Image Data Analysis - Project Report

Claudio Metelli

A.Y. 2024/2025

Abstract

In this work I will present the problem approach, the design, the implementation and evaluation of a CNN architecture which aims to recognize different surface patterns from lake Iseo in order to have a better understanding of the water hydrodynamics.

1 Introduction

The objective of the analysis is to study the hydrodynamics of Lake Iseo, which allows us to understand the movement of water in the basin. This is important because temperature, nutrients and oxygen diffuse into the lake along with the water. Therefore, knowing the hydrodynamics of the lake also helps to assess the quality of the lake ecosystem.

In order to reach this goal, I will provide 2 convolutional neural network (CNN) systems that classifies images of the lake with eight possible labels.

The eight possible labels are:

1. Clean Front (fronte_freddo)
2. Turbid Front (fronte_torbidi_in_moto)
3. Turbid Stain (macchie_circolari_torbide)
4. Vortex (vortici_slick)
5. Wooden Material (tronchi)
6. Turbid Lines (linee_torbide)
7. Turbid Discharge (immissione_torbidità)
8. Non Event (non_eventi)

This eight possible events differs between each other in some details; for example wooden material or turbid lines are similar phenomena where the floating material concentrates along packs or lines. Also, non event means that the photo is not easily classifiable.

In the next section (2) we will explore the dataset, in order to understand what type of data are we dealing with. Then in section 3 the solution architectures are presented; they are based on CNNs which represent the state of the art in image classification, in particular VGG19 [1] and ResNet50 [2]. After the training description, in section 4, the architectures training result are analyzed using different metrics and visualizing data. Finally, in section 5 there are conclusions.

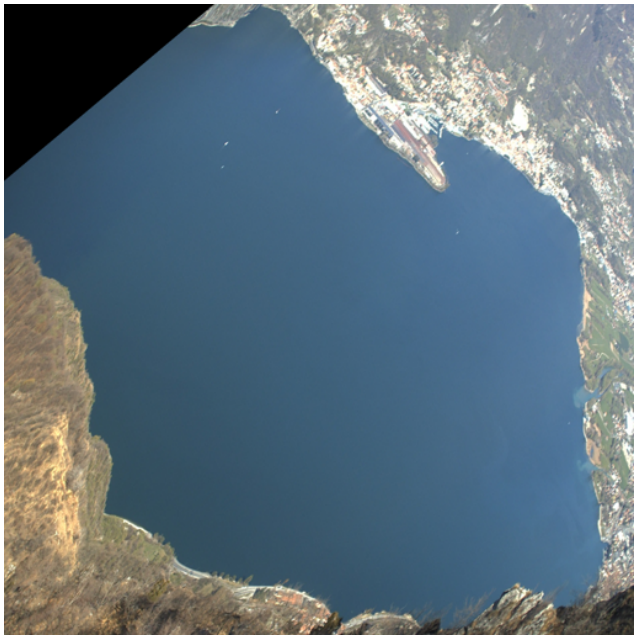


Figure 1: Example of image in the dataset.

2 Dataset Analysis

Our dataset is composed of 3600 RGB images, that have already been preprocessed, and every preprocessed image has a resolution of 512×512. Figure 1 is an example of image in the dataset.

The eight possible labels are not equally splitted: in fact we can see from figure 2 that some labels are more frequent (for example turbid front - fronti torbidi in moto), while others are less fre-

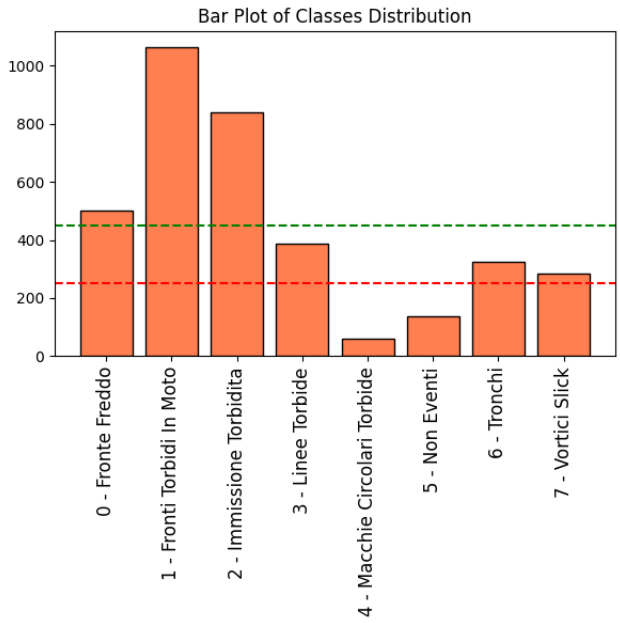


Figure 2: Label distribution in the dataset; green line represents the mean of the images examples per class (450), while red line (250) represents a good number of minimum examples per class.

quent (for example turbid stain - macchie circolari torbide).

In the image we can also see the mean of the images examples per class (450), represented by the green line, while the red line represents what should be a good number of examples for each class (250), chosen arbitrarily. Since the unbalancing given by the classes distribution could represent a problem during the training phase, I have chosen to augment the number of example in classes turbid stain and non events to 250. In this way the classes distribution becomes more regular.

Since we will need to implement a CNN, we will have to apply some transformations to the images in order to make them suitable as input for these CNNs. Also, we need to split our dataset in three different parts: training set, validation set and test set. The chosen distribution is 70% for training set, 15% for validation and 15% for testing.

Since both ResNet and VGG architectures are trained on ImageNet dataset and are structured to use 224×224 RGB images, we will need to resize them. To ensure that the system better learns the features of the different labels, data augmentation was applied, specifically:

- random rotation with 10% factor,
- random translation with both height and width factor of 5%,
- random flip

Finally, data is prepared using ad hoc methods for VGG and ResNet: images are converted from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset, without scaling.

For output, we transform each label in a one-hot encoded representation, since the output of the CNN will be a probability distribution of the possible classes.

Finally, data is batched with a batch size of 32 and prefetched in order to be faster during training phase. In figure 3 we can see a general schema of preprocessing for both images and labels.

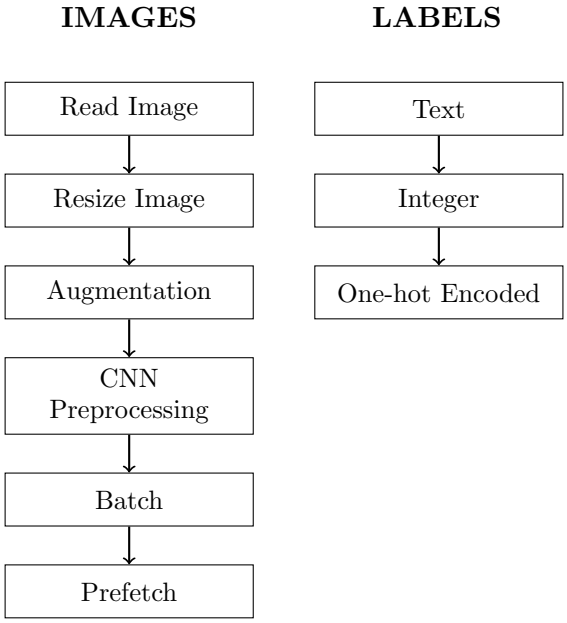


Figure 3: Preprocessing schema for images and labels.

3 Solution Architectures

In order to provide an accurate and robust solution, the chosen approach was to use well known and tested CNN architectures and modify them. The choice fell on VGG19 and ResNet50 as already described, excluding their top level and maintaining the base convolutional layers, and then build a series of ad hoc layers for our problem on top of the previous ones.

3.1 Base Models Overview

VGG19 and ResNet50 are the chosen base model. But why? Well, they have been two of the most used and well known CNN architectures in the last years.

VGG19, introduced by the Visual Geometry Group at Oxford University in 2014, has remained widely used due to their simplicity and effectiveness. This model is composed by 19 layers (16 convolutional layers + 3 fully connected layers), and has 3 more convolutional layer in order to capture more complex features respect to the smaller model VGG16.

ResNet50, on the other hand, is a deep convolutional neural network architecture that was developed by Microsoft Research in 2015. The primary problem ResNet solved was the degradation problem in deep neural networks. As networks become deeper, their accuracy saturates and then degrades rapidly.

Residual blocks are the key feature of the architecture: they are used to allow the network to learn deeper architectures without suffering from the problem of vanishing gradients. This block consists of classical convolution blocks, followed by a skip connection: this allows the unaltered input to be added directly to the output of the convolutional layers. This bypass connection ensures that essential information from earlier layers is preserved and propagated through the network, even if the convolutional layers struggle to learn additional features in that specific block.

With this addiction, network can use a lot more layers and learn more features, bringing potentially more accuracy for difficult image classification tasks.

3.2 Architectures Description

Based on the previously described networks, four models are built. A description follows, and the complete schema is reported in table 1.

VGG19_1: first model is based on VGG19. The upper part of the network consists of a flatten layer, which flattens all the extracted features, measuring $7 \times 7 \times 512$, so that two consecutive dense layers can then be added. The two layers in question have dimensions of 512 and 256, respectively, with a ReLU activation function, so that more complex functions can be encoded, and a layer with a dropout of 0.2 is added for each of them. Finally the last layer is an 8 elements layer with a softmax activation function in order to obtain a probability distribution for the labels. During training phase a learning rate of $1 \cdot 10^{-4}$ is used.

VGG19_2: second model is also based on VGG19. This is built in the same way as VGG19_1 but a different learning rate is used during training phase: $5 \cdot 10^{-5}$. Also, this model is fine tuned in a different way (see 3.3).

ResNet50_flat: this model is based on ResNet50. The upper part of the network is the same of VGG_1 and VGG_2. Differently from VGG based models, ResNet output is $7 \times 7 \times 2048$, using four times as many channels as previous models. This means that between flattening layer and the first dense layer there is four times as many weights. During training phase, learning rate is set to $5 \cdot 10^{-5}$.

ResNet50_avg: this model is based on ResNet50. The upper part is inspired from the original ResNet model: a global average pooling layer is added and flattened. Then there are two dense (dimension 1024 and 512) layers, with ReLU activation and 0.2 dropout, and the final layer for prediction, an 8 dimension vector with softmax activation function. The main difference is the global average pooling, which is fundamental for efficiency: using global average pooling before the first dense layer we try to preserve extracted features but reducing spatial dimension from $7 \times 7 \times 2048$ to $1 \times 1 \times 2048$, or simply 2048, the total number of weights used for the first dense layer is reduced to $(2048 + 1) \times 1024$, which is about 1/50 respect to ResNet50_flat. During training phase, learning rate is still set to $5 \cdot 10^{-5}$.

Also, all models shares some common characteristics: the loss function is categorical cross-entropy for all models, due to the nature of the problem, the chosen metrics for evaluation is accuracy, and the chosen optimizer is Adam.

Finally, during training phase, an early stopping call is used: we check the obtained loss on validation set, with patience of 5 epochs; in this way, also if we set 100 epochs for training, we usually do not reach this number of epochs. At the end of each epoch, also, weights are saved, in order to have optimal weights after each training and reuse them when needed without training again.

Model	Layer Type	Output Shape	Total Param #	Activation
VGG19_1 and **VGG19_2**	Input	(224, 224, 3)	0	—
	VGG19 (Base)	(7, 7, 512)	20 024 384	—
	Flatten	(25088)	0	—
	Dense	(512)	12 845 568	ReLU
	Dropout	(512)	0	—
	Dense	(256)	131 328	ReLU
	Dropout	(256)	0	—
	Dense (Pred)	(8)	2056	Softmax
ResNet50_flat	Input	(224, 224, 3)	0	—
	ResNet50 (Base)	(7, 7, 2048)	23 587 712	—
	Flatten	(100352)	0	—
	Dense	(512)	51 380 736	ReLU
	Dropout	(512)	0	—
	Dense	(256)	131 328	ReLU
	Dropout	(256)	0	—
	Dense (Pred)	(None, 8)	2056	Softmax
ResNet50_avg	Input	(224, 224, 3)	0	—
	ResNet50 (Base)	(7, 7, 2048)	23 587 712	—
	Global Average Pooling 2D	(2048)	0	—
	Flatten	(2048)	0	—
	Dense	(1024)	2 098 176	ReLU
	Dropout	(1024)	0	—
	Dense	(512)	524 800	ReLU
	Dropout	(512)	0	—
	Dense (Pred)	(8)	2056	Softmax

Table 1: Base structure of CNN based on VGG19 and ResNet50.

3.3 Fine Tuning

With base models already available, their weights (the ones used for feature extraction) have been frozen, in order to train only the top layers, which are the ones used for classifying. In this way the feature extracted from the base model can be used as input for classification phase.

Of course, this is a starting point, since we could need different features from the ones that are offered by the base model. After training the top layers, the fine tuning phase comes.

In this phase, the top layers of the base model are unfrozen, in order to find new features and reach better classification accuracy.

- In **VGG19_1** the unfrozen layers are 5, which correspond to the last convolutional block of VGG19.
- In **VGG19_2** the unfrozen layers are 10, which correspond to the last two convolutional blocks of VGG19.
- In **ResNet50_flat** and in **ResNet50_avg** the unfrozen layers are 94, which correspond to the last two convolutional blocks of ResNet50, counting also layers like ReLU activation layers that has zero parameters.

Adding this type of training after the first one we try to improve the total accuracy of the models. For each fine tuning training learning rate is set to $5 \cdot 10^{-5}$.

4 Results and Evaluation

After each training phase of each model, and after each fine tuning phase of each model, accuracy and loss have been tested on the test set, in order to obtain evaluation on data the model had never seen before.

All results are reported in table 2. We can notice from this table that the maximum accuracy has been reached by ResNet50_flat and ResNet50_avg and it is about 77.5%.

Training before Fine Tuning		
Model	**Loss**	**Accuracy**
VGG19_1	0.8696	68.26%
VGG19_2	0.7653	68.77%
ResNet50_flat	0.7152	70.14%
ResNet50_avg	1.0637	61.09%
Training after Fine Tuning		
Model	**Loss**	**Accuracy**
VGG19_1	0.6928	74.57%
VGG19_2	0.7714	72.53%
ResNet50_flat	0.5705	77.47%
ResNet50_avg	0.5756	77.65%

Table 2: Training performance before fine tuning and after fine tuning, for every model.

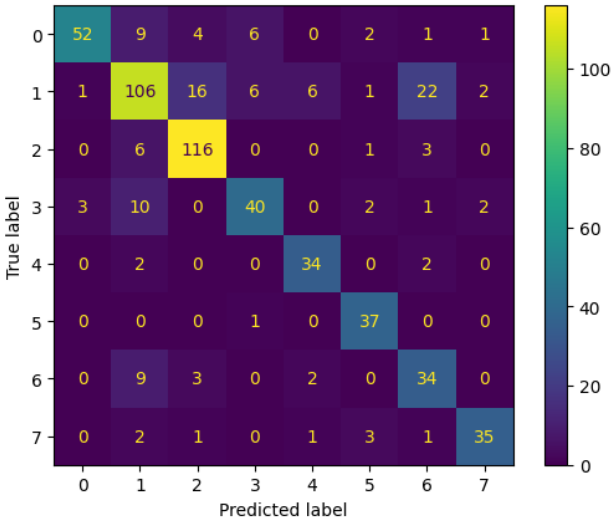


Figure 4: Confusion matrix for model ResNet_flat. Legend is the following:
- 0: Clean Front
- 1: Turbid Front
- 2: Turbid Discharge
- 3: Turbid Lines
- 4: Turbid Stain
- 5: Non Event
- 6: Wooden Material
- 7: Vortex

In this section I will analyze the results of ResNet50_flat, which are the bests along all models, with ResNet50_avg. As expected, the higher accuracy on test set is provided by ResNet-based networks. This result is expected due to some considerations: ResNet model is a more recent framework for image classification, compared to VGG; also, residual blocks have more complex structure and dimension, so they can understand more complex features, like the ones of water hydrodynamics.

Then, flat and average version of ResNet-based models have approximately the same loss and accuracy value after fine tuning, while having different performances in the first training phase. In fact, average pooling is sensitively less expressive than flat version if weights are frozen, since it uses a lot less weights. When lasts layers are unfrozen, they can adjust their weights in order to give more relevant features to the full connected layers and obtain better results.

I chose to analyze ResNet50_flat since it has a little lower loss, while the two models differ in accuracy for just one image across the whole dataset. Anyway, ResNet50_avg is a good model and performance are very similar to flat version.

Despite this, the difference in accuracy and loss across all models is not very high and it seems that none of the models are able to achieve excellent accuracy, regardless base model, hyperparameters or data augmentation.

4.1 Error Analysis with Quantitative metrics

In order to understand why the model is not able to clearly recognize classes, we use a confusion matrix. A confusion matrix, also known as error matrix, is a matrix where each row represents the instances in an actual class while each column represents the instances in a predicted class. The diagonal of the matrix therefore represents all instances that are correctly predicted.

Along with other metrics, like precision and recall, it is used to understand what type of error the network make. Confusion Matrix (based on test set) is presented in figure 4.

Observing confusion matrix, the errors are fairly uniform, but there are some exceptions.

- Class 6 (wooden material) is often predicted wrong, and the assigned class in this case is often class 1 (turbid front).
- Class 1 present bad classification: sometimes it is predicted in a wrong way, sometimes it is not recognized by the network.
- The system often fails in recognizing class 0

(clean front).

These observations can be verified using precise indices, like precision and recall. Precision is the fraction of examples classified as positive that are actually positive, while recall is the fraction of examples correctly classified among those actually belonging to the class.

Precision for class i is:

$$precision_i = \frac{TP_i}{TP_i + FP_i},$$

with TP_i the total number of true positive classified examples for class i (that can be found also in confusion matrix with index i on the diagonal) and FP_i the total number of false positive for the same class (the sum of one column in the confusion matrix, excluding the element on the diagonal).

If we also introduce the total number of false negative FN_i (the sum of one row in the confusion matrix, excluding the element on the diagonal), the recall value can be found:

$$recall_i = \frac{TP_i}{TP_i + FN_i}.$$

Class	Precision	Recall
0 - Clean Front	92.86%	69.33%
1 - Turbid Front	73.61%	66.25%
2 - Turbid Discharge	82.86%	92.06%
3 - Turbid Lines	75.47%	68.97%
4 - Turbid Stain	79.07%	89.47%
5 - Non Event	80.43%	97.37%
6 - Wooden Material	53.12%	70.83%
7 - Vortex	87.50%	81.40%

Table 3: Precision and Recall for each class.

We can observe from table 3 that the previous observations are confirmed: class 1 has a bad precision and recall, wooden material is often confused with turbid front, so it has a bad precision, and clean front has pretty low recall.

4.2 Qualitative Error Analysis

In this section I will report some of the wrong classified images, in order to understand why they have been classified in the wrong way.

As one of the previously highlighted problems was the prediction with wooden material class, I have observed some of them, and a subset is reported in figures 5 and 6.

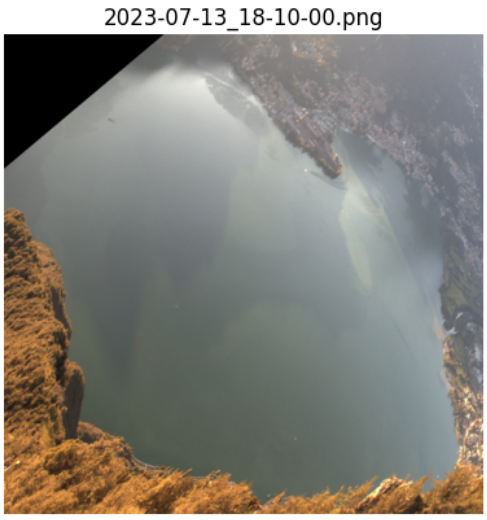


Figure 5: Image classified as wooden material with true class turbid front.

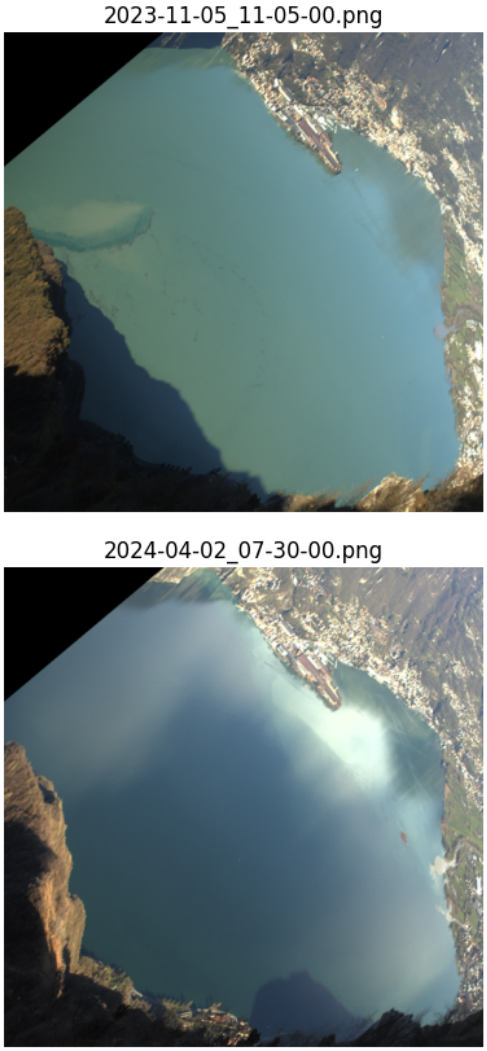


Figure 6: Example of two image with wrong classification.
Prediction is wooden material for each image.
Class of the upper image is turbid front.
Class of the lower image is turbid discharge.

Image 5 does not present any wooden material, so the model is wrong. On the other side, in image 6 we can observe two figures that presents wooden material. Since they represent correctly the classes turbid front and turbid discharge, but they also present wooden materials, they have been classified as such. Furthermore, to untrained eyes, such as those of people who are not experts in hydrodynamics, the presence of wooden material is much more immediate and recognizable due to the brown color of the material itself.

Regarding turbid front class, we can observe a similar problem: turbid fronts and wooden materials are present at the same time, so the inverse problem emerges: sometimes we got turbid fronts and wooden material at the same time, but the system predicts turbid fronts when the true label is wooden materials. An example is reported in figure 7.

Another problem with this class is the similarity between it and clear front class. Observing images in the dataset the difference is not always very clear: the model in this way makes a wrong classification. An example can be seen in figure 8.

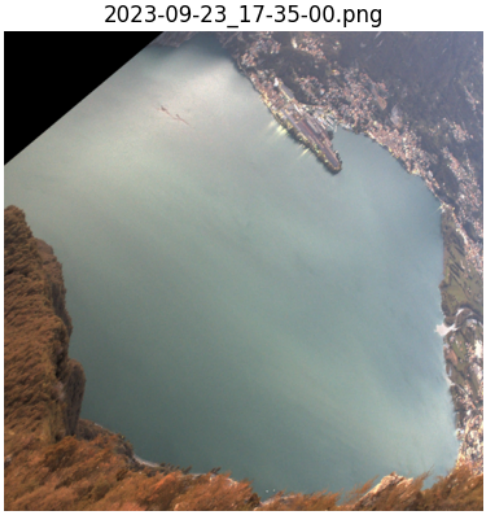


Figure 7: Image classified as turbid front with true class wooden material.

In the end, some image in the dataset is classified in a wrong way. An example is figure 9, classified as turbid front, but it is clearly a non event, since the water is not visible.

It should be noted that error analysis is made rather complicated by the combination of the number of classes available (8) and the similarity between them. The inability to classify images at first glance, without the help of an expert in the field, is also a factor that increases the difficulty of classification.

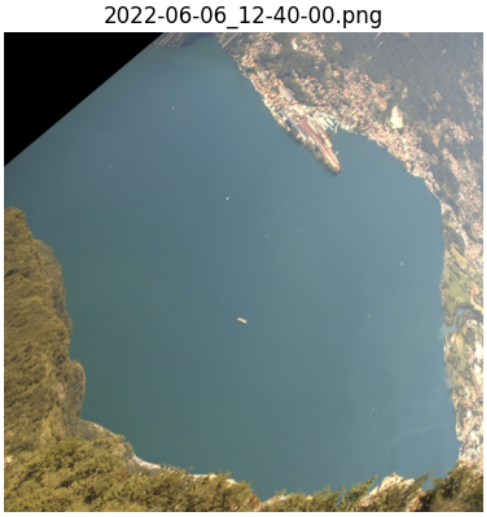


Figure 8: Image classified as turbid front with true class clear front.

Figure 9: Image classified as turbid front in the dataset.

5 Conclusion

The performance achieved by all the models tested is not very accurate. With the best models we can correctly classify about 3/4 of the available images.

The results obtained by the different architectures used are pretty similar, as evidence that there is some problem in the process. As a first instance, dataset should be revisited, in order to clean some wrong data. Then, it should be useful taking some decision about labeling wooden material class and turbid front class when they are presented together: a better labeling should bring better result.

Another possible solution is to combine some

classes: for example, if the expert of the domain notes that some separation between classes is not useful and they are similar from the image point of view, they should be merged. An example is clear front and turbid front.

Also, if after a dataset and labeling rework the accuracy is still low, more advanced and big models can be used and fine tuned: for example ResNet152V2 [3].

References

[1] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1409.1556.

[2] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[3] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks*, 2016. arXiv: 1603.05027 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1603.05027.

A Full Results Analysis

A.1 Accuracy and Loss Parameters during Training

During training phase accuracy and loss are continuously changing. The objective of the training, of course, is to make the model learn the task. At the same time, during training, we need to avoid overfit, using validation set. Figures 10 and 11 represent accuracy and loss parameters respectively during training of the previously described ResNet50_flat neural network architecture.

We need to consider that last five epochs for both base training and fine tuning are not considered, since the model implements early stopping with patience of five epochs. Figures have already been corrected, deleting last five epochs.

Considering this, in general there is no overfit in the accuracy graphics in base training. During fine tuning, overfit occurs, but also validation becomes better. The same happens for loss values.

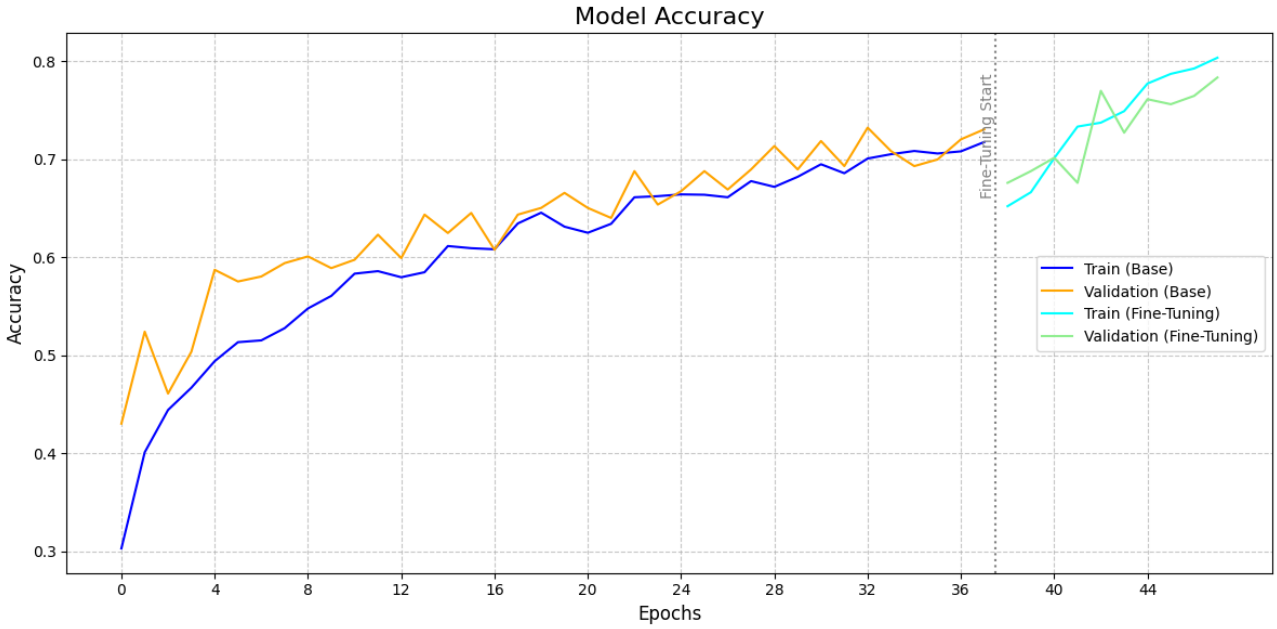


Figure 10: Accuracy during training for ResNet50_flat.

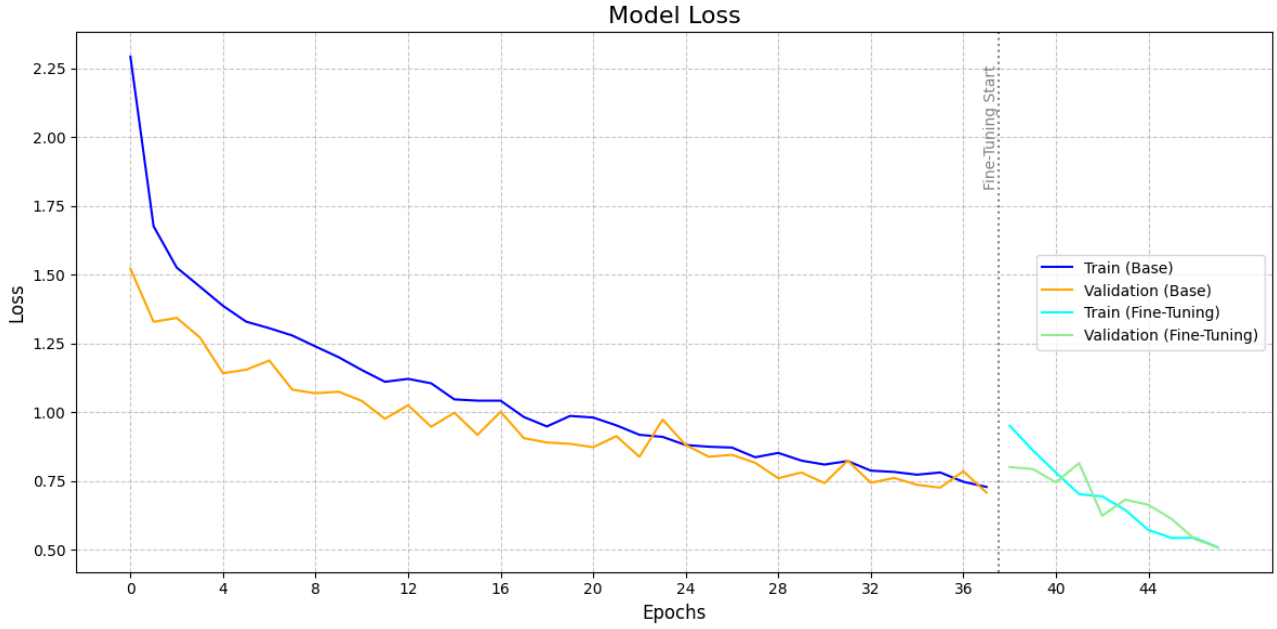
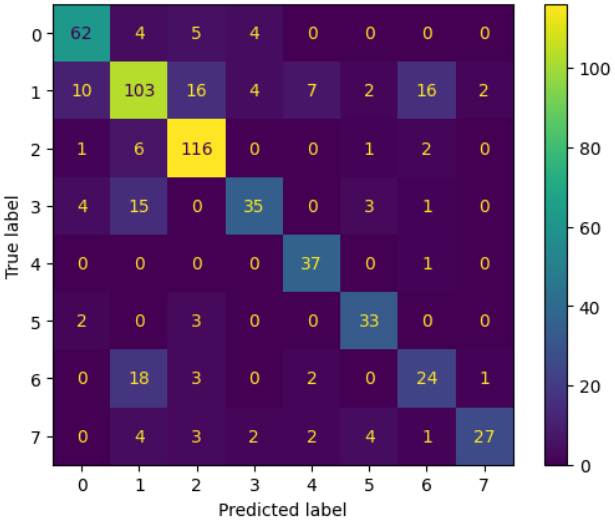
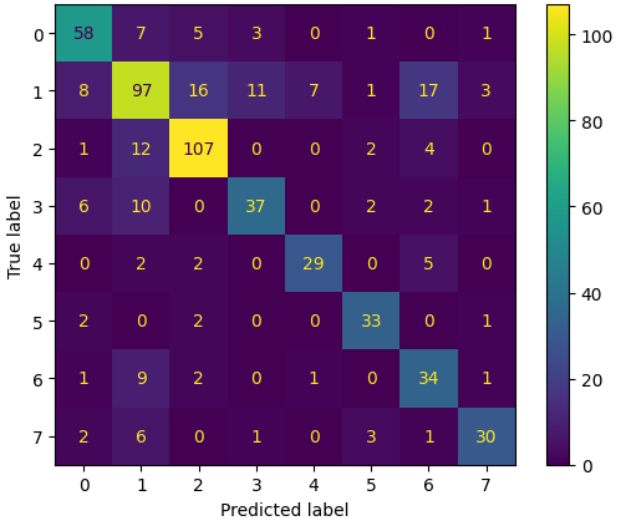


Figure 11: Loss during training for ResNet50_flat.

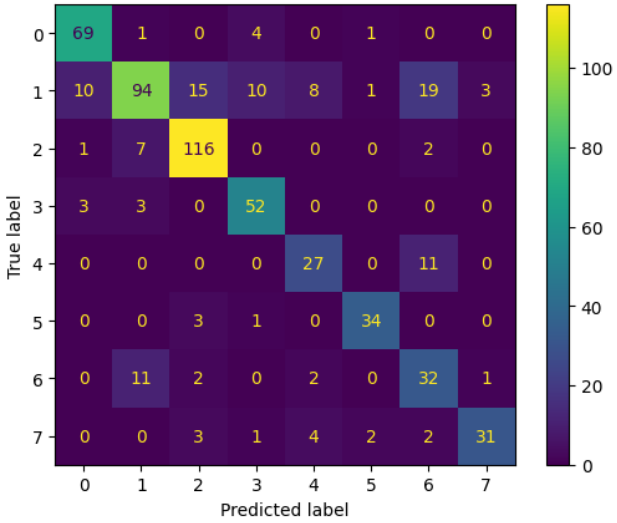
A.2 Error Report



(a) Confusion Matrix for model VGG19_1



(b) Confusion Matrix for model VGG19_2



(c) Confusion Matrix for model ResNet50_avg

Figure 12: Confusion matrices for VGG-based models and ResNet50_avg

In this subsection a report about errors in the VGG-based models and ResNet50_avg is presented. In figure 12 we can see error matrix for each model. All three error matrices are very similar to the one observed in section 4, confirming that some types of error regards dataset and labeling. For example, incorrect classification between class 1 and 6 occurs approximately in the same way. Table 4 confirms with precision and recall what we can see from confusion matrices.

Class	**VGG19_1**		**VGG19_2**		**ResNet50_Avg**	
	Precision	Recall	Precision	Recall	Precision	Recall
0 - Clean Front	78.48%	82.67%	74.36%	77.33%	83.13%	92.00%
1 - Turbid Front	68.67%	64.38%	67.83%	60.62%	81.03%	58.75%
2 - Turbid Discharge	79.45%	92.06%	79.85%	84.92%	83.45%	92.06%
3 - Turbid Lines	77.78%	60.34%	71.15%	63.79%	76.47%	89.66%
4 - Turbid Stain	77.08%	97.37%	78.38%	76.32%	65.85%	71.05%
5 - Non Event	76.74%	86.84%	78.57%	86.84%	89.47%	89.47%
6 - Wooden Material	53.33%	50.00%	53.97%	70.83%	48.48%	66.67%
7 - Vortex	90.00%	62.79%	81.08%	69.77%	88.57%	72.09%

Table 4: Precision and recall for each class of VGG19-based models and ResNet50_avg, expressed in percentages.