Solving wood heterogeneous texture classification: A deep learning approach with cropping data augmentation

Frank A. Ricardo[1], Martxel Eizaguirre[1], Desmond Moru[2], Diego Borro[1,3]

[1]CEIT-Basque Research and Technology Alliance (BRTA) and Tecnun (University of Navarra)
[2]School of Science and Technology (SST), Pan-Atlantic University (PAU)
[3]Institute of Data Science and Artificial Intelligence (DATAI), University of Navarra

Lecturer: Diego Borro

Computer Science PhD

Vision and Robotics group at CEIT

dborro@ceit.es

Index

Index

Introduction

- The oak used to make barrels for the ageing of wines and spirits is classified according to its origin and grain type

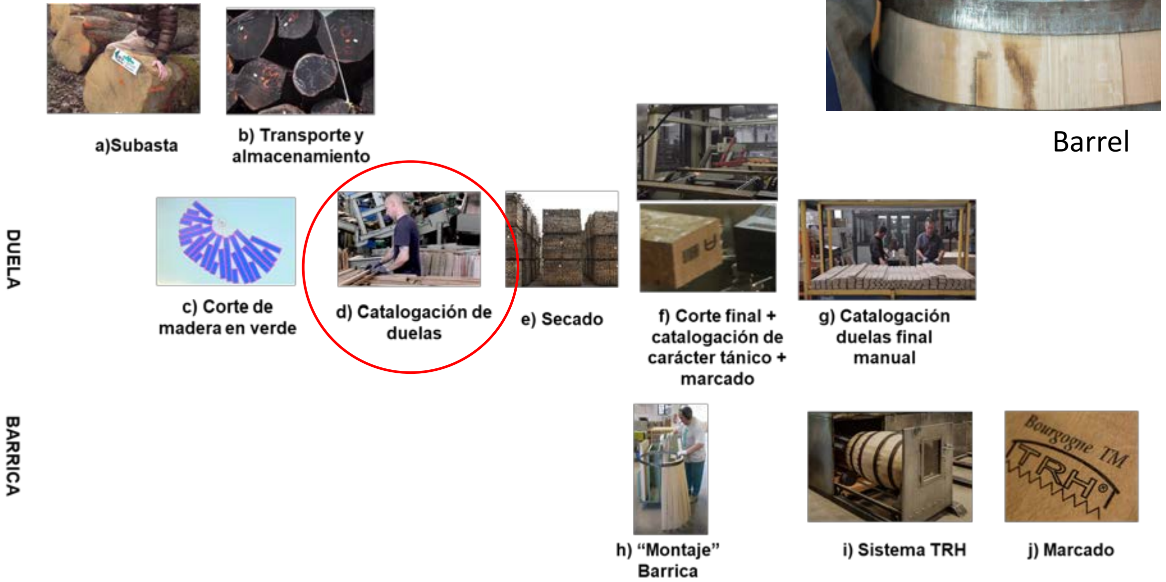
- Visual grading of wood surfaces is a problem that combines very formal grading rules, large natural variations in the target material, and highly subjective appearance criteria

- The current classification of wood is frequently done by trained human experts. This procedure is time-consuming, expensive and laborious

Wood warehouse

Staves

Barrel



Introduction

- Visual inspection is the main technique applied to determine the grain tightness of the wood and it is based on the average distance between annual growth rings
  - A growth ring is the annual increase in circumference or width of a tree from early spring to winter.
  - Older trees grow slower than younger trees, and trees grown in cooler climates have a tighter grain than trees that grown in warmer climates
  - The tightness of the grain in an oak barrel makes a huge difference to whether your wine will be delightfully aromatic or aggressively tannic. Super-fine grain releases more aromatics, while coarse grain gives more tannins

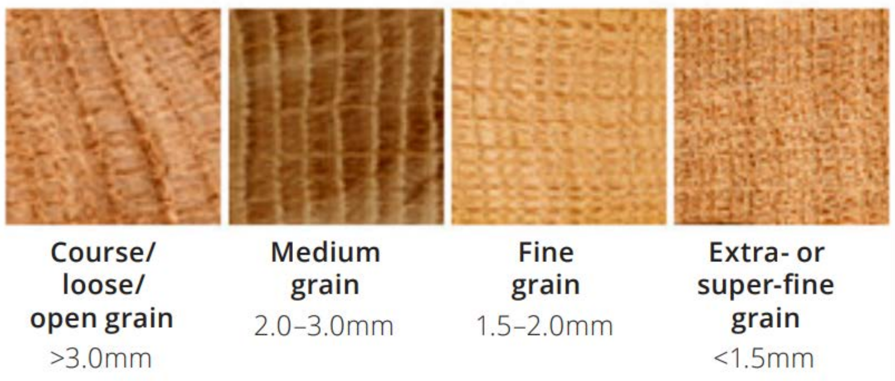


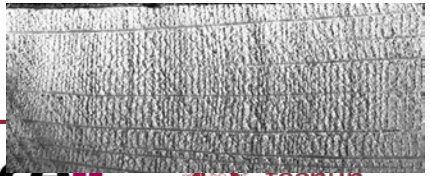
Image source: Vicard Cooperage, https://www.groupe-vicard.com/en

The classification range varies between manufacturers

Objetives

- Manual inspection does depend on field circumstances, labor perception, and performance, which are connected to factors like weariness, tension, and motivation
- It is essential to create a system that can evaluate product quality objectively and automatically

- Main goal: to automate the inspection of wood staves using computer vision-based systems
  1. Dimensional control (computation of the stave width)
     - Low tolerances (5-10mm)
  2. Classification
     - 3 categories
       - Coarse: <4 rings/cm
       - Fine: >=4 rings/cm y <7 rings/cm
       - Extrafine: >=7 rings/cm

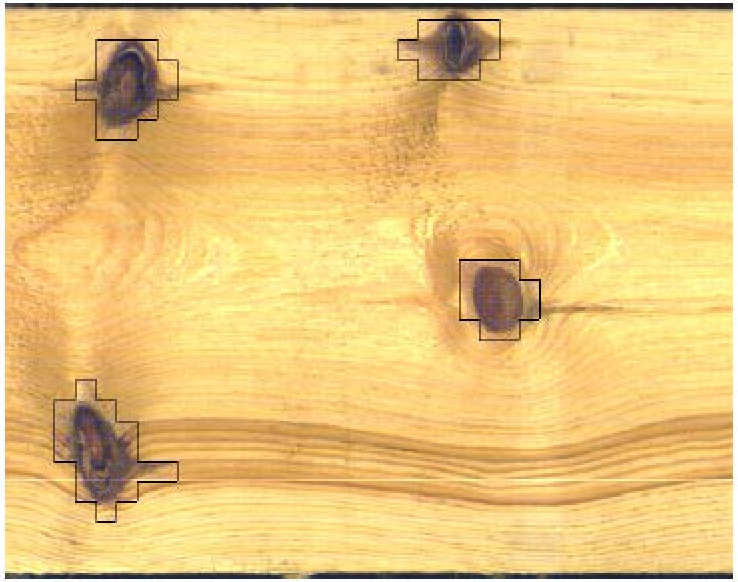


Index

Related works: defects inspection

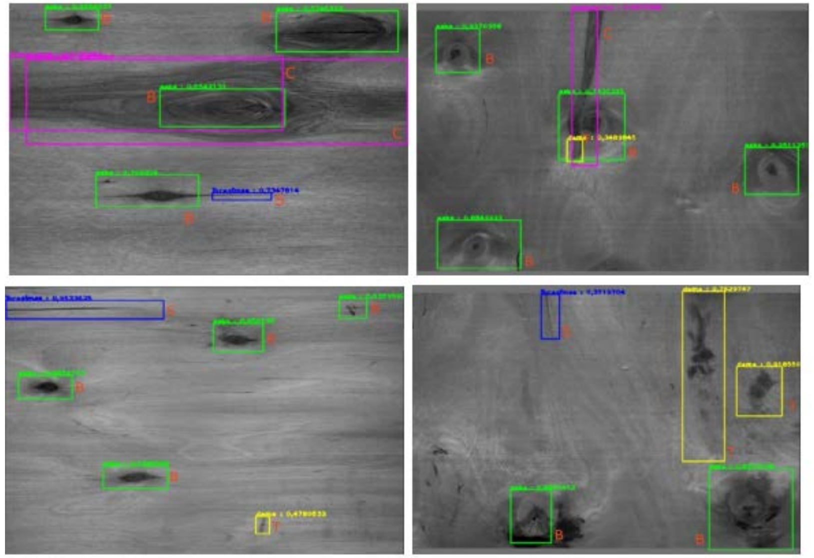
"COLOR AND TEXTURE BASED WOOD INSPECTION WITH NON-SUPERVISED CLUSTERING"[1]

- Knots and anomalies detection
- Un-supervised. Self Organized Map



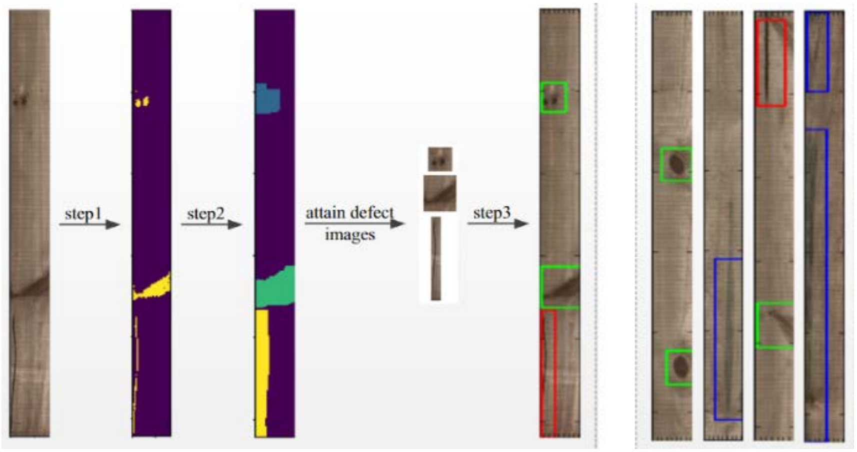
"Automated Identification of Wood Veneer Surface Defects Using Faster Region-Based Convolutional Neural Network with Data Augmentation and Transfer Learning " [2]

- Location and classification of defects
- Faster RCNN



"Application of deep convolutional neural network on feature extraction and detection of wood defects"[3]

- Location and classification of defects
- Original DCNN arquitecture

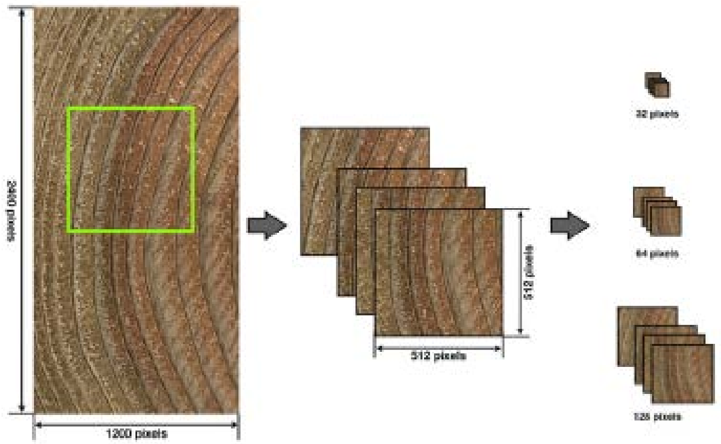


[1]Niskanen, M., Silvén, O., & Kauppinen, H. (2001, June). Color and texture based wood inspection with non-supervised clustering. In Proceedings of the scandinavian Conference on image analysis (pp. 336-342).
[2]Urbonas, A., Raudonis, V., Maskeliūnas, R., & Damaševičius, R. (2019). Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning. Applied Sciences, 9(22), 4898.
[3]He, T., Liu, Y., Yu, Y., Zhao, Q., & Hu, Z. (2020). Application of deep convolutional neural network on feature extraction and detection of wood defects. Measurement, 152, 107357.

Related works: classification

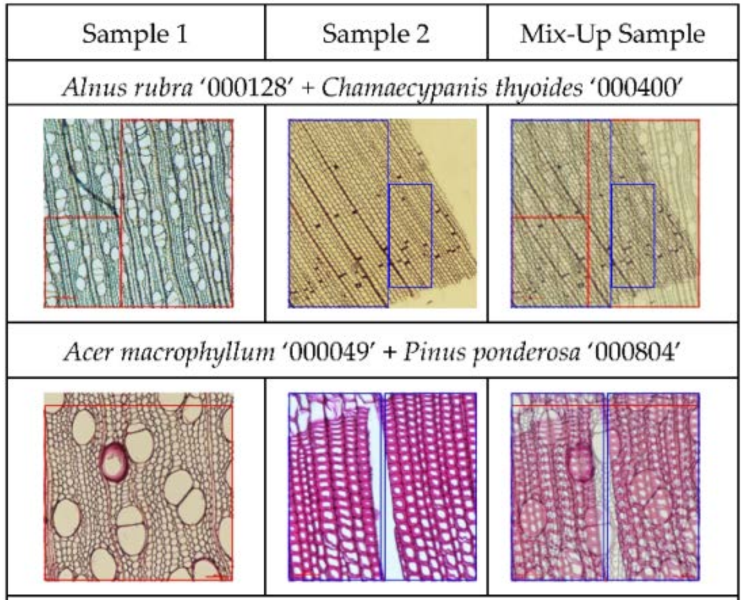
“Automatic Wood Species Identification of Korean Softwood Based on Convolutional Neural Networks”[1]

- Species classification: 5
- CNN: LeNet based architectures
- Clean and homogeneous images



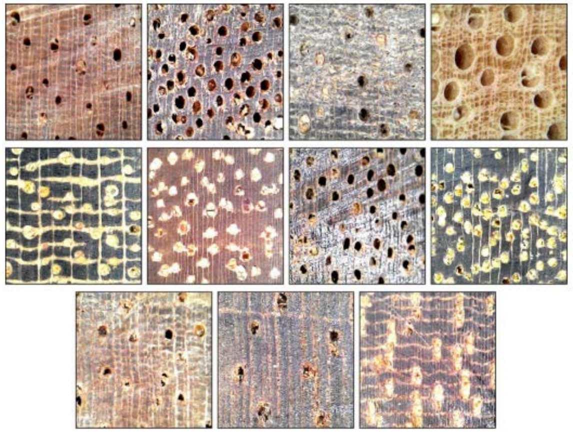
“Multi-Fusion Approach for Wood Microscopic Images Identification Based on Deep Transfer Learning”[2]

- Species classification: 10 labels
- Localization of features
- Faster RCNN with improvements
- Clean and homogeneous images



“Amazon wood species classification: a comparison between deep learning and pre-designed features”[3]

- Species classification: 11 labels
- Deep Learning models: ResNet, Inception V3, DenseNet, SqueezeNet.
- Deep Learning is better than classic algorithms (Particle Swarm Optimization)
- Clean and homogeneous images



[1] Kwon, O., Lee, H. G., Lee, M. R., Jang, S., Yang, S. Y., Park, S. Y., ... & Yeo, H. (2017). Automatic wood species identification of Korean softwood based on convolutional neural networks. *Journal of the Korean Wood Science and Technology*, *45*(6), 797-808.

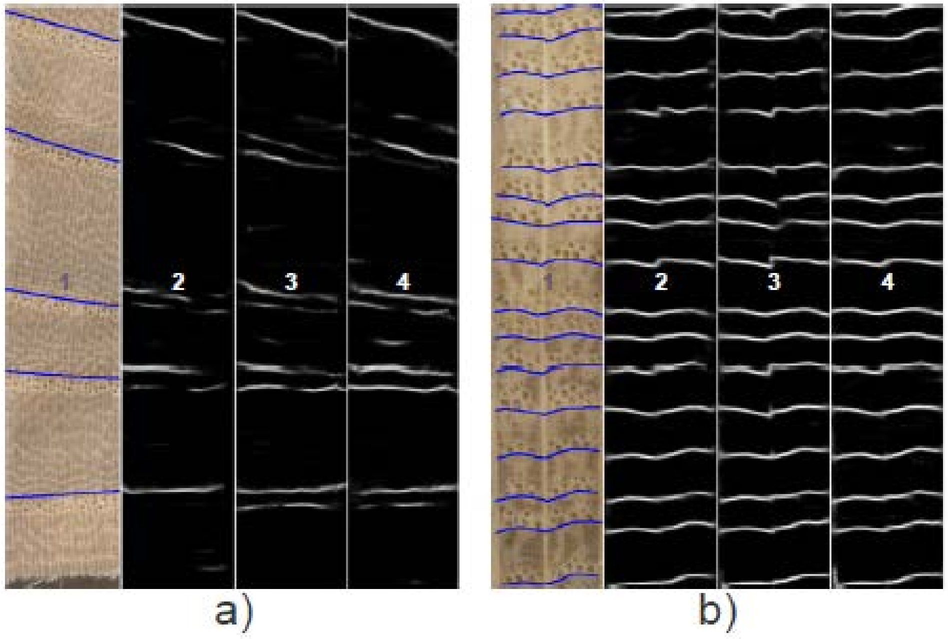
[2] Zhu, M., Wang, J., Wang, A., Ren, H., & Emam, M. (2021). Multi-fusion approach for wood microscopic images identification based on deep transfer learning. *Applied Sciences*, *11*(16), 7639.

[3] de Geus, A. R., Backes, A. R., Gontijo, A. B., Albuquerque, G. H., & Souza, J. R. (2021). Amazon wood species classification: a comparison between deep learning and pre-designed features. *Wood Science and Technology*, *55*, 857-872.

Related works: segmentation

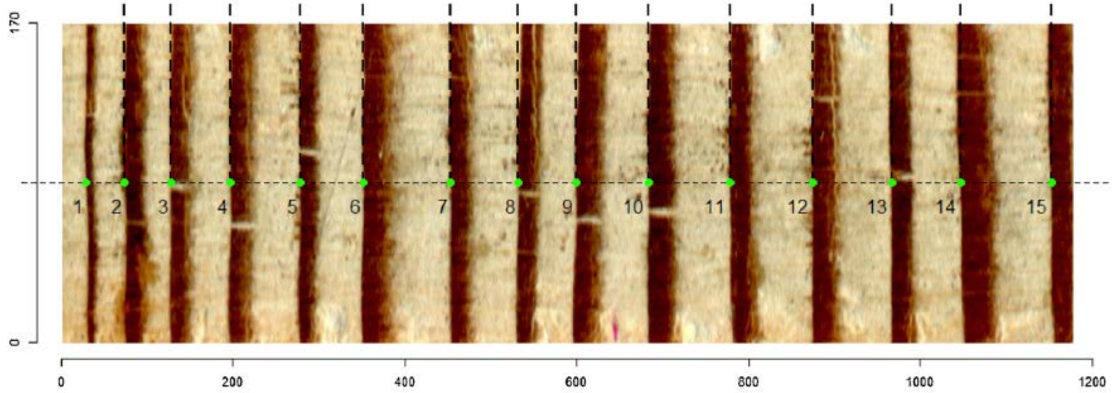
“DeepDendro – A tree rings detector based on a deep convolutional neural network”[1]

- Rings segmentation. U-Net based architecture
- (First approach with CNN to rings detection)
- Clean and homogeneous images



“MtreeRing: An R package with graphical user interface for automatic measurement of tree ring widths using image processing techniques” [2]

- Morphological sequential filters for noise reduction
- Three methods for rings detection:
  - Watershed algorithm for segmentation
  - Canny Edge detector
  - Linear detection algorithm
- Clean and homogeneous images



[1]Fabijańska, A., & Danek, M. (2018). DeepDendro–A tree rings detector based on a deep convolutional neural network. Computers and electronics in agriculture, 150, 353-363.
[2]Shi, J., Xiang, W., Liu, Q., & Shah, S. (2019). MtreeRing: An R package with graphical user interface for automatic measurement of tree ring widths using image processing techniques. Dendrochronologia, 58, 125644.

Index

1. Introduction and objectives
2. Related work
3. Algorithms
    1. Dimensional control
    2. Classification
        1. Why not classical approach?
        2. DL – preprocessing, labelling, data augmentation
        3. DL – architecture, training, test
        4. DL – voting system
4. Conclusions

Optical metrology: classical approach

$$H \cdot F = \overbrace{(F \oplus H)}^{\text{Dilation}} \underbrace{\ominus \, H}_{\text{Erosion}}$$

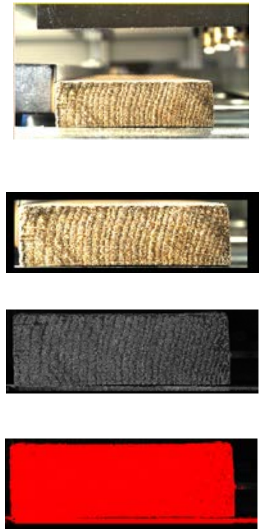
$$\begin{cases} H \oplus F = \{z | [(\hat{H})_z \cap F] \subseteq F\} \\ H \ominus F = \{z | (H)_z \subseteq F\} \end{cases}$$

The *edge strength* is given by the gradient magnitude

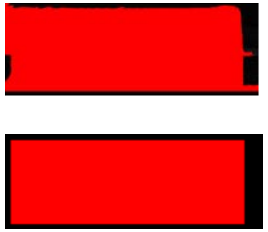
$$\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Parameters (2):
Filter, Selection

Parameters (2):
MinGray, MaxGray



Start → Adquire image → Create ROI → Pixel_on_focus → Threshold → Closing → Shape_trans → Region_features → Output anchura → Stop



Closing → Structured element: Circle
Parameters (1):
Radius

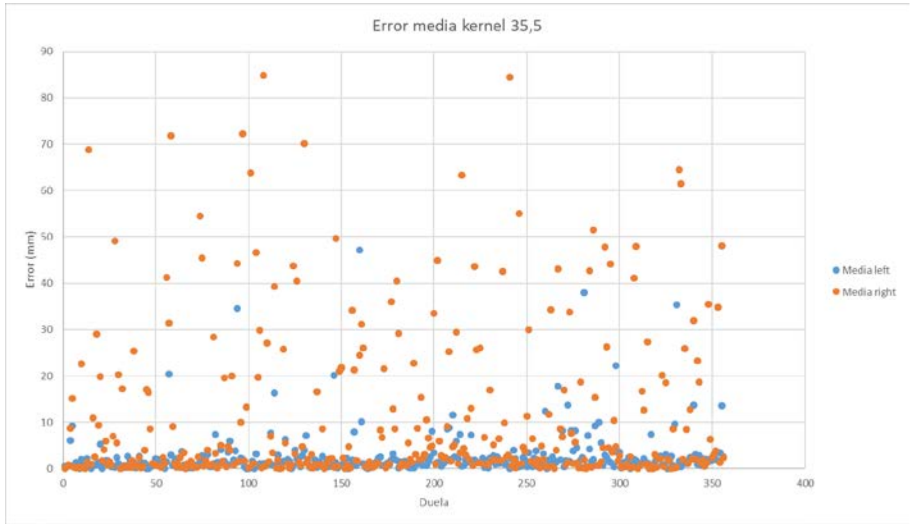
Shape_trans → 'inner_rectangle1'

Region_features → 'width'

Two problems solved:
1. Background removal
2. Joint staves problem

Each duela in a different focal plane

Optical metrology: classical approach (results)



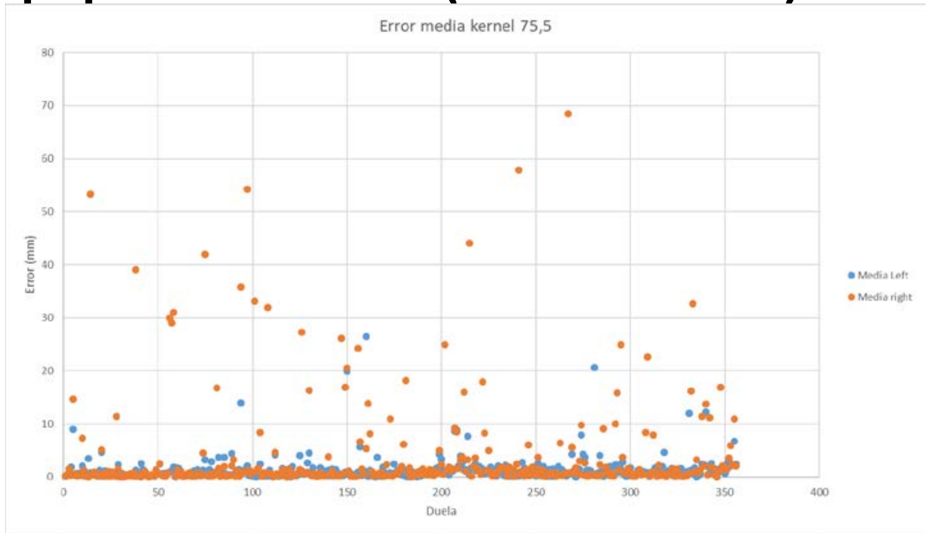
WD: 400mm

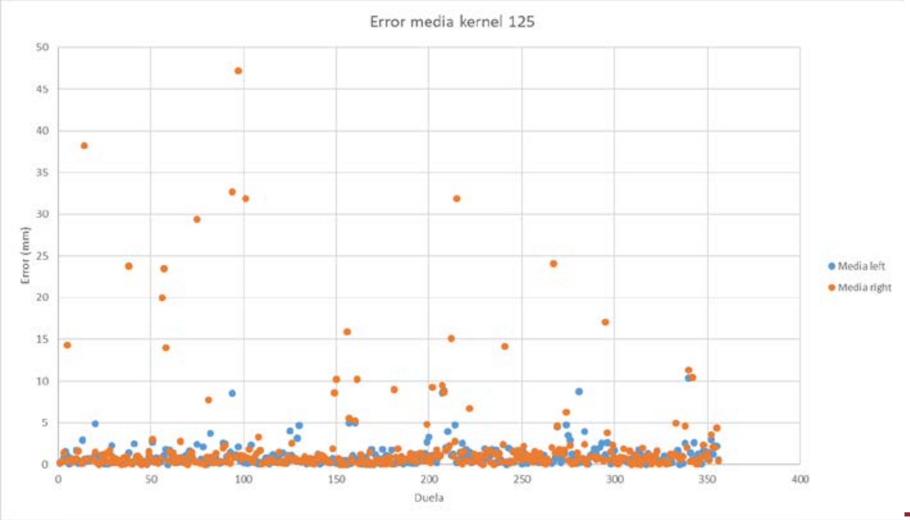
Endocentric lens

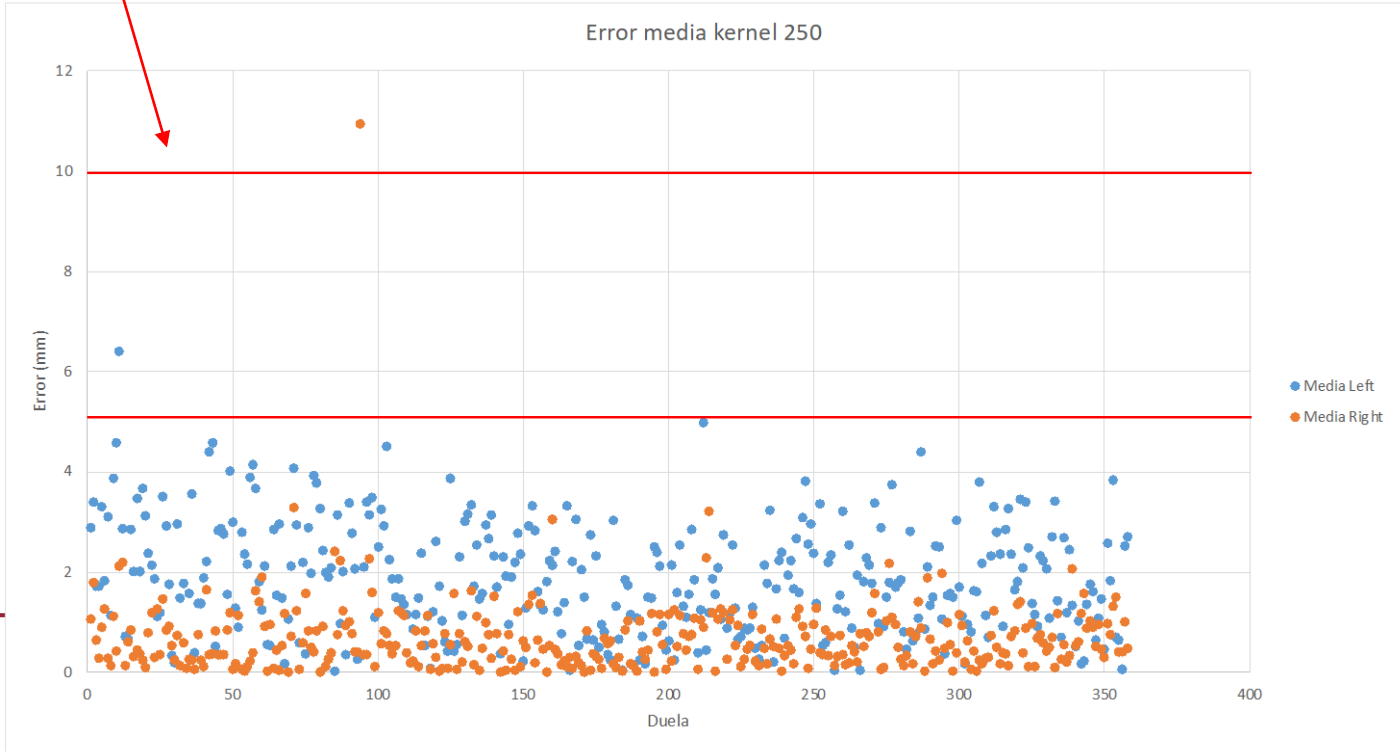
Tolerance: 5-10mm

358x2 measurements:

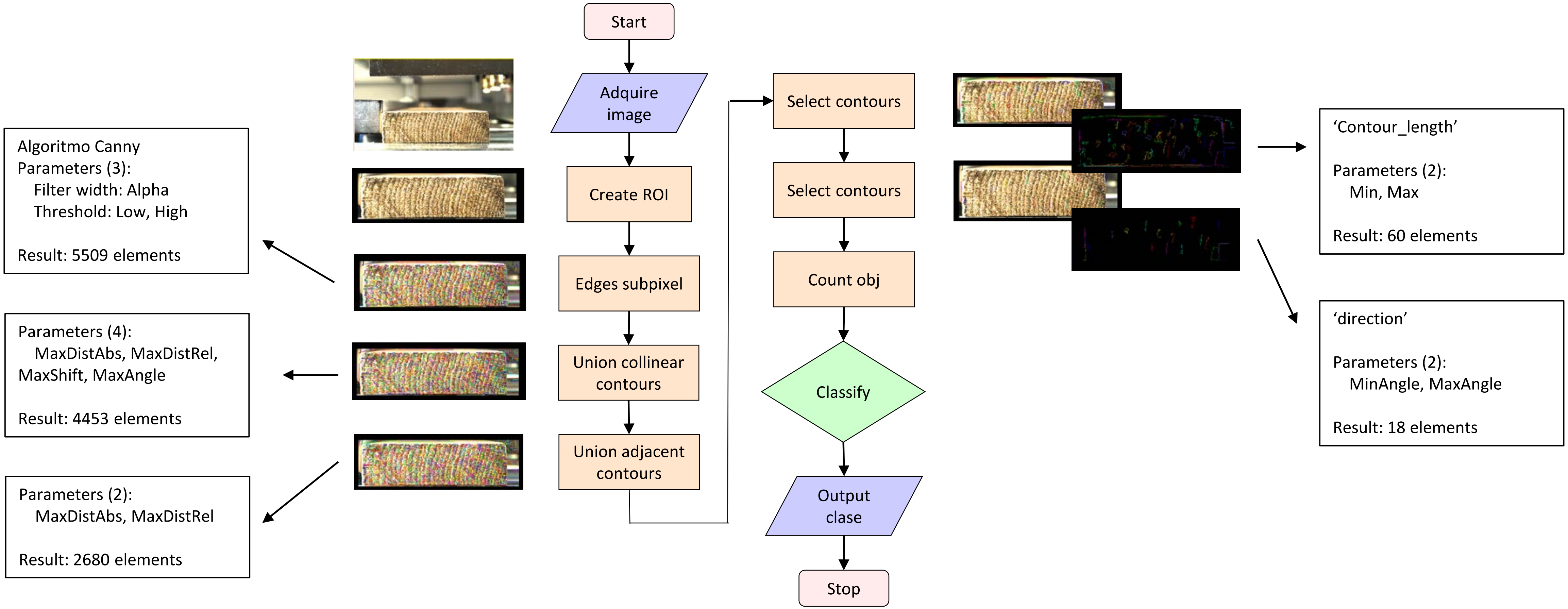
- Left camera: 0% over threshold (0)
- Right camera: 0,28% over threshold (1)







Classification *classical approach- Why not?*

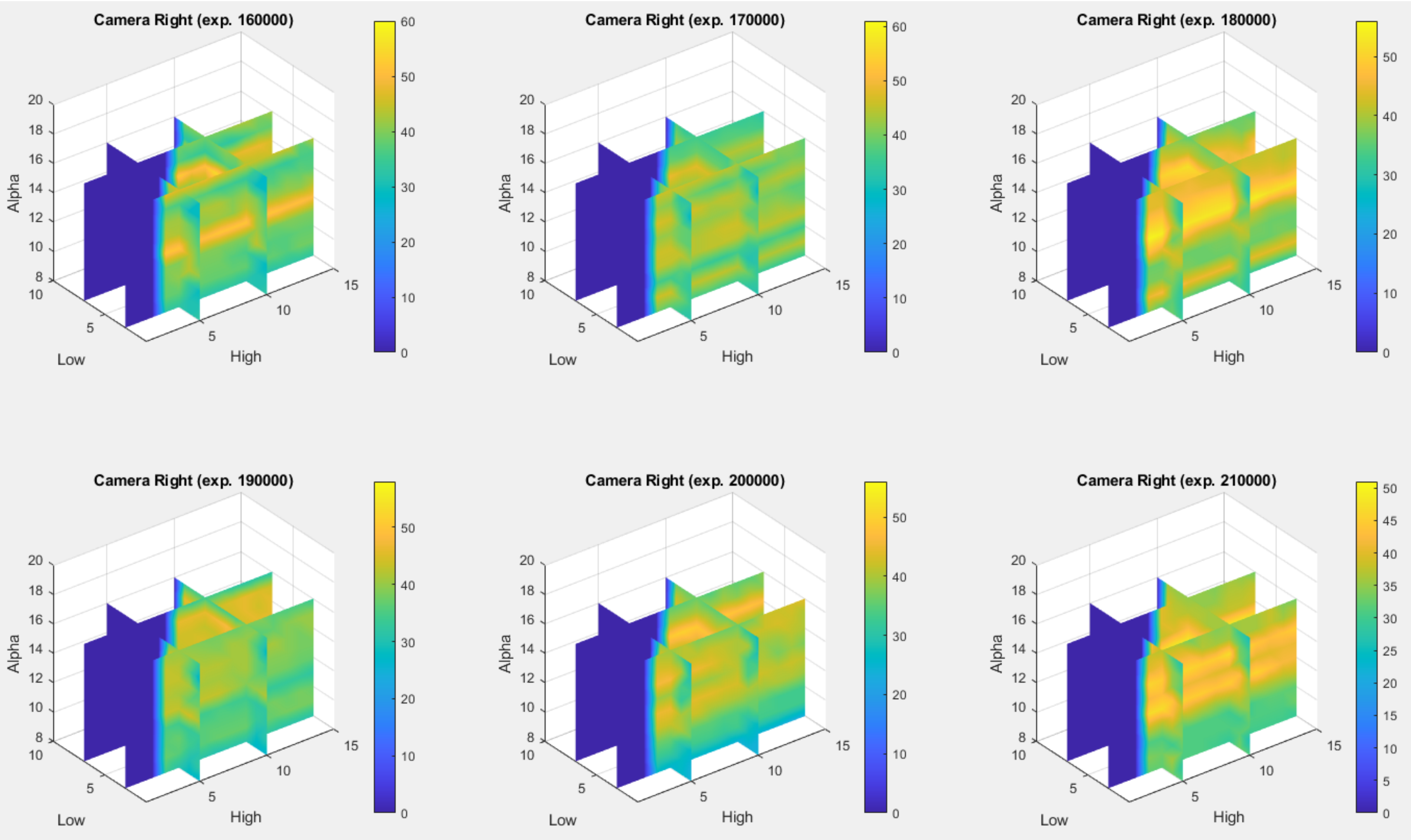


- First problem: 13 parameters
  - Optimal values? Iterative search?

Classification *classical approach- Why not?*

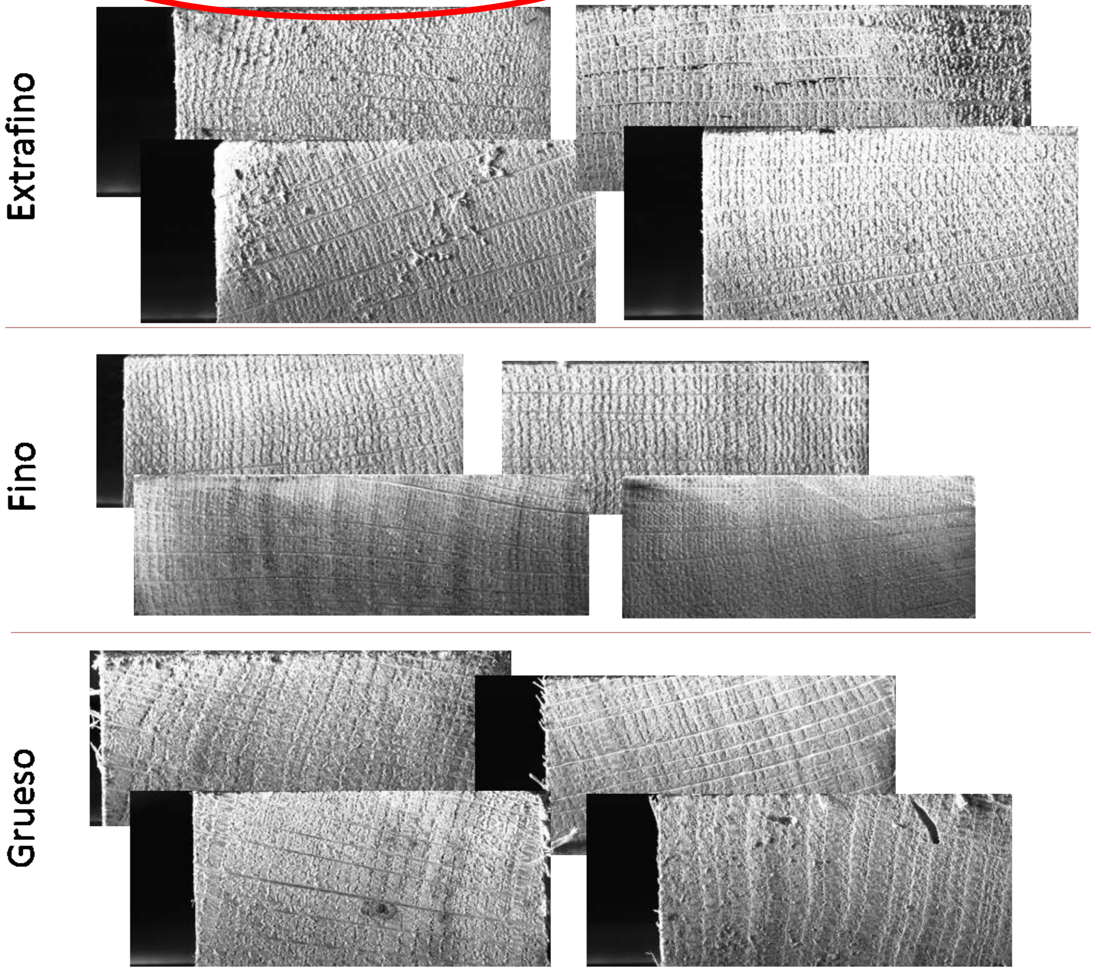
- Cámara rigth
- Búsqueda iterativa de 4 parámetros
    - Exposición (ms): [160:10:210]
    - Alfa: [8:16]
    - Low: [1:10]
    - High: [1:15]

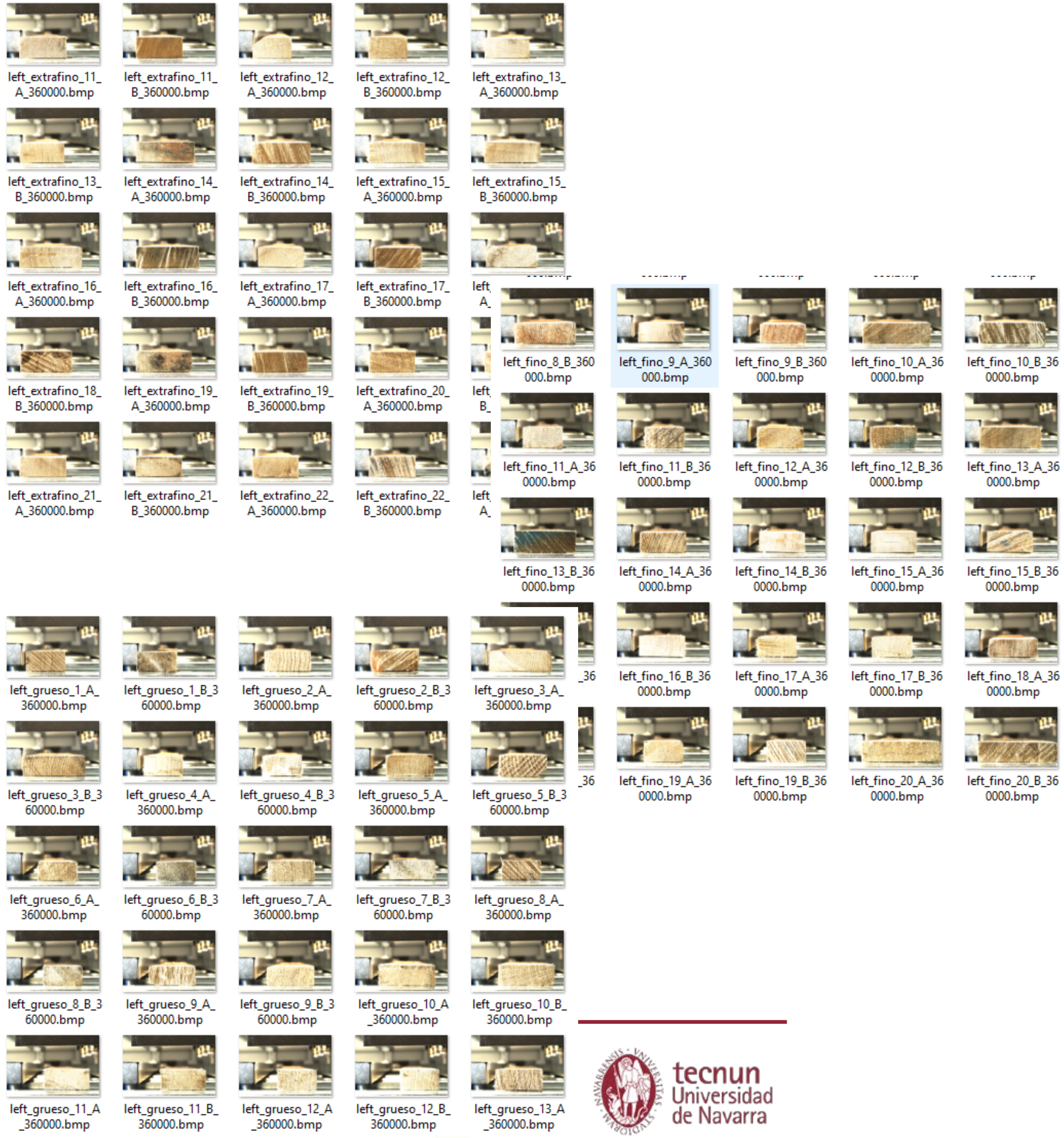
Valor óptimo para exp 160000: 60 (Low=9, High=10, Alpha=8)
Valor óptimo para exp 170000: 61 (Low=8, High=10, Alpha=12)
Valor óptimo para exp 180000: 56 (Low=9, High=10, Alpha=9)
Valor óptimo para exp 190000: 58 (Low=9, High=13, Alpha=10)
Valor óptimo para exp 200000: 56 (Low=6, High=7, Alpha=11)
Valor óptimo para exp 210000: 51 (Low=6, High=7, Alpha=9)



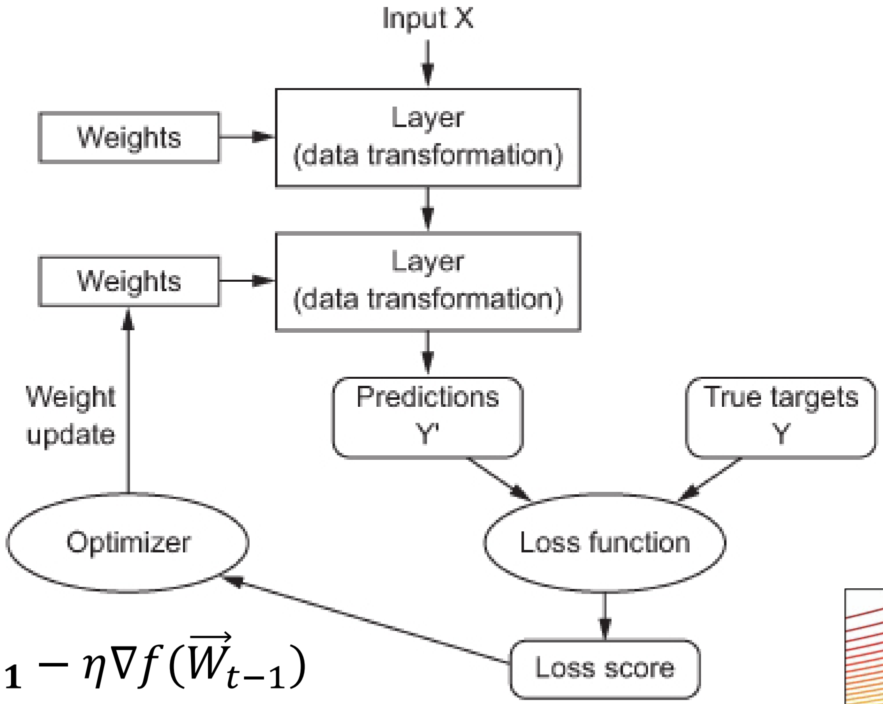
Classification *classical approach- Why not?*

- Second problem: heterogeneous samples!!
  - Deep Learning for sure!!
  - Classification vs Segmentation





Deep Learning scheme



$$\vec{W}_t = \vec{W}_{t-1} - \eta \nabla f(\vec{W}_{t-1})$$

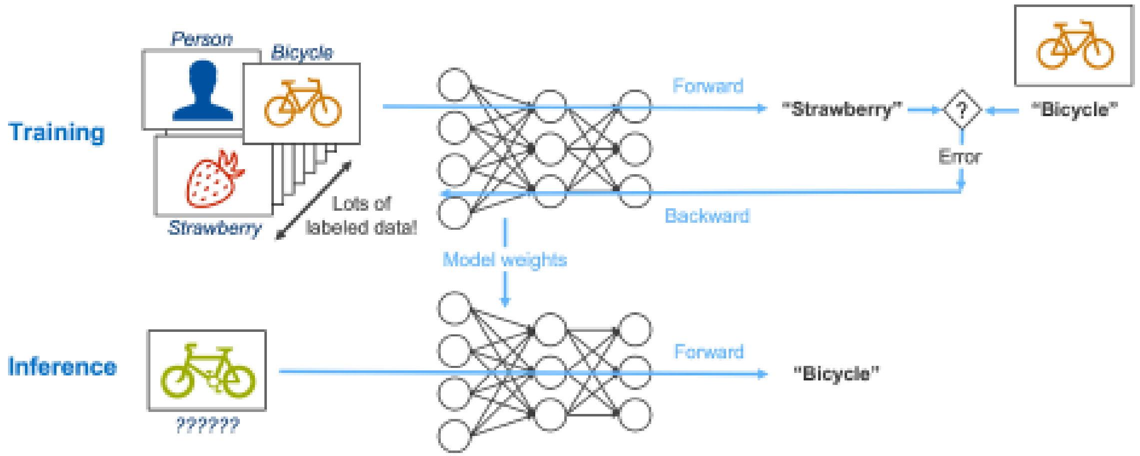
Actual weights

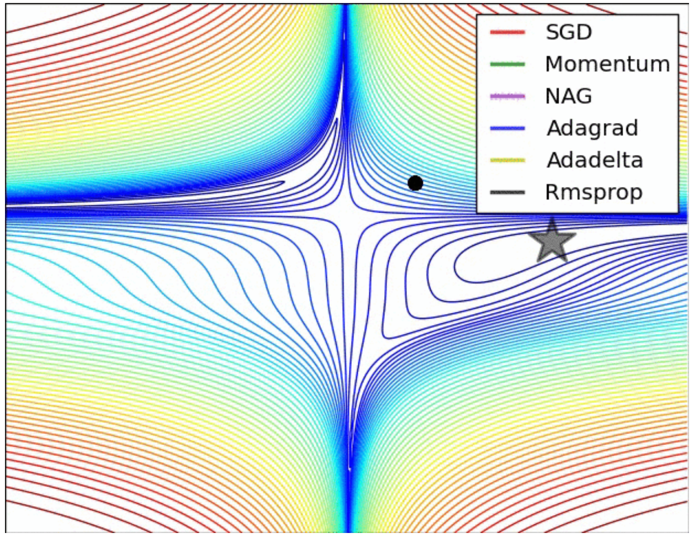
Previous step weights

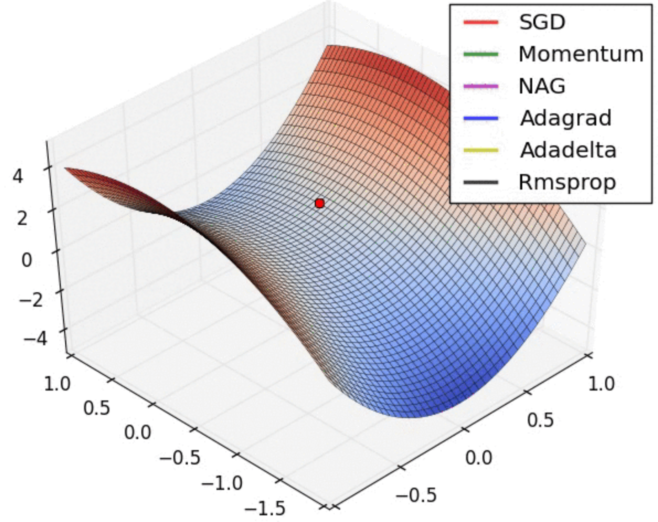
Learning rate

Gradient

Loss function

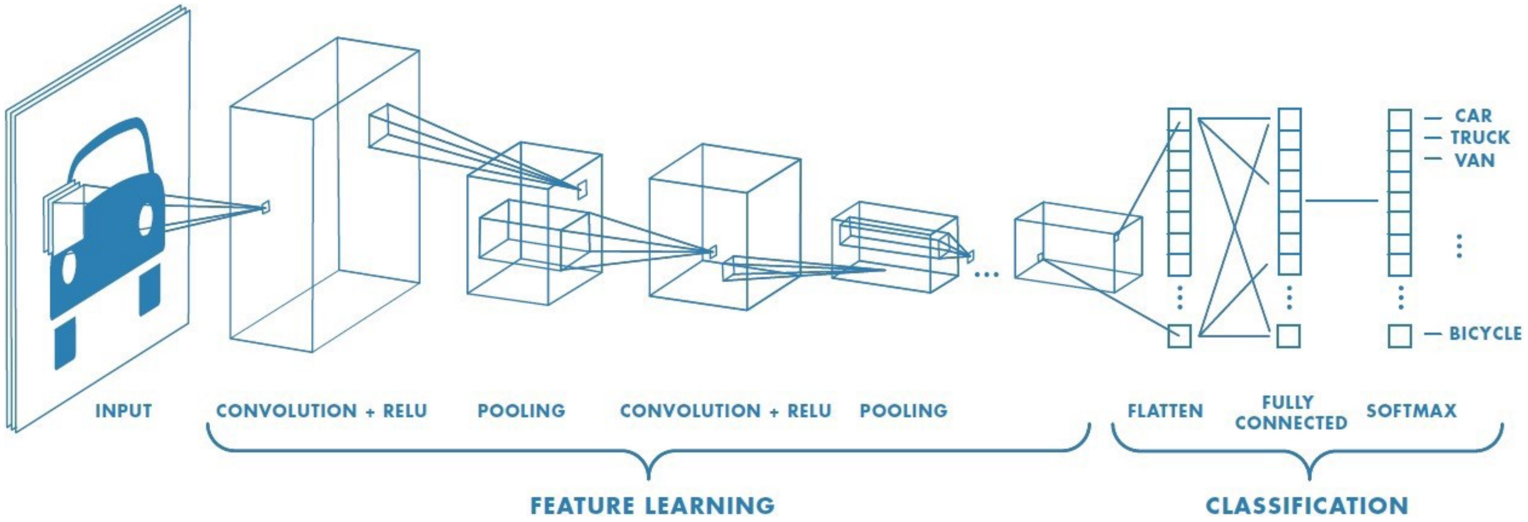


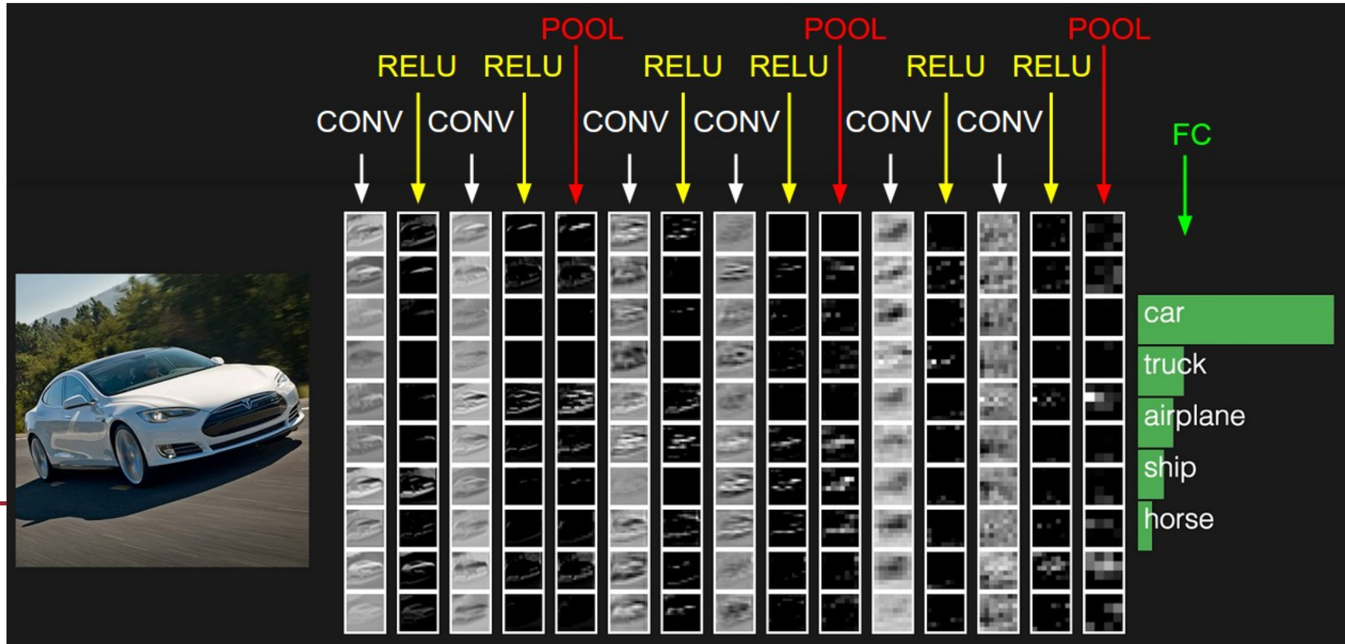




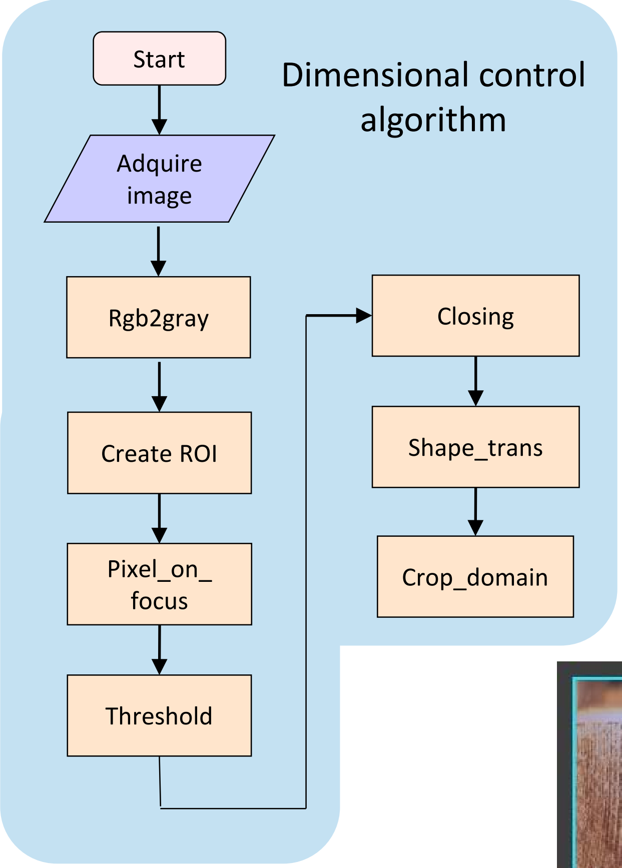
CNN scheme

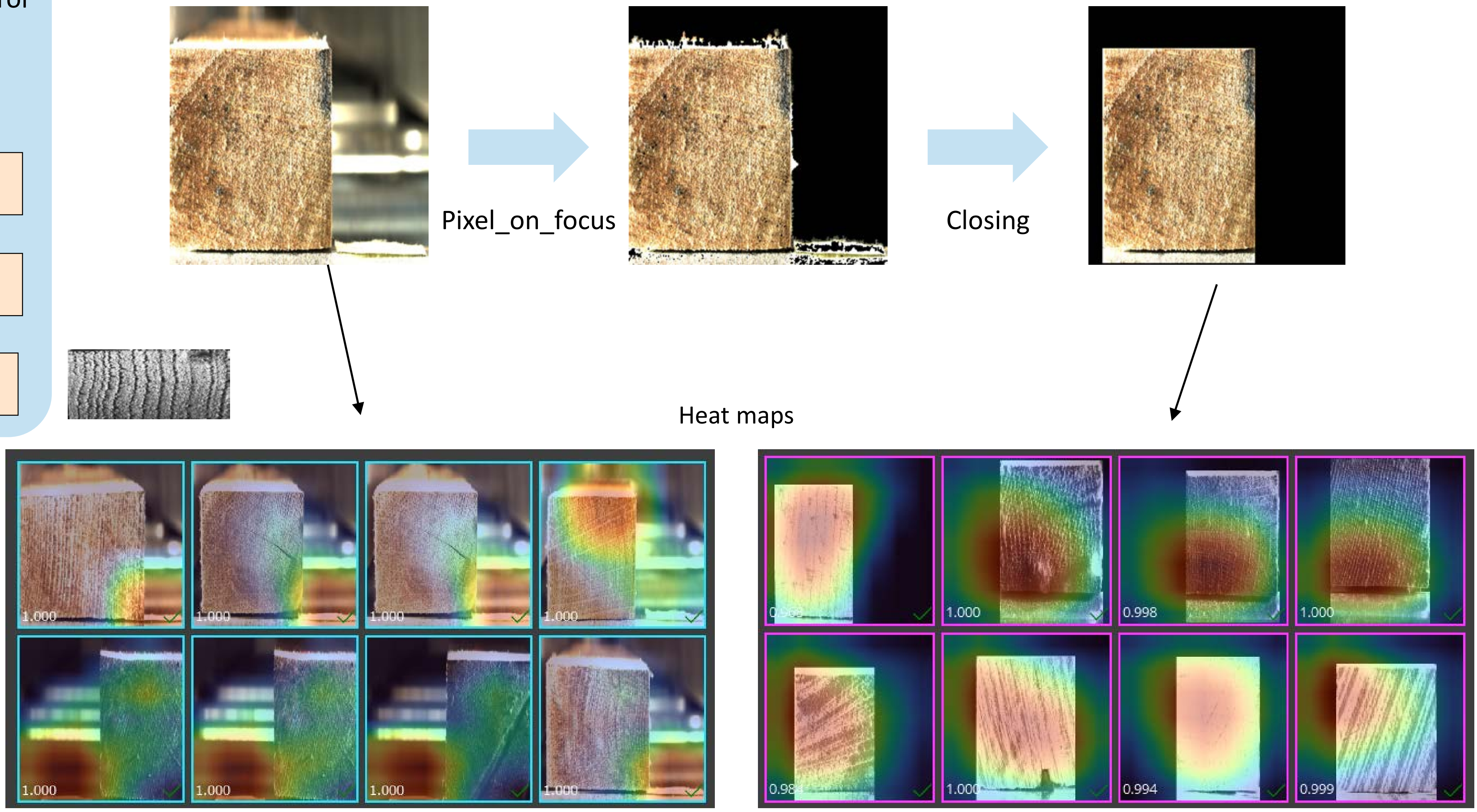
- Convolutional Neural Network (CNN):
  - Network architecture for deep learning algorithms
  - Specifically used for image recognition and tasks that involve the processing of pixel data
  - A lot of filters!!!!



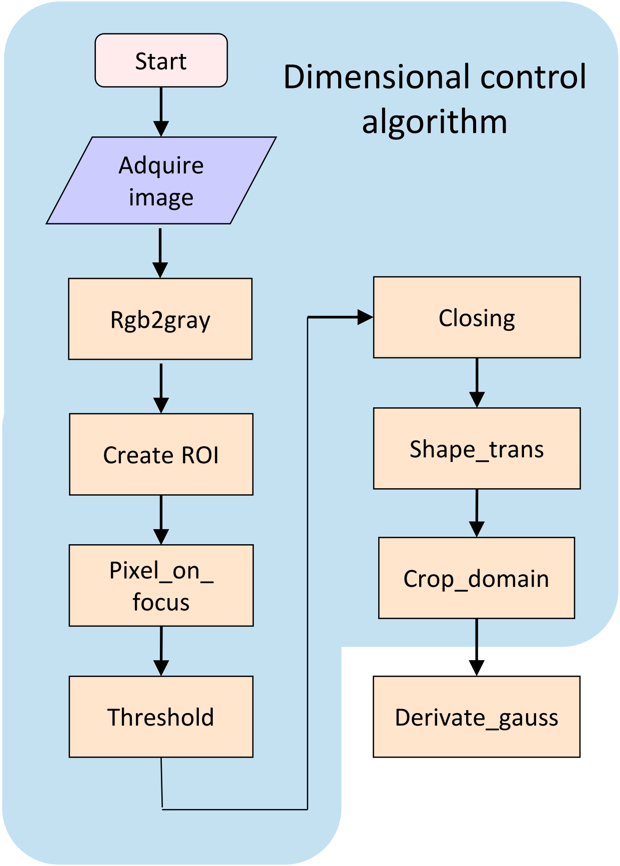


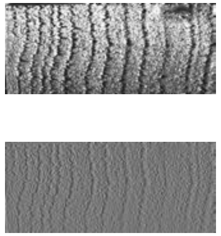
Preparing data (image preprocessing)





Preparing data (image preprocessing)

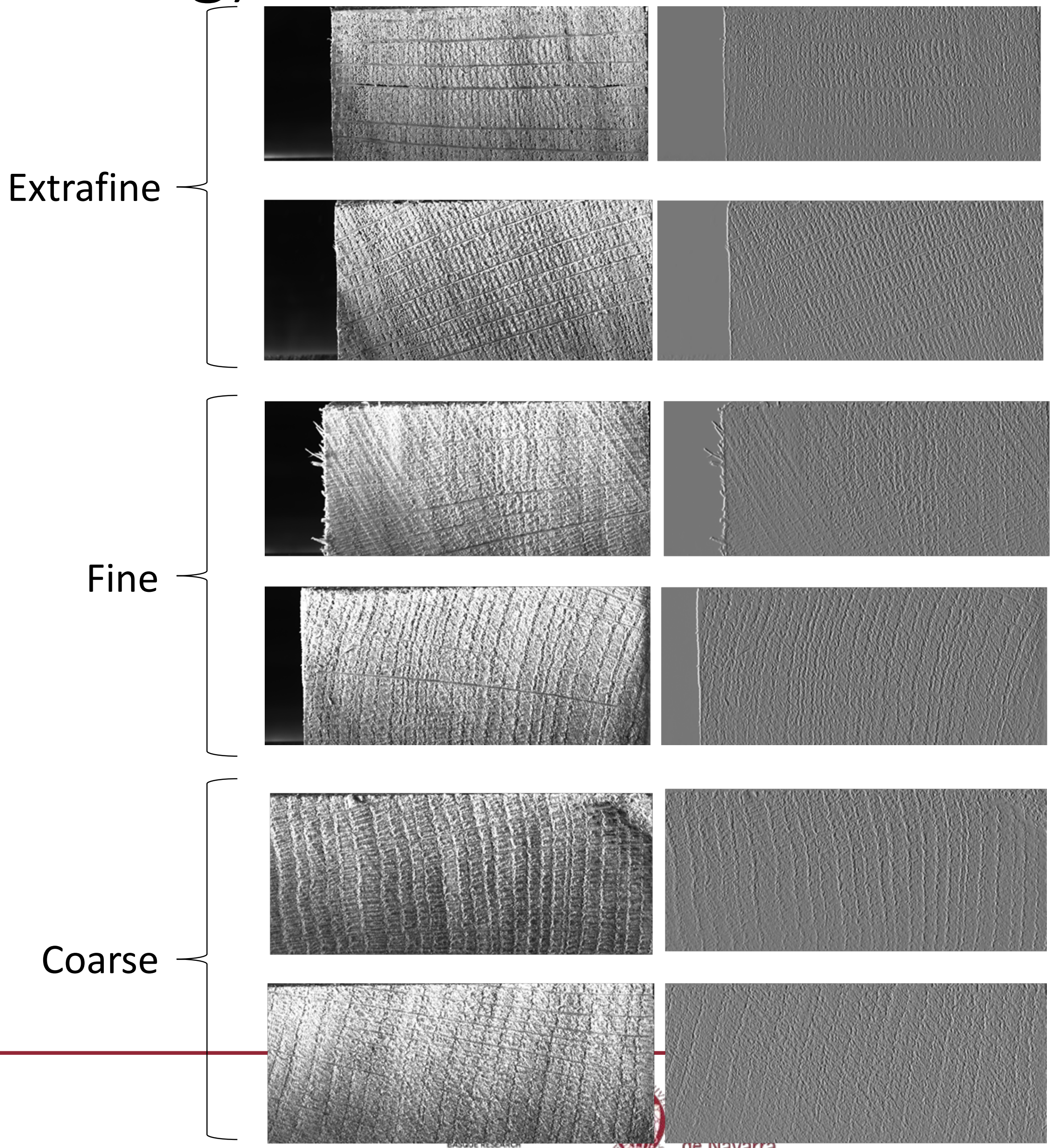




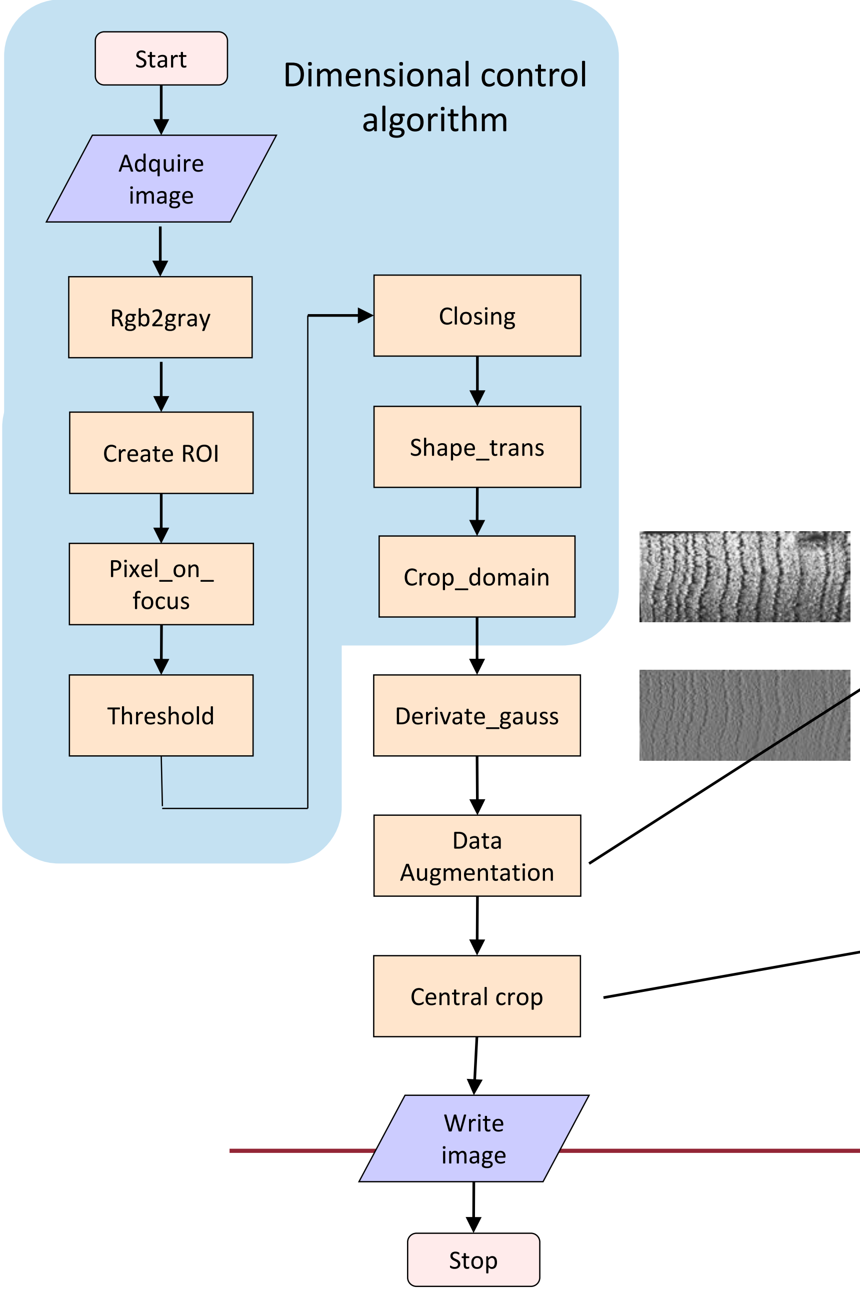
$$f'(u,v) = f(u,v) * g'(x,y)$$

$$g'(x,y) = \frac{\partial g(x,y)}{\partial x}$$

First Gauss derivative en “x”



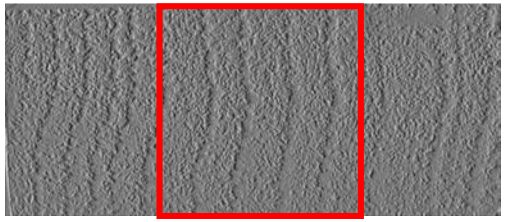
Preparing data (image preprocessing)



From 488 images, 3 transformations:

- Horizontal clip
- Vertical clip
- 180º rotation

2440 images to train



First models

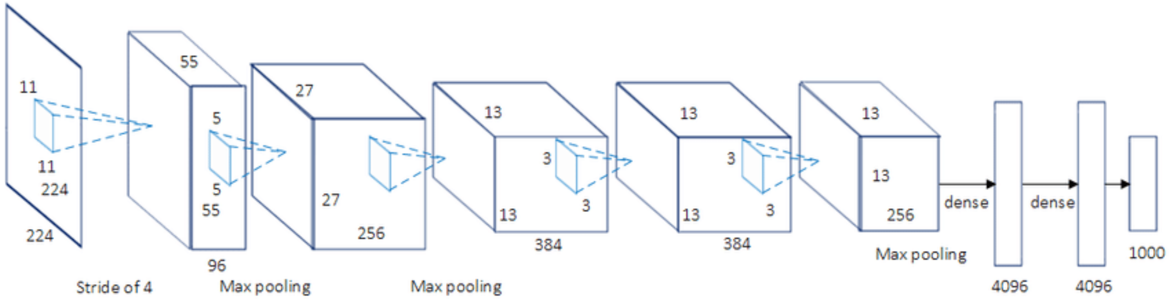
- The largest batch size that fits in memory is used
- Launch multiple iterations by changing parameters
  - Known architectures

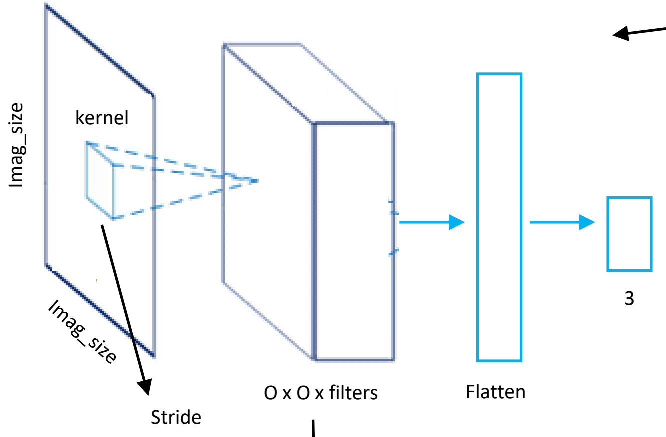
```
architecture=['resnet50', 'mobilenet', 'inception_resnet_v2']
imag_size_simu=[1024, 512, 224]
weights_simu=['imagenet', None] # Default weights with 'imagenet'. Weights from scratch with None
layer_trainable_simu=[False, True] # False: This will let us use the default weights used by the imagenet
lr_simu=[0.0001, 0.001]
wd_simu=[0, 0.001]
epoch_simu=[20]
```

- Iterations: architecture x size x weights x layer_trainable x lr x wd x epoch = 144

First models

- The largest batch size that fits in memory is used
- Launch multiple iterations by changing parameters
  - Own architecture based on AlexNet





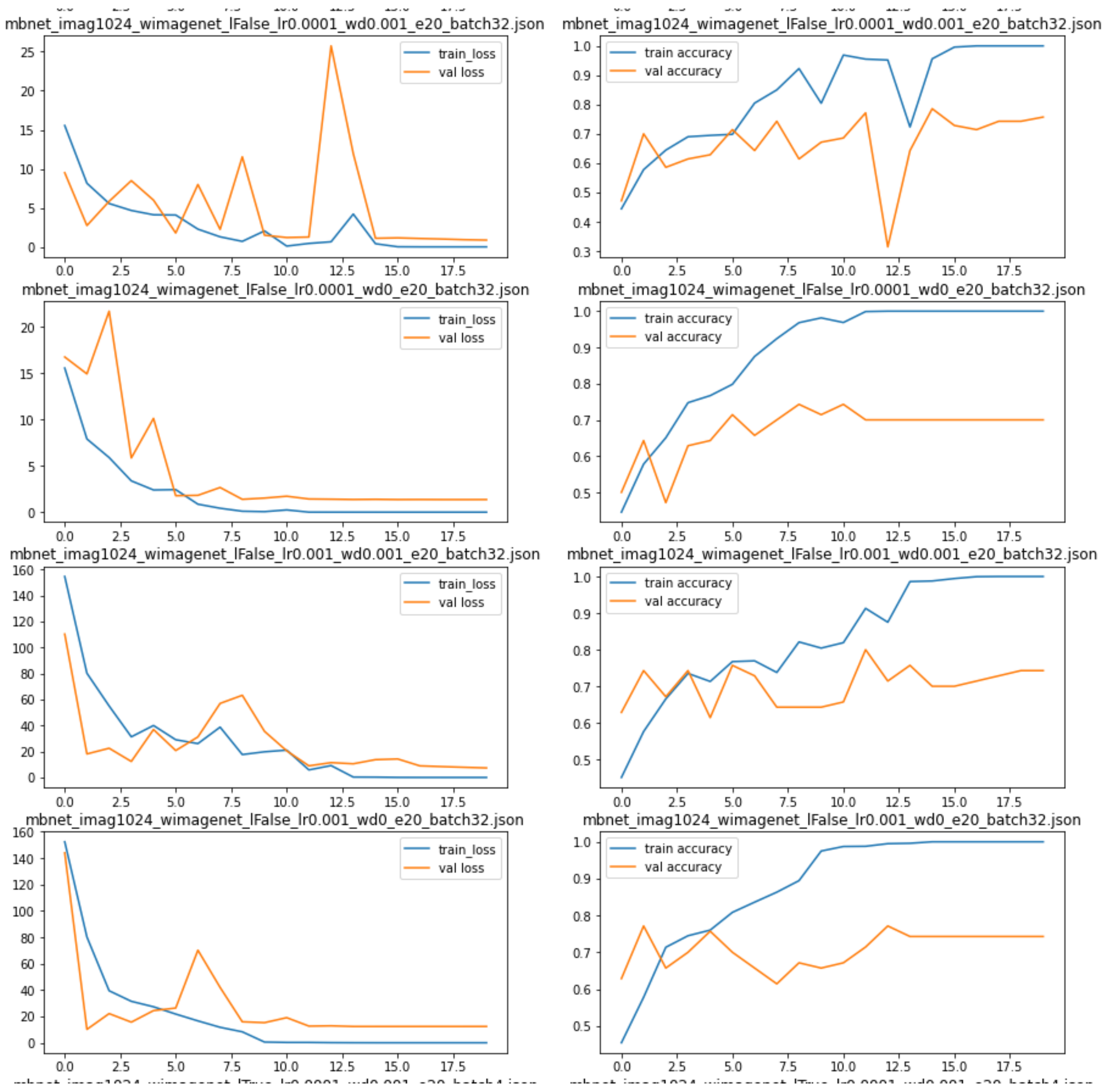
$$O = \frac{W - K + 2P}{S} + 1$$

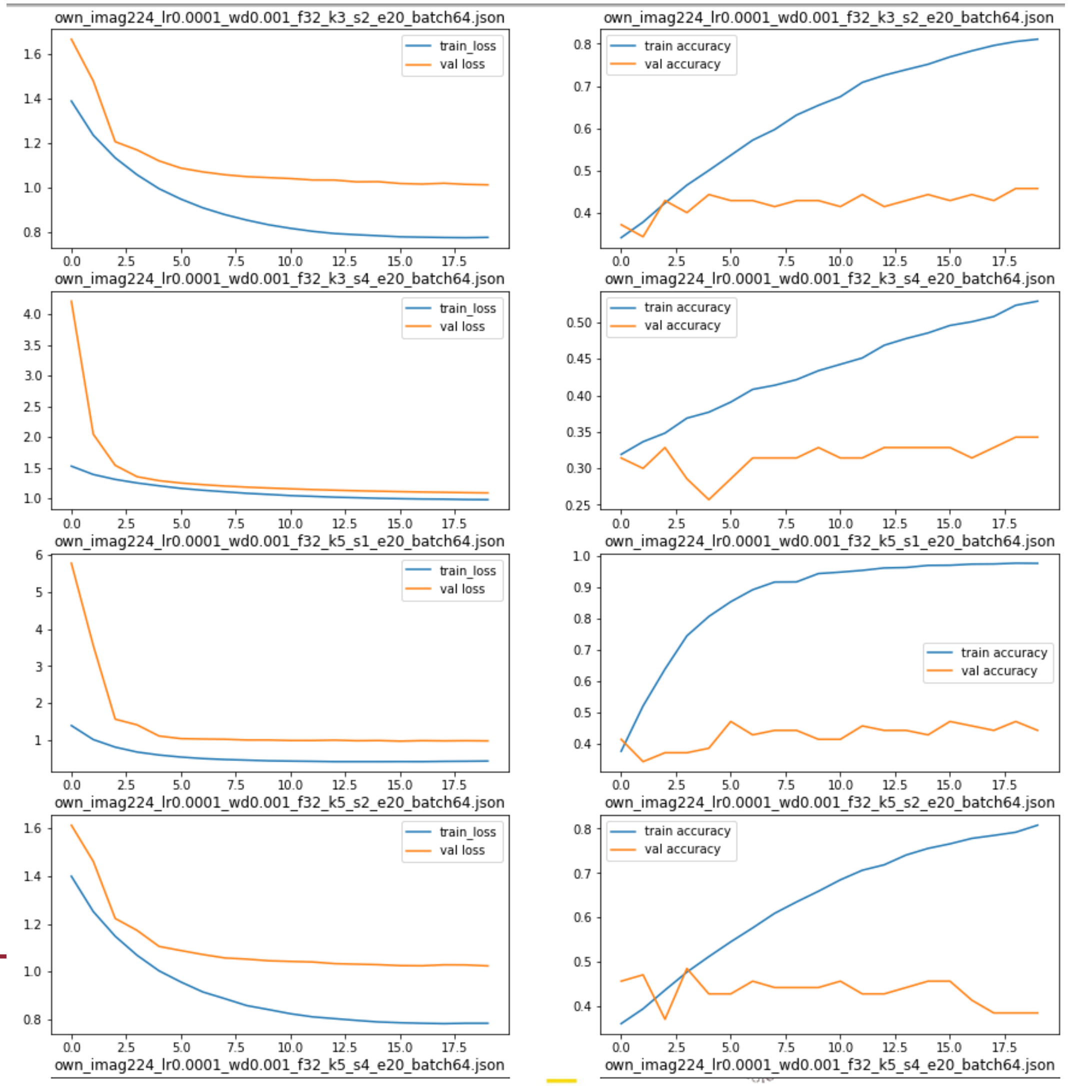
```
imag_size_simu=[1024, 512, 224]
lr_simu=[0.0001, 0.001]
wd_simu=[0, 0.001]
isCallback=False
filters=[16, 32, 64]
kernel=[3, 5, 7]
strides=[1, 2, 4]
epoch_simu=[20]
```

- Iterations: size x lr x wd x filters x kernel x strides x epoch = 324

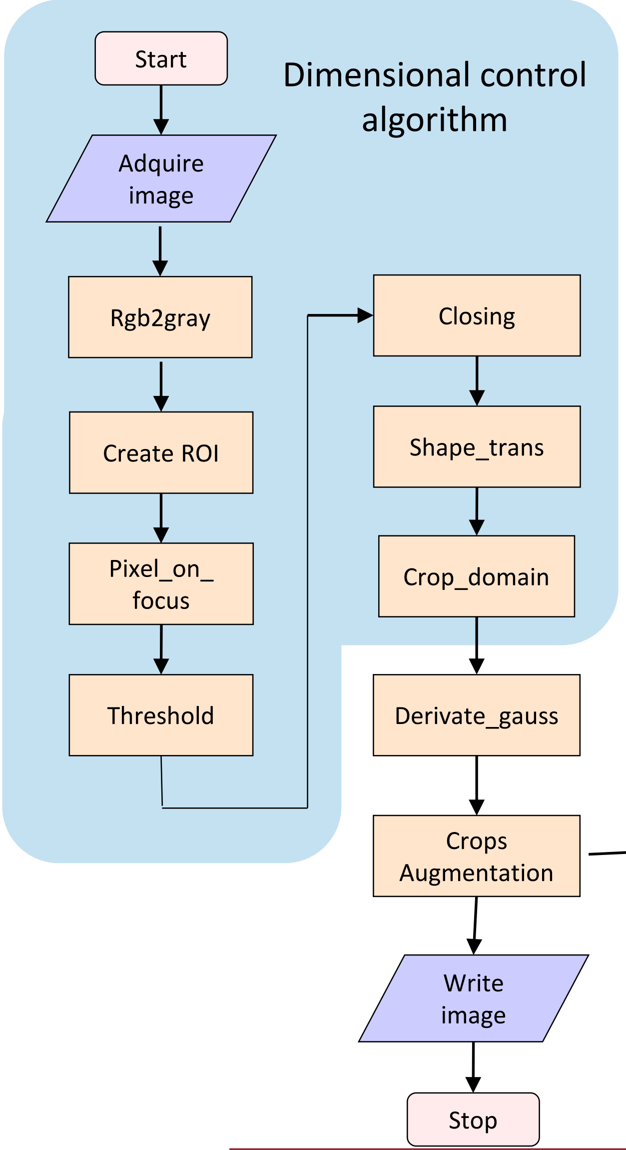
First models

- Total iterations: 144 + 324 = 468
- Time: 9 days

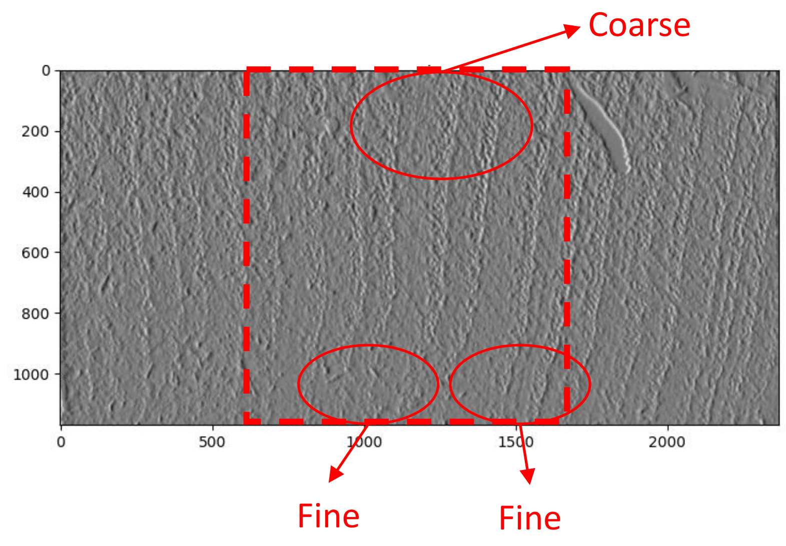


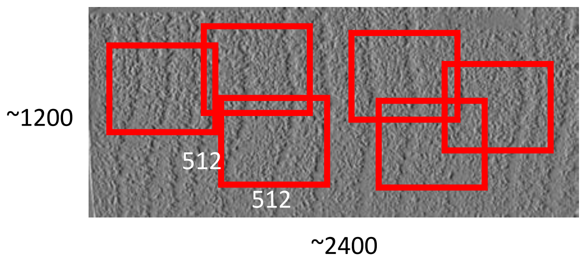


Preparing data (image preprocessing)



- Problem of intra-classes in the same sample!!



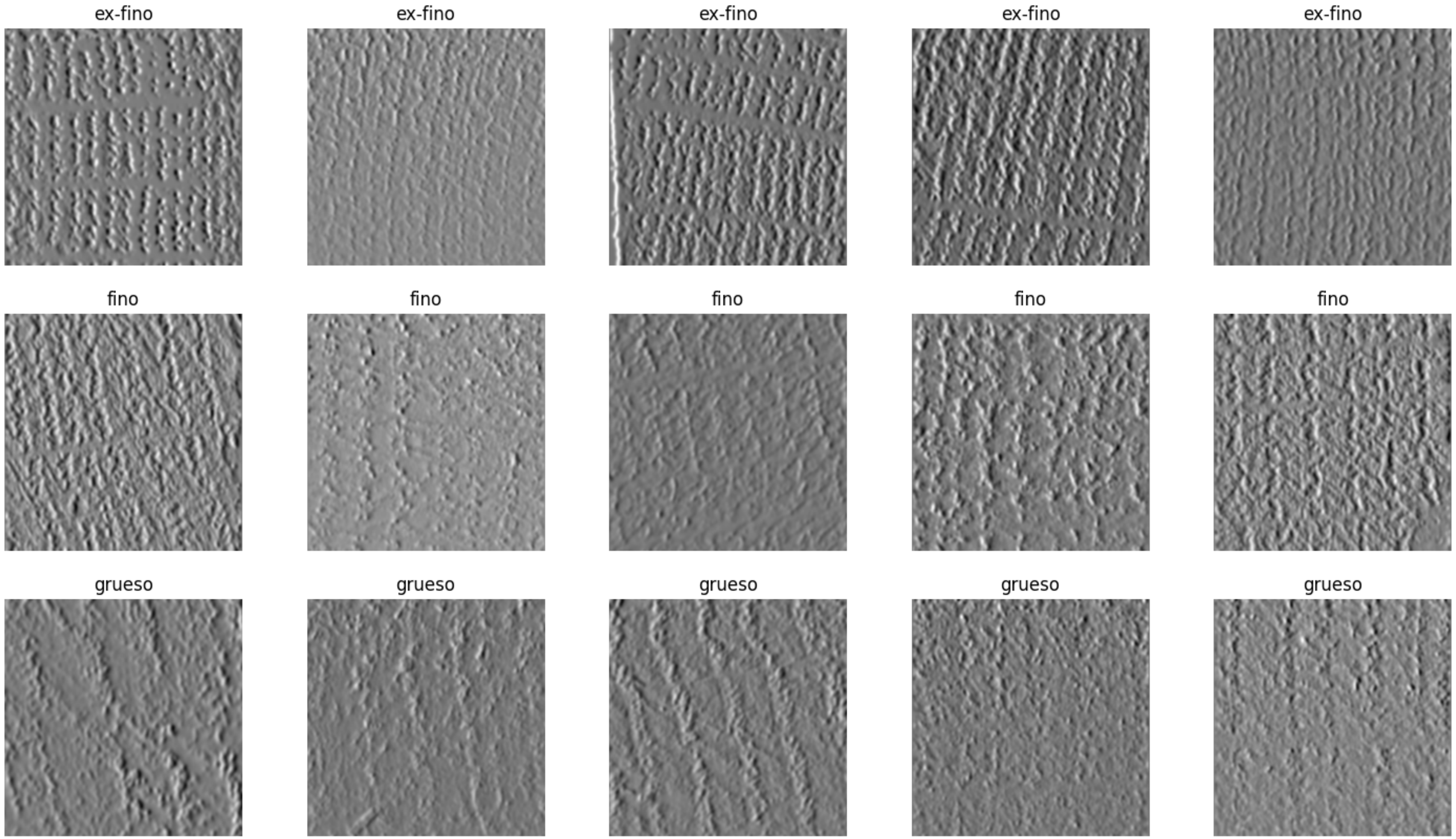


- Advantages
  - No intra-class problem
  - Increase DB without DA
  - Voting system
- Disadvantages
  - Labelling (counting rings)

Preparing data (labelling)

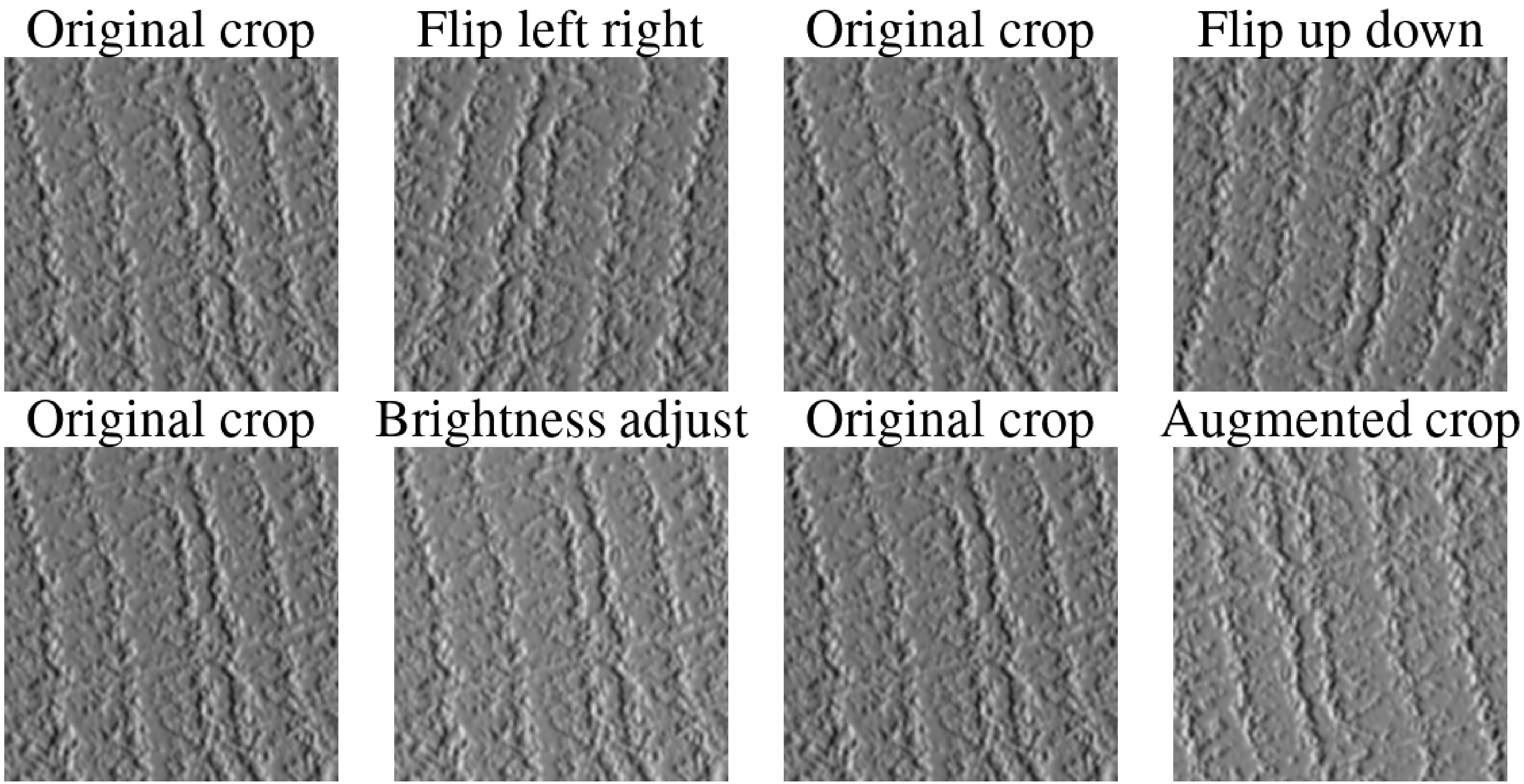
- Taking into account number of rings/cm and knowing stave width (mm), new classification rules for crops

	Number of rings/cm	Number of rings/crop (~1.3cm)
Coarse	$x < 4$	$x \leq 5$
Fine	$4 \leq x < 7$	$6 \leq x \leq 9$
Extrafine	$x \geq 7$	$x \geq 10$



Preparing data (data augmentation)

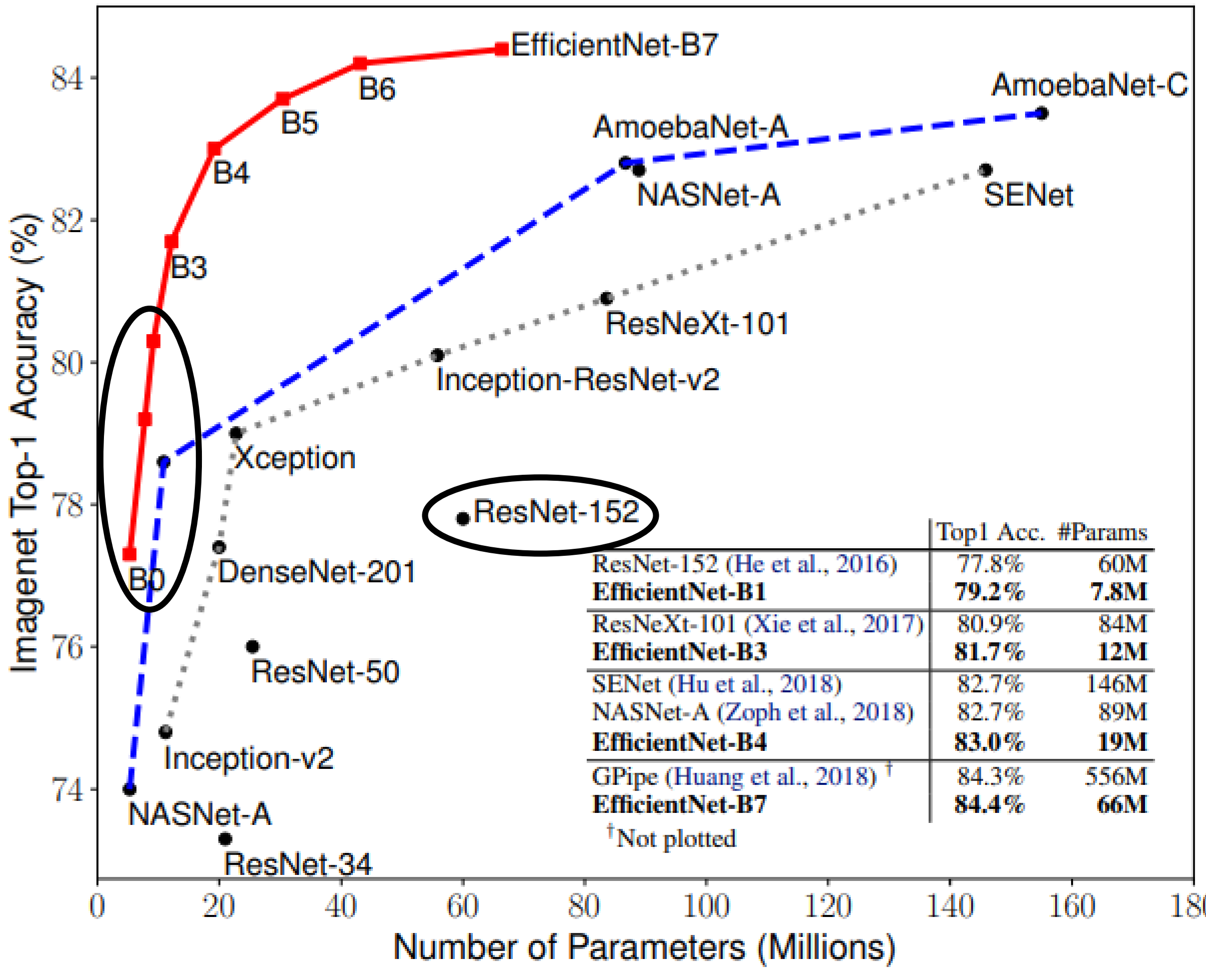
- Data augmentation in extrafine crops (unbalanced class)
  - Horizontal/vertical flips and brightness (0.05)



CNN architectures

Modelos disponibles en TensorFlow

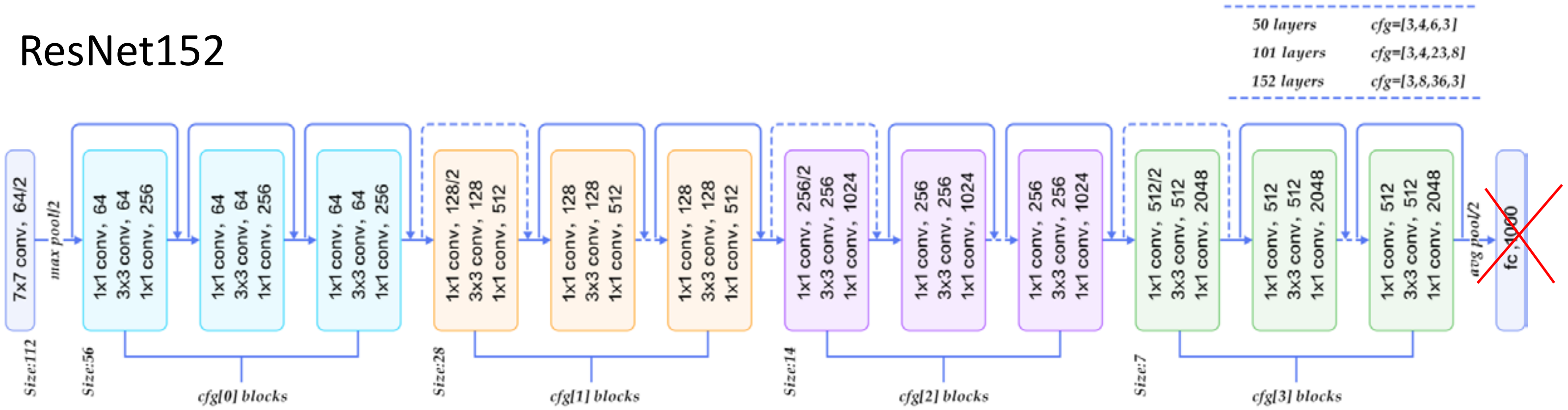
Nombre	Capas	Parámetros (MILLONES)	Nombre	Capas	Parámetros (MILLONES)
DenseNet121	427	7	RegNetX002	143	2,3
DenseNet169	595	12,6	RegNetX004	233	4,8
DenseNet201	707	18,3	RegNetX006	173	5,7
			RegNetX008	173	6,6
EfficientNetB0	238	4	RegNetX016	193	8,3
EfficientNetB1	340	6,6	RegNetX032	263	14,4
EfficientNetB2	340	7,8	RegNetX040	243	20,8
EfficientNetB3	385	10,8	RegNetX064	183	24,7
EfficientNetB4	475	17,7	RegNetX080	243	37,7
EfficientNetB5	577	28,5	RegNetX120	203	43,9
EfficientNetB6	667	40,9	RegNetX160	233	52,3
EfficientNetB7	814	64,1	RegNetX320	243	105,5
EfficientNetV2B0	270	5,9	RegNetY002	195	2,8
EfficientNetV2B1	334	6,9	RegNetY004	237	3,9
EfficientNetV2B2	349	8,8	RegNetY006	223	5,5
EfficientNetV2B3	409	12,9	RegNetY008	209	5,5
			RegNetY016	391	10,4
EfficientNetV2S	513	20,3	RegNetY032	307	17,9
EfficientNetV2M	740	53,1	RegNetY040	321	19,6
EfficientNetV2L	1028	117,7	RegNetY064	363	29,4
			RegNetY080	251	37,2
InceptionResNetV2	780	54,3	RegNetY120	279	49,7
			RegNetY160	265	80,7
InceptionV3	311	21,8	RegNetY320	293	141,5
MobileNet	86	3,2	ResNet101	345	45,7
MobileNetV2	154	2,3	ResNet101V2	377	42,6
MobileNetV3Large	263	2,99	ResNet152	515	58,4
MobileNetV3Small	229	0,9	ResNet152V2	564	58,3
			ResNet50	175	23,6
NASNetLarge	1039	84,9	ResNet50V2	190	23,6
NASNetMobile	769	4,3			
			ResNetRS50	271	33,7
			ResNetRS101	523	61,7
			ResNetRS152	781	84,7
			ResNetRS200	1087	91,3
			ResNetRS270	1471	128
			ResNetRS350	1887	162,2
			ResNetRS420	2255	190,2
			VGG16	19	14,7
			VGG19	22	20
			Xception	132	20,9



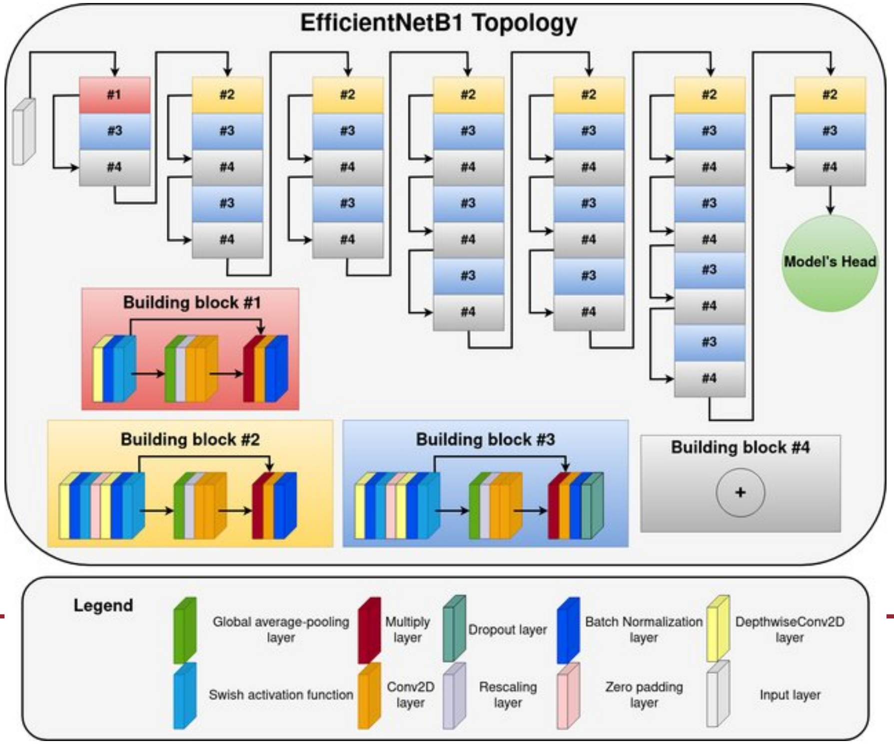
CNN architectures

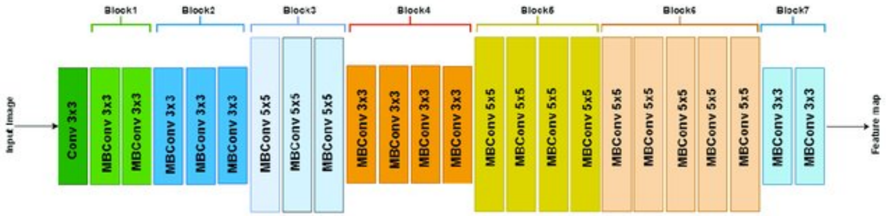
Modelos disponibles en TensorFlow					
Nombre	Capas	Parámetros (MILLONES)	Nombre	Capas	Parámetros (MILLONES)
DenseNet121	427	7	RegNetX002	143	2,3
DenseNet169	595	12,6	RegNetX004	233	4,8
DenseNet201	707	18,3	RegNetX006	173	5,7
			RegNetX008	173	6,6
EfficientNetB0	238	4	RegNetX016	193	8,3
EfficientNetB1	340	6,6	RegNetX032	263	14,4
EfficientNetB2	340	7,8	RegNetX040	243	20,8
EfficientNetB3	385	10,8	RegNetX064	183	24,7
EfficientNetB4	475	17,7	RegNetX080	243	37,7
EfficientNetB5	577	28,5	RegNetX120	203	43,9
EfficientNetB6	667	40,9	RegNetX160	233	52,3
EfficientNetB7	814	64,1	RegNetX320	243	105,5
EfficientNetV2B0	270	5,9	RegNetY002	195	2,8
EfficientNetV2B1	334	6,9	RegNetY004	237	3,9
EfficientNetV2B2	349	8,8	RegNetY006	223	5,5
EfficientNetV2B3	409	12,9	RegNetY008	209	5,5
			RegNetY016	391	10,4
EfficientNetV2S	513	20,3	RegNetY032	307	17,9
EfficientNetV2M	740	53,1	RegNetY040	321	19,6
EfficientNetV2L	1028	117,7	RegNetY064	363	29,4
			RegNetY080	251	37,2
InceptionResNetV2	780	54,3	RegNetY120	279	49,7
			RegNetY160	265	80,7
InceptionV3	311	21,8	RegNetY320	293	141,5
MobileNet	86	3,2	ResNet101	345	45,7
MobileNetV2	154	2,3	ResNet101V2	377	42,6
MobileNetV3Large	263	2,99	ResNet152	515	58,4
MobileNetV3Small	229	0,9	ResNet152V2	564	58,3
			ResNet50	175	23,6
NASNetLarge	1039	84,9	ResNet50V2	190	23,6
NASNetMobile	769	4,3			
			ResNetRS50	271	33,7
			ResNetRS101	523	61,7
			ResNetRS152	781	84,7
			ResNetRS200	1087	91,3
			ResNetRS270	1471	128
			ResNetRS350	1887	162,2
			ResNetRS420	2255	190,2
			VGG16	19	14,7
			VGG19	22	20
			Xception	132	20,9

ResNet152

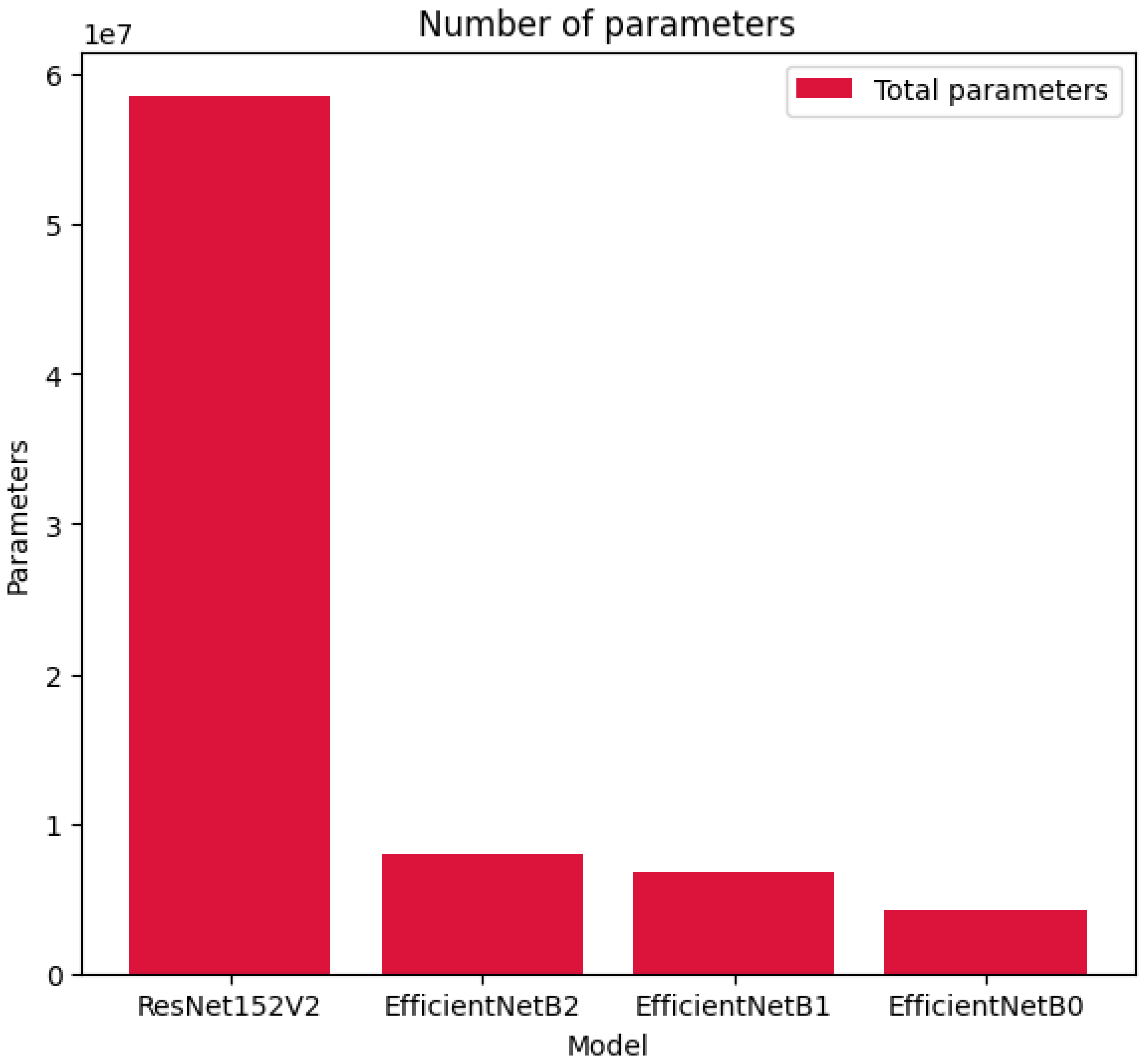


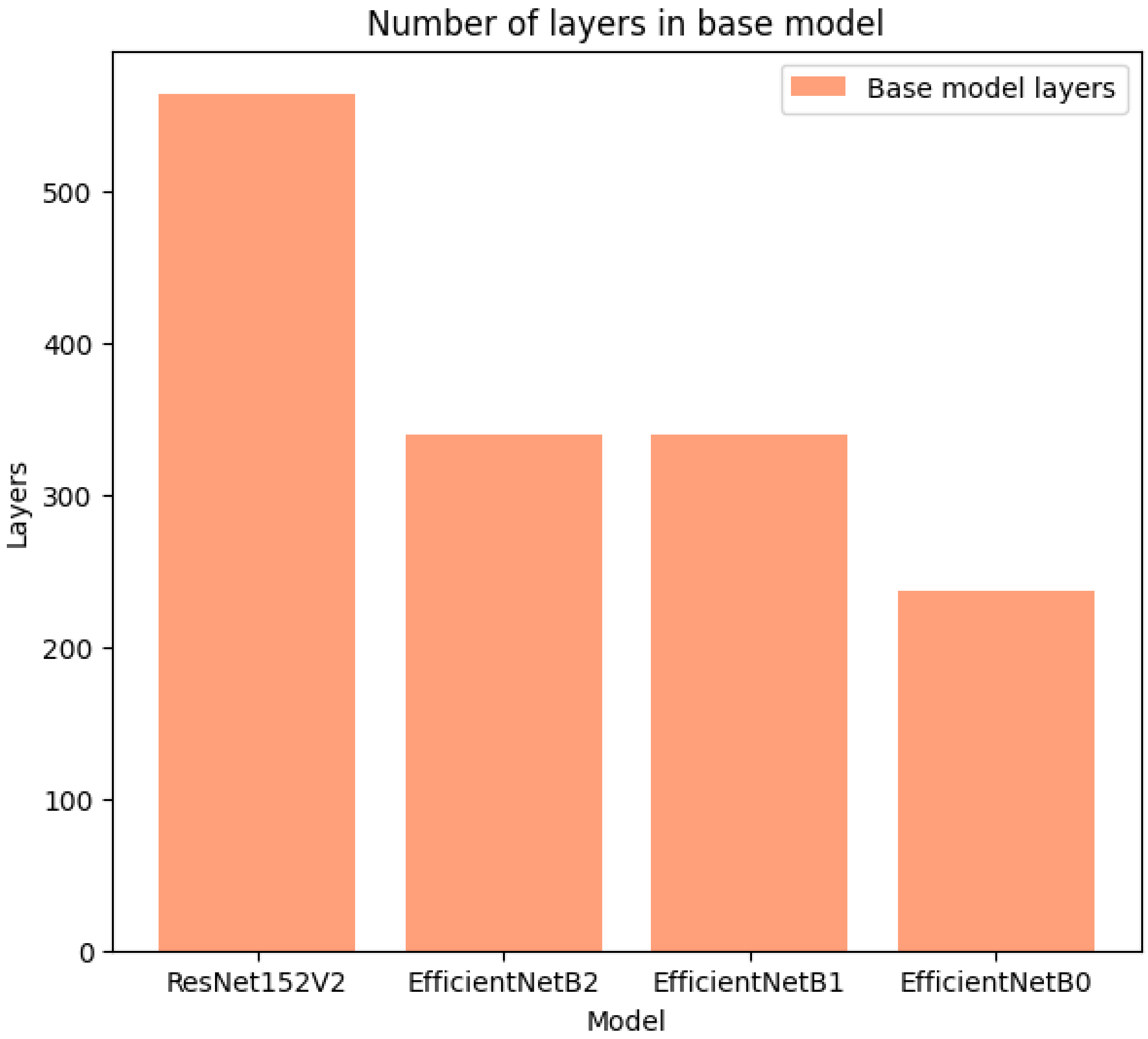
EfficientNet B1



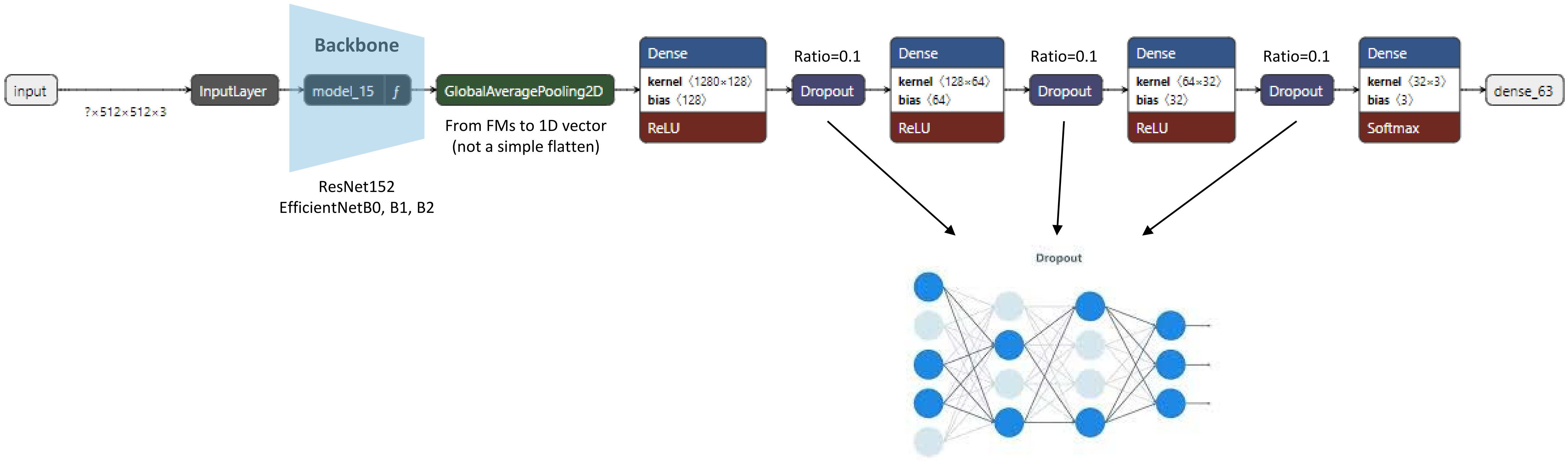


CNN architectures

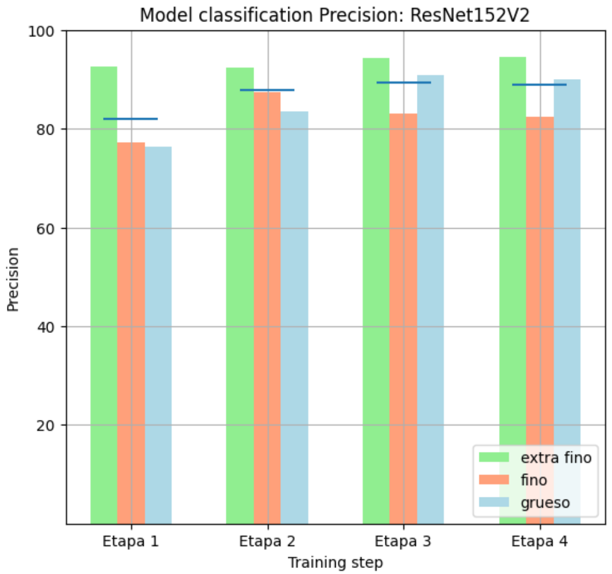


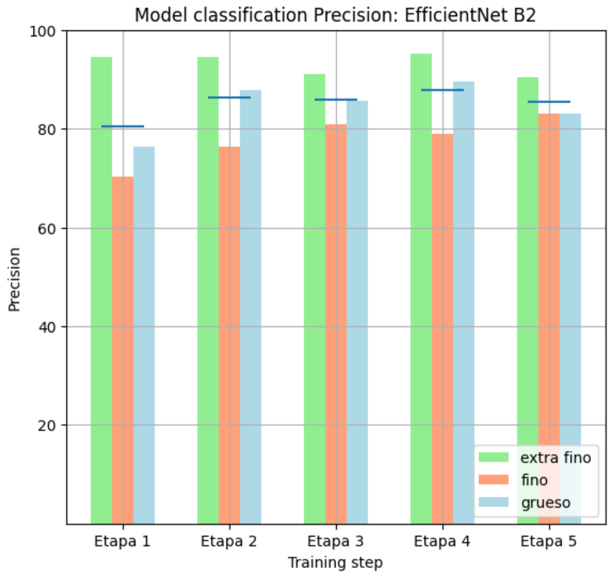


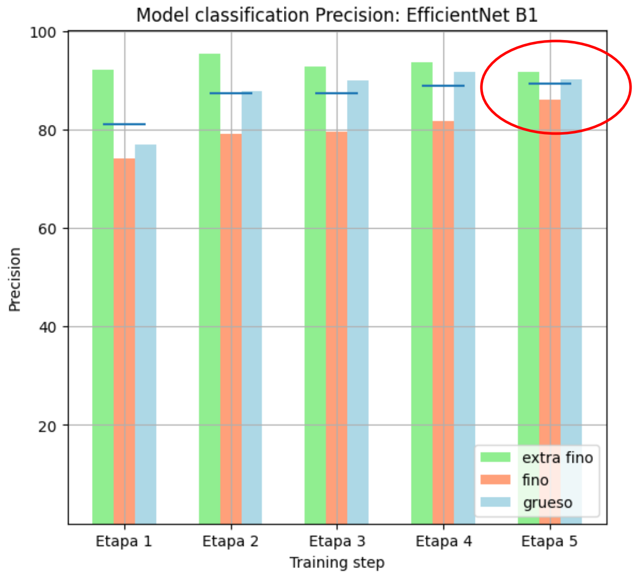
Classification - Our CNN arquitecture

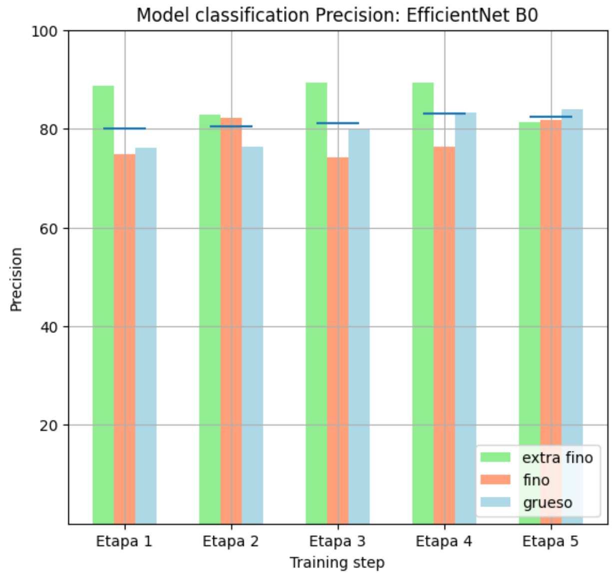


Classification - First results without tunning





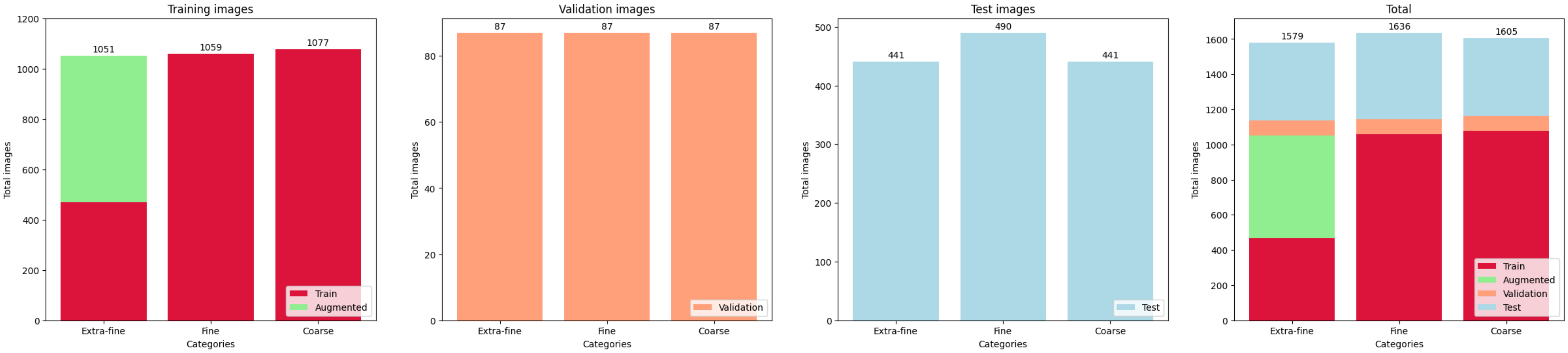




Classification - Dataset split

Voting system

		Crops			
	Duelas	Total	Train	Valid	Test
Grueso	550	1605	1077	87	441
Fino	403	1636	1059	87	490
Extrafino	852	1579	1051	87	441
			Split %		
			0,671	0,054	0,275
			0,647	0,053	0,300
			0,666	0,055	0,279



Classification - Train parameters

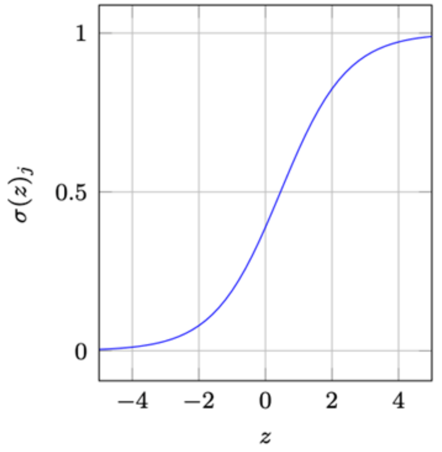
- Loss function: Cross-Entropy Categorical Multiclass
  - Generalization of Binary Cross-Entropy

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{i,k} \log \hat{y_{i,k}}$$

- Activation function (in dense layers): Softmax
  - Generalization of Sigmoid function

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{para } j = 1, \ldots, K.$$



- Optimizer: ADAM
  - It uses a more complex "Momentum" concept than SGD

Memory weight

$$\overrightarrow{GWM} = \beta(\overrightarrow{GWM}_{t-1}) + (1-\beta)\nabla f(\overrightarrow{W}_{t-1})$$

SGD optimizer: $\overrightarrow{W}_t = \overrightarrow{W}_{t-\mathbf{1}} - \eta\nabla f(\overrightarrow{W}_{t-1})$ Adding momentum $\overrightarrow{W}_t = \overrightarrow{W}_{t-1} - \eta(\overrightarrow{GWM})$

Classification - Train parameters: Adam/AdamW optimizers

- Adam
  - It calculates **individual adaptive learning rate** for each weight from estimates of first and second moments of the gradients in order to reduce the vanishing learning rates

First moment coefficient

$$\vec{W}_t = \vec{W}_{t-1} - \eta \frac{\hat{f}_t}{\sqrt{\hat{s}_t} + \varepsilon} \qquad \hat{f}_t = \frac{f_t}{1-\beta_1^t} \quad \& \quad \hat{s}_t = \frac{s_t}{1-\beta_2^t}$$

$$f_t = \beta_1 f_{t-1} + (1-\beta_1)\nabla f(\vec{W}_{t-1})$$

$$s_t = \beta_2 s_{t-1} + (1-\beta_2)\nabla f(\vec{W}_{t-1})^2$$

Coefficient (10^{-8})

Second moment coefficient

- Adam with L2 regularization
  - However, L2 regularization is not so effective in Adam

$$f = \mathcal{L}_{CE} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{i,k}\log \hat{y}_{i,k} + \lambda \cdot \left\|\vec{W}\right\|^2$$

Regularization coefficient

- Alternative: AdamW[1] (Adam with weight decay)
  - Weight decay is equally effective in both SGD and Adam

$$\vec{W}_t = \vec{W}_{t-1} - \eta \frac{\hat{f}_t}{\sqrt{\hat{s}_t}+\varepsilon} - \eta \cdot \lambda \cdot \left\|\vec{W}\right\|$$

[1] Loshchilov, I. and Hutter, F., (2019). Decoupled Weight Decay Regularization.

Classification - Training setup

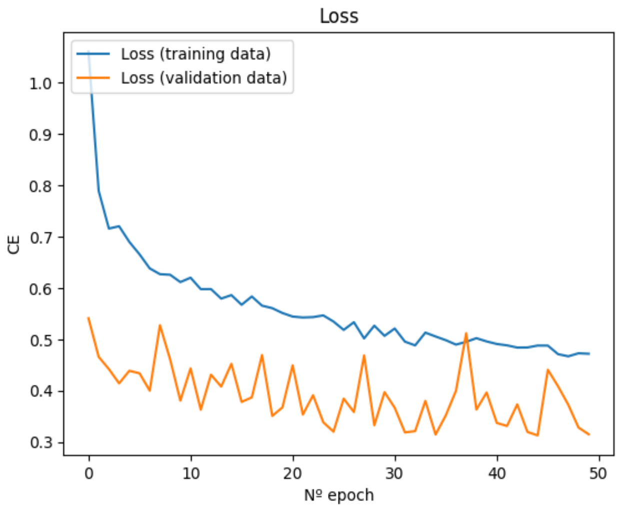
Parameter	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
Epochs	50			100		30
Loss	Categorical Crossentropy					
Metric	Categorical Accuracy					
Optimizer	Adam(LR = 10^{-3})	Adam(LR = 10^{-4})				AdamW(LR=10^{-4}, WD = $5 * 10^{-5}$)
Class_weights	{0:1. , 1:1., 2:1.}	{0:1. , 1:2., 2:1.}				{0:1. , 1:1., 2:1.}
Trainable	No	Trainable: 300 last layers				Whole trainable
Data Augment	rescale=1./255. rotation_range=10	rescale=1./255. rotation_range=10	rescale=1./255. rotation_range=15 width_shift=0.2 height_shift=0.2 channel_shift=0.2 zoom_range =0.2 shear_range=0.1 horizont_flip=True vertical_flip =True	rescale=1./255. rotation_range=15 width_shift=0.2 height_shift=0.2 channel_shift=0.3 zoom_range =0.2 shear_range=0.1 horizont_flip=True vertical_flip =True	rescale=1./255. rotation_range=25 width_shift=0.3 height_shift=0.2 channel_shift=0.4 zoom_range =0.2 shear_range=0.3 horizont_flip=True vertical_flip =True	rescale=1./255. rotation_range=15 width_shift=0.2 height_shift=0.2 channel_shift=0.2 zoom_range =0.2 shear_range=0.3 horizont_flip=True vertical_flip =True

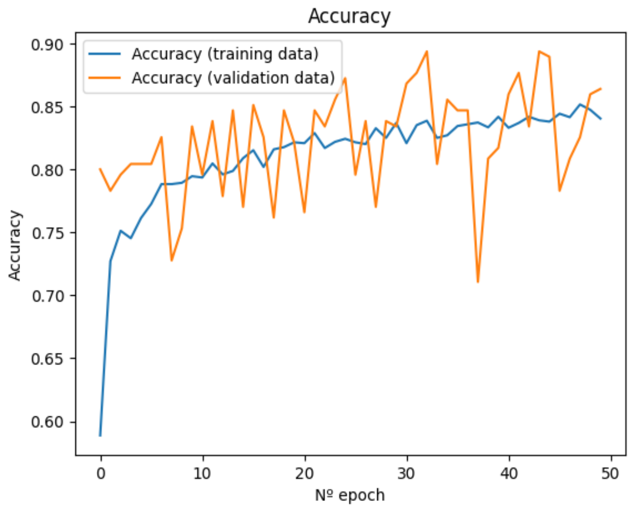
Classification - Training and test results

Training graphs

Test reports

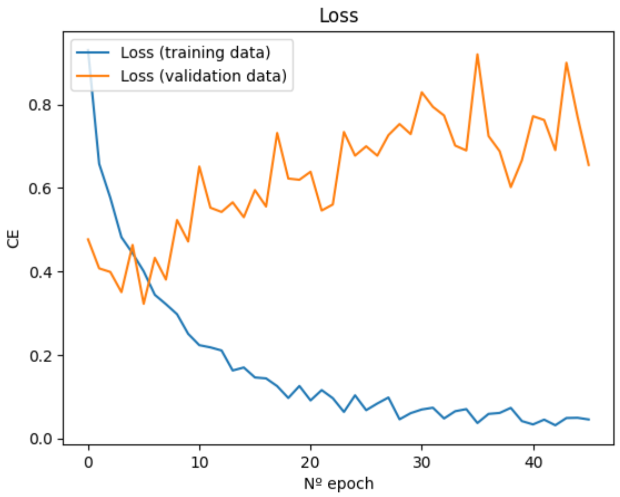
Stage 1

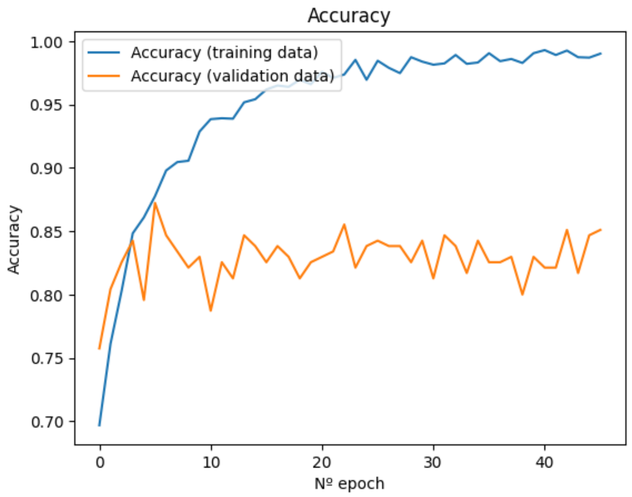




	precision	recall	f1-score	support
Extra-fine	0.97	0.88	0.92	441
Fine	0.71	0.80	0.75	490
Coarse	0.79	0.76	0.77	441
accuracy			0.81	1372
macro avg	0.82	0.81	0.81	1372
weighted avg	0.82	0.81	0.81	1372

Stage 2





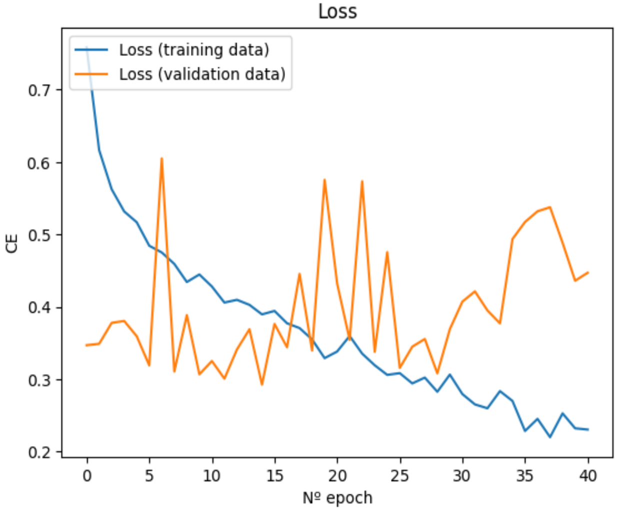
	precision	recall	f1-score	support
Extra-fine	0.96	0.85	0.90	441
Fine	0.73	0.81	0.77	490
Coarse	0.80	0.79	0.80	441
accuracy			0.82	1372
macro avg	0.83	0.82	0.82	1372
weighted avg	0.83	0.82	0.82	1372

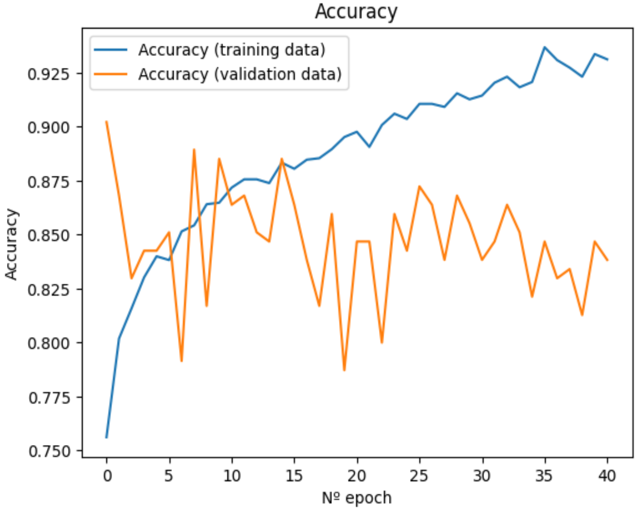
Classification - Training and test results

Training graphs

Test reports

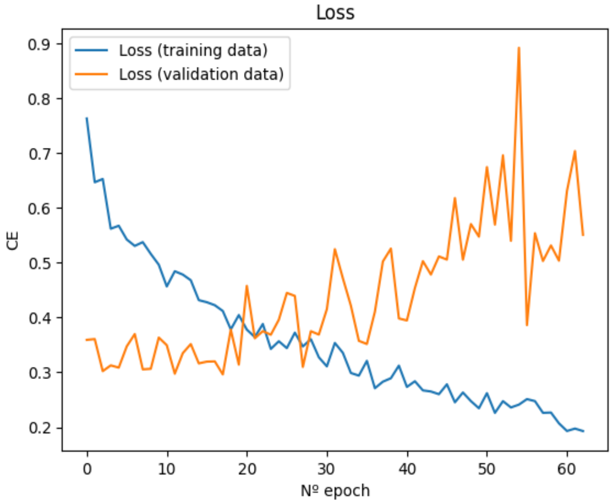
Stage 3

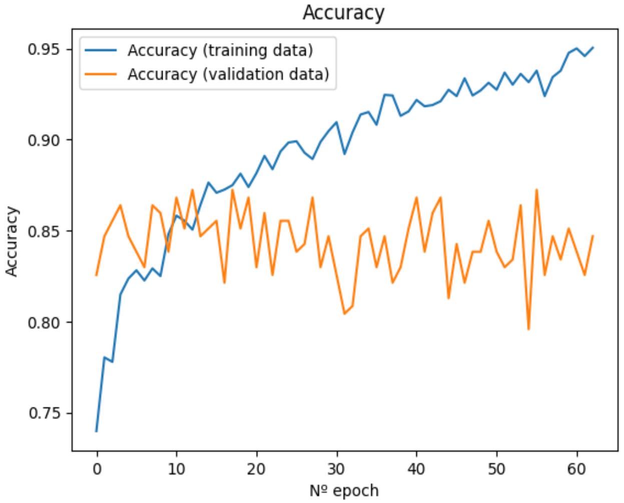




	precision	recall	f1-score	support
Extra-fine	0.98	0.86	0.91	441
Fine	0.79	0.75	0.77	490
Coarse	0.78	0.91	0.84	441
accuracy			0.84	1372
macro avg	0.85	0.84	0.84	1372
weighted avg	0.85	0.84	0.84	1372

Stage 4





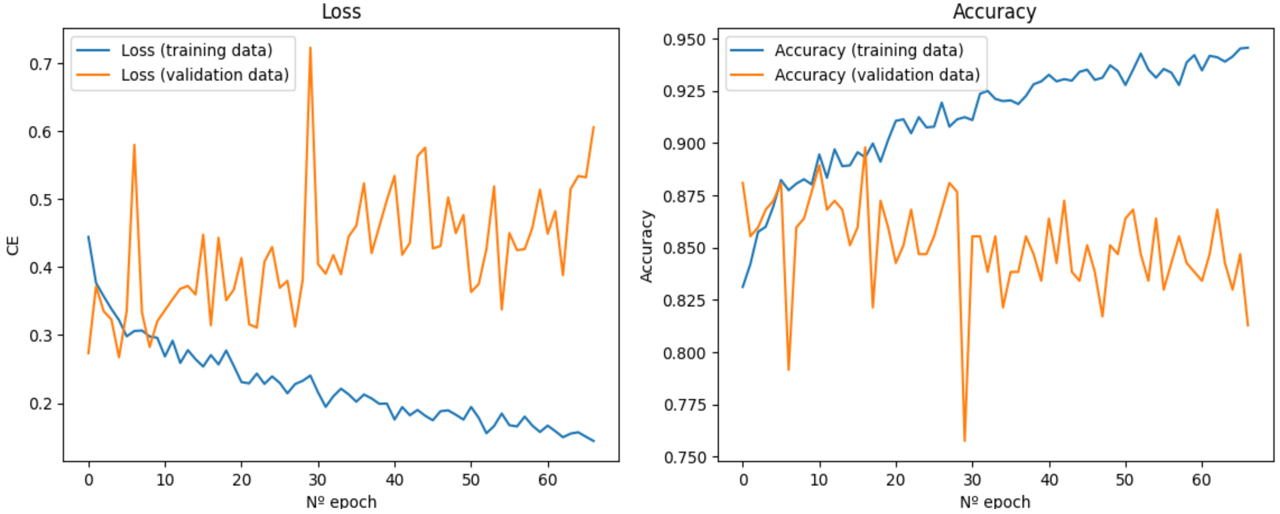
	precision	recall	f1-score	support
Extra-fine	0.95	0.88	0.92	441
Fine	0.79	0.75	0.77	490
Coarse	0.79	0.89	0.83	441
accuracy			0.84	1372
macro avg	0.84	0.84	0.84	1372
weighted avg	0.84	0.84	0.84	1372

Classification - Training and test results

Training graphs

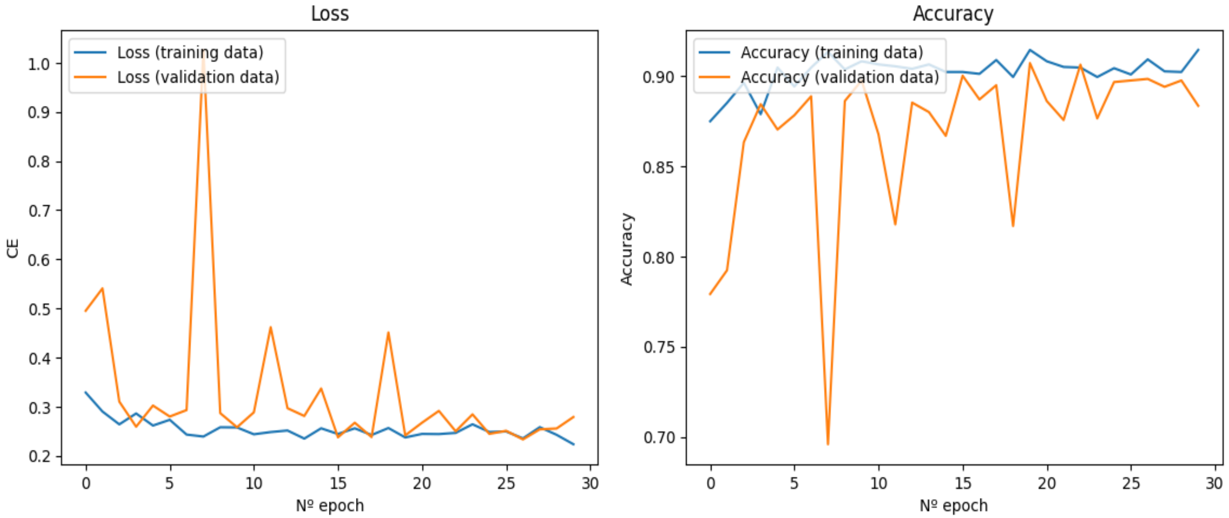
Test reports

Stage 5



	precision	recall	f1-score	support
Extra-fine	0.94	0.90	0.92	441
Fine	0.84	0.78	0.81	490
Coarse	0.83	0.92	0.87	441
accuracy			0.87	1372
macro avg	0.87	0.87	0.87	1372
weighted avg	0.87	0.87	0.87	1372

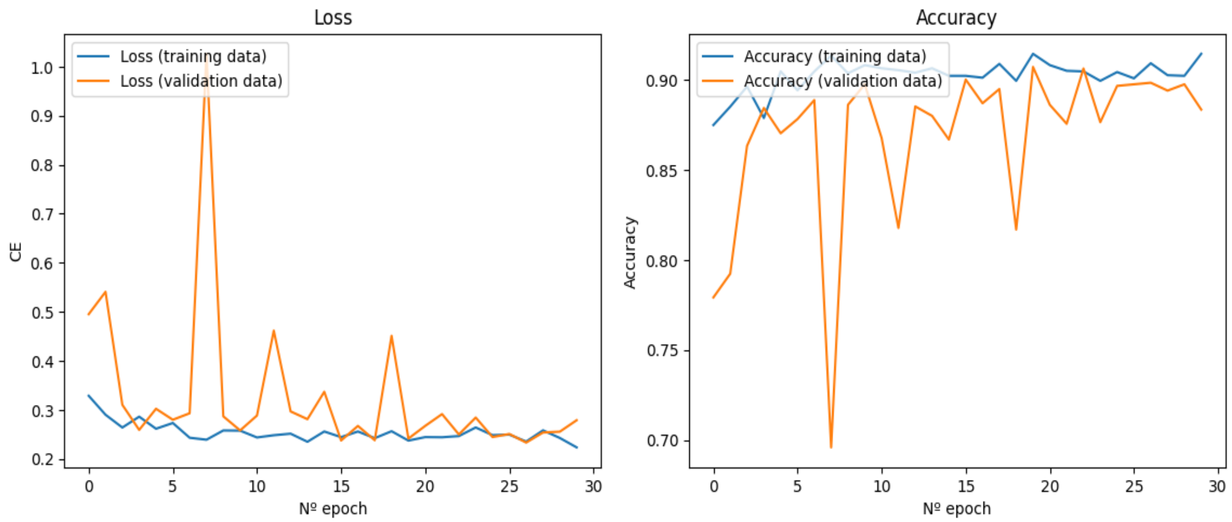
Stage 6



	precision	recall	f1-score	support
Extra-fine	0.97	0.96	0.96	441
Fine	0.88	0.91	0.90	490
Coarse	0.93	0.90	0.92	441
accuracy			0.92	1372
macro avg	0.93	0.92	0.93	1372
weighted avg	0.93	0.92	0.92	1372

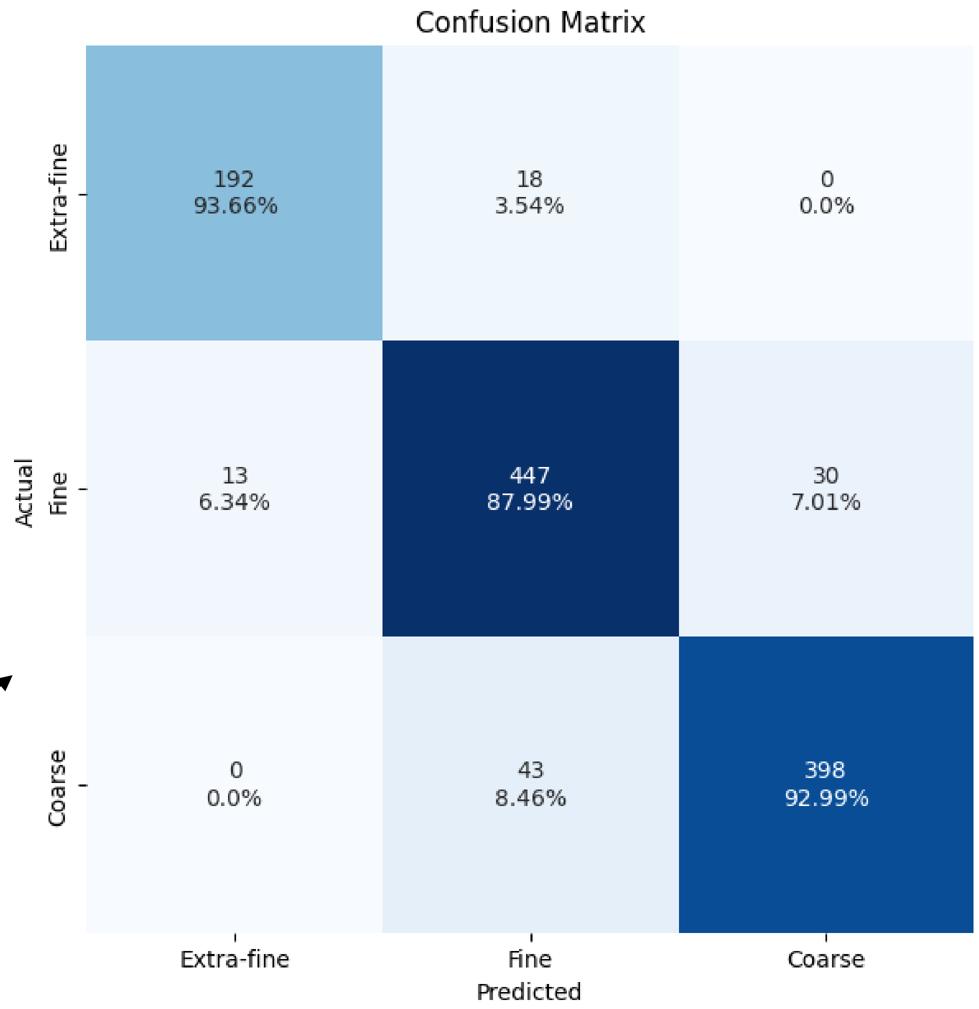
Classification - Training and test results (Stage 6)

Training graphs



	precision	recall	f1-score	support
Extra-fine	0.97	0.96	0.96	441
Fine	0.88	0.91	0.90	490
Coarse	0.93	0.90	0.92	441
accuracy			0.92	1372
macro avg	0.93	0.92	0.93	1372
weighted avg	0.93	0.92	0.92	1372

Test

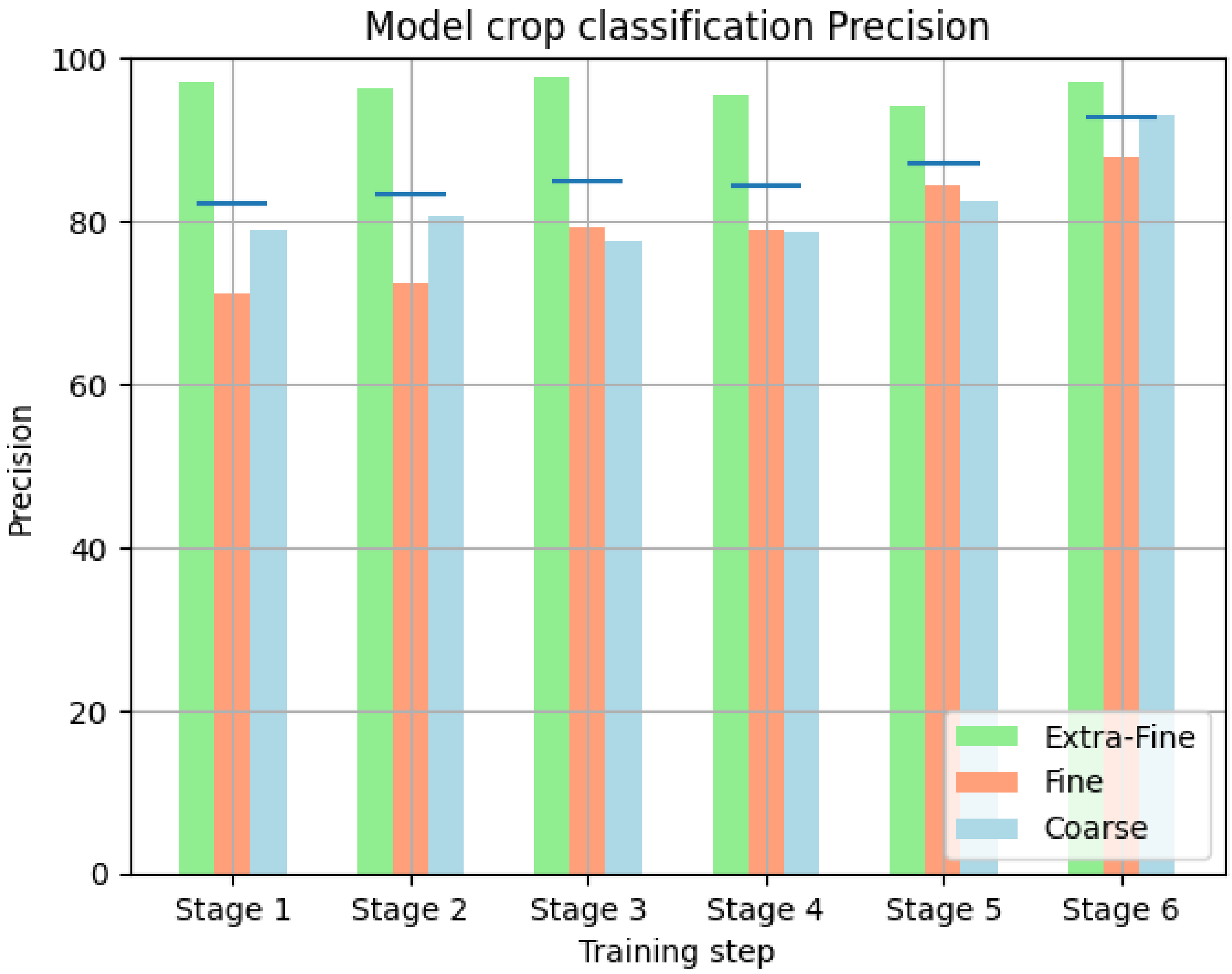


$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\_score = \frac{2 * Precision * Recall}{Precision + Recall}$$

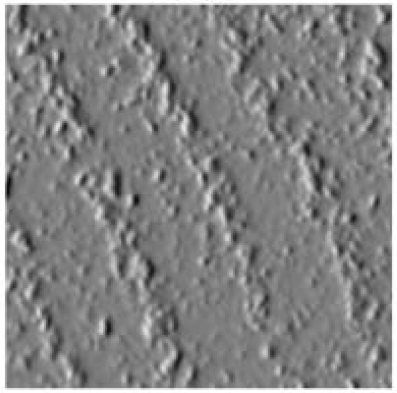
Classification - Test results comparison



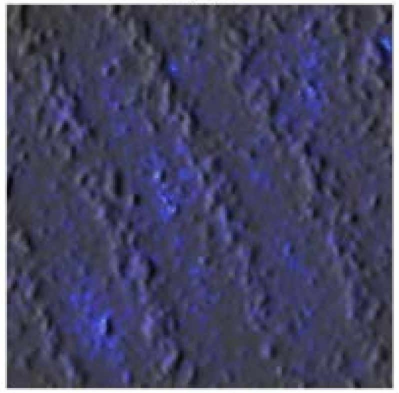
Classification - Visualizing results

- Saliency maps: pixels that have impact in the classification
  - Gradient of loss with respect to the input
  - Loss values changes with respect to small changes in the input

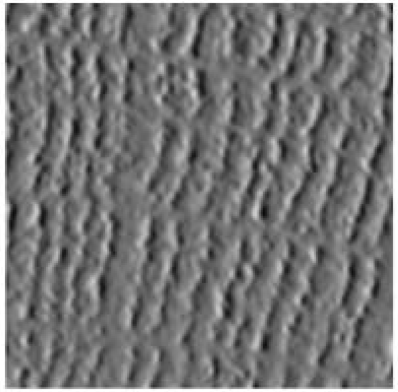
Coarse



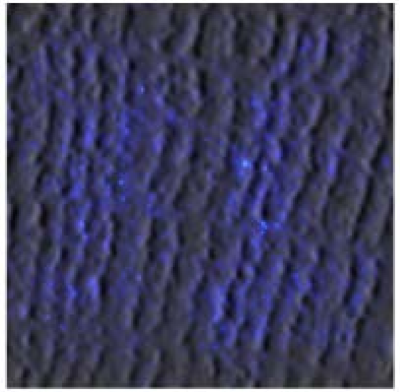
Coarse



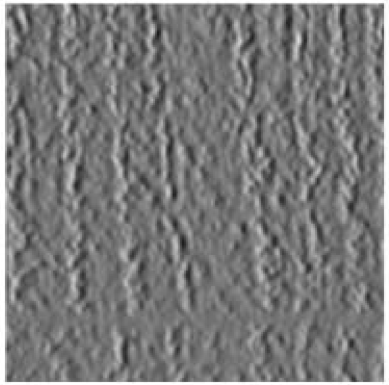
Extra-Fine



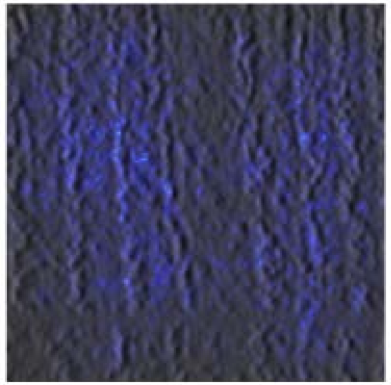
Extra-Fine



Fine

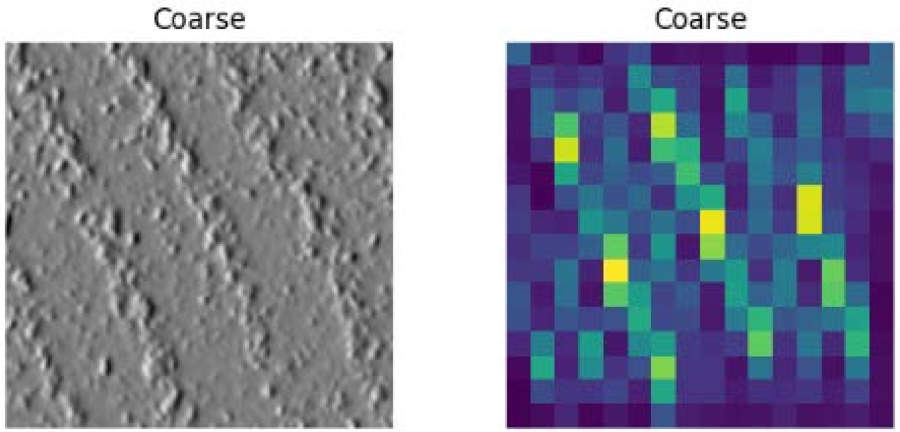


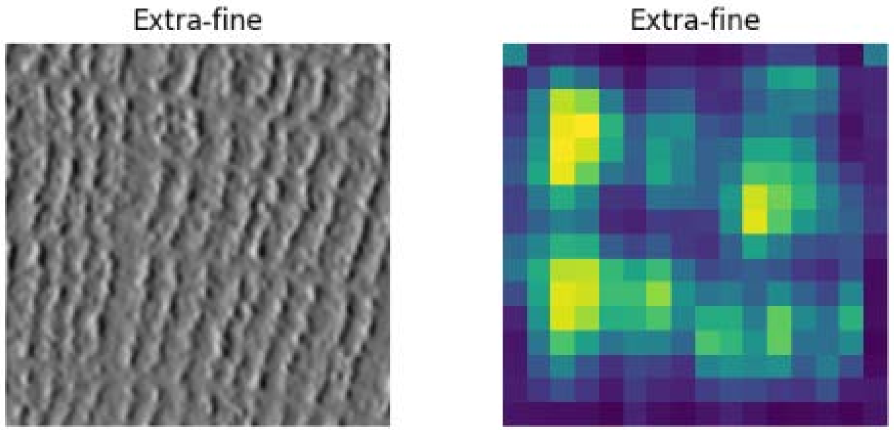
Fine

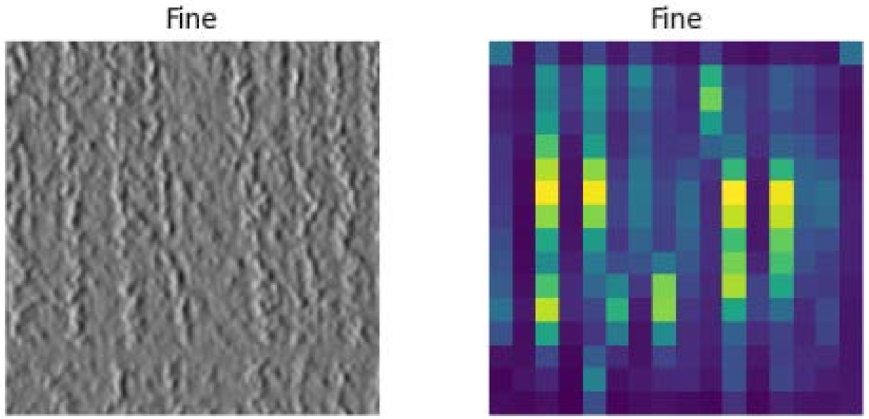


Classification - Visualizing results

- Layers visualization
  - Last conv layer of the model (1280 features maps of 16x16), and added all those maps

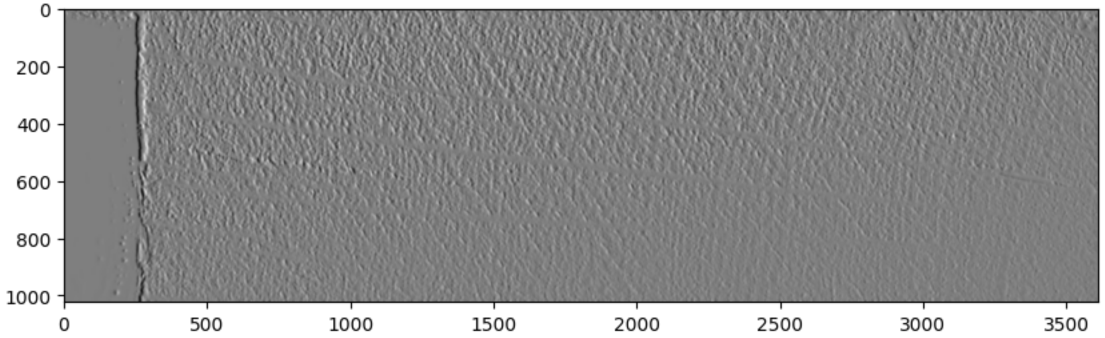


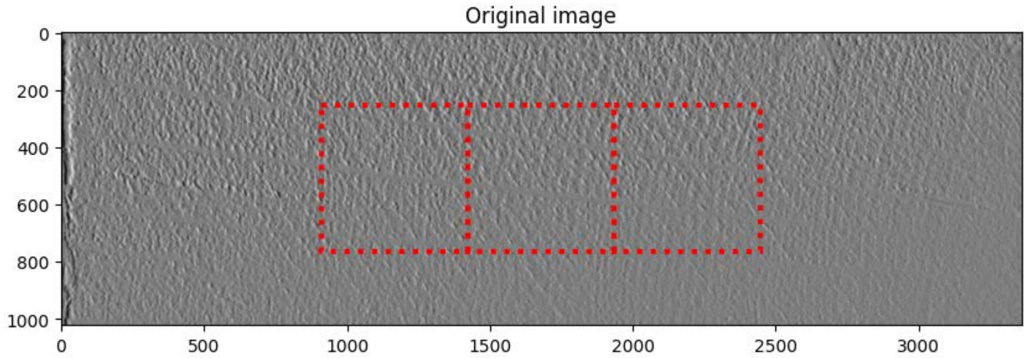


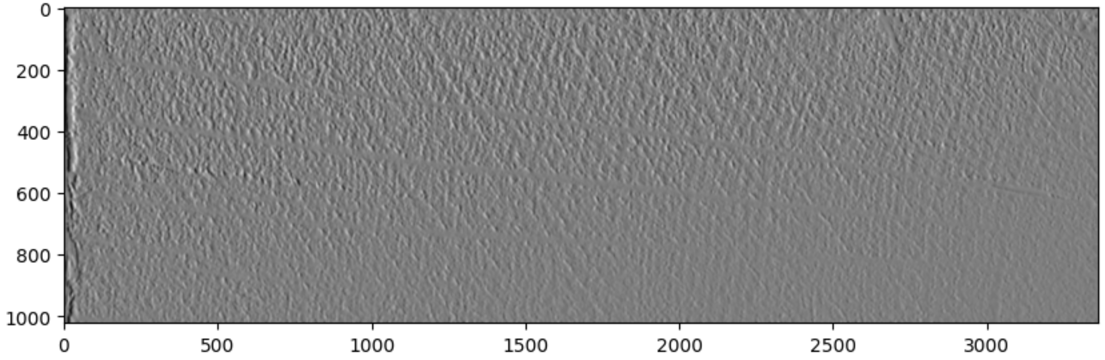


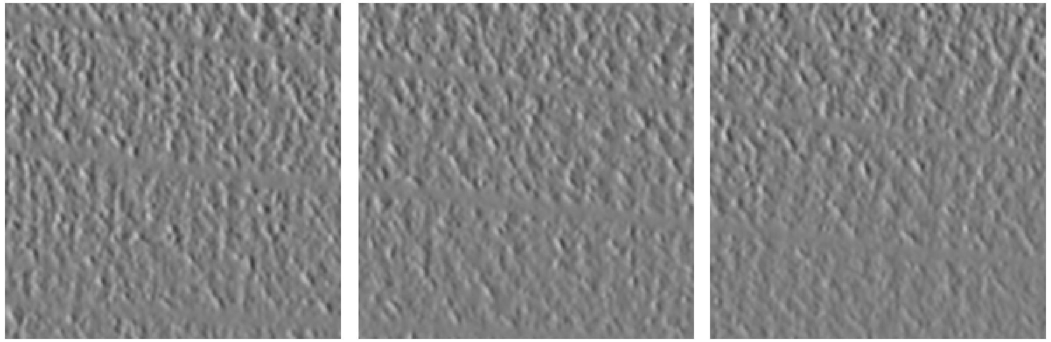
Voting system – Crops selection

- 3 central crops



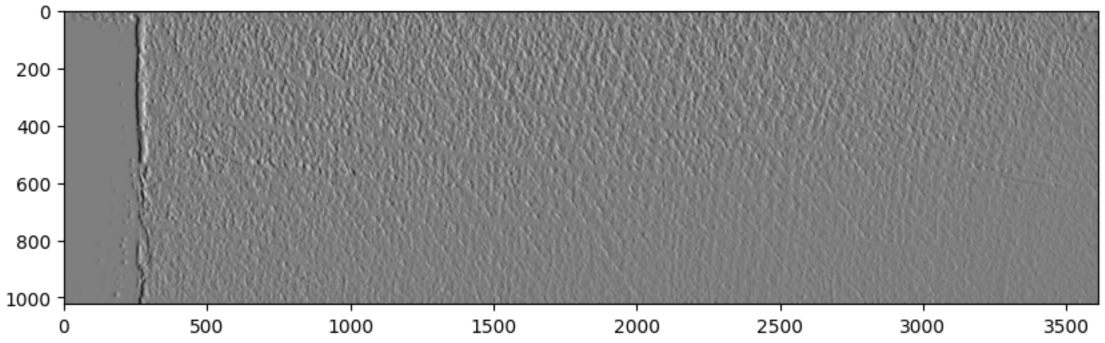


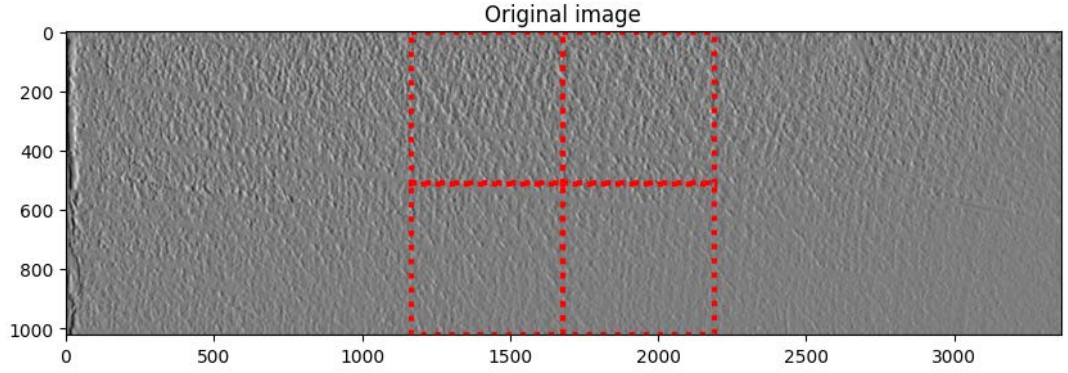


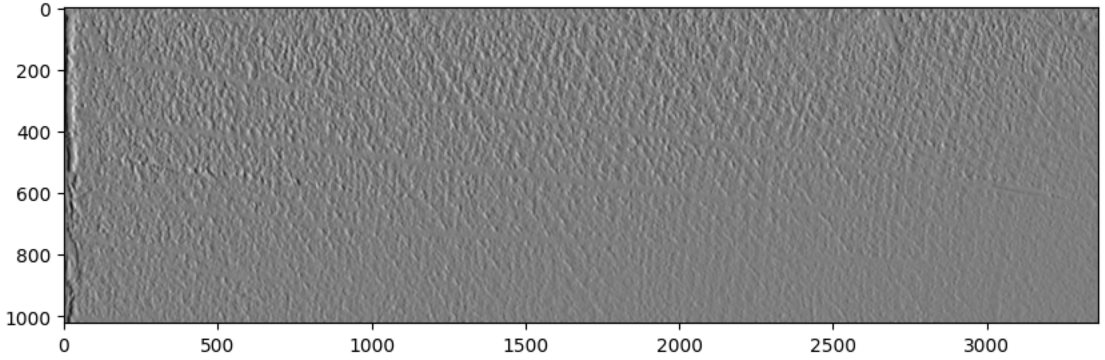


Voting system – Crops selection

- 4 central crops

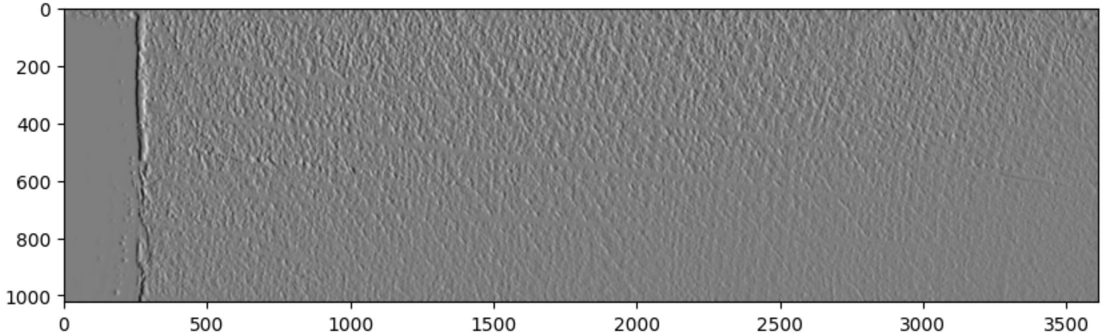


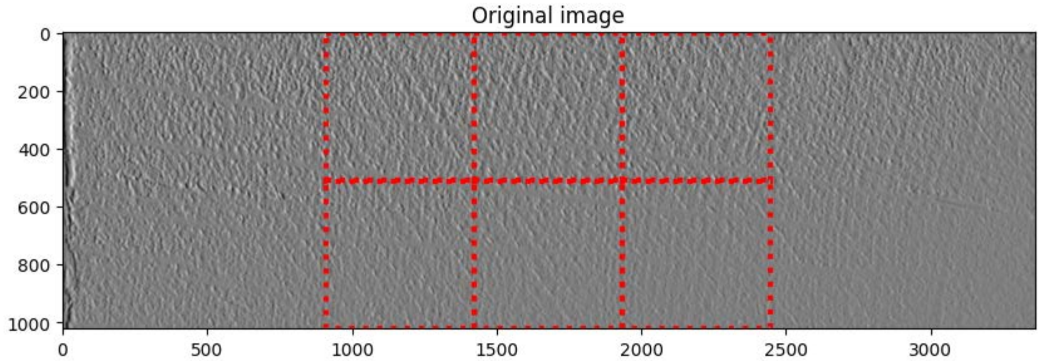


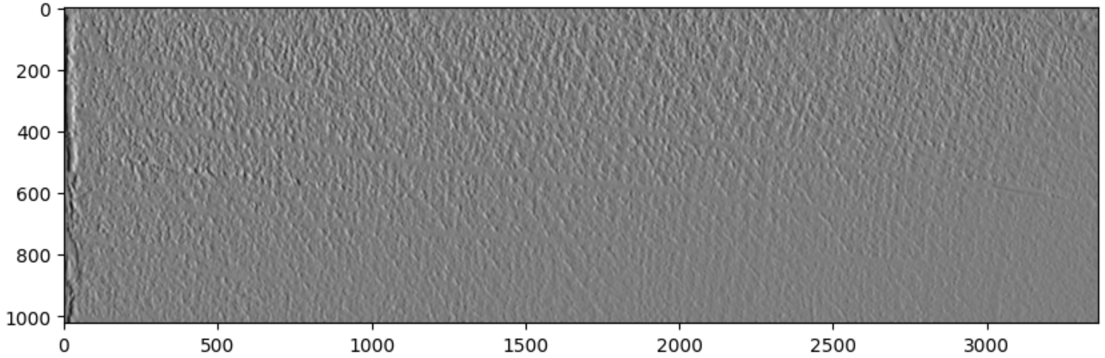


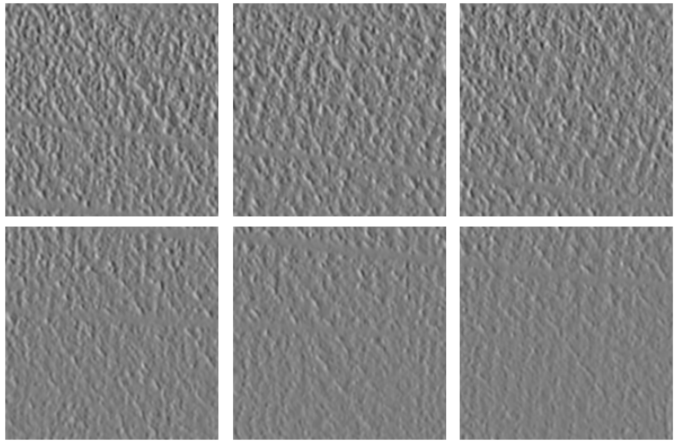
Voting system – Crops selection

- 6 central crops









Voting system - results

Staves for test

	Wood staves
Extra-Fine	852
Fine	403
Coarse	550

- Crops selection: 3, 4, 6
- Voting system:
  1. Sum of number of crops of each class, with threshold adjustment ([0,5..0,8])
     - In case of a tie, priority in this regard: [Extra-fine > Fine > Coarse]

Función	Acc para umbral 0.5	Acc para umbral 0.6	Acc para umbral 0.7	Acc para umbral 0.8
3 centrales	0.899	0.911	0.915	0.912
4 centrales	0.899	0.905	0.902	0.905
6 crops	0.927	0.930	0.936	0.956

  2. Sum of crop probs

Función	Acc
3 centrales	0.912
4 centrales	0.910
6 crops	0.935

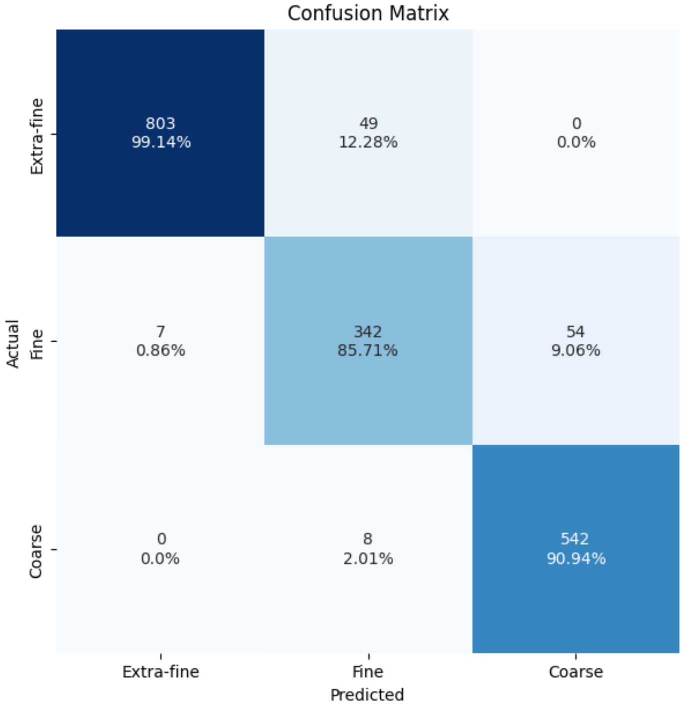
  3. Hybrid approach (6 crops, threshold 0.8): Accuracy=0.94
     - Taking a prediction $\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3]$

$$approch_h(\hat{y}) = \begin{cases} sum\_crop\_probs & if \quad \hat{y}_1 = \hat{y}_2 = \hat{y}_3 \\ sum\_number\_crops & if \quad \forall i,j, \quad \hat{y}_i \neq \hat{y}_j \; or \max(\hat{y}) > (sum(\hat{y}) - \max(\hat{y})), \qquad i \neq j \; and \; i,j = 1,2,3 \end{cases}$$

Voting system - results

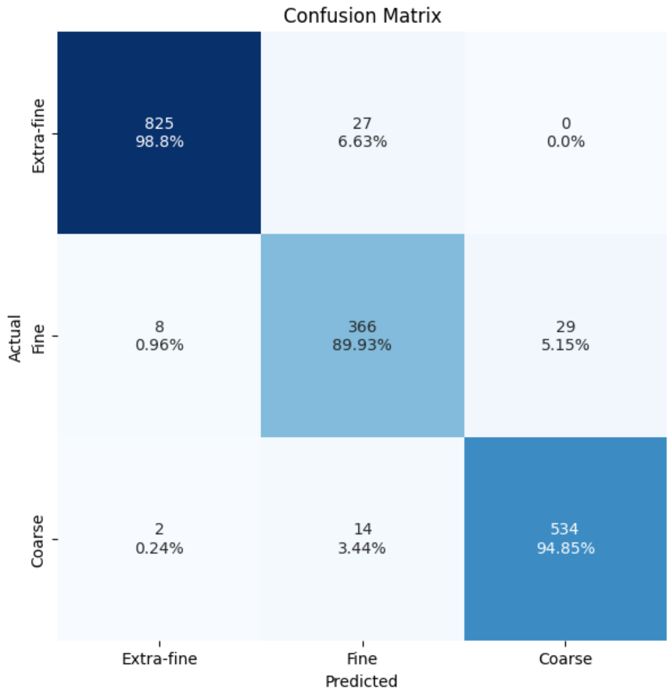
Sum of crop probs

	precision	recall	f1-score	support
Extra-fine	0.99	0.94	0.97	852
Fine	0.86	0.85	0.85	403
Coarse	0.91	0.99	0.95	550
accuracy			0.93	1805
macro avg	0.92	0.93	0.92	1805
weighted avg	0.94	0.93	0.93	1805



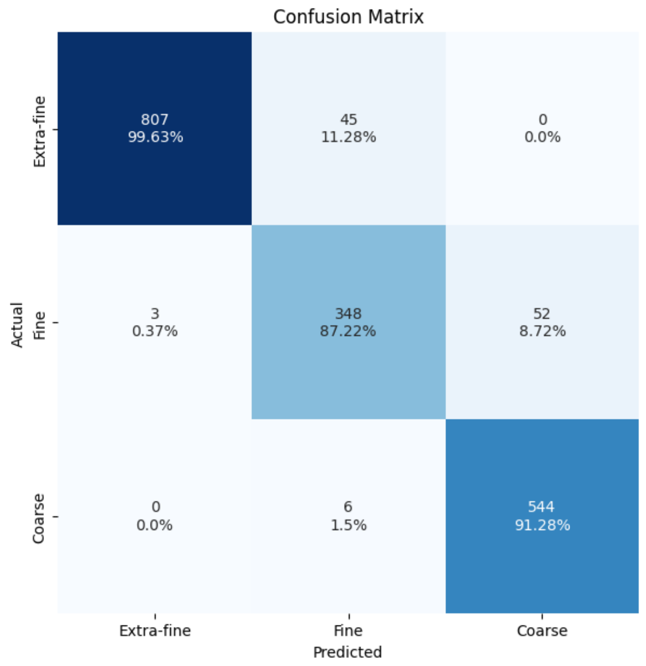
Sum of number of crops of each class, threshold 0,8

	precision	recall	f1-score	support
Extra-fine	0.99	0.97	0.98	852
Fine	0.90	0.91	0.90	403
Coarse	0.95	0.97	0.96	550
accuracy			0.96	1805
macro avg	0.95	0.95	0.95	1805
weighted avg	0.96	0.96	0.96	1805

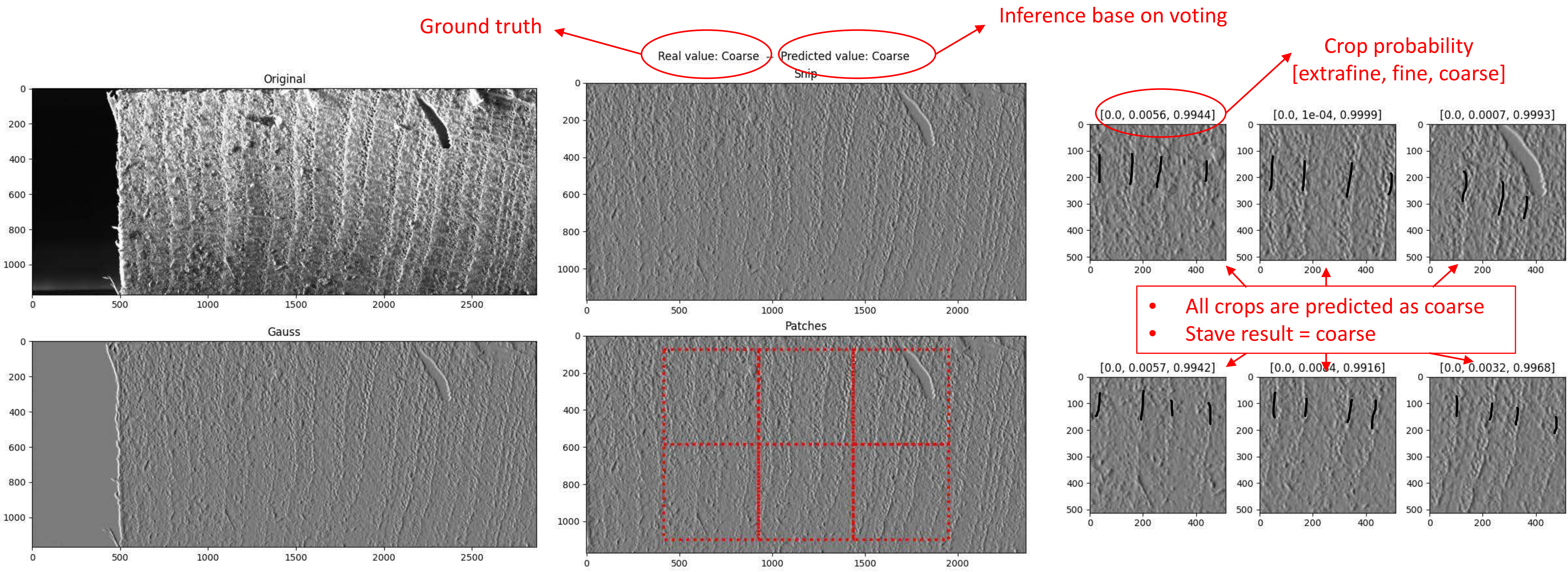


Hybrid approach, threshold 0,8

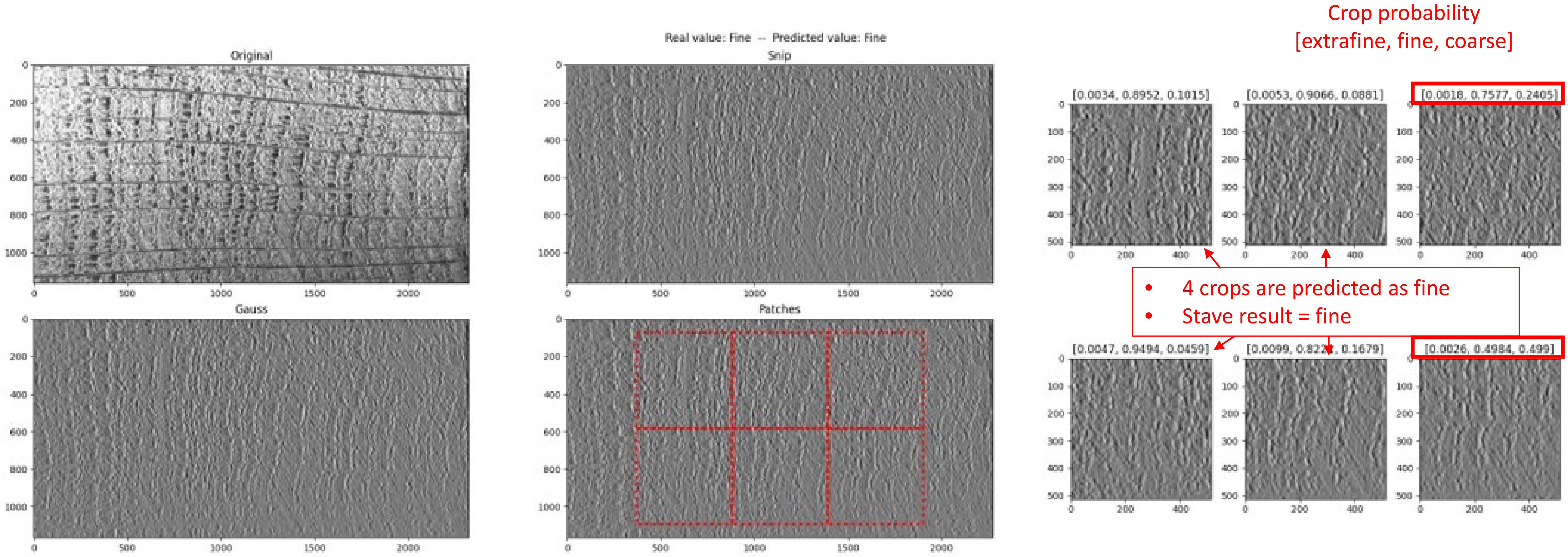
	precision	recall	f1-score	support
Extra-fine	1.00	0.95	0.97	852
Fine	0.87	0.86	0.87	403
Coarse	0.91	0.99	0.95	550
accuracy			0.94	1805
macro avg	0.93	0.93	0.93	1805
weighted avg	0.94	0.94	0.94	1805



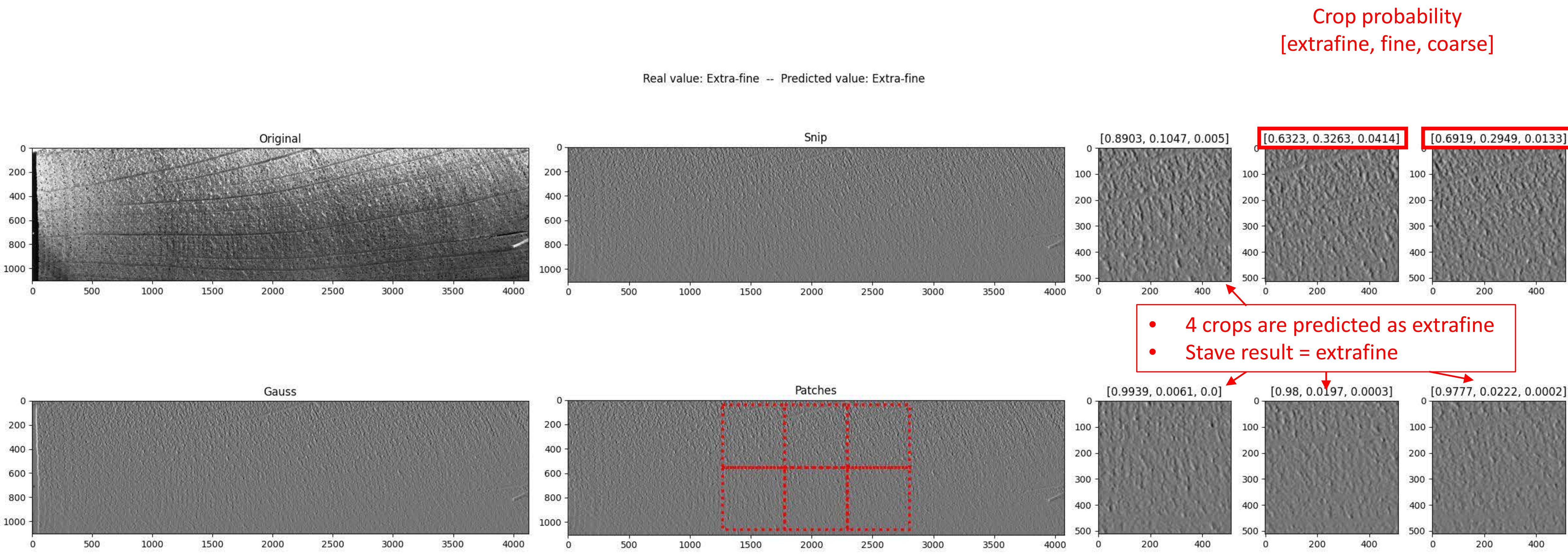
Voting system - Visualizing results



Voting system - Visualizing results



Voting system - Visualizing results



Index

1. Introduction and objectives
2. Related work
3. Algorithms
    1. Dimensional control
    2. Classification
        1. Why not classical approach?
        2. DL – preprocessing, labelling, data augmentation
        3. DL – architecture, training, test
        4. DL – voting system
4. Conclusions

Conclusions/learned lessons

- A more complex problem than it seems at first

- Dimensional control under tolerance
  - 716 samples, 0,28% error
  - Despite the poor optical set-up

- Classification
  - Related works focused on “clean” samples
  - Good divide-and-conquer approach for classification
    - Data and crops augmentation
    - Crops classification
    - Stave classification by means of voting system
  - Accuracies above 90%
    - Crops classification: test samples=1.372, acc=0,91
    - Stave classification: test samples=1.805, acc=0,94
  - Better than humans?

Solving wood heterogeneous texture classification: A deep learning approach with cropping data augmentation

Frank A. Ricardo[1], Martxel Eizaguirre[1], Desmond Moru[2], Diego Borro[1,3]

[1]CEIT-Basque Research and Technology Alliance (BRTA) and Tecnun (University of Navarra)
[2]School of Science and Technology (SST), Pan-Atlantic University (PAU)
[3]Institute of Data Science and Artificial Intelligence (DATAI), University of Navarra

Lecturer: Diego Borro
Computer Science PhD
Vision and Robotics group at CEIT
dborro@ceit.es