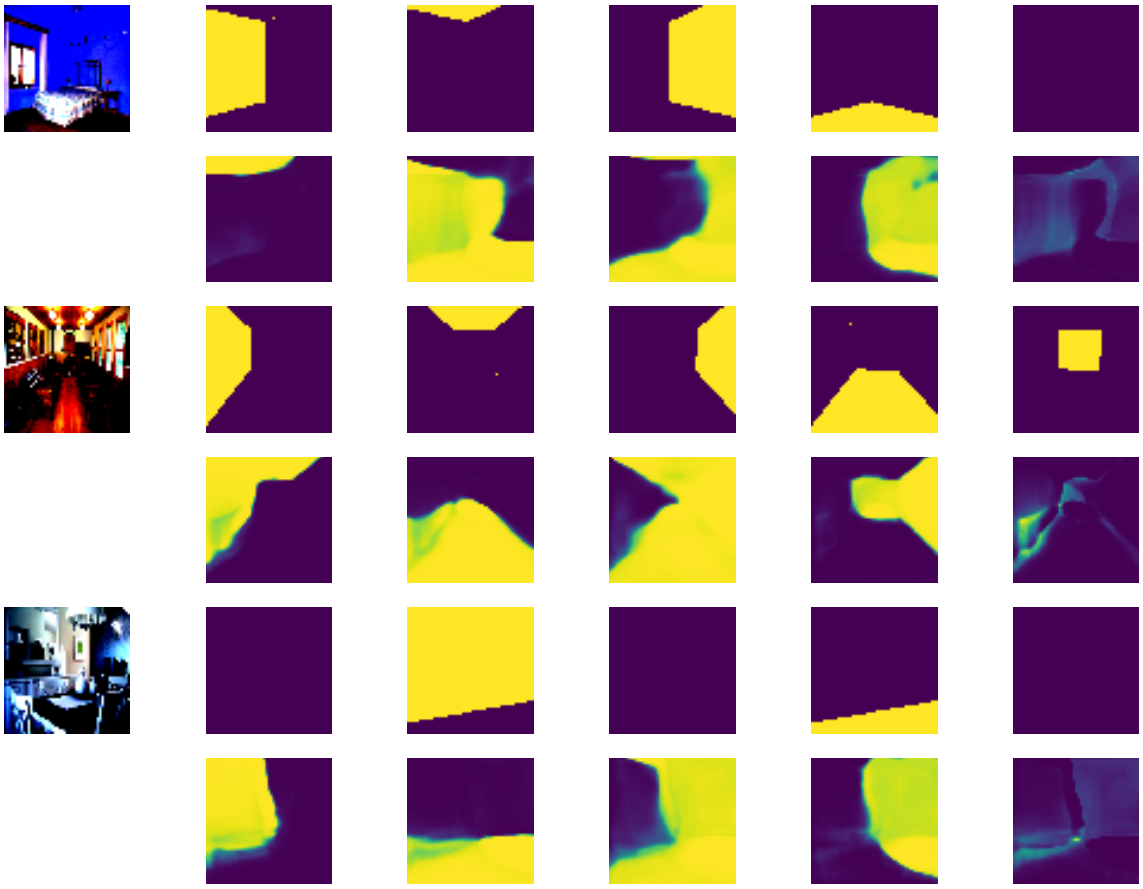

Figure: No-preprocessing test. Source: Original elaboration

Results: Visualization (2)

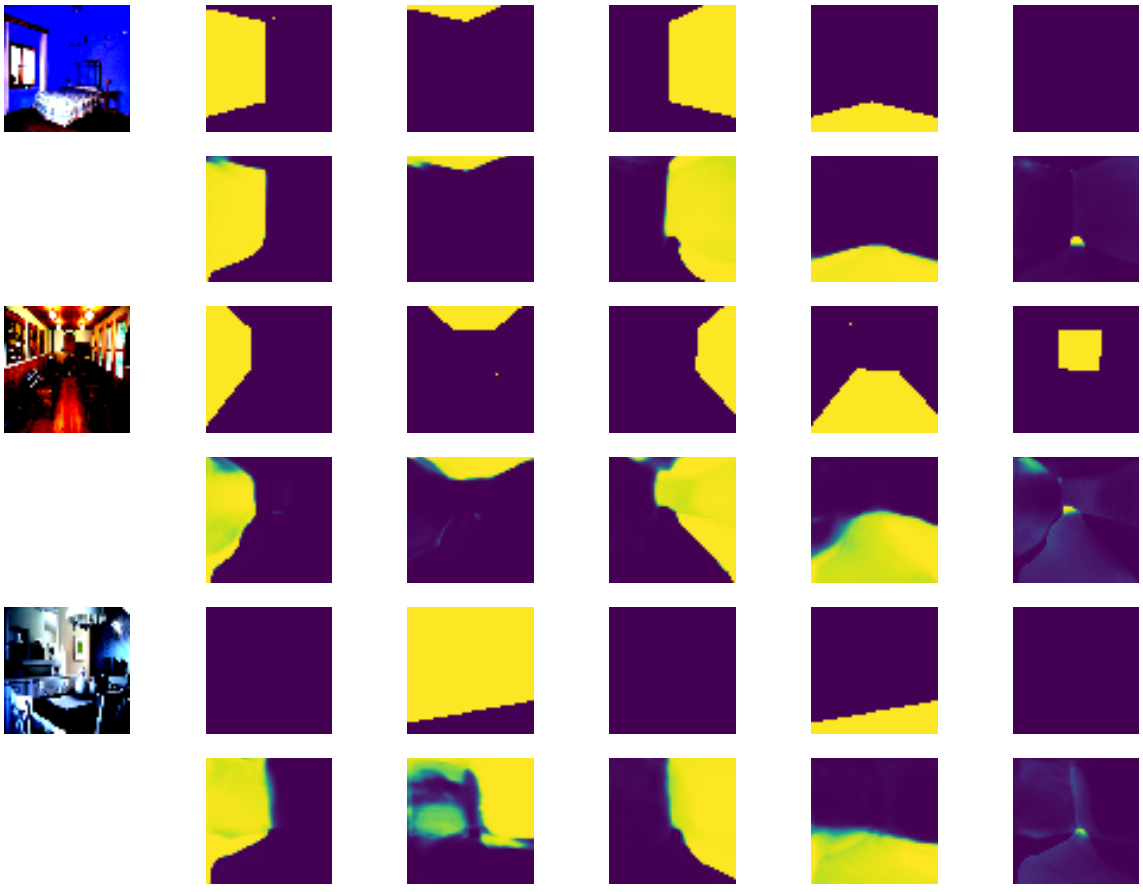

Figure: Preprocessing test. Source: Original elaboration

Results: Visualization (3)

Figure: Preprocessing and refinement test. Source: Original elaboration

CONCLUSIONS AND FUTURE WORK

Conclusions

- The neural network architecture known as U-Net, optimizes its learning process if a preprocessing of homogenization of metadata is applied to the set destined for its training.
- A neural network that classifies data into independent channels will improve its estimates if those channels uniquely order the target information.
- A training procedure oriented to deep learning about a U-Net architecture will be adaptable to any type of system, varying its depth and the size of its input data.

Future work

- Evaluate the use of the trained model under a continuous signal flow (such as video).

Bibliografía I


Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. cite arxiv:1603.07285. 2016. URL: http://arxiv.org/abs/1603.07285.


Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). DOI: 10.1109/cvpr.2015.7298965.


O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: LNCS 9351 (2015). (available on arXiv:1505.04597 [cs.CV]), pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.