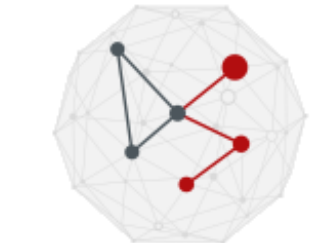
A Convolutional Neural Network for defects classification in steel sheet

- XAI for the steel industry -

Francesco Grimaldi

Danieli Automation & University of Padova



DATA SCIENCE
UNIVERSITY OF PADOVA

Topics

1. Introduction
2. Binary-class model (BM): evaluation and interpretation
3. Multi-class model (MM) : evaluation and interpretation
4. Conclusion

Problem

The aim of this project is trying to build a data-driven model which is able to detect the presence and the type of a defect in a steel sheet from labeled image data.

Once the model has been trained, interpretability methods are used to create an Explainable Artificial Intelligence (XAI) model, which can be used as a human-friendly recommendation system.

Dataset I

The dataset is a collection of 12568 steel sheets images of dimension 1600 x 256. For every image there is also an information about the presence of a defect.

The defects are divided in four types whose description could be summarized roughly in: multiple chips on the surface, single vertical cracks, multiple vertical cracks and multiple large surface patches.

Dataset II

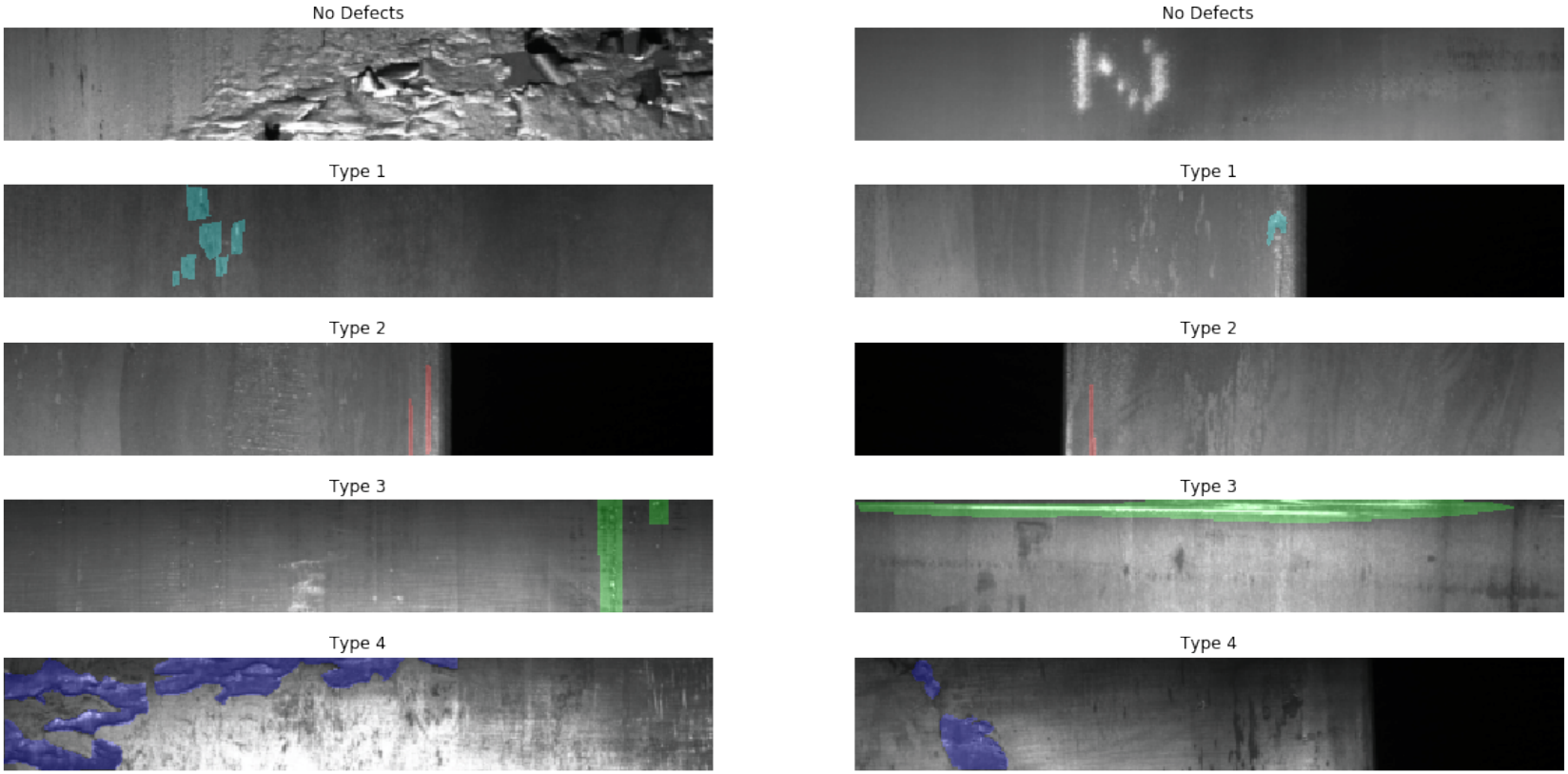


Figure: In each row it possible to see two example of a class. One class per row

Data Distribution

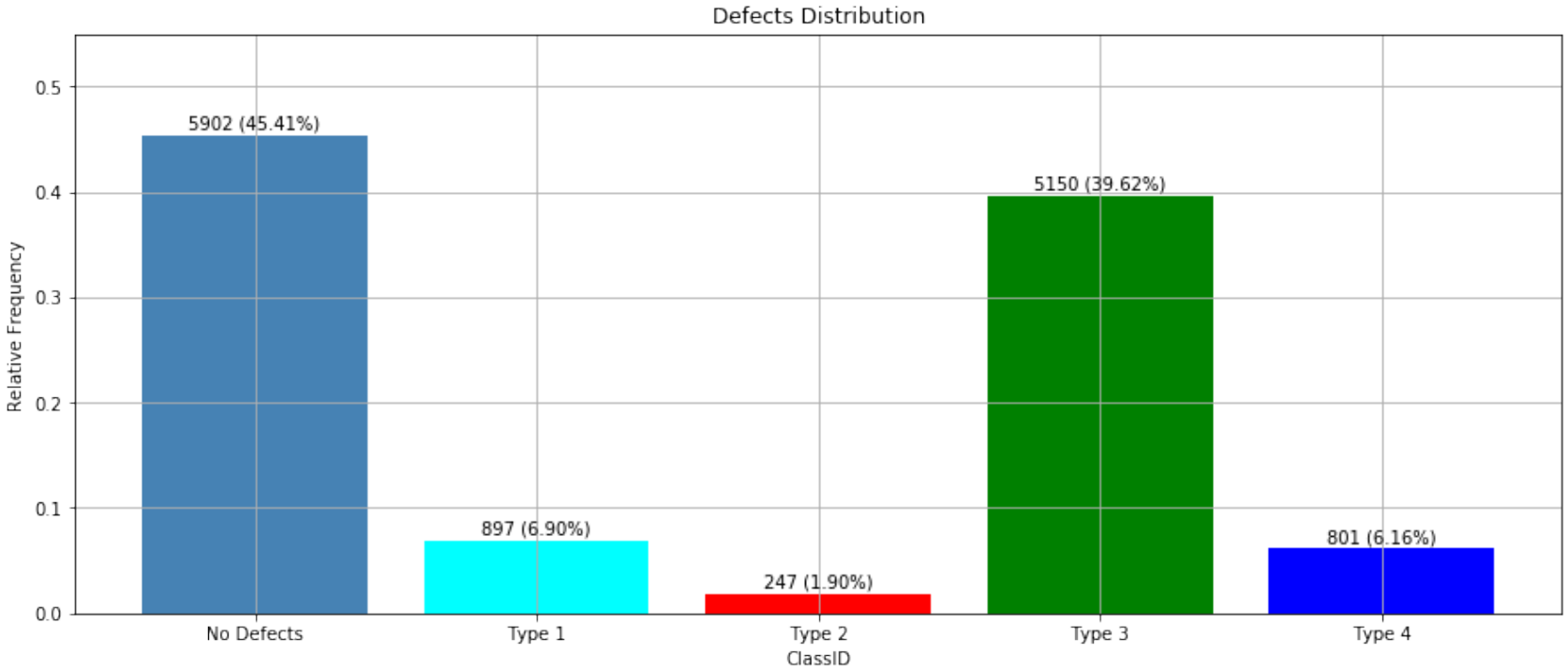


Figure: Distribution of the classes. We can see that the majority of the images have no defect or have type 3 defect, while only around 2% of the data belong to class *Type 2*

Data pre-processing

- The images have been resized by a factor of 16. From 1600x256 to 400x64. This was to reduce the amount of computational power needed to process the data.
- The data has been normalized across the 3 channels (standard pre-processing procedure).

Overview

Initially a binary model has been created in order to detect if a given steel sheet has a defect or not.

The models which were tried to solve this task are:

- An ad-hoc CNN
- Fine-tuned* and Pre-trained** VGG16 with Batch Normalization
- Fine-tuned* and Pre-trained** SqueezeNet v1.1

*The model has been retrained on our data.
** The model has been initialized with the weights of the pre-trained official versions trained on ImageNet.

Results

	Accuracy	F1	AUC	Time*
Ad-hoc CNN	0.887	0.895	0.951	**9.9ms**
VGG16	**0.941**	**0.944**	**0.985**	171.4ms
SqueezeNet	0.939	0.943	0.982	18.8ms
Reduced SqueezeNet	0.939	0.942	0.983	14.1ms

*Time to evaluate a batch size of 5 images with a GPU 850M. Models trained on PyTorch 1.2.0, Python 3.6, Windows 10.

N.B.

Results have been generated using augmented data X' defined as following: $X' = X + 0.15Z$, with $Z \sim \mathcal{N}(0, I_d)$

SqueezeNet I

The final model has been trained on a reduced version of Squeeze Net v1.1, with the last three FIRE modules taken out.

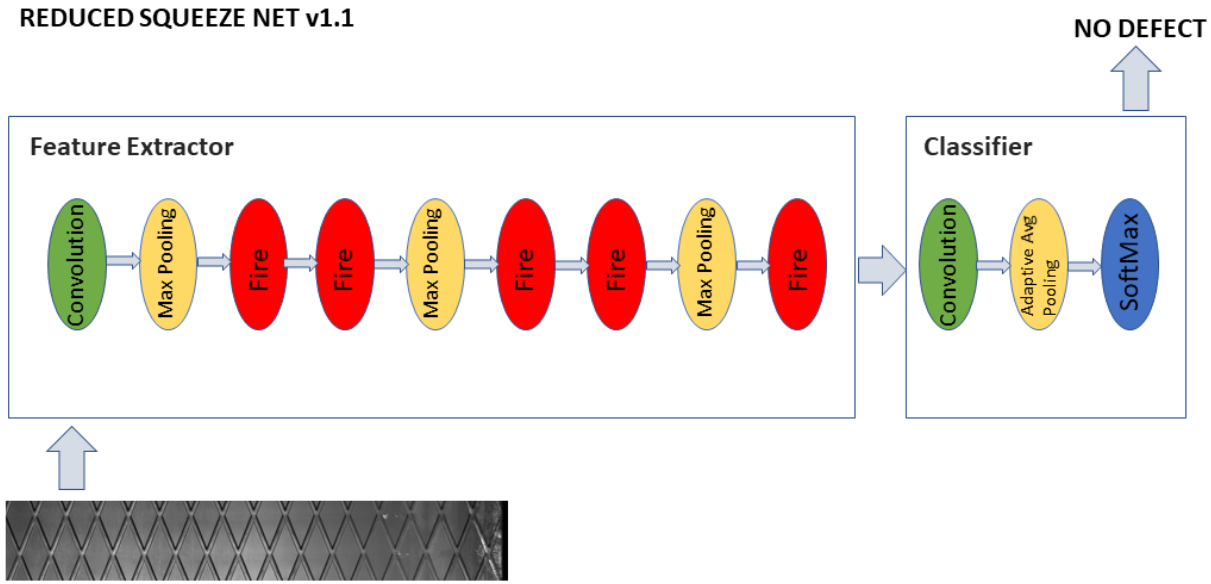


Figure: Architecture of the final model

SqueezeNet II

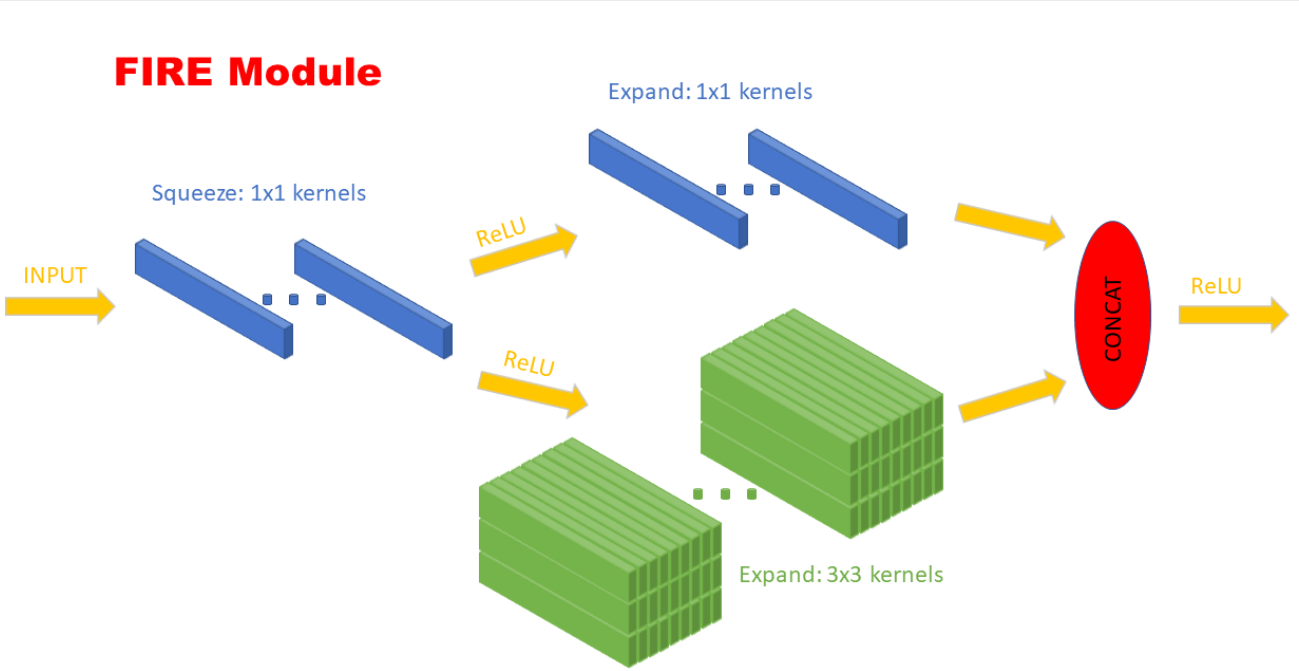


Figure: Inside the FIRE module: the squeeze layer decrease the number of filters while the expand layers increase it. The 3x3 kernel has the same output dimension of the 1x1 kernel thanks to a padding of 1.

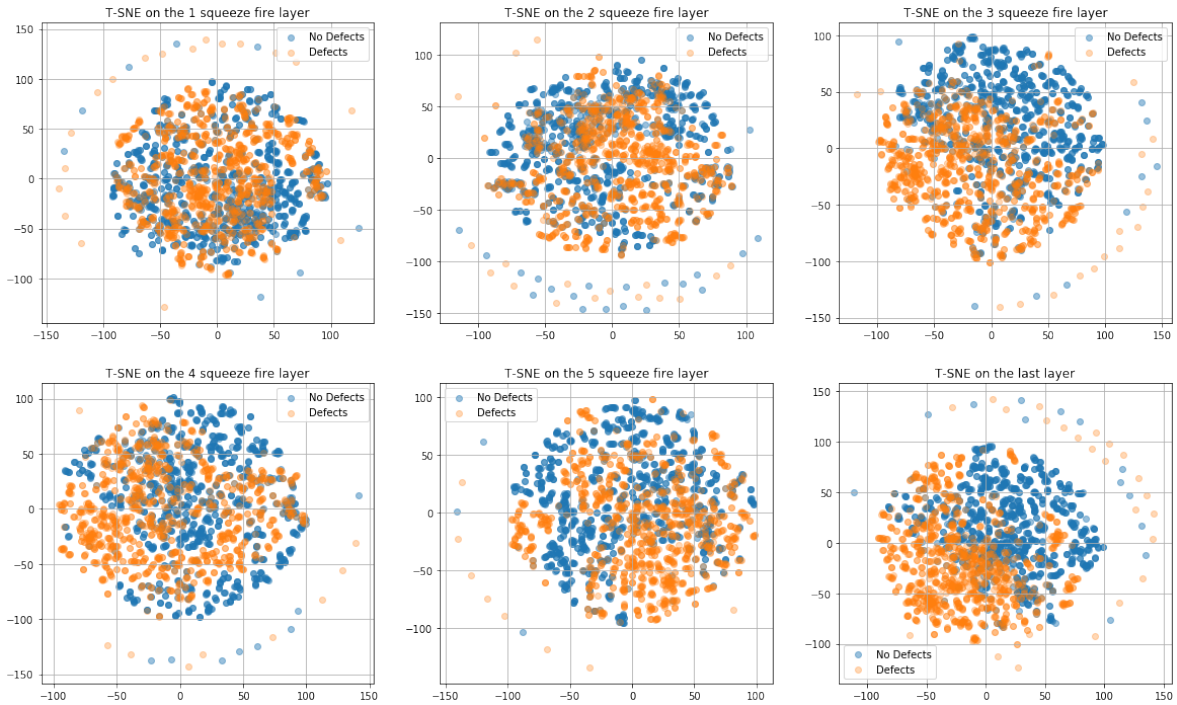
Layer Inspection

Quantitative and qualitative analysis have been used to inspect if the model was behaving as expected, in particular if the layer applied a manifold to the distributions in such a way that they become increasingly more linearly separable.

- Qualitative analysis with t-SNE
- Quantitative Analysis with a Logistic regression

Layer Inspection: t-SNE

t-SNE, a non linear dimensionality reduction technique, with as input the output of every layer, gives a visual idea of the manifold internal process.



Layer Inspection: Logistic Regression

The accuracy of a linear model (Logistic Regression), trained using as input the output of every layer, can be used as a proxy of the linear separability of the output at every layer.

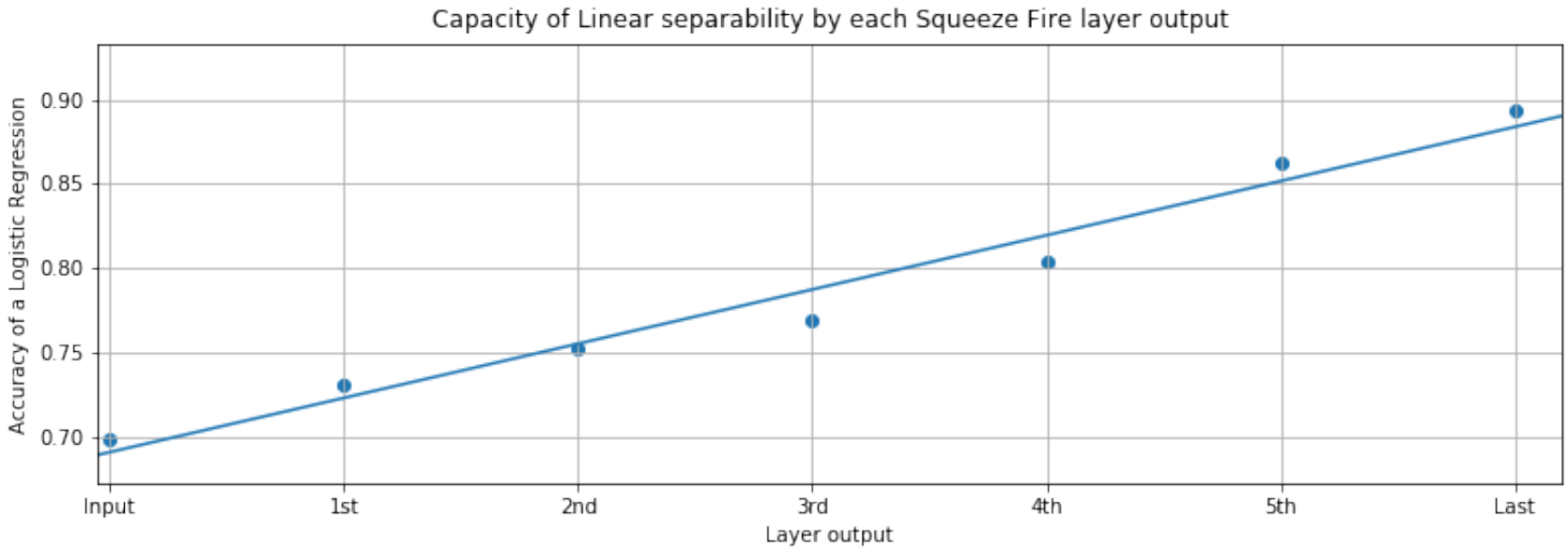


Figure: We can see how as the signal pass through the network it becomes more linearly separable

Neuron Inspection I

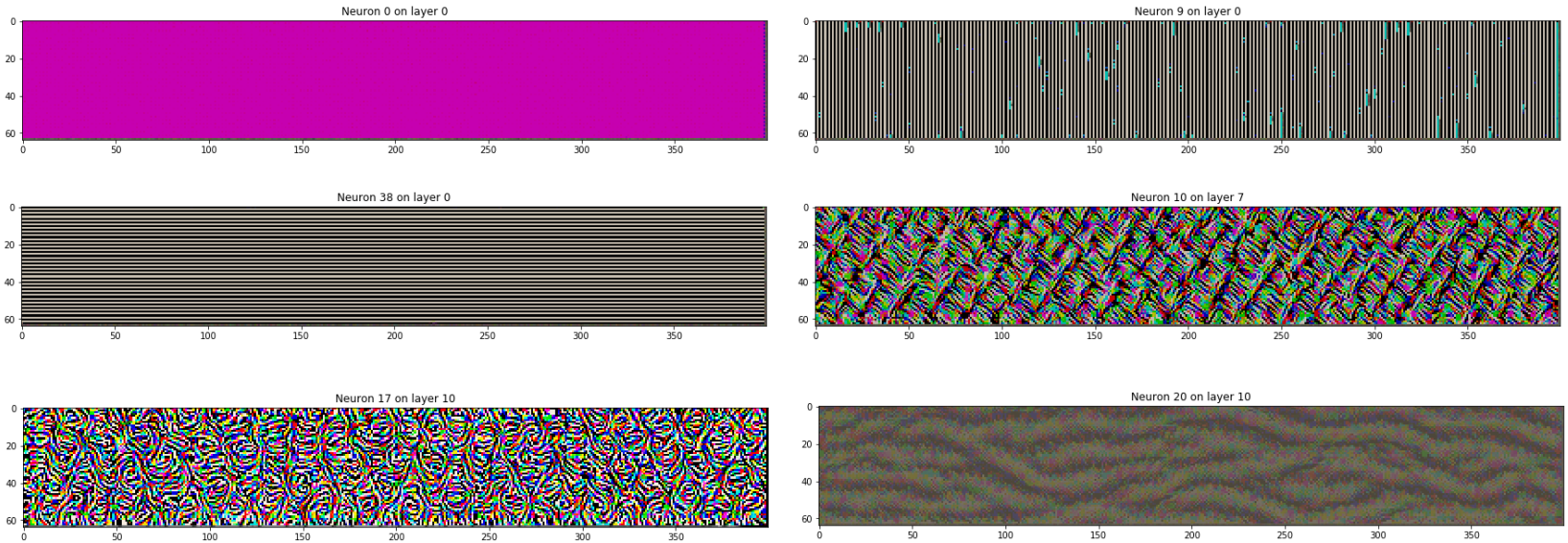
In order to dig inside the model and try understand which internal representations the model has learnt, the activation maps of the single neurons has been search through the following optimization process:

$$x' = \arg\min_{x \in \chi} -f_n^l(x; w) \tag{1}$$

where f_n^l is the output of neuron/filter n of layer l and χ is the image space.

Neuron Inspection II

Results showed that the model has implemented an hierarchical features representations, with simple features (colors/strips) detected in the first layer and more complex ones in the last layers.



Local Explanations

To infer the global policy and to try to build a more human-friendly and explainable model, local interpretability methods such DeepShap, DeepLift, GradCAM, GuidedGradCAM, GuidedBackpropagation, IntegratedGradients and LIME have been applied. In particular the following work has been done:

- Qualitative comparison of the interpretability methods
- Quantitative comparison of the interpretability methods using them as semantic segmentation models
- General application of the local methods to the dataset to infer global behaviour
- Analysis of the false positive and false negative

Local Explanations: Qualitative comparison

It has been seen that GradCAM localizes well the desired area but fail to give fine-grained details that others methods provide.

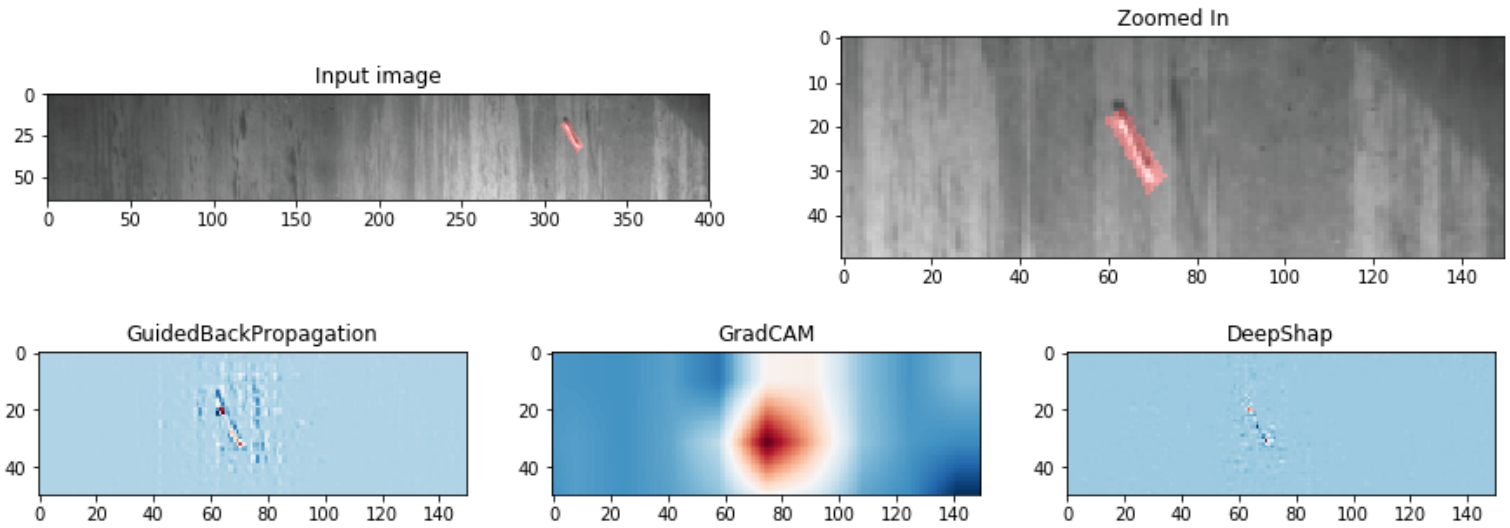


Figure: Visual comparison of GuidedBackpropagation, GradCAM and DeepSHAP. GradCAM gives a big visual clue of where the defect is located but, differently from GuididBackpropagation and DeepShap fails to be precise.

Local Explanations: Quantitative comparison I

The interpretability methods have been used as semantic segmentation model and evaluated using Recall, Precision, IoU and Dice Score.

	Recall	Precision	IoU	Dice	Time (ms/it)
DeepLift	0.647	0.121	0.101	0.170	102 ms
DeepShap	0.146	0.202	0.065	0.119	534 ms
GradCAM	0.588	0.160	0.124	0.207	**18 ms**
GuidedGradCAM	0.526	**0.298**	**0.195**	**0.308**	59 ms
GuidedBackpropagation	**0.670**	0.240	0.184	0.295	43 ms
IntegratedGradients	0.646	0.064	0.057	0.101	254 ms
LIME	0.269	0.100	0.040	0.070	641 ms

Table: *Evaluation results of semantic segmentation using the interpretability methods. The masks are given using the pixel above the 2 deviation standard and a blur filter.*

Local Explanations: Quantitative comparison II

A result of GuidedBackpropagation with the metrics equals to the average metrics of the methods. Despite the low scores we can see how the method is able to localize the area of the defects quite well.

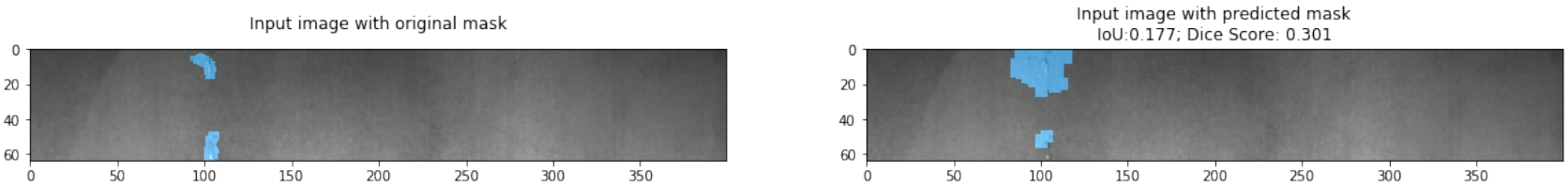


Figure: In the left we can see the image with the original mask, while in the right we can see the image with the predicted mask

Local Explanations: General applications

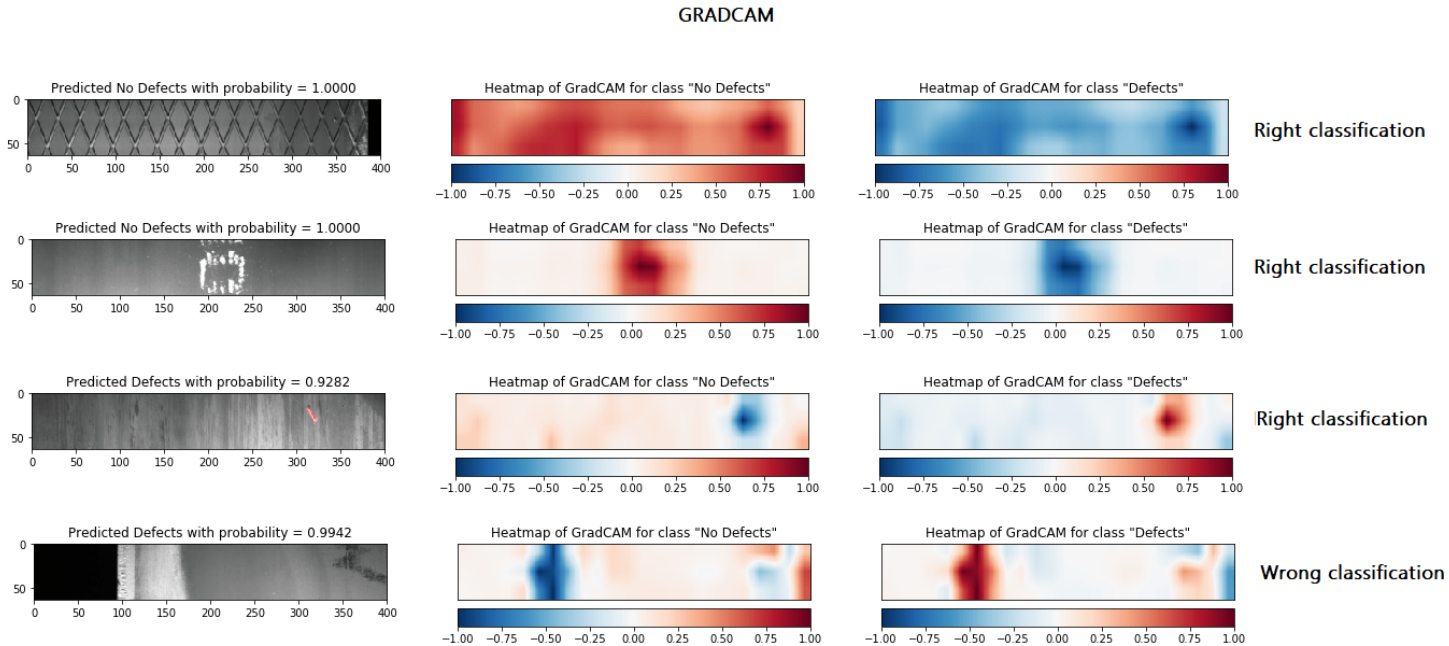


Figure: Four examples of GradCAM. In each row, from left to right, we have the original image (with mask, if any) with the predicted class and relative probability, the GradCAM output w.r.t the output neuron for class *No Defects* and w.r.t the output neuron of class *Defects*

Local Explanations: False Positive

By analysing the false positive (predicted class *Defects* but true class *No Defects*) with local explanations, it has been possible to see that most of the times the model considers as defects small impurities, dots and stripes which are not really defects.

An example of false positive can be seen in the fourth row on Fig. 8 (previous slide), where two regions of the image are considered as regions with defects.

White-Stain Effect I

By using local explanations tools, a possible spurious correlation effect has been noted in the data: the presence of white stain in the surface - the absence of defects in the surface.

This led the model to learn to classify a image as belonging to the class *No Defects* primarily based on the presence of these stains rather than the lack of defects.

White Stain Effect II

Moreover it has been noted that images with white stains were predicted with much more confidence than the ones without. To test this hypothesis, two groups of 50 samples have been created:

- First group: images with NO DEFECTS and NO WHITE STAINS
- Second group: images with NO DEFECTS and WHITE STAINS

White Stain Effect III

For each group it has been computed the vector of model outcome probabilities of belonging to class *No Defects*. If we define these two vectors, respectively, X and Y, with $X, Y \in [0, 1]^{50}$ we can use the Mann-Whitney U test to reject the following null hypothesis:

$$H_0 : P(x_i > y_j) \geq P(y_j > x_i) \tag{2}$$

where x_i, y_j are randomly sampled components (outcome probabilities) of vector X and Y respectively.

White Stain Effect IV

The Mann-Whitney test gave as result a p-value under 1e-10, giving us strong evidence to reject H_0.

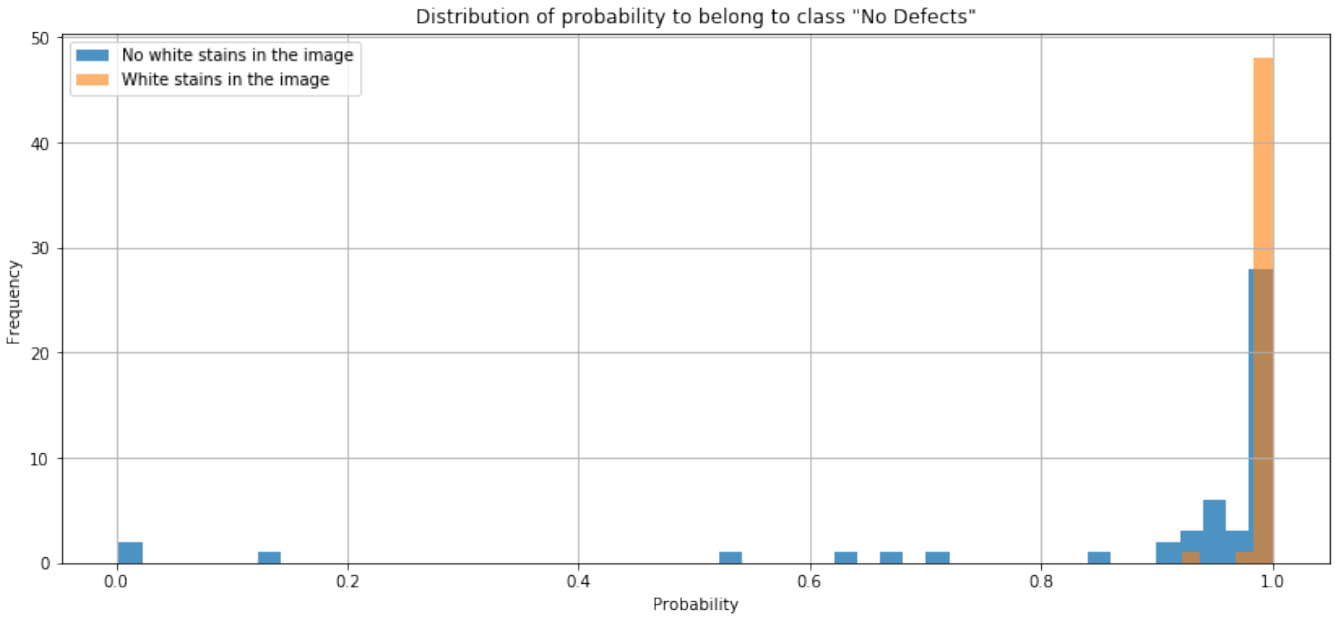


Figure: Distribution of the model outcome probabilities for the two groups: NO DEFECTS and NO WHITE STAIN (blue), NO DEFECTS and WHITE STAINS (orange).

White Stain Effect V

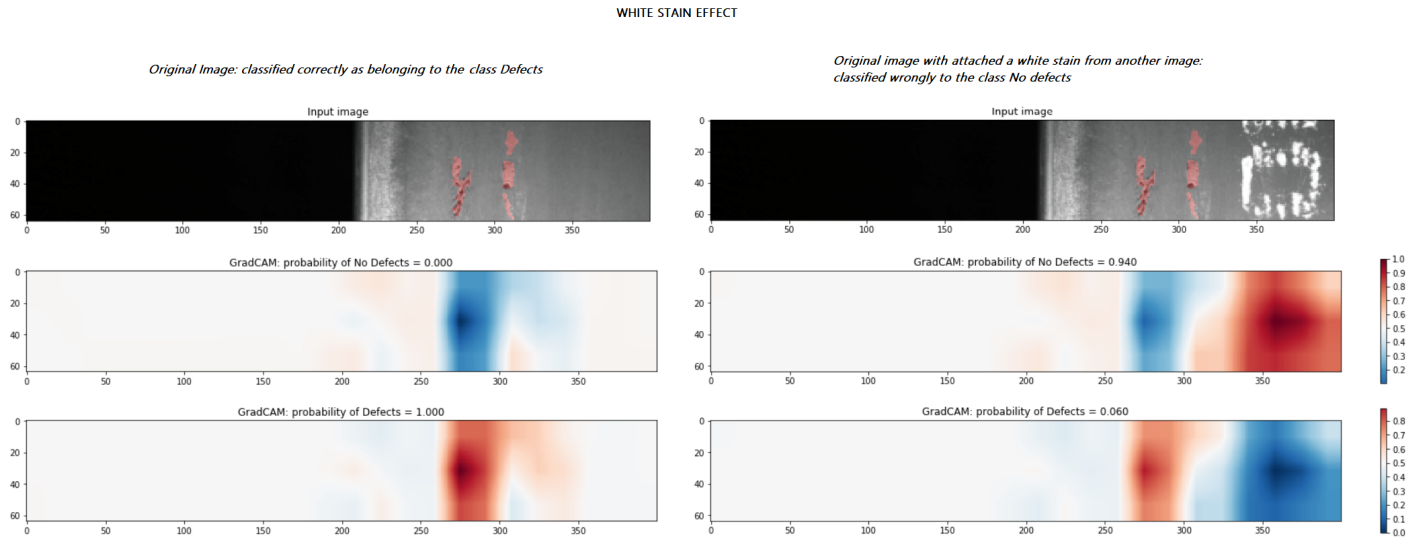


Figure: White-Stain Effect on an image with defects. In the top-left we can see an original image with a defect and rightfully classified and in the middle and bottom left we can see the GradCAM output for that image. Meanwhile in the top-right we can the original image with a white-stain attached to it. In the middle and bottom right we can see the GradCAM output and see how the model has changed its prediction even if the defects are still present in the surface.

Overview

After the binary model, a multi class model has been trained in order not only to detect the presence of defects but also their types. The model has a total of 5 classes: No defects, Type 1 (defect), Type 2, Type 3 and Type 4.

As before, an ad-hoc CNN, VGG16 with Batch Normalization and SqueezeNet v1.1 have been tested

Data Augmentation and other training techniques

Since we are dealing with a very unbalanced dataset (more than 85% of the samples belong to only class *No Defects* and *Type 3*) some different techniques have been tried to compensate for this factor:

- Use a weighted loss to favor the the most under-represented classes
- Increase the samples of these under-represented classes with data augmentation
- Train the model over patches instead of full images in order to oversample our dataset by a factor of 6.

Data Augmentation I

Dividing the images in patches decreased all the evaluation metrics, while using a weight loss increased the balanced accuracy (as expected) but decreased the others.

The only technique which was able to increase the balanced accuracy while also keeping intact the other metrics has been the data augmentation (with horizontal and vertical flip) of the under-represented classes.

In particular class Type 1 and Type 4 increased their samples of 200% while Type 1 of 600%.

Data Augmentation I

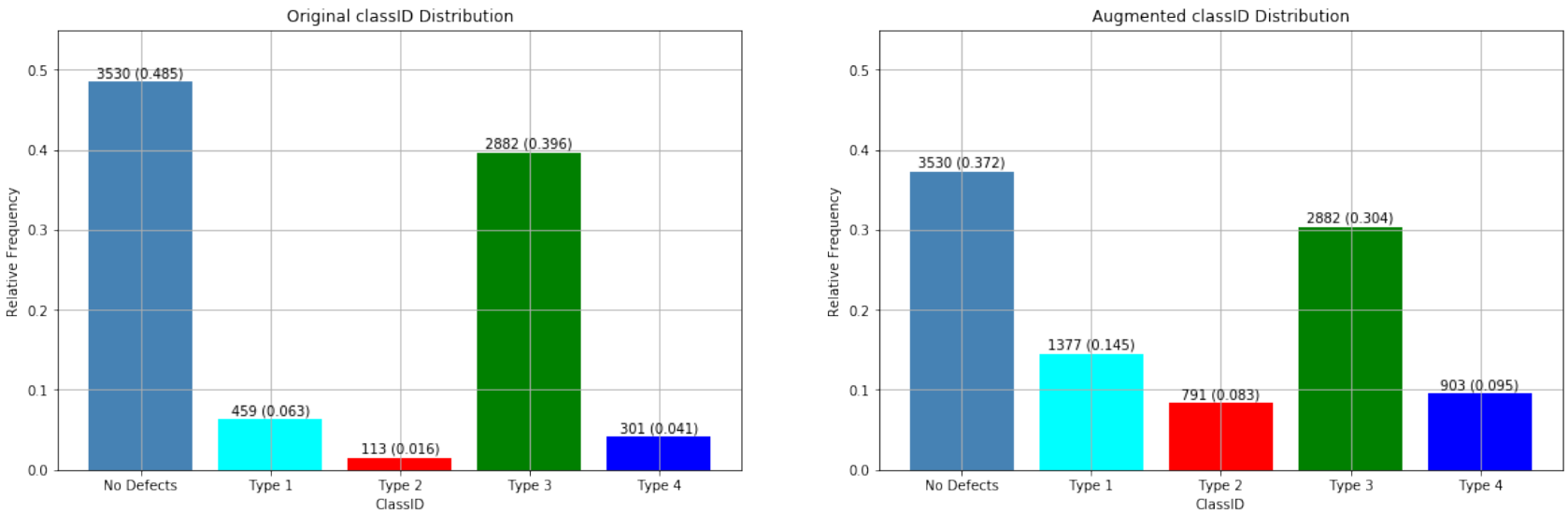


Figure: Before and after class distributions.

Data Augmentation II

The probability p of horizontal and vertical flip to apply to the image has been chosen to maximise the probability $f(p)$ that the original image X and the two augmented images X', X'' are different by each other ($X \neq X' \neq X''$).

$$\begin{aligned} p' &= \arg \max_{p \in [0,1]} f(p) \\ &= \arg \max_{p \in [0,1]} 2p^2(1-p)^2 + 4p^3(1-p) \\ &= \frac{1}{\sqrt{2}} \end{aligned}$$

Results

	Accuracy	Balanced Acc.	F1-Score	AUC	Time*
Aug. CNN	0.804	0 697	0.804	0.982	**10 ms**
Squeeze	0.920	0.860	0.920	0.986	19 ms
Weighted Squeeze	0.904	0.883	0.909	0.984	21 ms
Aug. Red. Squeeze	0.921	**0.904**	0.924	**0.987**	14 ms
VGG16	**0.929**	0.895	**0.929**	0.984	167 ms
Aug. VGG16	0.920	0.863	0.920	0.982	167 ms

Table: Acc. stands for Accuracy; Aug. (augmented) means that the model has been trained with the augmented data for the under-represented classes; Red. stands for Reduced, hence the version of SqueezeNet with only 5 Fire modules; weighted means that the loss to optimize has been weighted. **Time taken to evaluate a batch size (ms/batch) with a GPU 850M and a batch size of 5.*

Evaluation

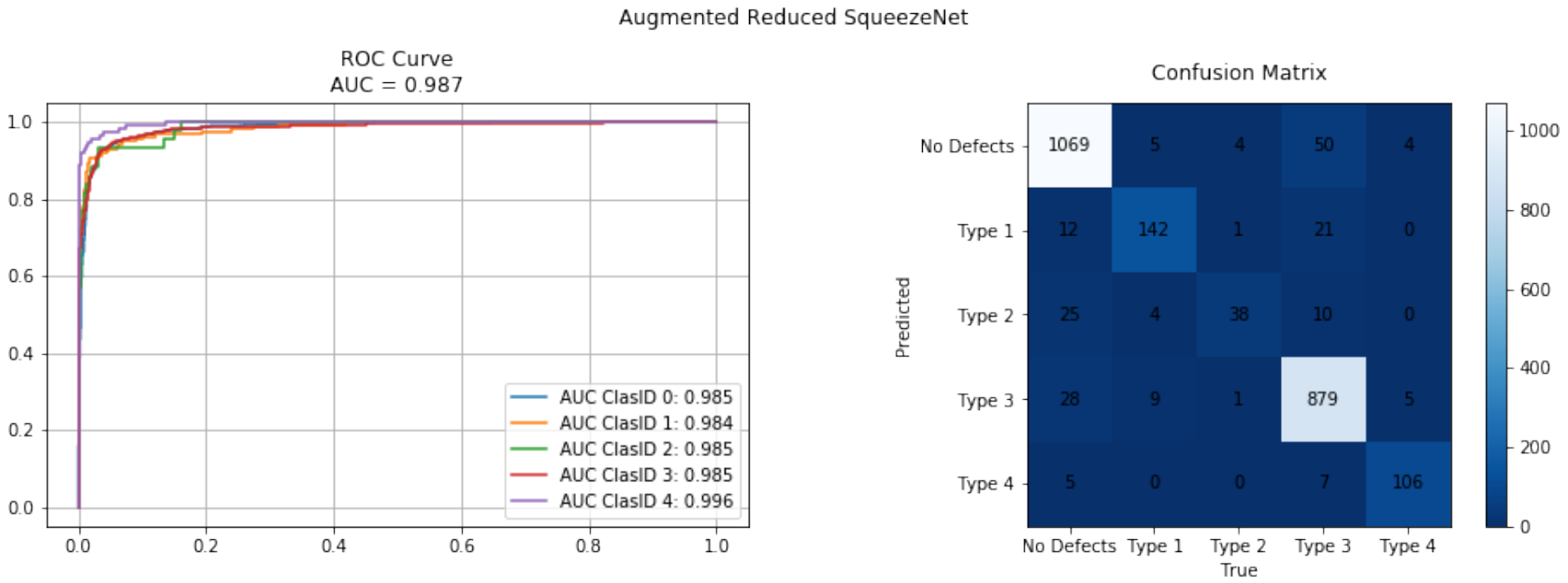


Figure: ROC curve and AUC score in the left and Confusion Matrix in the right. The model is the reduced version of SqueezeNet v1.1 trained on augmented data.

I don't know policy I

We can tell the model to do not take any decision unless its confidence (outcome probabilities) is higher than a certain threshold. So we can choose between an always available model (it always output a decision) and a not always available, but more accurate, model.

From a 92% accuracy and a 100% classification rate, we can raise the threshold up to 0.99 (the model predicts only when its confidence is close to 1). In this way the accuracy goes to 99% but the classification rate is almost halved.

I don't know policy II

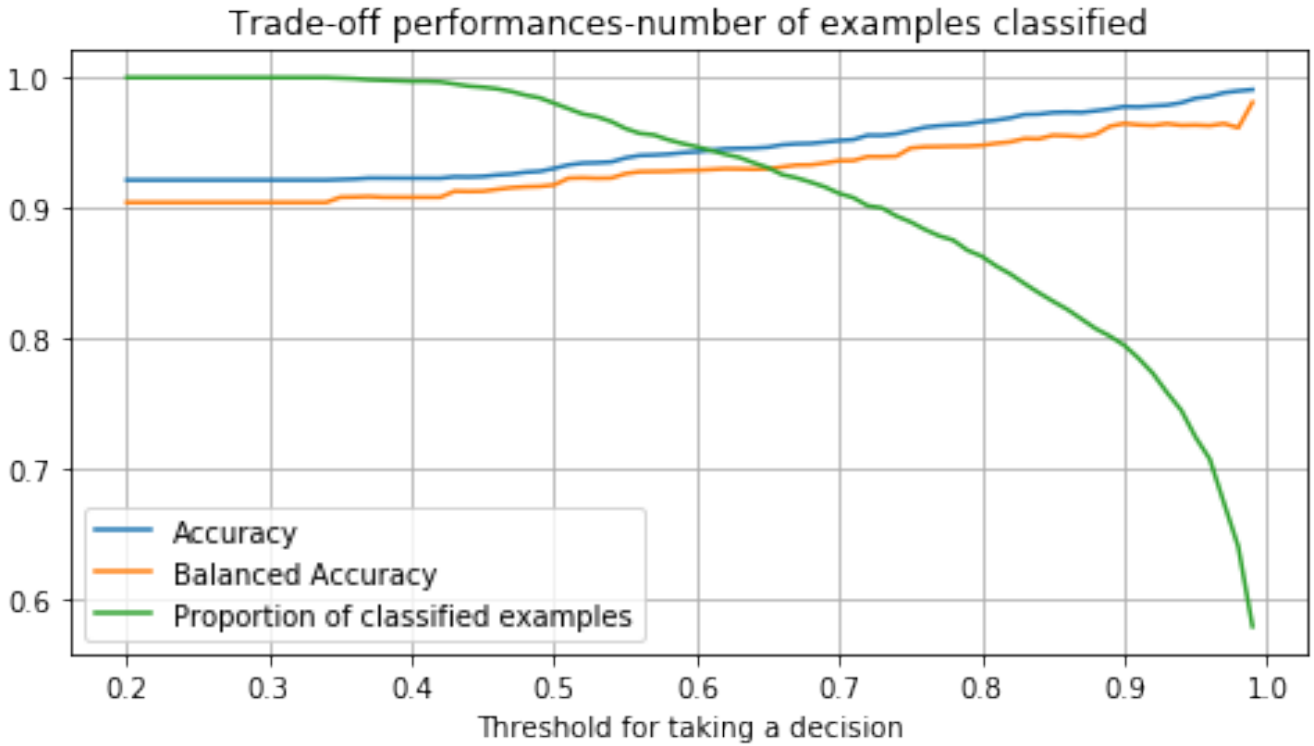


Figure: Accuracy, Balanced accuracy and number of classified example by the classification threshold value. With low threshold the model is the same of the standard one, while when the value grows higher, the model becomes more reliable but also less available (it does not take a decision)

Layer Inspection

Like in the binary case, qualitative (t-sne) and quantitative (logisitic regression) analysis has been used with the output of the model's layers. This time t-sne did not give good visually results,

while Logistic Regression did, showing how the accuracy increases as the signal goes through the network, a good index of proper manifold layer capabilities.

Layer inspection: t-sne

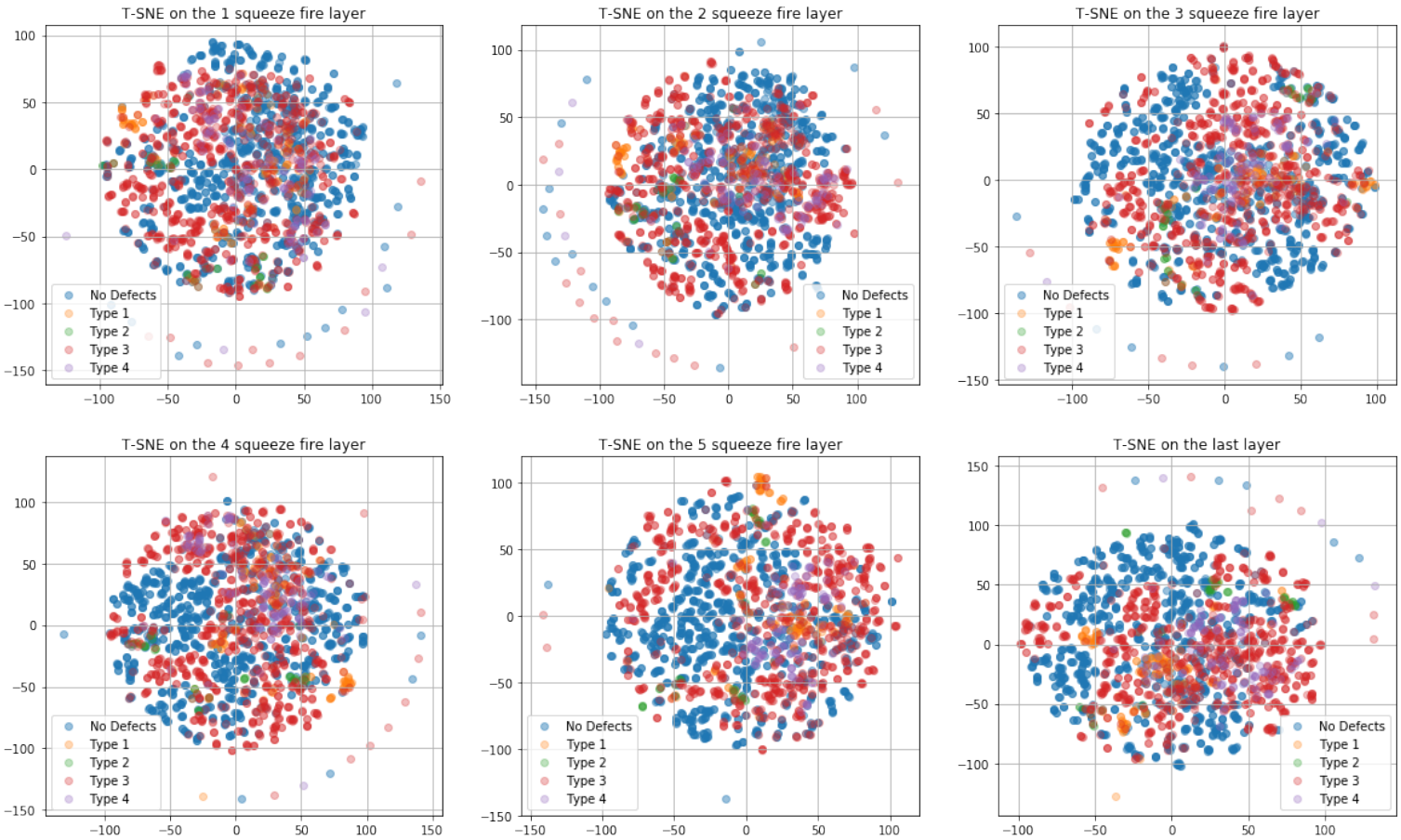


Figure: T-SNE results using the layers output as input.

Layer inspection: Logistic Regression

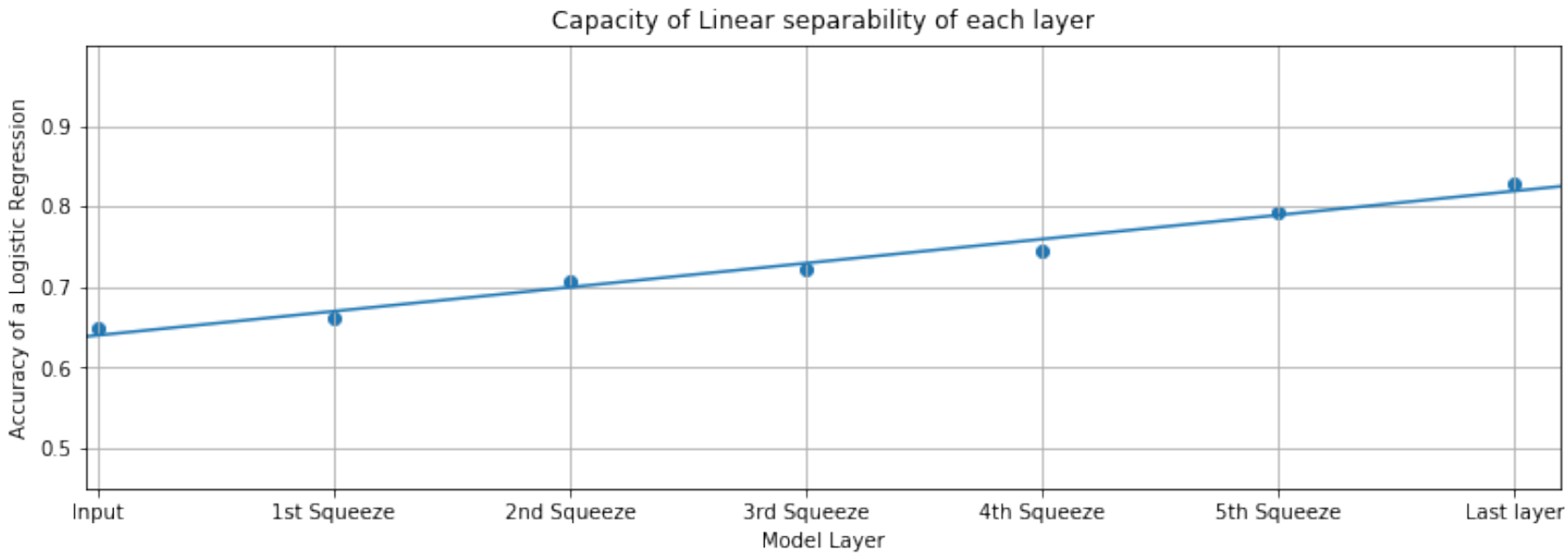


Figure: Accuracy of a Logistic Regression using as input the output of the model's layers.

Neuron Inspection I

This time to try to remove high frequency noise from the activation maps of the filters two techniques (resizing and blurring) have been used in the optimization process explained in Equation 1, slide 15.

This time, the process started using an input of lower size (64x64) and, every 50 optimization steps, the following operations have been used:

- Resize the image by a factor of 1.4
- Blur the image with a kernel of size from 3 to 10 (depends by nthe neuron)

Neuron Inspection II

Still, the procedure did not reveal insightful and complex patter, but only a hierarchical features complexity: simple features in the first layer, more non linear ones in the deepest layers.

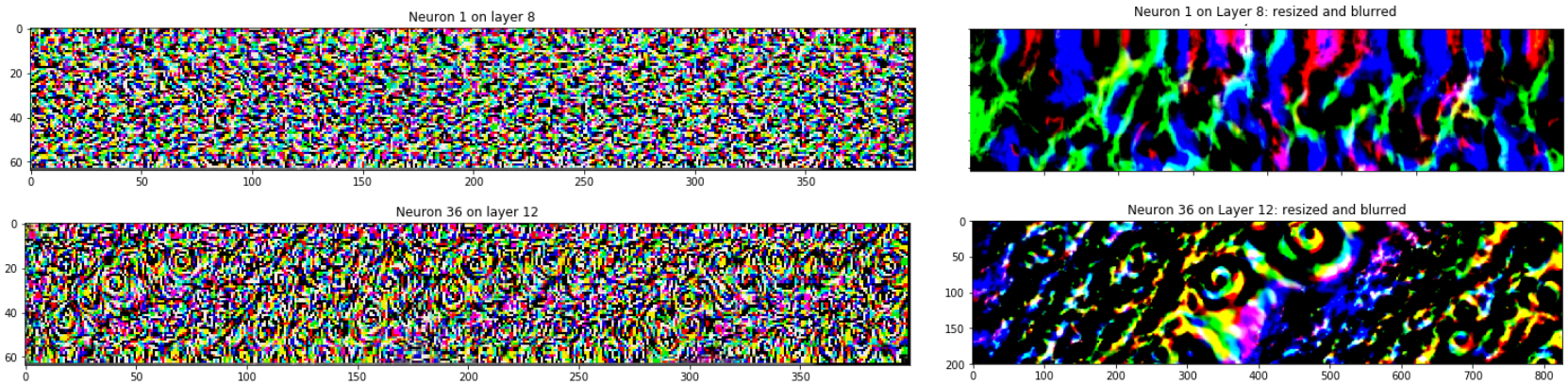


Figure: Differences between the activation maps generated by the standard procedure (left) and the ones generated with resizing and blurring (right).

Neuron Inspection III

Neuron inspection revealed that the features map of neurons of the multi-class model, were very similar to the ones of the binary model. A sign that the two models share the same weights.

To test this we switched, progressively, the weights of the multi model with the ones of the binary model, and tested the model accuracy.

Switching the weights of the features extractor super module did not decrease the model performance by much. However the metrics fell when we switched also the classifier super module weights.

Neuron Inspection IV

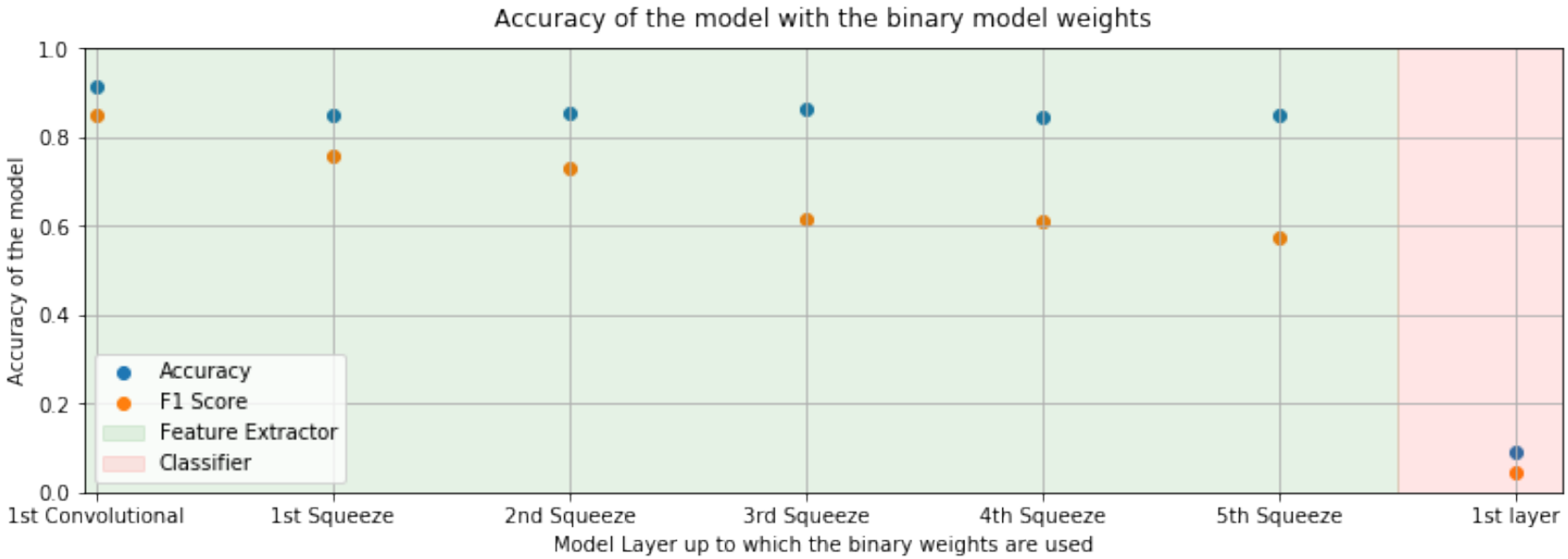


Figure: Accuracy and F1 Score of the multi class model, using the weights of the binary model. If we switch the weights of the features extractor (green area) the metrics do not decrease by much, while if we switch the weights of the classifier (red) the metrics decrease a lot.

Local Explanations

Like in the binary case we tested once again the local interpretability methods to see how well they could be used as semantic segmentation model, therefore providing insightful explanations of their decision process.

After this part we used the most suitable methods for our case, to analysis the prediction on the test set in order to understand the model policies

Local Explanations: Semantic Segmentation I

	Recall	Precision	IoU	Dice Score	Time (ms/it)
DeepLift	**0.760**	0.257	0.204	0.316	134ms
DeepShap	0.541	0.225	0.150	0.243	535ms
GradCAM	0.499	0.223	0.144	0.236	**22ms**
GuidedGradCAM	0.576	**0.360**	**0.236**	**0.363**	64ms
GuidedBackpropagation	0.638	0.285	0.199	0.315	44ms
IntegratedGradients	0.714	0.075	0.066	0.115	263ms
LIME	0.257	0.127	0.041	0.071	641ms

Table: *Evaluation results of semantic segmentation using the interpretability methods. The masks are given using the pixel above 3 std and a blur filter.*

Local Explanations: Semantic Segmentation I

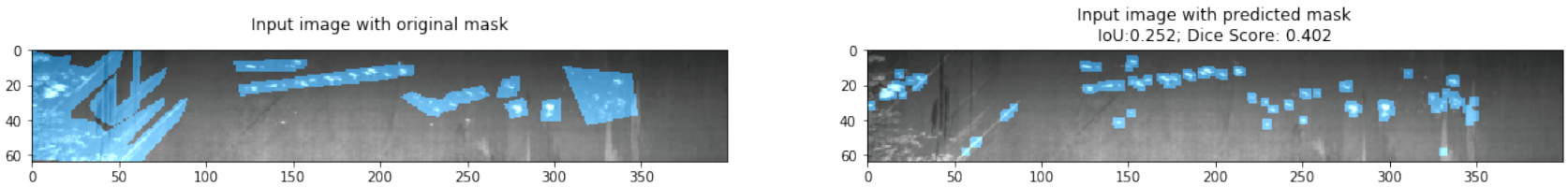


Figure: An example of output of GuidedGradCAM: in the left we can see the image with the true mask, while in the right we can see the image with the predicted one. The metrics of the predicted mask of this example are in line with the average metrics.

Local Explanations: GradCAM

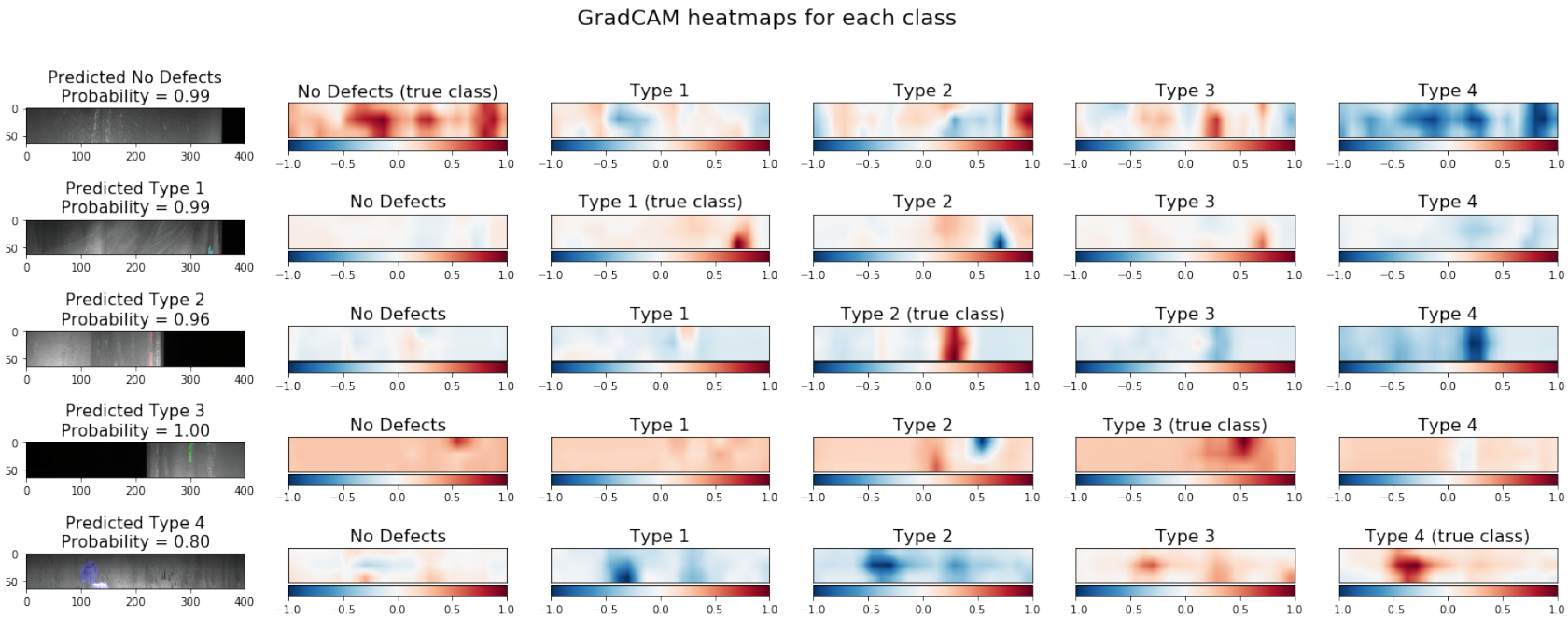


Figure: Example of application of GradCAM. In each row we can see an example (one per class). The image in the first column is the original one, while the others five are the GradCAM output w.r.t the 5 classes. Red areas excites the target neuron, while blue ones inhibits it

Local Explanations: Type II False Positive

Another fact that GradCAM enabled us to discovery is a similarity between the False Positive examples for Type 2 class (examples predicted type 2 defects but were not).

In fact, all false positive examples were images of the beginning or the end of steel sheets. This correlation is not spurious, since all the the type 2 defects are usually concentrated in these areas. Nevertheless, its a strong clue that the model, in order to classify an image as type 2 defects, it looks primarily on the presence of an edge rather than the presence of a real defect.

Local Explanations: Type II False Positive

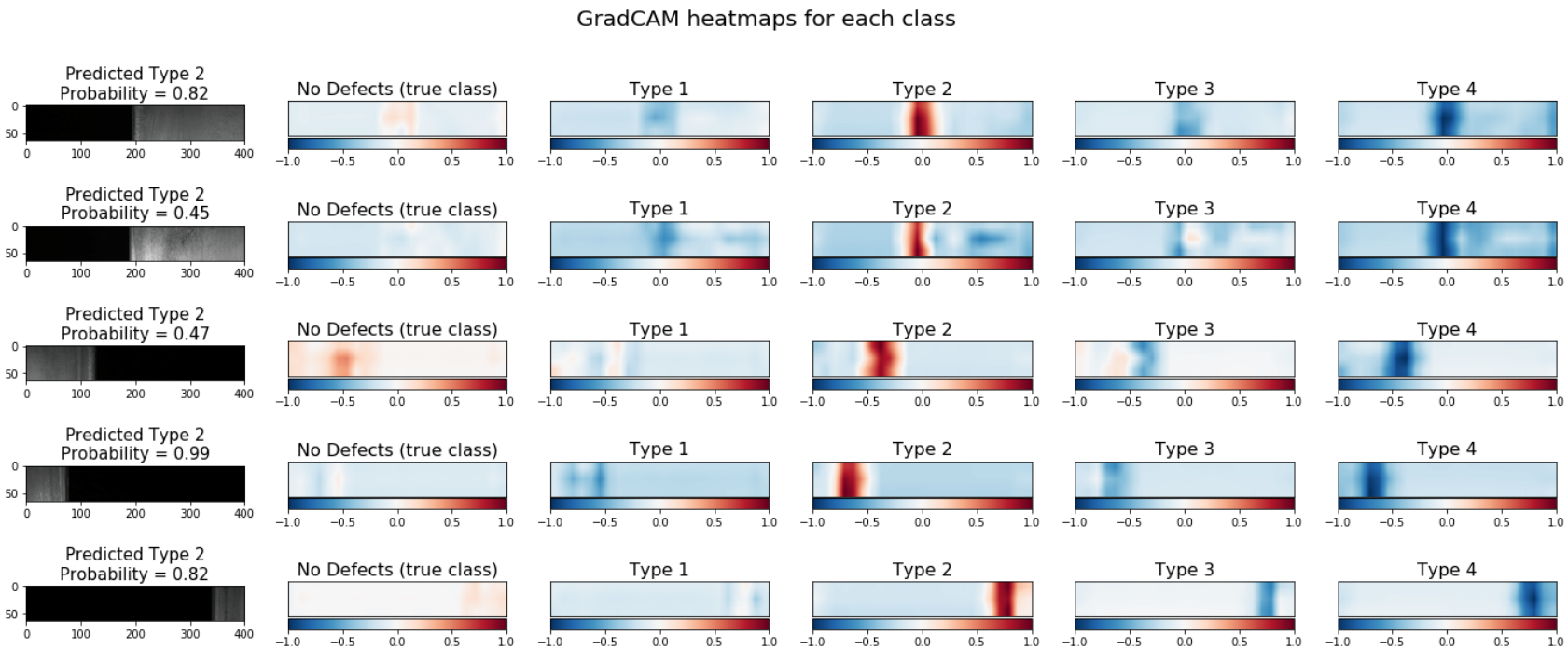


Figure: Examples of images predicted as *Type 2* defects but with true label *No Defects*. We can see that the reason of the prediction is due to the edge between the steel sheet area and the black area.

White-Stain Effect: episode II

The White-Stain effect, present in the binary case, appeared also for this multi-class model. The same analysis performed in the binary case (creation of 2 groups of images with no defects and with white stains and no white stains), had the same results.

The probability for an image with white stains to be classified as *No Defects* is much higher than the one with no white stains (Mann-Whitney U test; p-value $<$ 1e-19).

White-Stain Effect: episode II

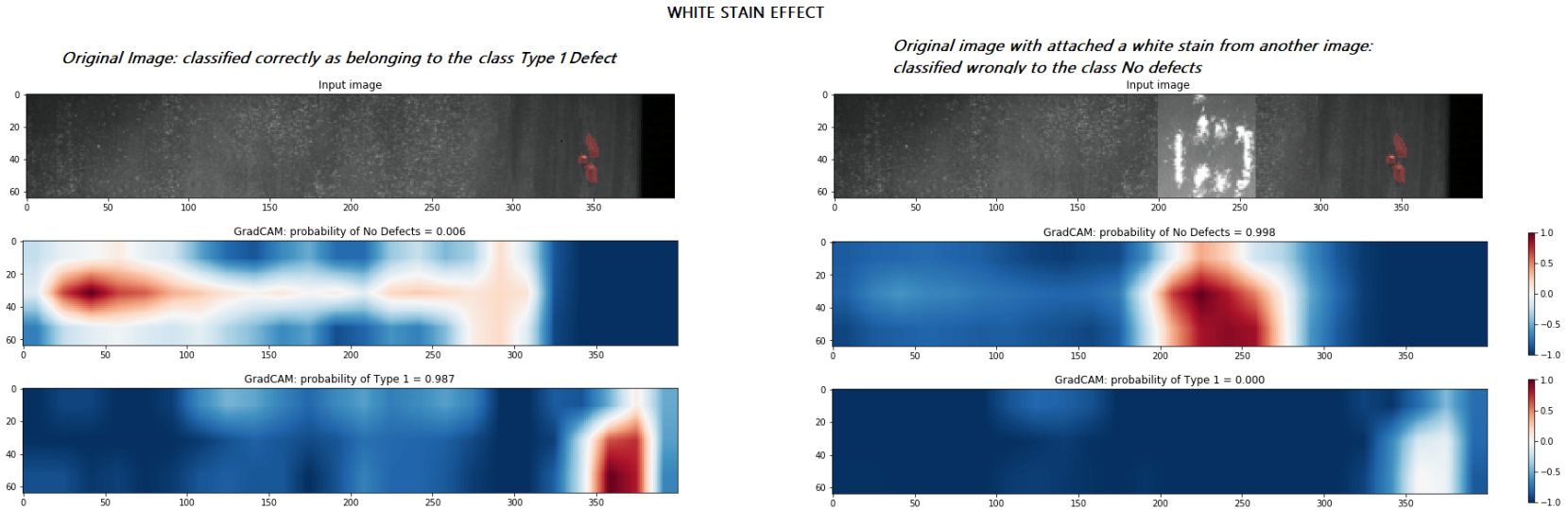


Figure: An example of how the white-stain is strong. An image rightfully labeled as with a defects, is classified as *No Defects* after a white-stain is attached over the image.

Conclusion: Overview

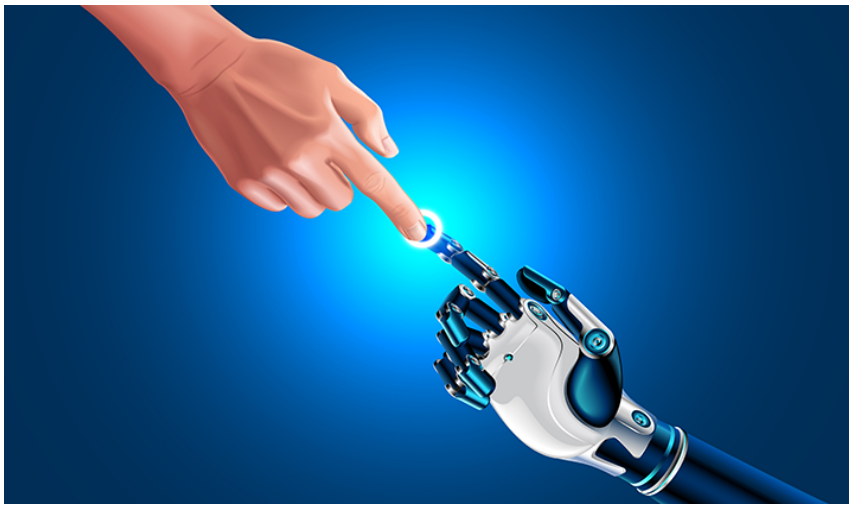
Using Machine Learning to classify defects in the steel sheet surface is possible in a both accurate (92-94% accuracy) and efficient (using a small model) way.

Moreover using inspection and interpretability techniques on the model give a lot of advantages:

- Assess the good behaviour of the model and its internal representations;
- Understand its weakness and its strength;
- Provide a tool for a human-computer interaction.

An Explainable Artificial Intelligence System

Interpretability techniques are a promising way to go in order to reach an XAI for this kind of task.



They offer good explanations of the AI decision process by highlighting the regions they considered important for their output, thus creating a procedure for achieving communication between human and AI.