Exploration of resolution specialization in neurons of deep convolutional neural networks

Statistics Project

Author - Andrey Gizdov*
Bocconi University
andrey.gizdov@studbocconi.it

Author - Ali Akhtari
Bocconi University
ali.akhtari@studbocconi.it

October 8, 2023

1 Introduction

Convolutional Neural Networks (CNNs) are frequently used machine vision systems for the task of object detection (classifying pixels in an image as belonging to a class of objects) [1]. It is not clear, however, how a CNN tackles the challenge of combining several resolutions within a single image to come up with a final prediction (Figure 1).

The human visual system spans 120 degrees and benefits tremendously from the ability to combine several resolutions. For example, to notice the subtle movements of an approaching predator in the vast forest, which might pose a danger, a large FOV is required. We commonly use our low-resolution peripheral vision for such tasks. On the other hand, guiding a thread through the eye of a needle requires exquisite resolution. The center of our visual field, the fovea, possesses a very high peak resolution, unlike the periphery, and allows us to do that too [2]. Both tasks have utility and we possess the visual capabilities to tackle both, simultaneously. Our visual cortex has internal adaptations to combine those different types of context - provided by the peripheral vision and the high-resolution fovea - at any given time in order to produce one "whole" perception of what is happening around us [3].

It is not clear, however, if and how CNNs are adapted when presented with multiple resolutions. The convolutional kernels are commonly applied uniformly across the entire image, and in this study, in part inspired by the human vision, we pose the question: what internal representations are learned in CNNs to account for the optimization problem of combining several resolutions within the input image?

2 Hypothesis

It is established in the literature that CNNs often dedicate their initial convolutional layers during optimization to the detection of simple shapes (edges, corners etc.) [4] (Figure 2).

It is not difficult to imagine, in the simple case of vertical edge detection, that the outer part of the right image in Figure 1 would require a slightly different convolutional kernel than the left image: after all, one kernel needs to produce a high dot product after being applied across a blurry edge, whereas the other across a sharp one (the convolutional operation is $g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x-dx, y-dy)$, where $g(x,y)$ is the generated feature map, $f(x,y)$ is the original image, and ω is the kernel).

However, we must consider that a CNN applies its convolutional kernels across the entire image and not selectively to areas of a particular resolution. Given an image containing several resolution types, it may be the case that it learns one single kernel, which is the average of the "ideal" kernels for the different edge resolutions.

At the upper-level layers of the network, where we typically deal with detecting more complex shapes [4], it is even less obvious whether the network keeps separate kernels for, for example, WLOG, detecting a low-resolution person vs

*Weizmann AI Center, Weizmann Institute of Science

a high-resolution person. It could be that it indeed keeps some form of separation between the kernels for low/high-resolution people, but it could also be that it learns a single set of kernels ideal for the detection of people at a "mid-level" resolution. A kernel as such would still work in a low/high-resolution context, but less precisely than a resolution-specific kernel would. However, considering that the network has to apply the learned kernels to the entire image and not selectively, and that it has a limited number of neurons, this scenario is certainly plausible.

An abstraction of this example is exactly what we explored in this study. In short, our research question is: *do the kernels of high-level convolutional layers, often responsible for detecting high-level features (body parts, object segments etc.) rather than edges and corners, specialize in detecting the same type of object in different resolutions?* **Or** *has the optimization process learned kernels that somehow combine low and high-resolution features into a similar feature map, regardless of what resolution a particular type of object occupied?*

3 Data extraction

To test this, we trained a Mask-RCNN model on the COCO Dataset [5], with a ResNet-101 backbone pre-trained on the ImageNet dataset [6]. Identical processing was performed on all COCO and ImageNet training images, corresponding to a filter of variable-resolution that decreases with eccentricity (right side of Figure 1). This in turn provides the desired optimization task: object detection on images of a continuum of resolutions. We then inferred $2,500$ images of variable-resolution with predominant high-resolution (*type 1*) and $2,500$ images of variable-resolution with predominant low-resolution (*type 0*) (Figure 3 - 5).

Note:

1. All inferred images were of different objects, so as to achieve independence of the extracted tensor samples.
2. *type 0* and *type 1* resolutions are both from the continuum of resolutions present in each image sample of the training set.
3. About 80 object types were inferred to the network. *type 1* and *type 0* resolution images both contained every type of object, in equal numbers (with different instances of that object, of course). This ensures that our statistical test differentiates the extracted tensors *only* based on their resolution (discussed later).

The inference of each image produced a feature map (tensor) of size varying from $1024 \times 50 \times 50$ to $1024 \times 70 \times 70$, depending on the size of the image. This feature map was extracted during inference from the output of layer:

$$backbone.body.layer3.block22.batchnorm3$$

As alluded to earlier, each "slice" in those 1024 indices is a matrix representing the dot product of a convolutional kernel with another feature map, produced by the previous layer. As such, the produced $5,000$ feature maps of average size $\approx 1024 \times 60 \times 60$ effectively represent the *activation values* of 1024 convolutional kernels when presented with opposing resolution types.

If the network indeed keeps a separation between the kernels dedicated to resolution *type 0* and resolution *type 1*, we would expect higher activation values for particular matrices (in the total of 1024) only when inputting a *type 1* or *type 0* resolution image into the network. On the contrary, if the network combines the low and high-resolution features regardless of what resolution an object occupied, we would expect an indistinguishable activation pattern in the matrices of the feature maps. The following section discusses how this question was tested.

Note: for the purpose of data-reduction, each of the $5,000$ feature maps of size $\approx 1024 \times 60 \times 60$ was reduced to a tensor of size $1024 \times 3 \times 1$, by taking the *average, median and maximum* of each individual matrix. Our results suggest that those descriptive statistics are sufficient.

4 Hypothesis Testing

Hypothesis Given a sample of $5,000$ tensors (each with size $1024 \times 3 \times 1$), generated by the two different types of source images (specifically, of *type 1* and *type 0* resolution), we want to establish whether there's a statistically significant difference between the activation pattern of the tensors solely depending on the resolution type. In the setting of our experiment, we have ensured that *all* other factors have been kept the same in the generation of the tensors (check the previous section).

Two-sample tests Let $X_1, \ldots, X_{n_0}$, $Y_1, \ldots, Y_{n_1}$ be independent, random vectors of real numbers, representing our tensors generated by resolution *type 1* and resolution *type 0* respectively, such that $X_i \sim \mathbb{P}_0$ and $Y_i \sim \mathbb{P}_1$, with $\mathbb{P}_0$, $\mathbb{P}_1$ probability distributions. The hypothesis test is then reduced to:

$$H_0 : \mathbb{P}_0 = \mathbb{P}_1 \text{ vs. } H_1 : \mathbb{P}_0 \neq \mathbb{P}_1$$

Most two-sample tests assume some extent of normality.

Non-parametric two-sample tests Lacking further information on the inner machinery of the training process and the distribution of kernel populations, we must establish a non-parametric, general approach where minimal assumptions are made about the data generation process. Many non-parametric models, however, such as kernel density estimation, are not typically feasible in high-dimensional settings [12]. Consequently, we need a model that doesn't require an intermediate density estimation to avoid the curse of dimensionality. Putting the pieces together, we could utilize a kernel two-sample test [13]. We will, instead, use a classification model as a proxy for a two-sample test.

Classification Let $\mathbb{P}_0, \mathbb{P}_1$ be two different probability distributions, without any assumptions made about them. Let $\mathcal{C}$ a classifier model. $\mathcal{C}$ is a simple function from the set of all possible samples taken from $\mathbb{P}_0$, $\mathbb{P}_1$ to 0,1, indicating whether the sample belongs to $\mathbb{P}_0$ or $\mathbb{P}_1$. We can denote the function as $\mathcal{C} : \mathbb{X} \to \{0, 1\}$.

Two-sample test based on classification accuracy Classification-based approaches have been proposed for two-sample testing on high-dimensional, complex data. [7, 8, 9, 10, 11] If a classification model can sufficiently discriminate between samples from two populations, it's reasonable to assume that the underlying data generation processes are different. Consequently, the accuracy of a classification model (preferably, if not imperatively, over out-of-sample data) must be a constituent of the test statistic. [9, 10] discuss the consistency and power of a classifier-based test compared to the minimax power, [8] proposes an estimation of the likelihood ratio by the odds ratio of the classification probability, and [7] uses the score assigned to each datapoint by the classifier to reduce the dimensions and run a single-dimensional two-sample test.

Assumptions We'll introduce assumptions under which the tests introduced in the next two chapters operate. Denoting the size of the samples taken from $\mathbb{P}_0$, $\mathbb{P}_1$ as n_0, n_1, respectively, tr as the size of the training sample, and defining $n \equiv n_0 + n_1$, as $n \to \infty$:

I. $\exists \lambda \in (0, 1) : n_0/n \to \lambda$

II. $\exists \kappa \in (0, 1) : tr/n \to \kappa$

Both assumptions can easily be satisfied and are introduced to avoid edge cases (particularly, the cases where all samples are drawn from one population or when the splitting ratio is 0 or 1). The final assumption refers to the classification error. We first need to introduce the apparatus necessary to study the accuracy of a classifier.
Let $\mathcal{C}_{X,Y}$ a model trained on $X_0, \ldots, X_{n_0}$ and $Y_0, \ldots, Y_{n_1}$. Denote

$$E_0 \equiv \mathbb{E}_n[P(\mathcal{C}_{X,Y}(Z) = 1 | Z \sim \mathbb{P}_0)]$$

and

$$E_1 \equiv \mathbb{E}_n[P(\mathcal{C}_{X,Y}(Z) = 0 | Z \sim \mathbb{P}_1)]$$

finally let $E \equiv (E_0 + E_1)/2$. We now need to estimate E_0, E_1 by splitting our sample into test and training. $\mathcal{C}$ is the trained model on $X_1, \ldots, X_{tr_0}$, $Y_1, \ldots, Y_{tr_1}$. tr_0, tr_1 are respectively the number of datapoints in the training sample drawn from $\mathbb{P}_0$, $\mathbb{P}_1$. Similarly, let te_0, te_1 the number of datapoints in the test sample drawn from $\mathbb{P}_0$, $\mathbb{P}_1$, $te \equiv te_0 + te_1$. Denote the ith datapoint in the test sample by X_i, Y_i.

$$\hat{E_0} \equiv \frac{1}{te_0} \sum_{i=1}^{te_0} \mathbb{1}(\mathcal{C}(X_i) = 1)$$

$$\hat{E_1} \equiv \frac{1}{te_1} \sum_{i=1}^{te_1} \mathbb{1}(\mathcal{C}(Y_i) = 0)$$

$$\hat{E} \equiv (\hat{E_0} + \hat{E_1})/2$$

Intuitively, if the null hypothesis holds, $\hat{E}$ converges to 1/2. The last assumption addresses this issue.

III. Under the alternative hypothesis, $\exists \epsilon > 0 : E(\mathcal{C}) = 1/2 - \epsilon$ and $\hat{E}(\mathcal{C}) = E(\mathcal{C}) + o(1)$

The third assumption, again, holds as long as the model is reasonably accurate. We simply state that if the null is rejected, then the accuracy must be better than a coin toss.

Asymptotic method With these three assumptions in mind, notice that the errors are essentially sums of random variables conditioned on the training set and can be modelled asymptotically by a normal distribution. Consider the following test statistic:

$$T \equiv \frac{2\hat{E}(\mathcal{C}) - 1}{\sqrt{\hat{E}_0(\mathcal{C})[1 - \hat{E}_0(\mathcal{C})]/te_0 + \hat{E}_1(\mathcal{C})[1 - \hat{E}_1(\mathcal{C})]/te_1}}$$

The hypothesis test defined by it is

$$\mathcal{H}_a \equiv \mathbb{1}(T < -z_\alpha)$$

[9] p. 66 proves that given I, II, III, $\lim_{n\to\infty} \mathbb{E}(\mathcal{H}_a|\mathbb{P}_0 = \mathbb{P}_1) < \alpha$ and $\lim_{n\to\infty} \mathbb{E}(\mathcal{H}_a|\mathbb{P}_0 \neq \mathbb{P}_1) = 1$. The test is therefore consistent, as asymptotically, under the alternative hypothesis, the power converges to 1.

Half-permutation method Following the instructions set out in [9], we divide the data into two parts, $\mathcal{X}_{tr_0}, \mathcal{Y}_{tr_1}$ ($\mathcal{X}_{tr_0} = X_1, \ldots, X_{tr_0}$, $\mathcal{Y}_{tr_1} = Y_1, \ldots, Y_{tr_1}$) and $\mathcal{X}_{te0}, \mathcal{Y}_{te1}$ and train a logistic classifier on $\mathcal{X}_{tr_0}, \mathcal{Y}_{tr_1}$ and measure its accuracy over $\mathcal{X}_{te_0}, \mathcal{Y}_{te_1}$. The accuracy of the model can be estimated by

$$\hat{\alpha_*} = \frac{1}{te} \sum_{i=1}^{te_0} \mathbb{1}(\mathcal{C}(X_i) = 0) + \frac{1}{te} \sum_{i=1}^{te_1} \mathbb{1}(\mathcal{C}(Y_i) = 1)$$

Where X_i, Y_i are the ith datapoint in $\mathcal{X}_{te0}$, $\mathcal{Y}_{te1}$. For convenience, we'll denote $\hat{\alpha_*}$ as α_*.
We then merge $\mathcal{X}_{te0}$, $\mathcal{Y}_{te1}$ into one pile, permute them, and split them into two equal halves to generate $\mathcal{X}_1$, $\mathcal{Y}_1$. By testing the trained model on the new sample, α_1 is obtained. We repeat the same process P times, for $P > (1 - \alpha)/\alpha$ (particularly, $P \geq 20$ for $\alpha = .05$).

Sorting $\alpha_*, \alpha_1, \ldots, \alpha_P$, we assign an order statistic to each index. Let $k = \lceil (1 - \alpha)(1 + P) \rceil$. The hypothesis test can be represented by the following indicator function:

$$\mathcal{H}_p \equiv \mathbb{1}(\alpha_* > \alpha^{(k)})$$

where $\alpha^{(k)}$ is the kth order statistic of the sample. [9] p. 67 proves the consistency of this test and that the size, asymptotically and under the null hypothesis, will be less than α.

5 Implementation

Classifier model Establishing the theoretical background behind the classification-based hypothesis test, we'll utilise a logistic regression classifier as a simple, flexible baseline model. Refer to Appendix, A for a theoretical discussion of the regression.

Setup [9] suggests $n \geq 400$ for $d \geq 200$, and a train-test ratio, κ of 1/2. We used $n = 5000$, $d = 2048$, and $\kappa = 1/2$.

Independence of datapoints We initially used the same picture twice in the database, once with constant and another time with variant resolution. However, this methodology was inconsistent with the assumption that datapoints are independent (particularly, for the errors of the classification model to converge to a normal distribution, we need that assumption). Therefore, we dropped half of the database and moved from $n = 10000$ to $n = 5000$.

Multicollinearity of features The mean and median activation value of the neurons in each layer were generally correlated. As a result, we removed the median to satisfy the requirements of the logit model, going from $d = 3072$ to $d = 2048$.

Logistic regression The logistic regression was run with an L2 penalty term and the Limited-memory BFGS solver algorithm, allowing no more than 500 iterations.

Results The model produced an accuracy of .9532, easily passed the Z-test with $\alpha = .01$, and generated the following confusion matrix:

$$\begin{bmatrix} 1213 & 51 \\ 66 & 1170 \end{bmatrix}$$

Note that the confusion matrix is defined as

$$CNF \equiv \begin{bmatrix} \sum_{i=0}^{te_1} \mathbb{1}(\mathrm{C}(Y_i) = 1) & \sum_{i=0}^{te_0} \mathbb{1}(\mathrm{C}(X_i) = 1) \\ \sum_{i=0}^{te_1} \mathbb{1}(\mathrm{C}(Y_i) = 0) & \sum_{i=0}^{te_0} \mathbb{1}(\mathrm{C}(X_i) = 0) \end{bmatrix}$$

Features Maximum value of activation tends to have a higher coefficient (Figure 6), as the top 100 features based on coefficients are max columns (Figure 7).

Permutation test Restricting $P > (1-\alpha)/\alpha$ for $\alpha = .01$, we have $P \geq 100$, where P is the number of times the labels are permuted. We set $P = 100$. The model, expectedly, passed the permutation test as well (Figure 8). Below you can find the ordered, minimized list of 100 permuted accuracies and the model accuracies.
0.4748, 0.4804, 0.482, ..., 0.5196, 0.5228, 0.5284, **0.9532**.

6 Conclusions

Given that *two* tests were rejected with very high certainty, we conclude that there indeed are significant differences in the activation patterns of the neurons in the network governed by resolution. This suggests that the network indeed learns separate convolutional kernels for processing objects at different resolutions, rather than a single set of uniformly applicable kernels with less precision for any given resolution type.

However, based on the statistical analysis, the maximum activation values of the neurons seem to be higher for images of *type 1*. This suggests that the network has dedicated more resources to the detection of high-resolution objects, rather than low-resolution ones. This could be due to the distribution of objects in the training set: indeed most objects in COCO are located around the middle, where we have higher resolution in the original training samples.

7 Topics for further investigation

It is interesting to see if the CNN keeps this separation into the higher-level convolutional layers. The tensors we extracted from the network were not from its highest-level layers, only about halfway depth-wise. One could further study whether there exists *any* point in the network architecture where it actually combines the low and high-resolution features into one "whole" indistinguishable feature map, similar to the human brain [3].

Additionally, it would be interesting to see if any of the convolutional kernels produce higher activation values for *type 0* images on average; this would suggest that the network has indeed devised a sharper separation between kernels for particular resolutions during optimization, at least at some level of the architecture.

For the hypothesis test, one could study the distribution of features and test certain assumptions on them to facilitate the choice of the two-sample test. With more time and computational power, one could run a full-permutation classification-based test on data, where both test and training data are merged and permuted P times. While both half-permutation and full-permutation methods offer a valid α, the full-permutation might be more powerful as it re-trains the model on a permuted training set. The odds ratio of the classification probability times a constant could be used to approximate a likelihood ratio test to increase the computational efficiency (by eliminating the need for cross-validation) and provide a more powerful test [8].

Finally, one could reduce the dimension of the features to one by assigning the output of the classifier's decision function (the classification score, $logit^{-1}(\beta \cdot X)$) to each datapoint and running a single-dimensional two-sample test on the scores [7]. However, the reader must be aware that the distribution of the classification scores was determined to deviate from the normal distribution by D'Agostino-Pearson Omnibus Test [14, 15] based on skew and kurtosis (Figure 9). The test produced a $s^2 + k^2$ of 408.3 (s is the Z-score of the skew test, k the Z-Score of the kurtosis test) and a two-sided chi-squared p-value of 2.1465653015784047e-89.

8 Appendix

A. Logit model

Expit function Consider a differential equation that models a linearly decreasing growth rate.

$$\frac{dP(t)}{dt} = P(t)r(1 - \frac{P(t)}{A})$$

for $r > 0$, $A > P(0)$. The solution is of form

$$P(t) = \frac{A}{1 + (A/P(0) - 1)e^{rt}}$$

which is noted as a logistic or expit function. Setting $r = 1, A = 1$, the solution is the standard logistic function. Geometrically the function can be re-written as

$$f(t) = \frac{L}{1 + e^{-k(t-t_0)}}$$

and forms a sigmoid curve with t_0 as the midpoint, L the supremum, and k the steepness of the curve. Assume t can be described by the design matrix, X, $t = \beta \cdot X = \beta_1 x_1 + \ldots + \beta_n x_n$. For $L = 1, k = 1$,

$$f(t) = p(x) = \frac{1}{1 + e^{-(\beta \cdot X)}}$$

where $p(x)$ in the logit model represents the probability of $Z \sim \mathbb{P}_1$, a success case.

Logit function Denote $q \equiv \frac{p}{1-p}$. Consider the inverse of the expit function, $logit(p) = \ln q = \ln(\frac{p(x)}{1-p(x)}) = \beta \cdot X$. Raising both sides to the exponential, $q = e^{\beta \cdot X}$.

Logit model The optimal β is found by using an MLE approach maximizing the log-likelihood function. Intuitively, we want p to return a value close to 1 when x is a case, and 0 when it's a failure.

$$\mathcal{L}(\beta) = \prod_{x_i \sim \mathbb{P}_1} p(x_i) \times \prod_{x_i \sim \mathbb{P}_0} (1 - p(x_i))$$

Taking the logarithm

$$\ell(\beta) = \sum_{i=1}^{n} y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))$$

where $y_i = \mathbb{1}(x_i \sim \mathbb{P}_1)$.

B. Figures

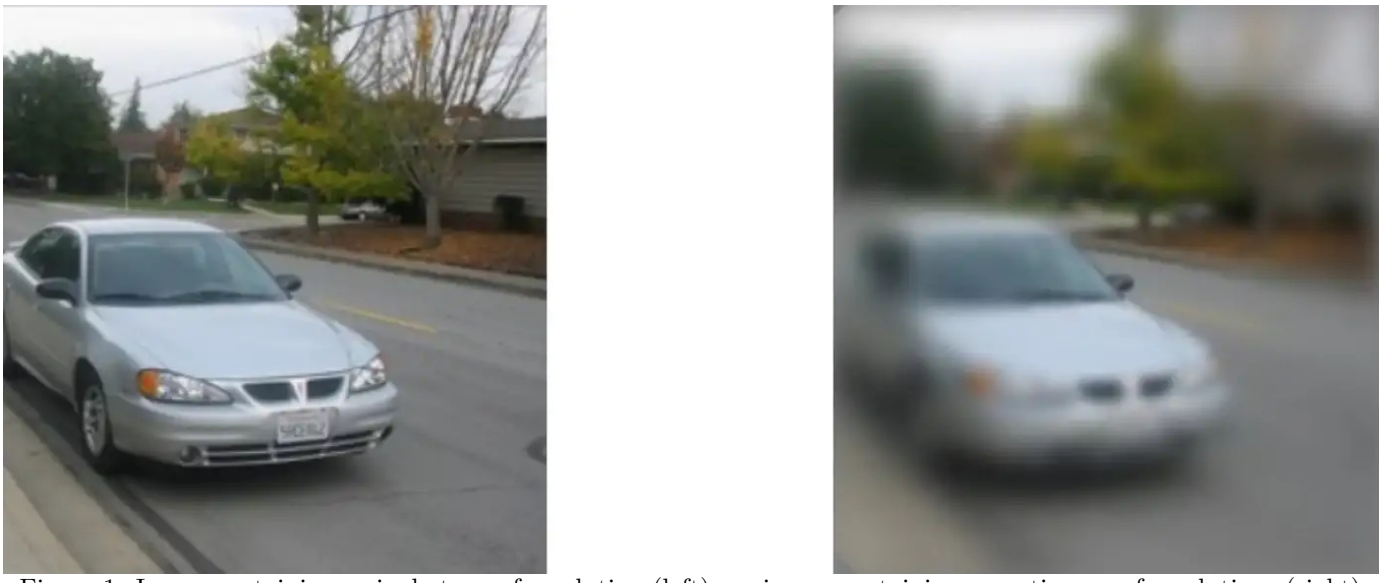


Figure 1: Image containing a single type of resolution (left) vs. image containing a continuum of resolutions (right)

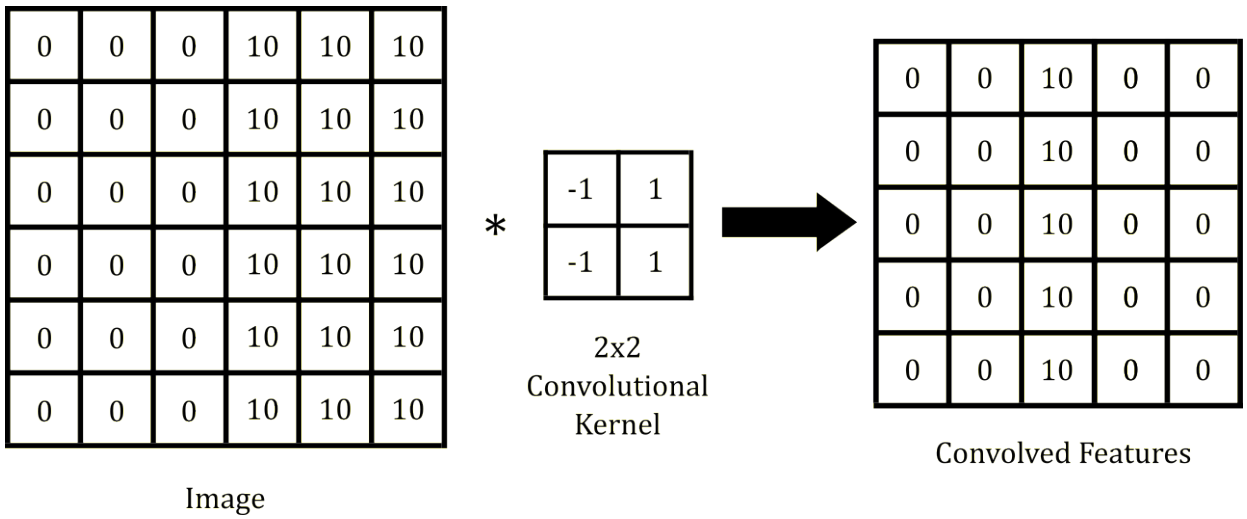


Figure 2: Simple image matrix and a convolutional kernel for vertical edge-detection

Figure 3: CNN training image

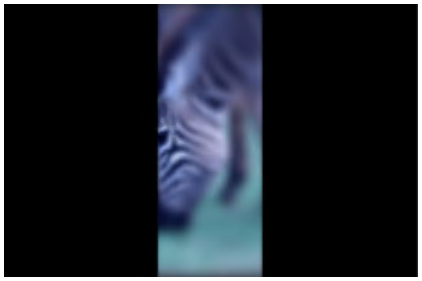


Figure 4: Type 1 resolution image

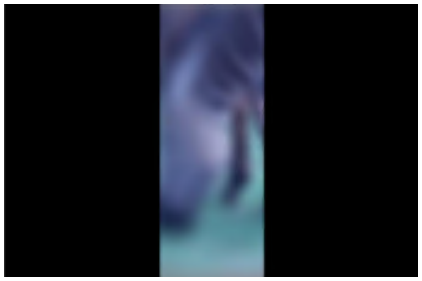


Figure 5: Type 0 resolution image

Figure 6: Regression coefficients assigned to mean and max columns.

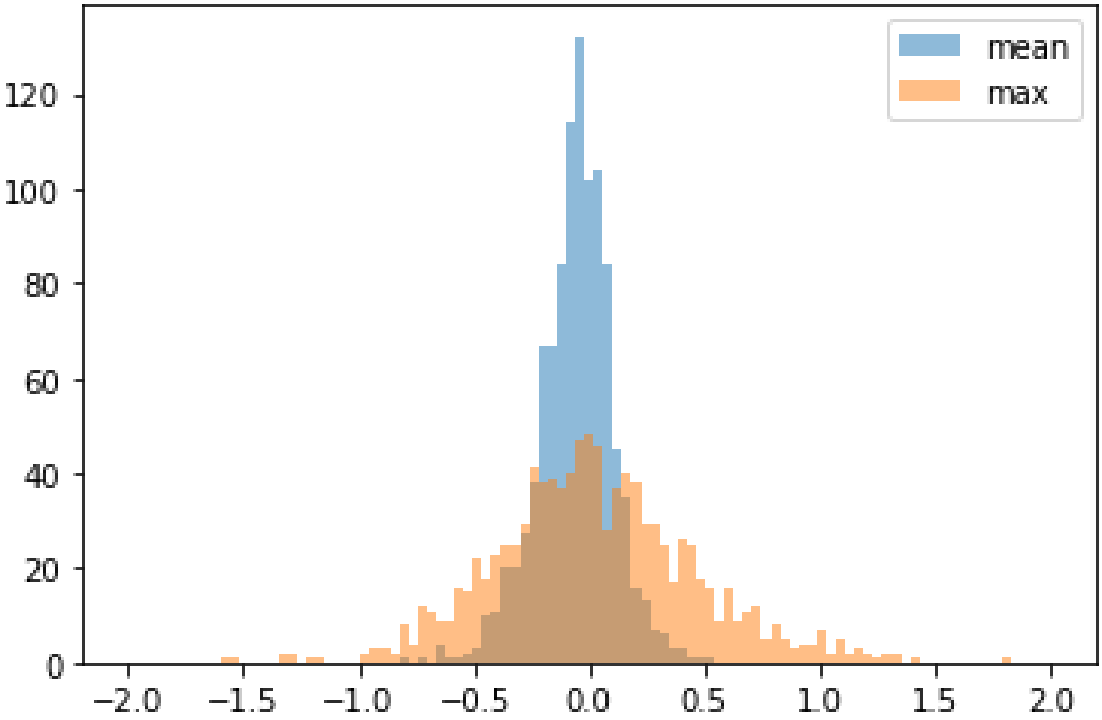


Figure 7: Top 9 features based on the absolute regression coefficients. Cases (type 1 resolutions) seem to have a higher maximum and a fatter right tail.

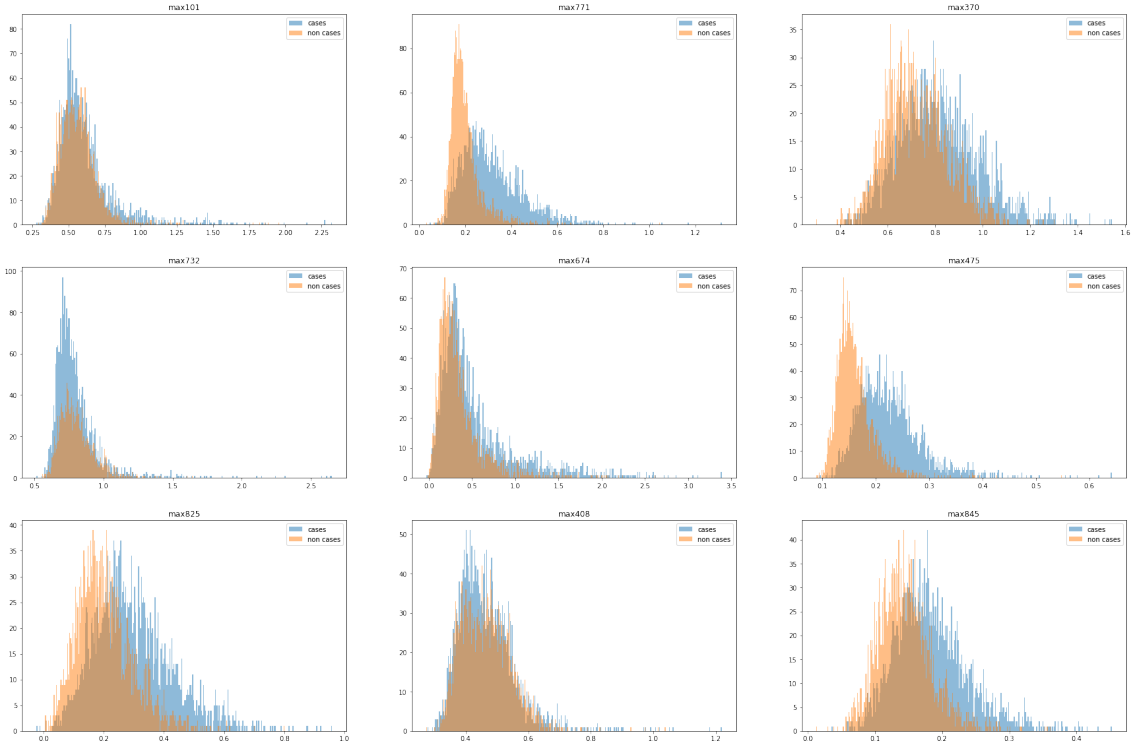


Figure 8: The model passed the permutation test.

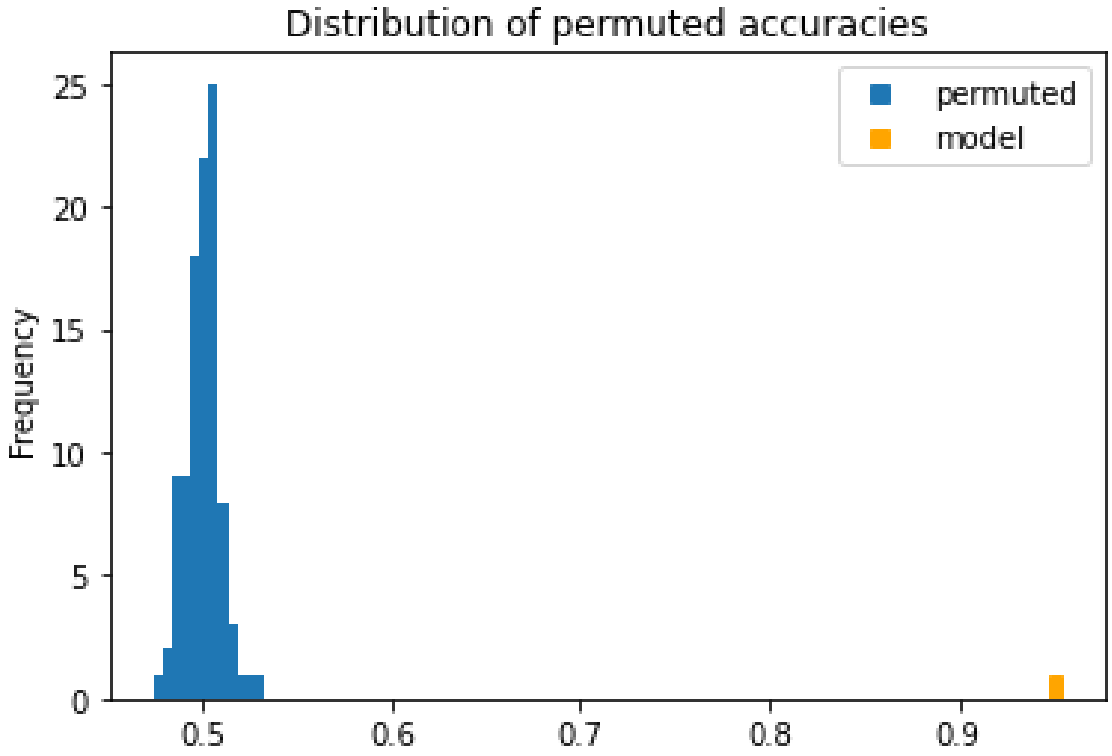
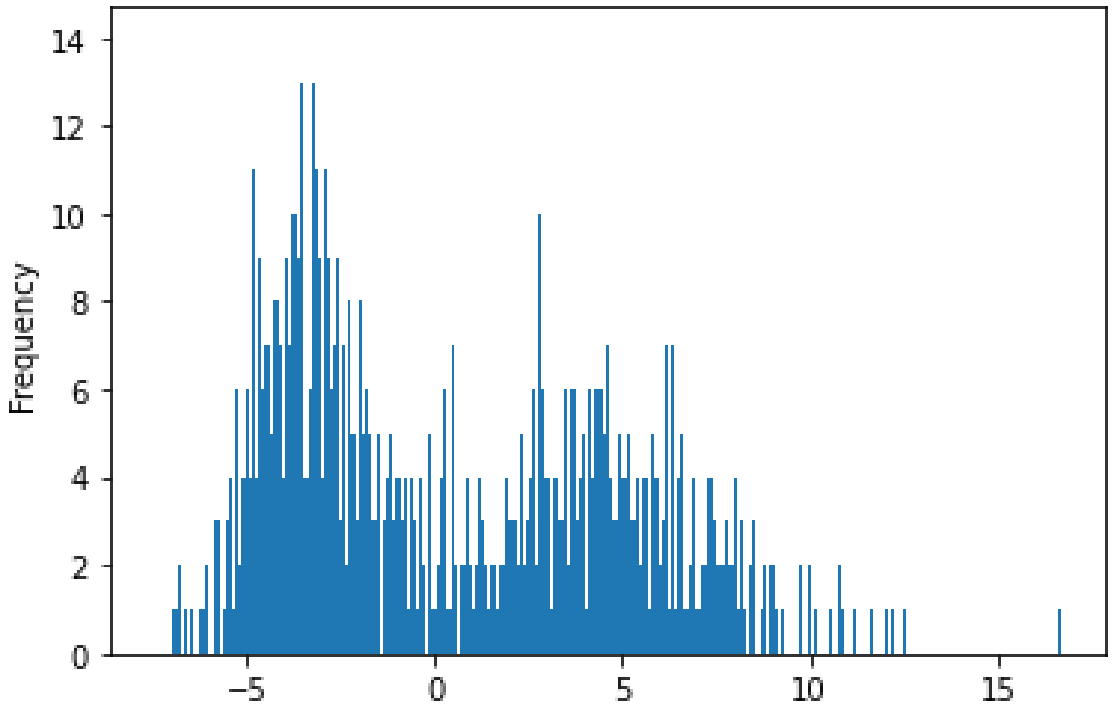


Figure 9: Classification scores are not normally distributed.



C. Code

- The code for the extraction of tensors from the Mask-RCNN may be found: `https://github.com/andics/maskrcnn-combined-build/tree/master/EXPERIMENTS/complete_crop_dataset_rec_featuremaps_var`
- The code for the hypothesis testing and statistical analysis can be found at: `https://colab.research.google.com/drive/1bfaZ5JRu2RaMtf8JuHlY6g2pm3IYQx0B?usp=sharing`

References

[1] Z. Q. Zhao, P. Zheng, S. -T. Xu and X. Wu "Object Detection With Deep Learning: A Review" IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11 (2019)

[2] Rossi, E., Roorda, A. "The relationship between visual resolution and cone spacing in the human fovea" Nature Neuroscience 13, 156–157 (2010)

[3] Emma E. M. Stewart, Matteo Valsecchi, Alexander C. Schütz "A review of interactions between peripheral and foveal vision" Journal of Vision Vol. 20, 2 (2020)

[4] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D. "Backpropagation applied to digit recognition" Neural computation, Vol. 1, Issue 4, 541–551 (1989)

[5] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context" European conference on computer vision (2014)

[6] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei "Imagenet: A large-scale hierarchical image database" IEEE conference on computer vision and pattern recognition (2009)

[7] Friedman, Jerome "On Multivariate Goodness–of–Fit and Two–Sample Testing" (2003).

[8] Cai, Haiyan, Bryan Goggin, and Qingtang Jiang. "Two-sample test based on classification probability." Statistical Analysis and Data Mining: The ASA Data Science Journal 13.1 (2020): 5-13.

[9] Kim, Ilmun, et al. "Classification accuracy as a proxy for two sample testing." arXiv preprint arXiv:1602.02210 (2016).

[10] Lopez-Paz, David, and Maxime Oquab. "Revisiting classifier two-sample tests." arXiv preprint arXiv:1610.06545 (2016).

[11] Blanchard, Gilles et al. "Semi-Supervised Novelty Detection." (2010).

[12] Altun, Yasemin et al. "Learning via Hilbert Space Embedding of Distributions." (2007).

[13] Gretton, Arthur et al. "A Kernel Two-Sample Test" (2012)

[14] D'Agostino, R. B. (1971), "An omnibus test of normality for moderate and large sample size", Biometrika, 58, 341-348

[15] D'Agostino, R. and Pearson, E. S. (1973), "Tests for departure from normality", Biometrika, 60, 613-622