Identifying Characteristics of False Positives from a Flower Detection Network

Isa Lykke Hansen

May 28th 2020

Abstract

This will soon be a paper about image classification ... I hope.

Contents

1 Introduction **3**

1.1 Image Recognition 3

1.2 A Flower Detection Network 4

2 Methods **4**

2.1 Sorting Images 5

2.2 Extracting Meta Data 5

2.2.1 The Q Complexity Measure 6

2.2.2 The IC_{LS} Complexity Measure 8

2.3 Statistical Analysis in R 9

2.4 Multicollinearity Checks 9

2.5 Cross-Validation 10

2.6 Evaluating Predictions 11

3 Results **11**

4 Discussion **14**

4.1 Interpreting Results 14

4.2 Further Analysis of the Data 15

5 Conclusion **17**

A The Q Complexity Measure **20**

B Interaction Plots **21**

C Multicollinearity Check **25**

1 Introduction

The method of using neural networks (NNs) to classify images have grown steadily in popularity over the last couple of years. These days, image classification algorithms help us solve diverse tasks such as detecting oil spills, improving early cancer detection, predicting the weather and even exploring the distant corners of our universe (Abhishek et al., 2012; Ekici and Jawzal, 2020; Krastev, 2020; Vestfalen, 2019).

Since 2018 researches at Universitetet i Tromsø (UiT) and Aarhus University (AU) have collaborated on a project to track the plant-pollinator interaction in selected species of Arctic flowers. This is done to investigate whether climate change is shifting the active seasons of the flowers and pollinating insects. Such a shift could potentially have very serious consequences, as it is ultimately the overlap between the pollinators and flowers that determine the outcome of food crops ({Danmarks Frie Forskningsfond}, 2019).

The project implements large scale video-surveillance of the plant-pollinator interaction by way of time-lapse images. For two years, time-lapse data was gathered from several cameras set up at various locations in the Arctic, collecting thousands of images for analysis. The idea then is to train a NN to automatically detect flowers and insects in the images. This creates the possibility for investigating changes in large areas and over long time periods with minimal amounts of equipment and manual labour.

1.1 Image Recognition

Broadly speaking, image recognition tasks can be divided into three sub-categories (Deeplearning.ai, 2017):

1. Classification
2. Classification with localization

3. Detection

When searching an image for an object, X, 1 answers the simple question "is X present in the image?". 2 in addition provides information on where in the image X is located. The exact location of X is specified by a so-called *bounding box*, which is parameterized by a vector containing the x,y coordinates for the midpoint of the box, as well as the height and width of the box. Finally, 3 allows for the localization of multiple objects from several classes within a single image. The output will be an array of bounding boxes each associated with a class and a probability for that class.

1.2 A Flower Detection Network

The images used in this paper were detections from a MASK R-CNN trained on data from four observation sites/periods (Narsarsuaq 2018, Narsarsuaq 2019, Thule 2018 and Ny Aalesund, Svalbard 2019). The network was trained on rectangular bounding boxes and obtained a precision of .84 and a recall of .92.

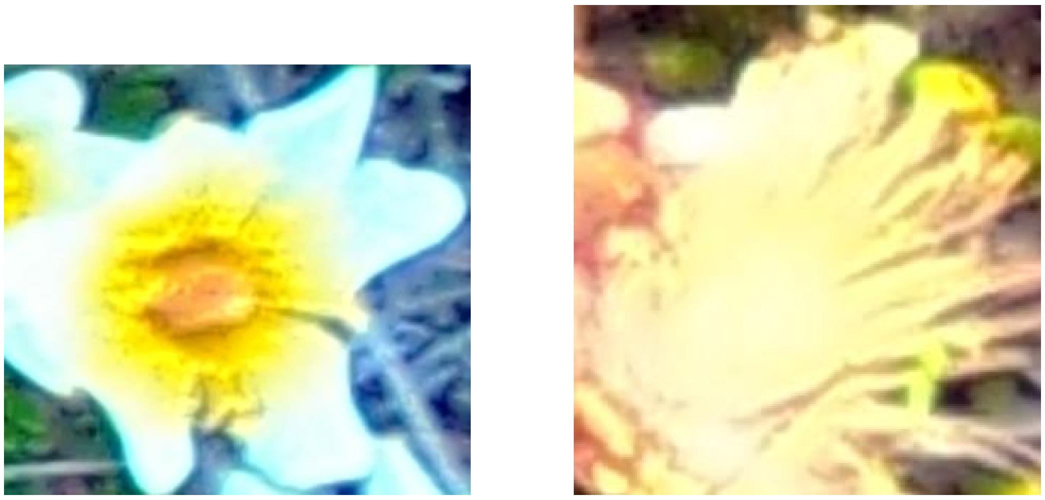
Thus, to put it more precisely, the images used here were crops created from the bounding boxes predicted by the network. The purpose of this paper is to investigate whether structural differences exist between true- and false positive predictions from the network. Incorporating such knowledge into the network could potentially lead to an increase in the recall and precision.

2 Methods

All scripts used for analysis can be found on the Github repository https://github.com/isalykke/Data-Science-Exam-2020

2.1 Sorting Images

The first task consisted of manually sorting the images into true- and false positives. From the original set of 7.420 images, 6.685 were manually sorted into two groups of true(n = 4.345) and false(n = 2.340)positives. The 735 images that could not clearly be defined as either true or false positives were excluded from the analysis. Example images from the two groups can be seen in Figure 1.



(a) (b)

Figure 1: Example images from a) the true positive group and from b) the false positive group

2.2 Extracting Meta Data

Secondly, meta data were extracted from all images using the script "extract_metadata.py". The script runs through the image directories and extracts image features for each image, before saving them to a .csv file for further statistical analysis (see section 2.3). Features extracted for each image included:

- Filename, location and label (true/false positive)

- Simple features such as height, width, size and ratio
- Complex features such as blurriness and complexity measures

Blurriness was calculated as variance of the Laplacian, a widely accepted standard in the field of image recognition (Bansal et al., 2016). The Laplacian filter performs edge detection, and the idea is that the sharper an image is, the more edges it will contain. In other words, in sharp images the variance of the Laplacian will be high, as there will be large differences in the gray-scale values of neighbouring pixels. Conversely, when the variance is smaller, we can interpret the image as being "blurry".

For complexity, since there is no widely accepted standard measure in the field, we calculated two different versions, in the hope that they would be able to explain some of the variance in our data. In sections 2.2.1 and 2.2.2 below we go more in depth with each of these.

2.2.1 The Q Complexity Measure

For the calculation of the Q complexity we implemented the method described in (Zanette, 2018). In short, the method creates several resized versions of a B/W image with dimensions NxM. These are then further divided into boxes of size LxL[1] and the sample variance of the gray-level for each box, b, is calculated as:

$$V_b = \frac{1}{L^2 - 1} \sum_{i\, in\, b} (g_i - \bar{g}_b)^2 \tag{1}$$

where g_i denotes the gray level of each pixel and

$$\bar{g}_b = \frac{1}{L^2} \sum_{i\, in\, b} (g_i) \tag{2}$$

[1]In theory L could be any number as long as its smaller than the dimensions of the resized image. In practice though, the best results are obtained for smaller values of L. This makes intuitive sense as less information is aggregated as L grows smaller. In this paper, as well as in the original, L was kept at 2.

The sample variance for each box is then combined to find the gray-level mean variance of the resized image:

$$V = \frac{L^2}{N'M'} \sum_b (V_b) \tag{3}$$

Where N'M' are the dimensions of the resized image. This process is repeated for all resized versions of the original image, and the associated scaling factor, S, is calculated as:

$$S = \frac{LN}{N'} = \frac{LM}{M'} \tag{4}$$

Finally, the complexity measure, Q[0:1], for the B/W image is calculated as:

$$Q = \frac{1}{s_{max} - s_{min}} \int_{s_{min}}^{s_{max}} \left[1 - \frac{1}{4} \left(\frac{dv}{ds} \right)^2 \right]_+ ds \tag{5}$$

Notes on Implementation Because the code used for calculating Q in the Zanette paper was unavailable, we implemented the method in python(v3.8.2)({Python Software Foundation}, 2020), by writing the functions **gray_level_mean_variance** and **Q_complexity** which can be found in the script 'extract_metadata.py'. Using this method we were able to very closely replicate findings from the original paper (see Appendix A). In the sections below we elaborate on some of the choices that were made during implementation of the Q complexity measure.

. In the original paper a series of values for N' were chosen to match the dimensions of the original image while at the same time maintaining the aspect ratio, so that $N'/M' = N/M$. As all of our images differed in dimensions, we instead implemented a method that downsized the images according to a percentage-wise scaling factor, P, such that $N' = N \cdot P/100$. The values of P were given by the set {3,4,5,7,10,12,15,17,20,25,30,40,50,60,70,80,90}. This we believe, makes the method more broadly applicable, while at the same time still satisfying the requirement of maintaining the aspect ratio.

. In addition, as 1 requires the division of images into 2x2 boxes, and all our images were of varying sizes, we decided to crop the images prior to the calculation of V_b. In practice, this was done by downsizing the dimensions of the resized image by one on the instances where the height or width did not satisfy $x\, mod\, 2 = 0$.

. Finally, as described in the article, in order to calculate 5 we need to integrate over the derivative dv/ds. Since we have only calculated discrete values of V(S) an interpolation function is required for the calculation of the derivative. Here we used zero-smoothed Bspline interpolation of the fourth degree utilizing the scipy funcitons **BSpline** and **splrep** (Virtanen et al., 2020).

2.2.2 The IC_{LS} Complexity Measure

The IC_{LS} is another proposed measure of complexity described in (Yu and Winkler, 2013). The IC_{LS} measures the compression ratio, CR, as the file sizes of uncompressed images relative to the file size of the compressed images, such that:

$$CR = \frac{s(I)}{s(C(I))} \tag{6}$$

Here, $s(I)$ denotes the file size (measured in bytes) of the uncompressed image and similarly, $s(C(I))$ is the file size of the image after compression. The lossless compression ratio of the image, IC_{LS}, can then be calculated as:

$$CR = \frac{1}{CR} \tag{7}$$

Notes on Implementation Similar to the Q-measure, calculation of the IC_{LS} was implemented in python(v3.8.2) via the function

ICLS_complexity defined in 'extract_metadata.py'. Below are some notes on the implementation of the IC_{LS}.

- The name "lossless compression ratio" is rather ill-fitting in our case, as the images we are investigating have already been compressed to .jpgs prior to analysis. Since we did not know the exact compression rate of the images, but were told they had been saved with standard jpg formatting, we accepted this as being roughly equivalent to lossless compression. In any case, since IC_{LS} is a ratio the true size matters less than the differences between sizes. We calculated three measures of the IC_{LS}, namely the IC_{LS10}, IC_{LS30} and IC_{LS50} where C(I) was set to 10, 30 and 50, respectively.

- The original paper uses gray-scale versions of the images for the calculation of IC_{LS}. Here we chose instead to use the full RGB versions of the images, as we wished to maintain as much information from the original images as possible.

2.3 Statistical Analysis in R

All statistical analysis was carried out in R(v.3.6.1) ({R Core Team}, 2020). Meta data were extracted for a total of 6.683 images[2]. For 73 of these (FP = 60, TP = 13), the calculation of Q resulted in NAs. In order to include as much data as possible in the model selection, in these instances we chose to assign the group average as Q.

2.4 Multicollinearity Checks

In order to decide which parameters to include in the model selection we did a multicollinearity check of all predictors, the results of which can be seen in Figure 11, Appendix C.

[2]Two of the false positives were excluded due to errors in the computation of Q

Since the three IC_{LS} measures were generally highly correlated with each other (all $\rho > .6$) we chose to include only IC_{LS50} in the analysis, since this parameter also had the lowest correlation with the width, height and size parameters. Similarly, and perhaps unsurprisingly, the width and height parameters were also both highly correlated ($\rho > .8$) with the size parameter. For simplicity, and to lower the computational cost, both of these were therefore left out of the cross-validation process. The 5 parameters included in the final model selection process were:

- Size
- Ratio
- Blurriness
- IC_{LS50}
- Q-complexity

2.5 Cross-Validation

To evaluate which model best predicted the data, we ran a series of cross-validations using the R packages groupdata2 (Olsen, 2020b) and cvms (Olsen, 2020a). Model formulas were determined with the function **combine_predictors** which allows for unique combinations of all predictors. Due to computational restraints we allowed for a maximum of two-way interactions. We ran a 10-folds cross-validation on each of the 1.268 model combinations and extracted the best 100 models as ranked by a Balanced Accuracy measure. These were then run though a 10-folds 5-times-repeated cross-validation. The top ten models from this process along with their Balanced Accuracy scores can be seen in Table 1. The model with the highest Balanced Accuracy score was expressed as:

$$\begin{aligned} false\ positives \sim\ & blur * Q + blur * size + ICLS50 * Q \\ & + ICLS50 * size + Q * size + ratio * size \end{aligned}$$

The parameter estimates for the model can be seen in Table 2 and additional model statistics can be seen in the first row of Table 3.

2.6 Evaluating Predictions

Because our goal ultimately is to be able to classify images from various research stations, it is important that the selected model generalises to data from more than one location. As we observed systematic structural differences (see Figure 2) between the false positives from location one (Narsarsuaq, n = 5681) and two (Thule, n = 1002) we suspected that the model might not generalize well across locations. In order to test this we fitted the model to a subset of the data collected at location one and tested it against data collected at location two. The results of this comparison can be seen in the second row of Table 3.

3 Results

Table 1: The ten highest scoring models from cross-validation

Model formula ($false\ positives \sim$)	Balanced Accuracy
$blur * Q + blur * size + ICLS50 * Q + ICLS50 * size + Q * size + ratio * size$	0.766
$blur * Q + blur * size + ICLS50 * Q + Q * ratio + Q * size + ratio * size$	0.763
$blur * size + ICLS50 * Q + ICLS50 * size + Q * ratio + Q * size + ratio * size$	0.763
$blur * Q + blur * size + ICLS50 * Q + Q * size + ratio * size$	0.763
$blur * size + ICLS50 * Q + ICLS50 * size + Q * size + ratio * size$	0.763
$blur * ICLS50 + blur * Q + blur * size + ICLS50 * Q + Q * size + ratio * size$	0.762
$blur * Q + blur * size + ICLS50 * Q + ICLS50 * ratio + Q * size + ratio * size$	0.762
$blur * ICLS50 + blur * size + ICLS50 * Q + ICLS50 * size + Q * size + ratio * size$	0.762
$blur * size + ICLS50 * Q + ICLS50 * ratio + ICLS50 * size + Q * size + ratio * size$	0.762
$blur * ratio + blur * size + ICLS50 * Q + ICLS50 * size + Q * size + ratio * size$	0.761

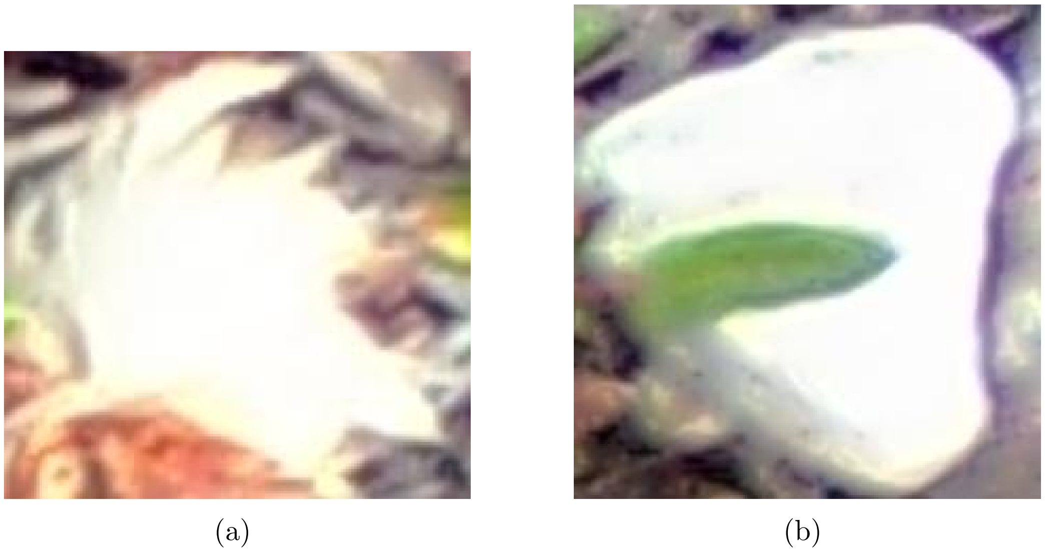


Figure 2: Representative examples of false positives from a) location one, Narsarsuaq and b) location two, Thule

Table 2: Model coefficients for the highest scoring model

Coefficient	Estimate	SE	p
Intercept	-69.62	19.57	<.001***
Blurriness	-0.04	0.02	0.02*
Q	99.61	38.95	0.01*
Size	-0.0006	0.00009	<.001***
$ICLS50$	100.50	24.19	<.001***
Ratio	5.03	0.45	<.001***
Blurriness:Q	0.12	0.03	<.001***
Blurriness:Size	-0.000006	0.00000009	<.001***
Q:$ICLS50$	-160.70	48.31	<.001***
Size:$ICLS50$	0.0004	0.0001	<.001***
Q:Size	0.0006	0.00003	<.001***
Size:Ratio	-0.00009	0.000007	<.001***

Table 3: Model parameters for the best model trained on all data vs a subset of the data (see Section 2.6 for details)

Data	Balanced Accuracy	AUC	Lower CI	Upper CI
All	0.77	0.86	0.85	0.87
Subset	0.35	0.33	0.30	0.37

4 Discussion

4.1 Interpreting Results

Table 2 displays the model coefficients for the model that best predicts the differences between true and false positives from the MASK R-CNN, as determined by the cross-validation process. We see that both Blurriness(-0.036), Q(99.61), Size(-0.0006), IC_{LS50}(100.50) and Ratio(5.03) were significant predictors of false positives.

Similarly, the interaction terms blur:Q(0.12), blur:size(-0.000006), IC_{LS50}:Q(-160.7), IC_{LS50}:size(0.0004), Q:size(0.0006) and ratio:size(-0.00009) were all significant predictors of false positives. However, according to the literature, given the presence of significant interaction effects the interpretation of main effects is meaningless (Field et al., 2012).

In Figures 5 though 10 in Appendix B the 6 different interaction effects of the model have been plotted. An interaction effect encodes a difference of differences for the effect of a predictor on an outcome. In other words, the significant interaction terms in our model suggest that these specific combinations of predictors differently affect the likelihood of an image being a false positive.

WRITE MORE ABOUT INTERACTIONS HERE

We also see that the best model (as predicted by the cross-validation) contains quite a lot of interaction terms, meaning that the effect of the predictors is not the same for true and false positives. In other words, there is a difference in differences between the predictive powers of the meta-data measures.

Table 3 shows the predictive power of the model when it is fitted to all data and when it is fitted to a subset of the data. We see that all measures drop substantially when the model is fitted to data from a new location. In fact, the model that was fitted to the subset from Narsarsuaq was worse than chance when it came to predicting false positives from Thule (Balanced Accuracy = 35 %). This suggests that our model is very prone to over-fitting to one specific location.

4.2 Further Analysis of the Data

Because of the exploratory nature of this study and due to time constraints much still remains to be said about these data. Below we go more in detail with some considerations for future investigations of the data.

One Model to Predict Them All In order to most effectively distinguish between true- and false positives we desire a model that is able to generalise across all locations, and not over-fit to one location as we saw in Table 3. One way to solve this could be to include location as a random effect in the model, or to simply make sure that the model is trained on a more balanced dataset. As was the case here that almost all of the false positives from Thule consisted of variants of the same image seen in Figure 2b, whereas the false positives from Narsarsuaq contained a much greater variety of images. Location independence is not the only desirable feature of a future model however. One could also imagine integrating features such as

Other factors that would be interesting to include in future models could be metadata such as the time of day, which to a certain extent could code for we should also look at light conditions, time of day/year (lightning) - baybe include as random predictors where is the sun (lengde breddegrader/position of camera) Taking factors into account about where you are when images are taken!

First of all, the measures investigated here are far from being exhaustive. As is the case with any model, the models presented here are a simplification of reality and are only allowed to explore the model space provided via the predictors. There might be other factors not included here which are even better at explaining the differences in the network predictions. As we discovered in our model cross-validation

The Aspect of Time During any test/train regime such as the cross-validation performed here it is important that the is no leakage of information between the training and testing sets. This is to make sure that the model is not trained on the same data it is evaluated against, as this would lead to over-fitting. The image data presented in this article could be argued to be serially correlated (so-called *autocorrelation*), as it consists of images of the same flower over time. Therefore, in principle, it is important to make sure that images containing the same flower are kept in separate folds during cross-validation. However, due to the way the original images were encoded we did not have this information available, which unfortunately causes our results to be less robust. Future studies should strive to include time-sensitive measures in the analysis so as to avoid over-fitting and increase generalisability of the results. This could also be done by including time as a random effect in the linear model.

Adding Anoter Layer Another thing that might be interesting is to train a CNN to distinguish between the true and false positives. If a high enough accuracy was obtained, information from such an additional network could be added as an extra layer to the already existing MASK R-CNN to improve performance. However, all the considerations we have discussed here with regards to the input data still apply, and, one might argue, to an even greater extent, as the inner workings of a neural networks cannot be meaningfully investigated with methods currently available. It is therefore highly important that we gain a deeper understanding of the data which is fed into the network, in order to better understand the predictions made by the network. Some might argue that what exactly the network is doing is not important as long as it ”works”, meaning that it makes good predictions. However, we argue that it is ill-advised to blindly trust the (the field known as Explainable AI)

5 Conclusion

In this paper we investigated systematic differences between true- and false positives from a MASK R-CNN detecting flowers from time-lapse images. We implemented novel measures of complexity in python and were able to replicate results from an original paper. Using both simple and complex image features we ran large scale cross validations in order to find the model parameters that best described the variance in our data.

We found that neither of the complexity measures were very good at explaining the data alone, but that in combination with simpler image features such as size and ratio they contributed with valueable information.

We concluded that

References

Abhishek, K., Singh, M., Ghosh, S., & Anand, A. (2012). Weather Forecasting Model using Artificial Neural Network. *Procedia Technology*, *4*, 311–318. https://doi.org/10.1016/j.protcy.2012.05.047

Bansal, R., Raj, G., & Choudhury, T. (2016). Blur image detection using Laplacian operator and Open-CV, In *2016 International Conference System Modeling Advancement in Research Trends (SMART).* https://doi.org/10.1109/SYSMART.2016.7894491

{Danmarks Frie Forskningsfond}. (2019). Aarets originale idé 2018.

Deeplearning.ai. (2017). C4W3L01 Object Localization.

Ekici, S., & Jawzal, H. (2020). Breast cancer diagnosis using thermography and convolutional neural networks. *Medical Hypotheses*, *137*, 109542. https://doi.org/10.1016/j.mehy.2019.109542

Field, A. P., Miles, J., & Field, Z. (2012). *Discovering statistics using R / Andy Field, Jeremy Miles, Zoë Field.* [Accepted: 2018-11-05T09:29:48Z]. London ; Thousand Oaks, Calif. : Sage,

Krastev, P. G. (2020). Real-time detection of gravitational waves from binary neutron stars using artificial neural networks. *Physics Letters B*, *803*, 135330. https://doi.org/10.1016/j.physletb.2020.135330

Olsen, L. R. (2020a). Cvms.

Olsen, L. R. (2020b). Groupdata2.

{Python Software Foundation}. (2020). Python.

{R Core Team}. (2020). R: A Language and Environment for Statistical Computing. *Vienna, Austria.*

Vestfalen, J. (2019). Aarhus: Hele byen er digitalt testcenter.

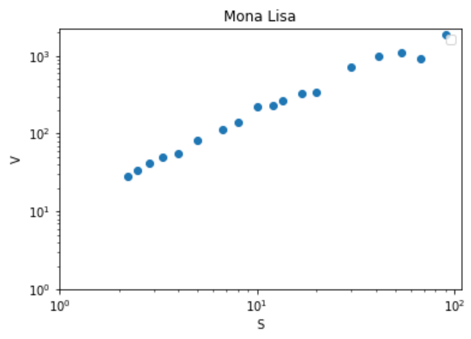
Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Yu, H., & Winkler, S. (2013). Image complexity and spatial information, In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, Klagenfurt am Wörthersee, Austria, IEEE. https://doi.org/10.1109/QoMEX.2013.6603194

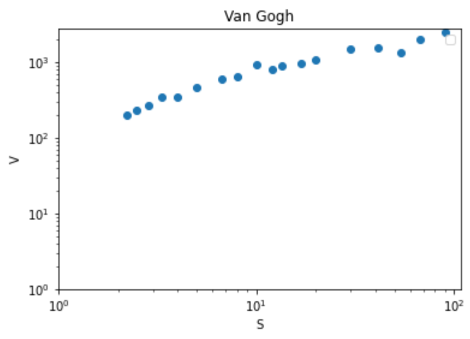
Zanette, D. H. (2018). Quantifying the complexity of black-and-white images (D. R. Chialvo, Ed.). *PLOS ONE*, *13*(11), e0207879. https://doi.org/10.1371/journal.pone.0207879

A The Q Complexity Measure

In Figure 3 we present two example variance profiles of the images from (Zanette, 2018) recreated using the methods described in 2.2.1. Compare these to the original variance profiles seen in Figure 4.



(a)



(b)

Figure 3: Variance profiles of the a) Mona Lisa and b) Van Gogh images recreated in python

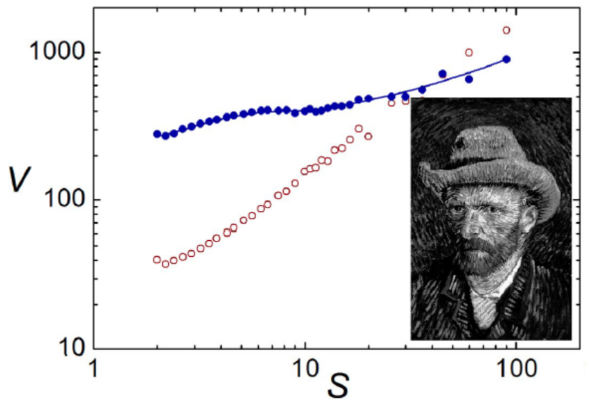


Figure 4: Original variance profiles of the Mona Lisa (red) and Van Gogh (blue) images. Image is an exempt from (Zanette, 2018)

B Interaction Plots

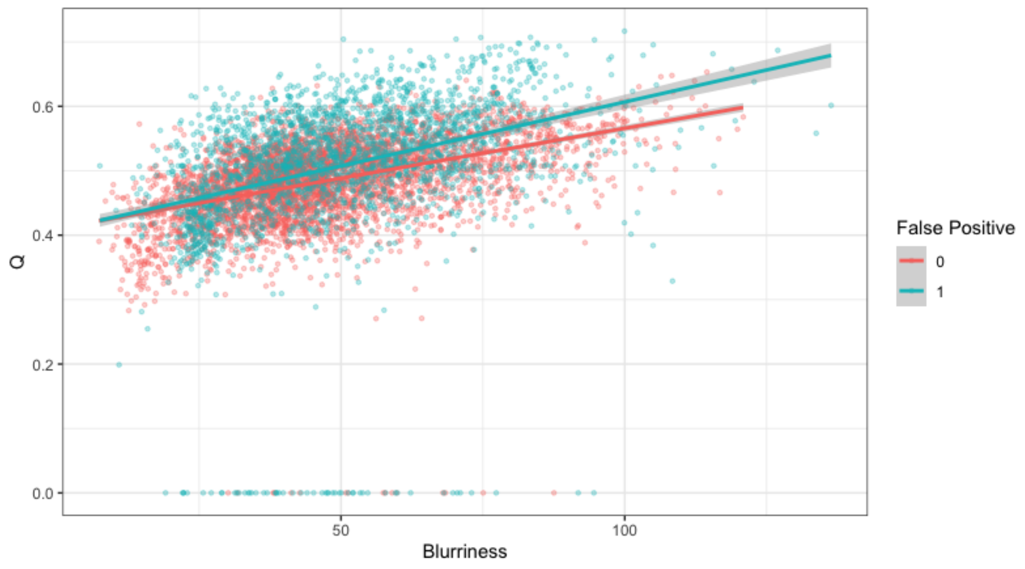


Figure 5: Interaction effect between the parameters Blurriness and Q

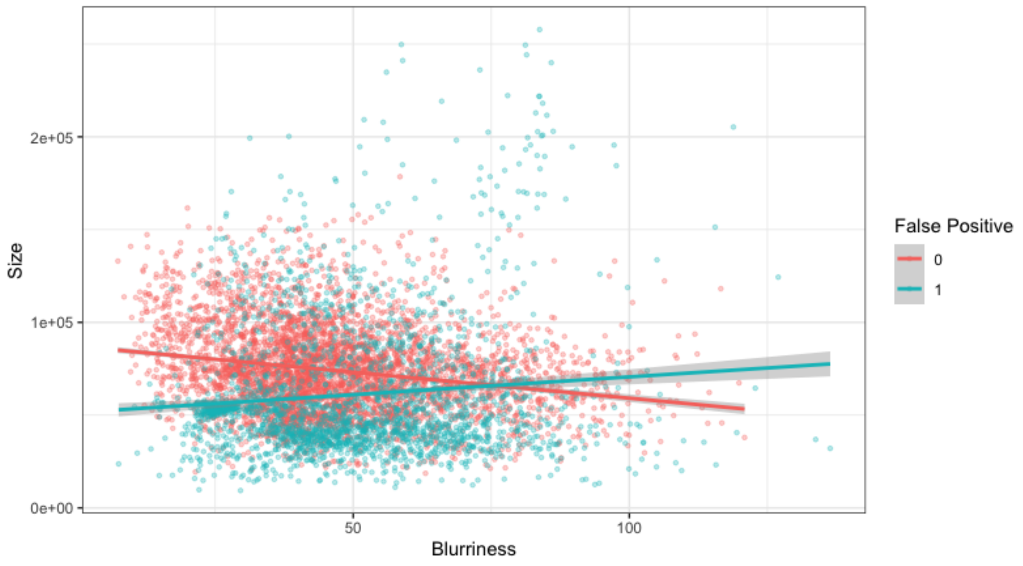


Figure 6: Interaction effect between the parameters Blurriness and Size

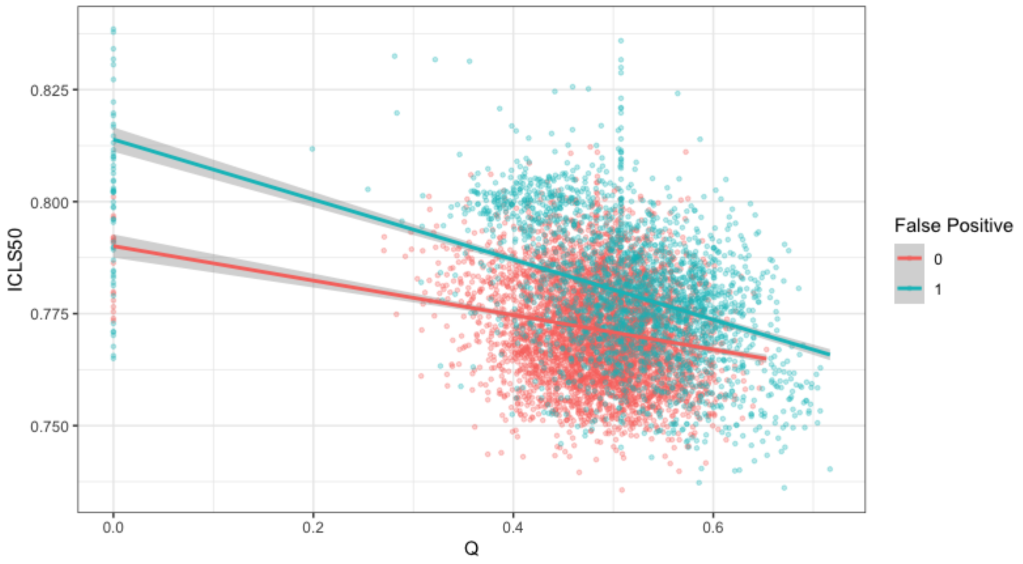


Figure 7: Interaction effect between the parameters Q and ICLS50

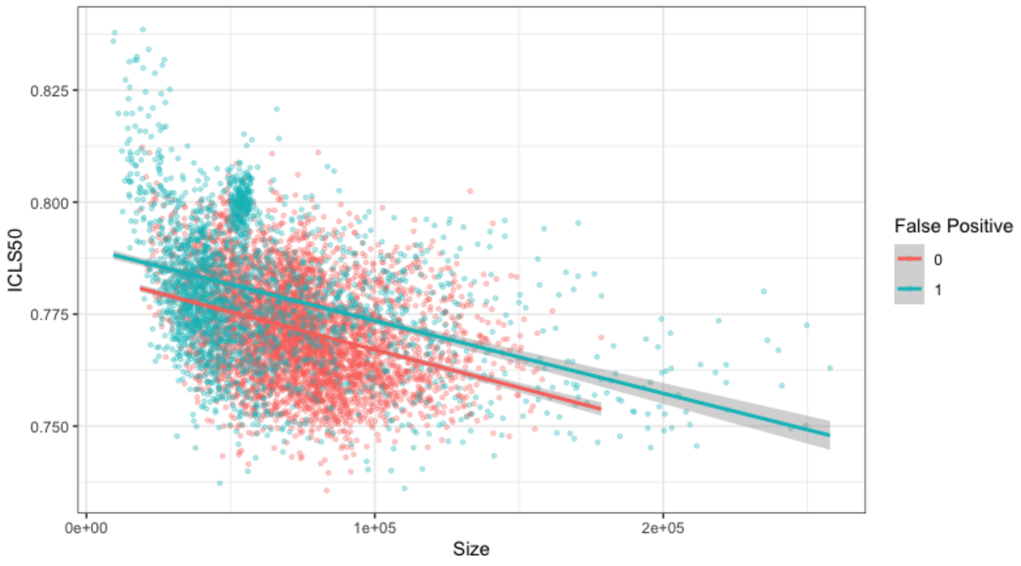


Figure 8: Interaction effect between the parameters Size and ICLS50

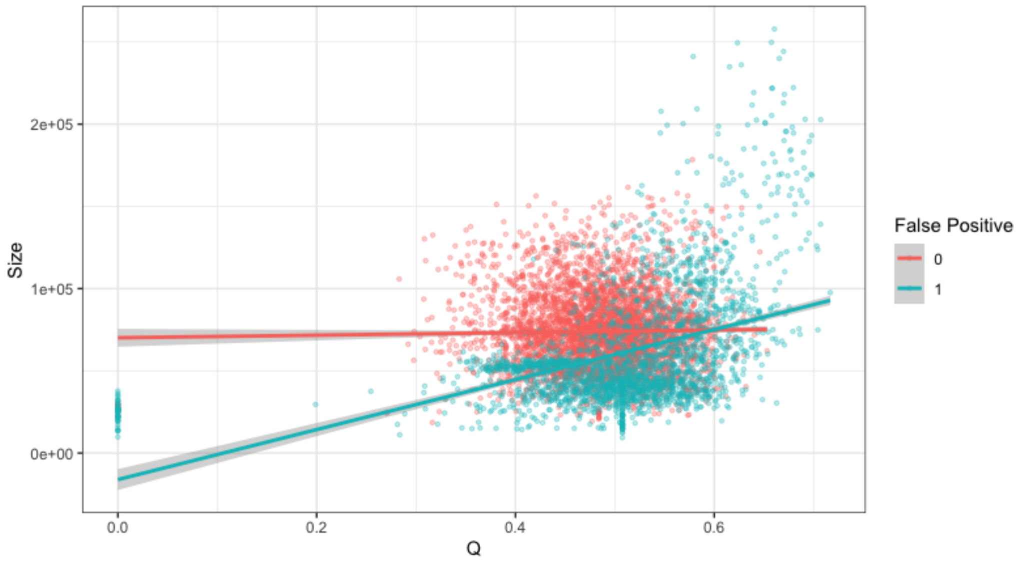


Figure 9: Interaction effect between the parameters Q and Size

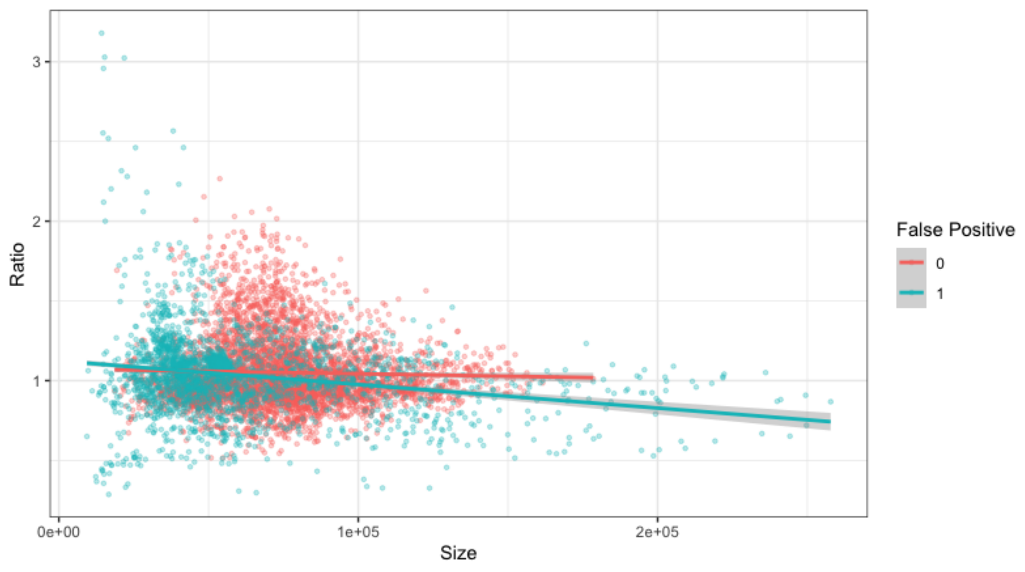


Figure 10: Interaction effect between the parameters Size and Ratio (height/width)

C Multicollinearity Check

Table 4: VIF stats for the highest scoring model

	VIF	1/VIF
blur	84.63	0.01
Q	6544.67	0.00
size	4715.35	0.00
ICLS50	97.29	0.01
ratio	9.19	0.11
blur:Q	132.37	0.01
blur:size	22.39	0.04
Q:ICLS50	5627.61	0.00
size:ICLS50	4126.69	0.00
Q:size	157.25	0.01
size:ratio	35.07	0.03

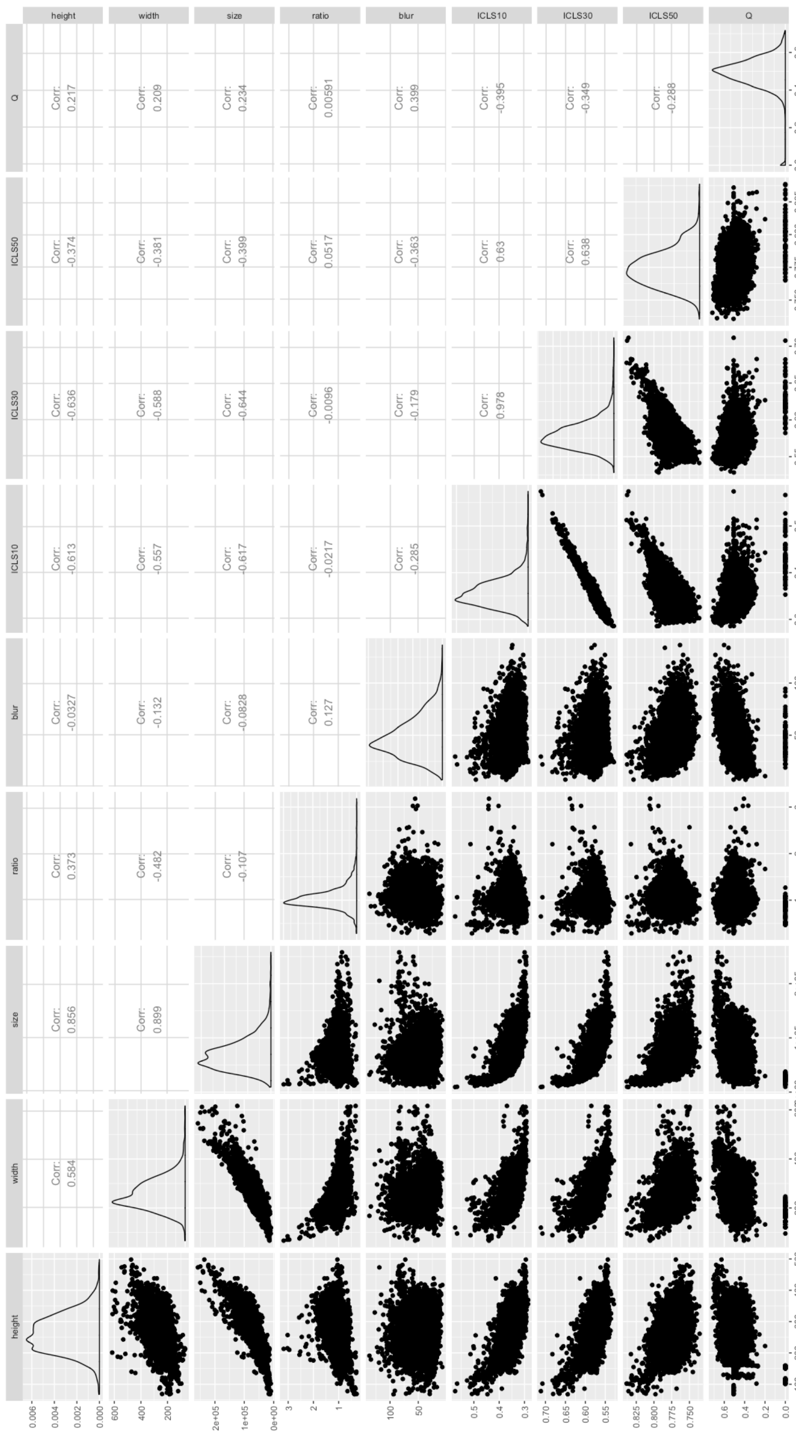


Figure 11: Multicollinearity matrix for the metadata parameters