TECHNISCHE UNIVERSITÄT DARMSTADT

Stimulus Selection and Image Processing for Enhancing Object Discrimination in Retinal Prosthetics

Bachelor Thesis by

Noah Schlegel

Bachelor thesis
Stimulus Selection and Image Processing for Enhancing Object Discrimination in Retinal Prosthetics

submitted by:
Noah Schlegel
Student ID: 2991405
Study program: Bachelor of Science Psychologie in IT

Heinheimer Straße 61
64289 Darmstadt
noah.j.schlegel@web.de

Submitted: March 31, 2018

Adviser: Prof. Jakob H. Macke

Human Perception
Institute of Psychology
Technische Universität Darmstadt
Alexanderstraße 10
64283 Darmstadt

All relevant code and data can be found on GitHub:
`https://github.com/sleep-yearning/artificial-retina-enhancement`

Declaration of Academic Integrity

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Noah Schlegel, die vorliegende Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden. Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

I herewith formally declare that I, Noah Schlegel, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once. In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content.

Date: Signature:

– *Abstract* –

Retinal implants that work by stimulating the retinal ganglion cells are an improvement for many patients with impaired vision due to retinal damage. However, due to low information resolution the detection of important features in complex scenes remains a difficult task. To develop appropriate image pre-processing methods for the prosthetics which can help patients in their everyday life, a framework was build to test different image manipulation techniques. Therefore, fitting visual stimuli were selected and processed by a simple simulated retina and finally discriminated with a machine learning algorithm. In this way, ease of feature detection and object recognition could be measured for different image processing methods. The results suggest improvements through background suppression methods, supporting findings from other studies.

Contents

1 Motivation and Previous Work

Over 15 million people worldwide suffer from either retinitis pigementosa or age-related macula degeneration, the two most prominent diseases of retinal degeneration eventually leading to blindness (Congdon *et al.* 2004), while retinal damage as a whole is the cause for 50% of blindness cases (Krumpaszky & Klauss 1996). Treating those diseases thus has been motivated for long and continues to be of growing relevance as life expectancies improve and geriatric diseases also get more frequent. Through increasing efforts in the last two decades, treatment of photoreceptor degeneration diseases through electronic implants has made substantial progress (Ong & da Cruz 2012; Zrenner, Stett, *et al.* 1999; K. Gekeler *et al.* 2018). Different implant types have been designed, build and tested, each with their own advantages and drawbacks. Electrodes directly stimulating the visual cortex for example can be applied in a multitude of diseases, not only retinal damage but also for optic nerve or similar damaged areas. They pose however even more surgically risks and have to do a lot of computation the visual pathway would normally do. This is a very complex task, motivating the earliest possible intervention to use the existing neurons' computing potential. (K. Gekeler *et al.* 2018) This thesis places focus on prosthetics attaching to the retina as those permit the incorporation of eye movements and the remaining retinal neurons. While the cell count does decrease in later stages (Marc & B. W. Jones 2003; Marc, B. Jones, *et al.* 2008), those inner retinal cells like amacrine, horizontal and ganglion cells can continue to function for many years after the onset of blindness due to degeneration of the photoreceptors (B. W. Jones & Marc 2005; Humayun *et al.* 1996). With such a transplant patients are able to relearn sight in the respective eyes from the resulting activation in the visual cortex they perceive. Two main approaches are used to develop retinal implants: subretinal and epiretinal. Subretinal implants are placed at the location of the degenerated photoreceptors, activating mainly bipolar cells that propagate the information through the remaining retinal network. Artificial photoreceptors are amplified and determine the output of electrodes connecting to the bipolar cells. Through the fixed positioning of the photoreceptors in the eye, eye movements can be used for visual searching per design. This is not true for all epiretinal implants, which have a lower pixels count also. Another big advantage is the use of existing spatial mapping of the visual pathway which doesn't have to be relearned. In contrast, epiretinal implants are placed on top of the retina and connect to its 1.2 million highly specialized ganglion cells or axons directly, so more computing has to be done by the prosthetic. As axons can be activated far from their ganglion cell's position, the activation patterns are not the same as the healthy eye would produce and vision has to be relearned slowly. Epiretinal prosthetics can be implanted easier, though. They also often have more possibilities to influence output current behavior based on input after the surgery. (Fine *et al.* 2015; K. Gekeler *et al.* 2018; Zrenner, Stett, *et al.* 1999; Zrenner 2002; Rizzo III *et al.* 2001)

"However, they all have low resolution, limited visual field, and can display only few gray levels (limited dynamic range), severely restricting their utility." as Jung *et al.* 2015 puts it.

The currently most advanced prosthetics all have under 2000 photoreceptors/electrodes and very limited numbers of discriminable gray levels. While generating useful information for the patient, the resolution is just too small to rely on visual information for many daily activities. The low resolutions are especially problematic when trying to recognize objects of interest out of crowded and cluttered scenes. Even non-invasive methods like auditory or tactile assistance systems suffer from the same resolution problems. While Zrenner, Bartz-Schmidt, *et al.* 2011 achieved promising results where patients recognized letters and objects and where partly able to succeed in typical acuity tests this was still in ideal experimental conditions.

One approach to improving the recognizing abilities turns to image processing methods used prior to electrical activation decision in the prosthetics. The goal is to deliver the most effective images to the remaining retinal cells. Jung *et al.* 2015 used a distance based method for this, removing all the background information which they identified by blurring through the specific camera focus point ('active confocal decluttering'). Thereby patients could scan through different distances and put clear focus of the vision on objects of interest. Their results are promising as the object recognizability improves a lot even for low resolutions. This method is however based on an external camera and thus not applicable for subretinal implants which have built in photoreceptors. But the underlying principle of suppressing background clutter could be realized in a different manner. Zhao *et al.* 2010 also tested the recognizability of different image settings, incorporating edge detection & binary threshold, different phosphene types (induced visual shapes in patients) and images sizes. They tested the recognizability on persons with normal vision to achieve minimal conditions for recognition.

This work compares 7 different methods on an image set that has been segmented in beforehand also building a theoretic test framework for additional possible methods. At first a fitting image set had to be selected and prepared for the image

processing methods and the simulated retinal processing, which followed, resulting in spike output per image. For each of the proposed processing methods (image conditions) a learning classifier was trained on the spike outputs and the true category the images came from. In our case the resolution reduction happened directly through the number of receptive fields available for spiking. The underlying assumption is that an image processing method that enhances the recognizability of objects should analogically also be able to improve the classifier performance by making the discrimination task easier. This would qualify the classifier prediction accuracy as estimate for 'easiness'.
As part of a larger research project "Image-processing computations in artificial vision" in the collaborative research center "Robust Vision" at the University Eye Hospital Tübingen and the Natural and Medical Sciences Institute Reutlingen with Prof. Macke, Prof. Zrenner, Dr. Stingl and Prof. Zreck, the results will be used for decisions about which stimuli and decoding approaches are going to be tested in neurophysiological experiments. In the end, patients might hopefully benefit from better object recognizing abilities and thus an easier life.
An illustration of the complete processing is shown in Figure 1.1.

Figure 1.1.: Illustration of the complete computation process. The whole image set is processed with 7 different methods resulting in 8 different conditions (1 unchanged). For each condition all the images are multiplied by the simulated ganglion receptive fields. The 10% with the highest activation are selected and a binary array of resulting activation generated. A multinomial logistic regression classifier finally tries to learn the image categories from those arrays. The same is done for a smaller image subset with balanced category frequency.

2 Stimuli

2.1 Criteria

To be able to compare the results of this simulation with real patient data, the stimuli have to be the same. That means they have to be realistic items that patients encounter in everyday life and also motives that are easily distinguishable. Quantitatively we aimed at getting 4 to 10 different categories with around 500 Stimuli each. To apply the image processing methods, object segmentation annotations had to be provided for the images. Also fixed sizes and object type information were needed for practicable standardized supervised discrimination learning. For the assumption, a learning classifier would be able to discriminate the pictures, it is useful to have datasets which are known to give appropriate classification results with computer vision algorithms.

2.2 Dataset Candidates

A few openly accessible labeled image datasets come with a segmentation of image components. Through an extensive online search including dataset compilations 6 sets of segmented images with everyday objects could be identified. From those one had to be selected as most fitting.

2.2.1 BSDS500

The Berkeley Segmentation Dataset and Benchmark is a selection of 500 natural images with manual pixel-wise annotations made by 5 subjects (Arbelaez *et al.* 2011). Segmentations are precise and adequate for our purposes. All pictures have the same number of pixels (321*481), differing only in orientation (horizontal or vertical). The downside of this image set is the small number of images contained. Also they are not categorized. Doing a rough categorization there are about 29 categories. Not all images could be associated with only one category which have sizes from 3 to 123 instances. With a count of 123 the dataset mainly provides pictures of humans while there are many kinds of animals with few instances. Also some depicted objects could only be summarized with 'man made' or 'other animals' since they occurred only once. The BSDS500 was clearly created for segmentation training and not categorization tasks.

Figure 2.1.: Instances of the BSDS500 dataset. Using boundary (left), segmentation (middle) and both (right) annotations to depict annotation quality.

2.2.2 Caltech101

Another Set of Images, originally constructed at the Caltech University in 2003 and annotated in 2006 (Fei-Fei *et al.* 2006), is called Caltech101 and consists of over 9000 images. Images were sorted into 101 categories and the outlines for the depicted objects were annotated manually with lines. Every image has about 300*200 pixels and shows a clearly identifiable natural thing, animal or human. Pictures from Caltech101 could well be used for our task, and while it is considered 'too easy' for today's computer vision algorithms, in our case the classifier would have a harder time anyways since only confronted with simulated ganglion spikes instead of image data. The successor of Caltech101 was Caltech256, which has however no segmentation annotations and was only created as classification challenge.

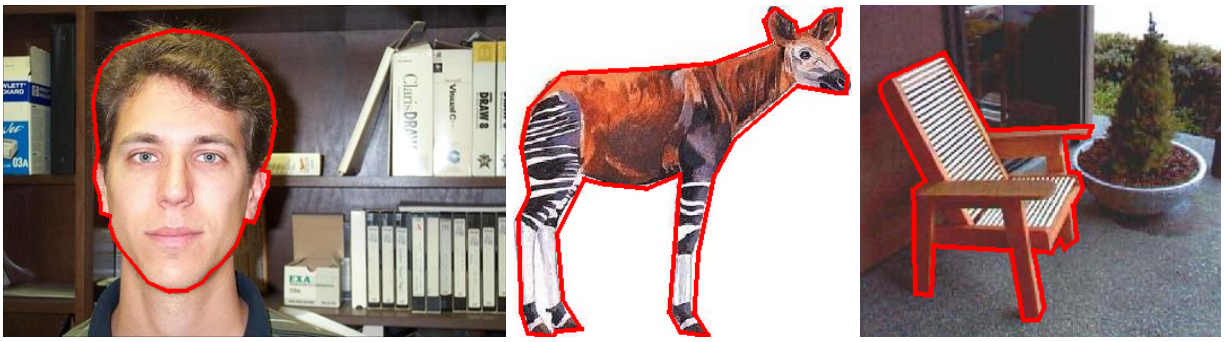


Figure 2.2.: Instances of the Caltech101 dataset with outlines.

2.2.3 EDUB-Obj

The EDUB-Obj is a dataset comprised of pictures taken from a first-person perspective of persons. Images where collected by 4 persons on 8 days, taking one picture every 30 to 60 seconds. Recorded activities include for example shopping, eating, riding a bike, working, attending meetings or commuting to work. Annotations provided contain information about object outlines and category for each object in the image (Bolaños & Radeva 2015). Containing 4912 images in total, relevant object categories are inside at least 130+ images, which are around 512*512 pixels big (some slightly cropped in one dimension). As the images depict the same scenes that users of retinal implants live in, the dataset could be used for an even more realistic approach. This is also reflected in the pictures by a large number of consecutive similar images. While boundary annotation is not bad, it is not as precise as with the previous two sets of pictures. Faces have partly been blurred.

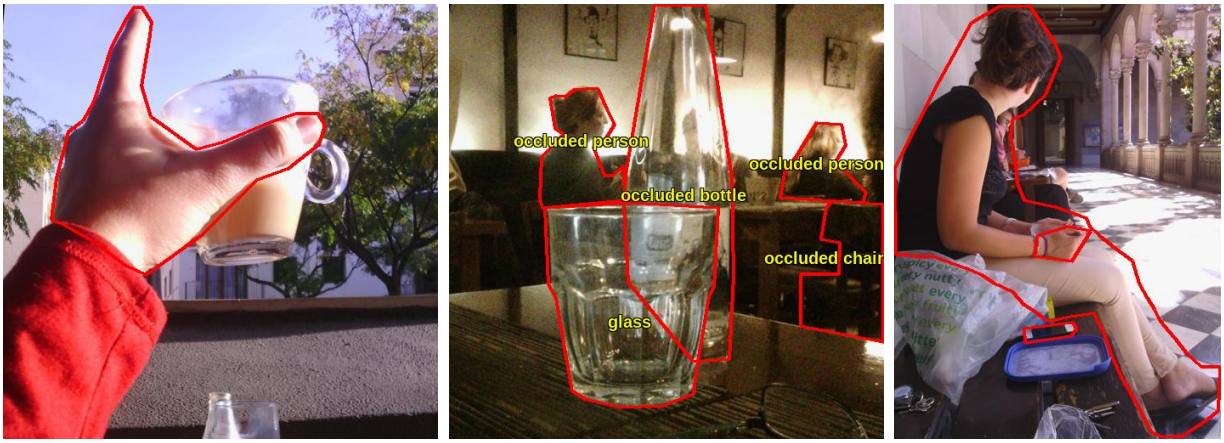


Figure 2.3.: Instances of the EDUB-Obj dataset with annotated outlines. Object categories are additionally given for annotated items as well as the information, whether an object is occluded in some way.

2.2.4 GRAZ-02

In GRAZ-02, which was compiled at the TU Graz (Opelt *et al.* 2006), there are four categories of objects: cars, bikes, people and background. The over 300 images in each category have a size of 640*480 pixels. A very precise annotation for this images exists under the name IG-02 (Marszalek & Schmid 2007) and even provides the expected location of occluded parts in the image. Since cars, bikes and persons are often seen in everyday life, this dataset could be useful. However, 3 object categories could turn out to be a too easy task, they also only represent traffic objects.

Figure 2.4.: One image of each of the categories of the GRAZ-02 dataset (except background).

2.2.5 Microsoft COCO

Microsoft's 'Common Objects in Context' is a huge dataset consisting of about 2,000,000 labeled images. As per the name, objects are depicted in their usual context and with their usual surrounding objects (Lin *et al.* 2014). While providing an unmatched number of images, object boundaries are rather coarse and consist of straight lines. Also images are of very variable size and objects marked inside an image can be very small or insignificant in that image. COCO therefore is a fairly complex image source.

Figure 2.5.: Instances of the Microsoft COCO image set. Images are complex, contain multiple annotated objects and often truncated items.

2.2.6 PASCAL VOC

The Visual Object Challenge was a widely known contest for computer vision algorithms. They provided a big set of images which was reused and augmented since 2009, resulting in the 2012 dataset containing all images since 2008 and therefore this last version is the largest of them. (Everingham, Eslami, *et al.* 2015). However, only limited segmentation annotations have been published with the training and validation data. In this 2012 set there are segmentation annotations for only 2913 of the 17125 total pictures. But there has been created a ground-truth annotation for the set of 2010 under the name 'PASCAL-Part', providing even a distinction between different parts of the visible objects (Chen *et al.* 2014). These pixel-wise annotations provide a precise object mask and enable an image set with precise object outlines and over 10,000 instances. Most of the images have a size of 500*375 pixels while some deviate slightly from those metrics (but all have one side which is 500 pixels long). Objects tend to be more prominent than in Microsoft COCO but are not as isolated as in Caltech101. About 400 to 1000 annotated pictures in each category (which are 20) are present in this dataset totaling to over 10,000 images.

Figure 2.6.: Instances of the PASCAL VOC 2010 dataset. All occurring objects in a picture are annotated and labeled separately.

Table 2.1.: Tabular comparison of some important properties of the dataset candidates.

Dataset	**Object types**	**Annotation type**	**Number of images per relevant category**
BSDS500	diverse common objects	pixel-wise	500 in total
Caltech101	diverse common objects	boundary points	>100
EDUB-Obj	things encountered in the days of collecting participants (ego perspective)	boundary points	179-1274
GRAZ-02 with IG-02	cars, bikes, persons & background	pixel-wise	311-420
Microsoft COCO	diverse common objects with complex common surroundings	boundary points	1200-66000
VOC 2010 + Pascal-Part	diverse common objects	pixel-wise	>400

2.3 Used Dataset

The properties of each of the dataset then led to the decision which one should be used in our simulation. The image dataset BSDS500 is not applicable due to few images per category and overlapping categories in many images. GRAZ-02 has too few categories which also do not reflect a broader range of daily objects to recognize. EDUB-Obj has many images without objects of interest and blurred faces. Object outline precision is also not sufficient which can be said about Microsoft COCO, too. The latter also had many different types of objects in most images which would complicate a discrimination too much. The remaining image sets are Pascal VOC 2010 and Caltech101, which both could be used. Here the decision fell on Pascal VOC 2010 (Everingham, Van Gool, *et al.* 2010), having the advantage of more precise, pixels wise annotations. Too cluttered or complicated images had to be filtered, which wouldn't have been necessary for Caltech101 but there many painted images would have needed to be sorted out. Another advantage of the Pascal images is the annotation of object parts, which allowed for an additional processing method to be applied.

Since we didn't want to use all categories of the set and some images seemed very difficult, fitting images were sorted and, if necessary, rotated and cropped to a common resolution of 500*333 pixels. Difficult images had objects representing only a small fraction of the picture, multiple overlapping objects or otherwise very complex scenes. They were omitted to reduce the possibility of learning failure due to too difficult input. Not all images had an annotation by Pascal-Parts and so 3586 images remained in the 10 categories:

- aeroplane (386)
- bird (366)
- boat (243)
- bottle (117)
- car (371)
- chair (198)
- dog (665)
- horse (202)
- person (903)
- tvmonitor (135)

To get useful batches of images for patient study, the images considered easily distinguishable where selected from each category. The easiest 10 of those build the 'easy' images for each category. From the remaining images not considered very easy 100 were selected and represent the 'difficult' stimuli. In contrast to the entirety of the 3586 pictures, the combined easy and difficult batches provide a balanced set where every category is represented in the same quantity (110 each).

2.4 Pre-Processing Variants

The ultimate goal of this work is the identification of useful image pre-processing methods, which then can be implemented into the retinal prostheses. Rosenholtz *et al.* 2007 states that visual clutter reduces performance of object segmentation, recognition and search through crowding and masking. For natural environments these effects of crowding and masking by background clutter are very relevant (Jung *et al.* 2015). Thus processing methods that suppress the background are plausible candidates. Edge extraction and binary threshold have also been proposed and used by Zhao *et al.* 2010 in a similar context. To find methods resulting in higher discriminability, different versions of the stimuli had to be compared. Using the annotations from Pascal-Part, the following methods have been used to alter the raw stimuli:

- blurring of background
- contrast reduction in background
- darkening of background
- mean-color segmentation of object parts and background
- mean-color segmentation of object and background
- black-white (binary) segmentation of object
- outline of object

Where multiple objects where on an image, objects of a different category where treated as background in processing. Using those methods, eight different variants of stimuli have been generated for comparison:

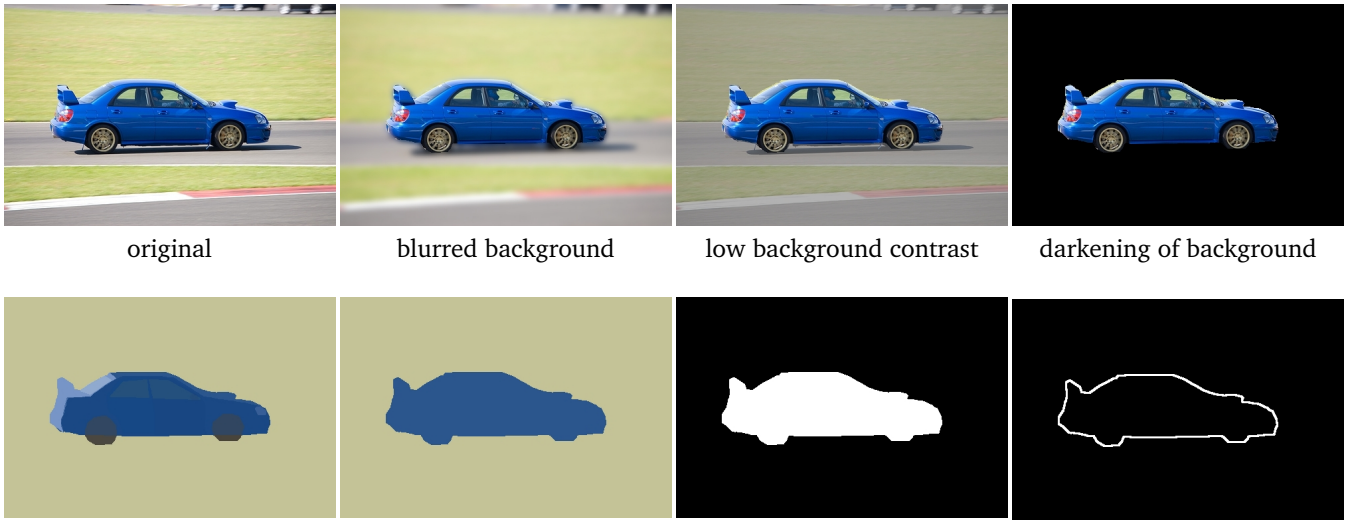


Figure 2.7.: The 8 stimuli variants depicted for an example image

There were no distinctive part annotations for the categories boat and chair. Therefore the stimuli here are the same as in the object segmentation condition. The python code for some of the processing methods can be found in appendix A.1 while the complete code and the selected image sets can be found on GitHub (A).

3 Simulated Ganglion Cell Activity

To assess whether the processing methods can bring recognition enhancement in a retinal prosthesis, the processing of those and the visual pathway had to be simulated.

3.1 Retinal Processing

The exact ways of retinal processing are still not fully understood, but many structural properties and effects have been found already. Some models incorporate contrast adaption, coupling of neurons, different receptive fields and temporal selectivity (Pillow *et al.* 2008). While scientifically plausible and important for the proper understanding of real retinal computation, those aspects would have complicated the discrimination in our task in an unnecessary portion while hardly contributing to answering the research question. Thus a rather simple Difference-of-Gaussians model of retinal computation was used for mapping respective stimuli to retinal ganglion cell spikes. Similar models are often used to simulate retinal behavior, even though a realistic model would be more complex. (Nonnenmacher *et al.* 2017)

3.2 Setup

On an area matching the fixed image size receptive fields where distributed randomly. Each receptive field consists of a difference of Gaussians area representing ON cells (center activates, surround inhibits). Difference of Gaussians is an often used model for receptive fields in retinal ganglion cells or LGN (Rodieck 1965; Tadmor & Tolhurst 2000). They often are attributed with the capability of basic edge detection. The outer Gaussian's covariance matrix was set to twice the covariance matrix of the inner one. To improve computational efficiency points farther away than 5/10 Mahalanobis distances form the receptive field's center were not taken into account. This way most of the probability mass was still considered but many very small values could be ignored. Contribution values of points lying outside of the image were added to the value at the edge to avoid activation/inhibition because of truncated receptive fields. In one test we projected 5000 receptive fields with a variance of 20 (and 5 Mahalanobis distances cutoff threshold) on the area of 500*333 points. The number of 5000 was chosen to enable a big amount of information for the classifier while staying in computational limits. Since this would probably result in overfitting of the model, another spike output computation was conducted with only 250 receptive fields (variance of 70, Mahalanobis distance of 10). The distribution of activation and inhibition of the combined receptive fields is shown in Figure 3.1

To determine the retinal ganglion cells which spike at a given stimulus, the 3 color channels of each image were first added and normalized to a range between 0 and 1, reflecting in some way the monochromatic limitation of the retinal prostheses. When multiplied with each receptive field an activation value was returned. To avoid a learning on the number of spikes, only the ganglion cells with activation values above the 90th percentile were accepted as spiking to achieve constant spike counts. At the same time it serves as some kind of contrast adaption so that darker images did not automatically have lower spike counts than lighter ones. In most conditions this spike count normalization worked well, in those conditions where big areas of the image were set to zero however, too many activation values resulted in a

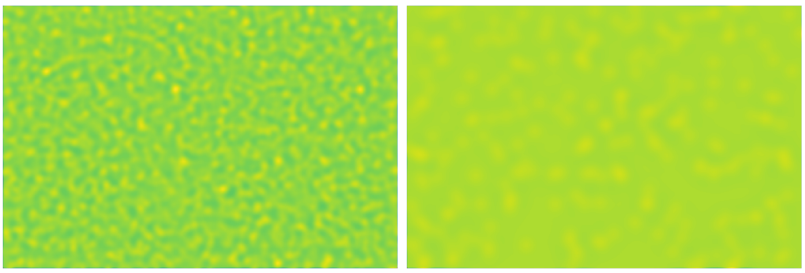


Figure 3.1.: Combined activation topography of the used receptive fields for the set of 5000 and the set of 250 receptive fields. To achieve an incorporation of all image points, the field variance had to be higher in combination with less receptive fields.

zero value which lead to fewer spike counts than 10%. While this effect holds for all image categories in the respective conditions, there could be relevant connections between spike count and category in those. The code for receptive field computation and spike computation can be found in the appendix A.2.

4 Discrimination of Ganglion Responses

To assess the grade of discriminability for the image manipulations, a simple multiclass classifier was needed. We implemented a multinomial logistic regression with a single layer neural network. For each receptive field it was trained separately in each of the image conditions. Due to 5000/250 computed different receptive fields and 10 image categories the network had 5000/250 input neurons and 10 output neurons fully connected with each other. The Cross-Entropy Loss function was used to compute the output error and the Adadelta optimizer with a starting learning rate of 0.3. This resulted in good training accuracy and thus information extraction. To achieve comparable results the training and test set were split by 11-fold cross validation since both the set of full images (3586) and the combination of 'easy' and 'difficult' selections (1100) were divisible by 11. Each training consisted of 200 epochs to ensure exhaustive training data incorporation and convergence, while batch size for gradient determination was 5/3. Higher batch sizes tended to ignore smaller categories so that categories with good distinctions and many training instances were well predicted while other output classes were never chosen. So batch sizes of 5 and 3 increased the variance over the output classes while a lower batch size would also result in a higher bias towards some classes and slow convergence. The 11-fold cross validation was repeated 5/3 times and the results of the 55/33 trainings combined. This procedure was applied for every stimulus condition including original images and for comparison on the whole image set and the 'easy + difficult' combination to allow insights into any effects of balanced training instances for the categories. While the loss took a weight matrix into account to correct for unbalanced category frequency, testing on a real balanced set seemed still useful.

Eight conditions with two sets each trained 55/33 times resulted in 880/528 trainings that had 200 epochs each. For each epoch the prediction accuracy on the test set was saved so that 200*55/200*33 prediction accuracies could be analyzed for each condition and set. Every training resulted in a final training data accuracy and class wise prediction performance which were saved also.

Python code for the learning network implementation in PyTorch is included in appendix A.3.

5 Effects of Different Image Manipulations

In Figure 5.1 the activated receptive fields of the simulated retina (with 5000 receptive fields) are illustrated for an example image in each condition. It can give an idea of the output activation array in the respective conditions.

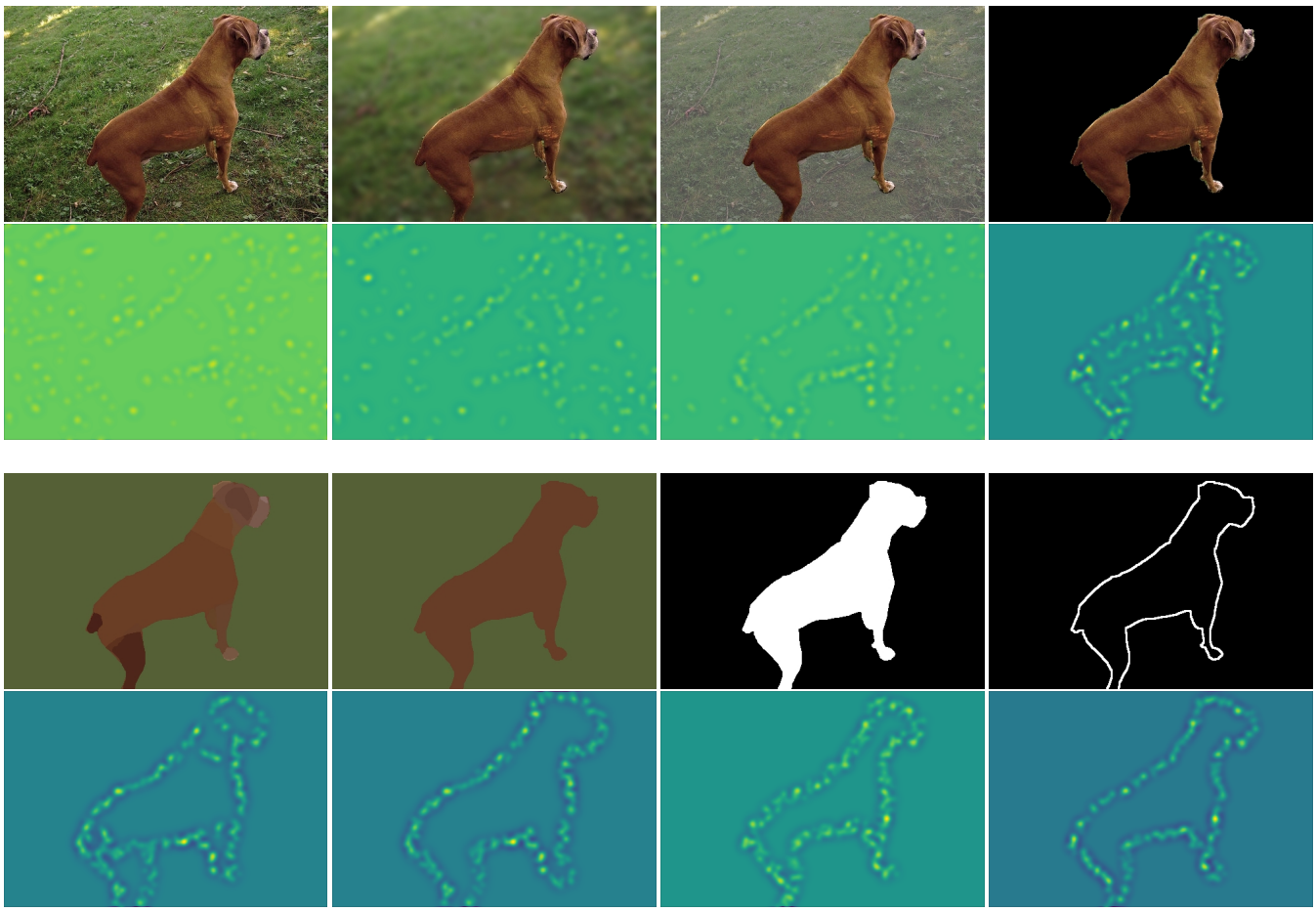


Figure 5.1.: Sum of activated receptive fields for each condition.

To examine meaningful informations about the learning performance under different conditions, the prediction strengths for every training epoch were recorded and combined. Labatut (2011) recommends Overall Success Rate and True Positive Rate as the most useful and interpretable quality metrics. Thus it can be assumed that with the prediction accuracy (scaled OSR) and the True Positive Rates of the classes, a basic reflection of prediction quality can be obtained through the numbers. In Tables 5.1 to 5.4 the resulting classification capabilities (for 5000 receptive fields) are visualized separately for the whole and balanced image set with the 8 different conditions respectively juxtaposed. Tables 5.5 to 5.8 list the outcomes for the smaller number (250) of receptive fields.

[1] rough fraction size of training and test data

[2] rough fraction size of training and test data

Table 5.1.: Overview of prediction strength of the model (5000 inputs) per condition in the *non-balanced* image set.

Condition	[1]	original	background blurred	background contrast low	background dark
Example					
mean prediction accuracy		0.23768	0.24381	0.24607	0.27433
max prediction accuracy		0.34663	0.33436	0.34049	0.38037
mean training accuracy		99.155%	98.458%	98.696%	96.785%
mean TPR aeroplane	11%	0.4277	0.4493	0.4386	0.4546
mean TPR bird	10%	0.0808	0.0978	0.1031	0.1918
mean TPR boat	7%	0.0590	0.0471	0.0518	0.1075
mean TPR bottle	3%	0.0037	0.01265	0.0038	0.0241
mean TPR car	10%	0.1226	0.1372	0.1229	0.1282
mean TPR chair	5%	0.0232	0.0103	0.0348	0.0800
mean TPR dog	19%	0.2573	0.2718	0.3021	0.2896
mean TPR horse	6%	0.0237	0.0260	0.0159	0.0196
mean TPR person	25%	0.4527	0.4531	0.4440	0.4756
mean TPR tvmonitor	4%	0.0214	0.0307	0.0269	0.0568

Table 5.2.: Overview of prediction strength of the model (5000 inputs) per condition in the *non-balanced* image set.

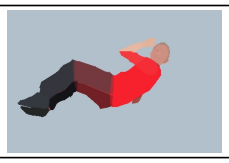
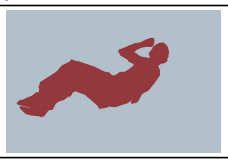
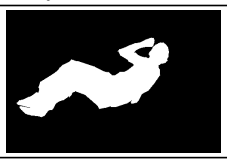
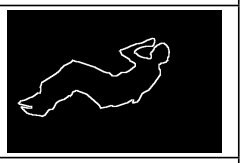
Condition	[1]	segmentation part	segmentation object	segmentation binary	object outline
Example					
mean prediction accuracy		0.25054	0.26176	0.27761	0.27968
max prediction accuracy		0.36503	0.34970	0.40184	0.38344
mean training accuracy		96.417%	94.817%	94.216%	97.669%
mean TPR aeroplane	11%	0.4007	0.4387	0.4356	0.4190
mean TPR bird	10%	0.1878	0.1681	0.2328	0.2198
mean TPR boat	7%	0.0728	0.0575	0.1085	0.1022
mean TPR bottle	3%	0.0617	0.0255	0.0219	0.0203
mean TPR car	10%	0.1512	0.1654	0.1935	0.2068
mean TPR chair	5%	0.0613	0.0705	0.0798	0.0671
mean TPR dog	19%	0.2668	0.3133	0.2970	0.3031
mean TPR horse	6%	0.0321	0.0352	0.0445	0.0492
mean TPR person	25%	0.4297	0.4303	0.4427	0.4594
mean TPR tvmonitor	4%	0.0491	0.0565	0.0967	0.0795

Table 5.3.: Overview of prediction strength of the model (5000 inputs) per condition in the *balanced* image set.

Condition	[1]	original	background blurred	background contrast low	background dark
Example					
mean prediction accuracy		0.16662	0.17438	0.17591	0.20285
max prediction accuracy		0.29000	0.29000	0.29000	0.32000
mean training accuracy		99.927%	99.976%	99.882%	99.604%
mean TPR aeroplane	10%	0.5420	0.6151	0.5542	0.5279
mean TPR bird	10%	0.2081	0.2166	0.1349	0.1950
mean TPR boat	10%	0.1281	0.1122	0.1503	0.1236
mean TPR bottle	10%	0.1540	0.0994	0.1084	0.1731
mean TPR car	10%	0.1256	0.1322	0.1945	0.1734
mean TPR chair	10%	0.0926	0.0977	0.1331	0.1544
mean TPR dog	10%	0.1004	0.1305	0.1527	0.1275
mean TPR horse	10%	0.1565	0.1063	0.1133	0.1257
mean TPR person	10%	0.0467	0.1067	0.0582	0.1939
mean TPR tvmonitor	10%	0.1497	0.1653	0.1991	0.2501

Table 5.4.: Overview of prediction strength of the model (5000 inputs) per condition in the *balanced* image set.

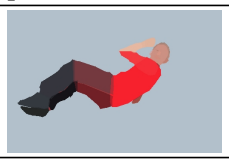
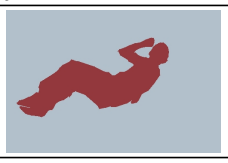
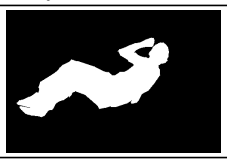
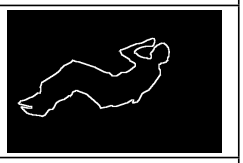
Condition	[1]	segmentation parts	segmentation object	segmentation binary	object outline
Example					
mean prediction accuracy		0.20116	0.20440	0.22296	0.20063
max prediction accuracy		0.33000	0.32000	0.36000	0.29000
mean training accuracy		99.598%	99.347%	98.845%	99.925%
mean TPR aeroplane	10%	0.4000	0.5053	0.5142	0.5300
mean TPR bird	10%	0.1832	0.1844	0.1731	0.2086
mean TPR boat	10%	0.1723	0.1552	0.1335	0.0953
mean TPR bottle	10%	0.1992	0.2043	0.1889	0.1634
mean TPR car	10%	0.1361	0.2248	0.3013	0.2759
mean TPR chair	10%	0.2310	0.1617	0.1320	0.1214
mean TPR dog	10%	0.1633	0.1571	0.1566	0.1365
mean TPR horse	10%	0.2046	0.1046	0.1540	0.1314
mean TPR person	10%	0.1483	0.0702	0.2198	0.1398
mean TPR tvmonitor	10%	0.1947	0.2705	0.2802	0.2422

Table 5.5.: Overview of prediction strength of the model (250 inputs) per condition in the *non-balanced* image set.

Condition	[2]	original	background blurred	background contrast low	background dark
Example					
mean prediction accuracy		0.22963	0.23779	0.24537	0.28108
max prediction accuracy		0.32209	0.31595	0.36503	0.36503
mean training accuracy		70.305%	69.664%	68.880%	64.943%
mean TPR aeroplane	11%	0.3394	0.3600	0.3765	0.4587
mean TPR bird	10%	0.0959	0.0832	0.1053	0.2730
mean TPR boat	7%	0.0898	0.0793	0.0467	0.0628
mean TPR bottle	3%	0.0134	0.0080	0.0084	0.0101
mean TPR car	10%	0.0951	0.0741	0.0723	0.1377
mean TPR chair	5%	0.0334	0.0173	0.0424	0.0288
mean TPR dog	19%	0.2539	0.2674	0.3092	0.2807
mean TPR horse	6%	0.0356	0.0384	0.0380	0.0417
mean TPR person	25%	0.3882	0.4223	0.4155	0.4700
mean TPR tvmonitor	4%	0.0289	0.0325	0.0182	0.0317

Table 5.6.: Overview of prediction strength of the model (250 inputs) per condition in the *non-balanced* image set.

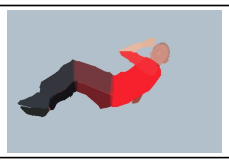
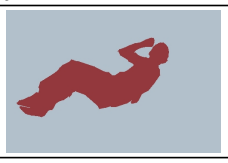
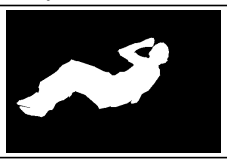
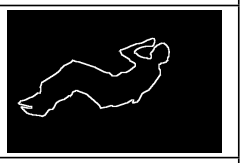
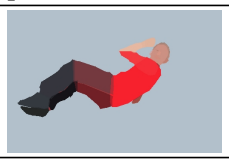
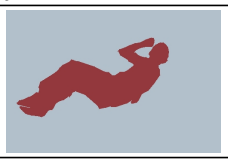
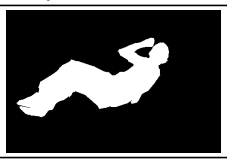
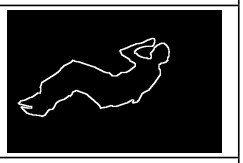
Condition	[1]	segmentation part	segmentation object	segmentation binary	object outline
Example					
mean prediction accuracy		0.25947	0.27006	0.30914	0.30020
max prediction accuracy		0.34049	0.35890	0.39877	0.38650
mean training accuracy		67.173%	68.607%	66.959%	69.026%
mean TPR aeroplane	11%	0.4069	0.4341	0.4827	0.4180
mean TPR bird	10%	0.1123	0.1781	0.3514	0.2482
mean TPR boat	7%	0.0386	0.0703	0.0151	0.0826
mean TPR bottle	3%	0.0287	0.0045	0.0254	0.0196
mean TPR car	10%	0.1379	0.1385	0.1554	0.1765
mean TPR chair	5%	0.0470	0.0596	0.0848	0.0495
mean TPR dog	19%	0.2869	0.2764	0.3404	0.3152
mean TPR horse	6%	0.0172	0.0273	0.0107	0.0236
mean TPR person	25%	0.4570	0.4512	0.4730	0.4956
mean TPR tvmonitor	4%	0.0418	0.0318	0.0646	0.0662

Table 5.7.: Overview of prediction strength of the model (250 inputs) per condition in the *balanced* image set.

Condition	[1]	original	background blurred	background contrast low	background dark
Example					
mean prediction accuracy		0.14472	0.13292	0.15241	0.19443
max prediction accuracy		0.25000	0.24000	0.25000	0.32000
mean training accuracy		92.367%	92.091%	91.615%	85.879%
mean TPR aeroplane	10%	0.3792	0.3208	0.4258	0.4923
mean TPR bird	10%	0.1402	0.0886	0.1418	0.2087
mean TPR boat	10%	0.1086	0.0909	0.0958	0.1200
mean TPR bottle	10%	0.0922	0.1046	0.1396	0.1761
mean TPR car	10%	0.1115	0.0971	0.1160	0.2125
mean TPR chair	10%	0.1166	0.1173	0.1140	0.0845
mean TPR dog	10%	0.1276	0.1317	0.1653	0.1888
mean TPR horse	10%	0.1414	0.0856	0.0779	0.0959
mean TPR person	10%	0.1333	0.0953	0.0870	0.1596
mean TPR tvmonitor	10%	0.1343	0.1160	0.1730	0.1328

Table 5.8.: Overview of prediction strength of the model (250 inputs) per condition in the *balanced* image set.

Condition	[1]	segmentation parts	segmentation object	segmentation binary	object outline
Example					
mean prediction accuracy		0.16672	0.20313	0.21104	0.21042
max prediction accuracy		0.29000	0.37000	0.32000	0.38000
mean training accuracy		88.527%	89.479%	87.145%	90.382%
mean TPR aeroplane	10%	0.3067	0.3938	0.5150	0.4332
mean TPR bird	10%	0.1222	0.1599	0.2047	0.1760
mean TPR boat	10%	0.1599	0.1430	0.1522	0.1285
mean TPR bottle	10%	0.2003	0.1309	0.1185	0.1708
mean TPR car	10%	0.2011	0.2795	0.2247	0.1994
mean TPR chair	10%	0.1540	0.1164	0.1688	0.1307
mean TPR dog	10%	0.0967	0.2141	0.1058	0.2335
mean TPR horse	10%	0.1199	0.1775	0.1322	0.1633
mean TPR person	10%	0.1129	0.1130	0.1899	0.1211
mean TPR tvmonitor	10%	0.1998	0.1983	0.2255	0.2566

6 Implications

Except in one condition (balanced image set, 250 receptive fields, background-blurring) every image processing method resulted in a higher mean prediction rate. This makes intuitive sense, since those parts irrelevant for the classification are reduced in complexity or completely removed while at least the object outline remains. The overall power to improve the classification ease is therefore safely assumable. All of the 7 processing conditions resulted in a higher mean prediction strength of the tested model, implying that the discrimination task was made easier. The magnitude of those effects however differs between the tested methods. Background blurring and contrast reduction improved the prediction accuracy only by a small margin or in one case even decreased the accuracy. Image processing methods producing stimuli with notably lower complexity produced bigger improvements. The strongest prediction power was achieved for binary segmentation, object outline and background-darkening, where irrelevant information is discarded completely, object segmentation also yielded comparable good classification performance. Training accuracy does not correlate in an expected way with prediction accuracy or maximum. In the 5000 input training this could very well be due to overfitting. Since the resulting mean predictive power lies between 13 and 31%, with maximum occurrences of up to 40%, it's safe to assume at least some generalizability through comparison with a chance baseline of 10% correctness. Indeed overfitting is highly probable since the input neurons number exceeds the number of images to train on. However when training with only 250 input neurons prediction accuracies per condition where not very different from the more complex model. In each category and dataset there were trends to ignore or specify on certain categories, which were mainly true in both cases. It can be seen that category prevalence in the training and test data heavily influences the TPR per category. But there are also other factors that play a role there. Aeroplanes for example could be detected very well regardless of their 'normal' contribution to the dataset with 11%. The aeroplane TPR was even better in the balanced training, suggesting structural properties that could be learned well. Without spike count normalization this could have been attributed to high background activation based on lots of sky but the effects also hold for conditions without spike count deviation. Also in conditions with less normalized spike counts there are no extraordinary class differences in prediction capability compared to other conditions, which would have hinted at a learning based on those spike count deviations.

It seems overall very plausible after this simulation that image processing methods can improve the recognizability of objects for the patients of retinal implants. Especially those removing the background overall (outline, binary segmentation & background darkening) have the highest discriminability and could prove useful for incorporation in retinal implants. This fits well with the results of Jung *et al.* 2015 and their approach to suppress background information completely. Object mean color segmentation had decent effects also while object part segmentation performed overall a bit worse than simple object segmentation. Since outline recognition is a difficult task already this would probably be true for implementation in prosthetics, too. Also in the monochromatic setting of the implants a mean color segmentation has not really the prospect of being better than binary segmentation. The classifier struggled a bit more in this condition, too. In computational limited retinal prostheses this methods could prove a feasible way of object recognition enhancement, since more advanced algorithms are even more difficult to implement in them. A major challenge even for those methods will be an online segmentation algorithm that is the foundation of implementing all here tested methods.

7 Shortcomings and Future Work

To achieve a more efficient computation of the receptive fields, activation factors farther than 5/10 Mahalanobis distances were ignored. This induced the problem of too many zero valued receptive fields in conditions with zeroing of large areas (background darkening, outline, binary segmentation). Thus a learning on spike count instead of positional combination cannot be ruled out in those conditions. Incorporating the activation factors across the whole image could resolve this issue. In combination with a more efficient presentation of receptive fields where only the positions and one computed receptive field is memorized, the memory requirements could be reduced in the end. This would however result in longer computation times per receptive field, a problem in real time prosthetics but not simulation. The discrimination classifier has much potential for improvement, too. High training accuracy and low prediction accuracy could be the symptom of an overfitting, mostly due to the many input neurons. The strength of this effect is however questionable as the drastically simpler model with 250 input neurons achieved very similar prediction accuracies. Another running of the retinal activation and classification with even less receptive fields could result in different outcomes. The same holds for different variations of the receptive field covariance matrix and the classification hyperparameters. Since the receptive field spikes basically still represent image properties, a visual specific convolutional neural networks could prove to be more useful than simple logistic regression. For this receptive fields close to each other would have to be sorted next to each other and steps like pooling and sampling would intuitively extract image information. To achieve better classification accuracy a subset of categories could be used to limit output options and take complexity from the model. Also the category prevalence weights did not achieve a proper learning balance, a stronger penalty could show better results. The processing simulation would be even more realistic when image annotations had no available segmentation and would come from a time critical algorithm, similar to what would be employed in a prosthetic. A classic strategy to tackle the overfitting and attaining more generalizability would be more data, which could come from Caltech101 or other databases. In Pascal VOC 2010 are even pictures left which were not included due to their deviations from the normalized resolution. With a more tolerant cropping process, the numbers could slightly increase that way. In the case of an on-the-fly segmentation algorithm proposed above, many many more images could be used since segmentation annotations were no longer a criterion. With a better working classifier, possibly achievable through above proposals, a more realistic spike computation would incorporate noise on the simulated perception. When training with the more restricted 250 input classifier, the prediction accuracies differed much between the balanced and the whole image sets. The balanced set had really weak performance while the whole set was notably better. This could be another indicator for data sparsity and motivates a bigger dataset. One issue with the object part segmentation were the missing annotations for boats and chairs. The results in that condition would be more reliable with added annotations for those categories. With further development of the implants the possibilities will increase and enable more preprocessing assistance for the patients. The basic methods described in this work could be a stepping stone on the pursuit to detailed vision replacement.

List of Figures

List of Tables

Bibliography

1. Arbelaez, P., Maire, M., Fowlkes, C. & Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **33,** 898–916. ISSN: 0162-8828 (May 2011).
2. Bolaños, M. & Radeva, P. Ego-object discovery. *CoRR* **abs/1504.01639.** arXiv: 1504.01639. http://arxiv.org/abs/1504.01639 (2015).
3. Chen, X. *et al. Detect what you can: Detecting and representing objects using holistic models and body parts* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), 1971–1978.
4. Congdon, N. *et al.* Causes and prevalence of visual impairment among adults in the United States. *Archives of Ophthalmology (Chicago, Ill.: 1960)* **122,** 477–485 (2004).
5. Everingham, M., Eslami, S. M. A., *et al.* The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* **111,** 98–136 (Jan. 2015).
6. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. *The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results* http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 2010.
7. Fei-Fei, L., Fergus, R. & Perona, P. One-Shot Learning of Object Categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28,** 594–611. ISSN: 0162-8828 (Apr. 2006).
8. Fine, I., Cepko, C. L. & Landy, M. S. Vision research special issue: Sight restoration: Prosthetics, optogenetics and gene therapy. *Vision Research* **111.** Sight restoration: prosthetics, optogenetics and gene therapy, 115–123. ISSN: 0042-6989 (2015).
9. Gekeler, K. *et al.* Implantation, removal and replacement of subretinal electronic implants for restoration of vision in patients with retinitis pigmentosa. *Current opinion in ophthalmology* (2018).
10. Humayun, M. S. *et al.* Visual perception elicited by electrical stimulation of retina in blind humans. *Archives of ophthalmology* **114,** 40–46 (1996).
11. Jones, B. W. & Marc, R. E. Retinal remodeling during retinal degeneration. *Experimental eye research* **81,** 123–137 (2005).
12. Jung, J.-H., Aloni, D., Yitzhaky, Y. & Peli, E. Active confocal imaging for visual prostheses. *Vision research* **111,** 182–196 (2015).
13. Krumpaszky, H. G. & Klauss, V. Epidemiology of blindness and eye disease. *Ophthalmologica. Journal international d'ophtalmologie. International journal of ophthalmology. Zeitschrift fur Augenheilkunde* **210,** 1 (1996).
14. Lin, T.-Y. *et al. Microsoft coco: Common objects in context* in *European conference on computer vision* (2014), 740–755.
15. Marc, R. E. & Jones, B. W. Retinal remodeling in inherited photoreceptor degenerations. *Molecular neurobiology* **28,** 139–147 (2003).
16. Marc, R. E., Jones, B., *et al.* Extreme retinal remodeling triggered by light damage: implications for age related macular degeneration. *Molecular vision* **14,** 782 (2008).
17. Marszalek, M. & Schmid, C. *Accurate object localization with shape masks* in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), 1–8.
18. Nonnenmacher, M., Behrens, C., Berens, P., Bethge, M. & Macke, J. H. Signatures of criticality arise from random subsampling in simple population models. *PLoS Computational Biology* **13,** e1005718 (2017).
19. Ong, J. M. & da Cruz, L. The bionic eye: a review. *Clinical & experimental ophthalmology* **40,** 6–17 (2012).
20. Opelt, A., Pinz, A., Fussenegger, M. & Auer, P. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28,** 416–431 (2006).
21. Pillow, J. W. *et al.* Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature* **454,** 995 (2008).
22. Rizzo III, J. F. *et al. Retinal prosthesis: an encouraging first decade with major challenges ahead* 2001.
23. Rodieck, R. W. Quantitative analysis of cat retinal ganglion cell response to visual stimuli. *Vision research* **5,** 583–601 (1965).
24. Rosenholtz, R., Li, Y. & Nakano, L. Measuring visual clutter. *Journal of vision* **7,** 17–17 (2007).

25. Tadmor, Y. & Tolhurst, D. Calculating the contrasts that retinal ganglion cells and LGN neurones encounter in natural scenes. *Vision Research* **40,** 3145–3157. ISSN: 0042-6989 (2000).

26. Zhao, Y. *et al.* Image processing based recognition of images with a limited number of pixels using simulated prosthetic vision. *Information Sciences* **180,** 2915–2924 (2010).

27. Zrenner, E., Stett, A., *et al.* Can subretinal microphotodiodes successfully replace degenerated photoreceptors? *Vision research* **39,** 2555–2567 (1999).

28. Zrenner, E. Will retinal implants restore vision? *Science* **295,** 1022–1025 (2002).

29. Zrenner, E., Bartz-Schmidt, K. U., *et al.* Subretinal electronic chips allow blind patients to read letters and combine them to words. *Proceedings of the Royal Society of London B: Biological Sciences* **278,** 1489–1497 (2011).

A Python Code

The full code can be found on
https://github.com/sleep-yearning/artificial-retina-enhancement.

A.1 Image Processing Methods

Some exemplary methods to show, how stimuli generation was done.

```
import os
import scipy.io as sio
import numpy as np
from PIL import Image, ImageFilter

#general file access
src_folder = ".../IMAGE-SELECTION/SELECTION/combined/"
dest_folder = ".../IMAGE-SELECTION/processed/"
annotation_folder = ".../pascal-parts/Annotations_Part/"
types=os.listdir(src_folder)
```

A.1.1 Image Outline

```
#get image outline for single image
def image_outline(imagenumber,image_class,factor=3):
    #load annotation and access mask data
    mat_content = sio.loadmat(annotation_folder+imagenumber+'.mat')
    annotation = mat_content['anno'][0,0]
    objects = annotation[1][0]

    #collect object annotations that match current category
    object_pos=[]
    for obj in objects:
        if obj[0][0]==image_class:
            object_pos.append(obj[2])
    #create combined outlines from interesting object masks
    outline=np.zeros((333,500))
    for pos_field in object_pos:
        if pos_field.shape[0]>pos_field.shape[1]:
            pos_field=np.rot90(pos_field,-1)
        #as images are cropped to the fixed size, annotations have to be, too
        location=pos_field[:333,:]
        #check for background pixels near object pixels-> outline
        for j in range(0,333):
            for i in range(0,500):
                if location[j,i]==1:
                    if np.any(location[max(0,j-factor):min(333,j+factor+1),
                                       max(0,i-factor):min(500,i+factor+1)]==0):
                        outline[j,i]=1
    #outline is white, everything else black
    return Image.fromarray(outline*255).convert('RGB')
```

A.1.2 Background Contrast Reduction

```
#reduce contrast of a single image
def reduce_contrast(imagenumber,image_class,factor=0.4):
    #load annotation and access mask data
    mat_content = sio.loadmat(annotation_folder+imagenumber+'.mat')
    annotation = mat_content['anno'][0,0]
    objects = annotation[1][0]

    #collect object annotations that match current category
    object_pos=[]
    for obj in objects:
        if obj[0][0]==image_class:
            object_pos.append(obj[2])

    #create combined mask of pixels where contrast should stay normal from interesting object masks
    outline=np.zeros((333,500))
    for pos_field in object_pos:
        if pos_field.shape[0]>pos_field.shape[1]:
            pos_field=np.rot90(pos_field,-1)
        #as images are cropped to the fixed size, annotations have to be, too
        outline+=pos_field[:333,:]

    #sub function to apply on the points
    def contrast_reduction(pix):
        return 128 + factor * (pix - 128)

    #image access
    im = Image.open(src_folder+image_class+"/"+imagenumber+".jpg")
    pixels = im.convert('RGB')
    imarray = np.array(pixels)
    #image with reduced contrast to take pixels from where background is
    reducedarray = np.array(im.point(contrast_reduction).convert('RGB'))

    #decision matrix for 'where' function contains information where background is
    background= np.where(outline<1,1,0)
    imarray[:,:,0]=np.where(background,reducedarray[:,:,0],imarray[:,:,0])
    imarray[:,:,1]=np.where(background,reducedarray[:,:,1],imarray[:,:,1])
    imarray[:,:,2]=np.where(background,reducedarray[:,:,2],imarray[:,:,2])
    return Image.fromarray(imarray)
```

A.1.3 Object Part Segmentation

```
#segment single image including object parts
def segment_parts(imagenumber,image_class):
    #load annotation and access mask data
    mat_content = sio.loadmat(annotation_folder+imagenumber+'.mat')
    annotation = mat_content['anno'][0,0]
    objects = annotation[1][0]

    #collect object annotations that match current category
    part_pos=[]
    for obj in objects:
        if obj[0][0]==image_class:
            part_pos.append(obj[3])
    #some categories do not have part segmentation annotations
```

```
    #and return just the object segmentation result
    if (part_pos[0].shape==(0,0)):
        return segment_objects(imagenumber,image_class)
    #image access
    im = Image.open(src_folder+image_class+"/"+imagenumber+".jpg")
    pixels = im.convert('RGB')
    imarray = np.array(pixels)

    #outlines for 3 color channels separately
    outlinesa=[]
    outlinesb=[]
    outlinesc=[]
    #create matrix with number of parts annotated in each pixel
    #(to divide corresponding added colors by that number in the end)
    overlap=np.zeros((333,500))
    for part in part_pos:
        #some rare image annotations strangely contain empty part annotations, those have to be ignored
        if len(part)!=0:
            for pos_field in part[0]:
                part_array=pos_field[1]
                #rotate annotation matrix if portrait shaped
                if part_array.shape[0]>part_array.shape[1]:
                    part_array=np.rot90(part_array,-1)
                #as images are cropped to the fixed size, annotations have to be, too
                location=part_array[:333,:]
                #when a part of the object of interest is inside current annotation matrix
                #compute mean color of respective pixels and put those at the position of the part
                if np.sum(location)!=0:
                    midval = np.sum(np.where(np.expand_dims(location,2),imarray,np.zeros((333,500,3))),
                                                                                    axis=(0,1))/np.sum(location)
                    outlinesa.append(location*midval[0])
                    outlinesb.append(location*midval[1])
                    outlinesc.append(location*midval[2])
                    overlap+=location


    ##add object outlines for pixels that are not in any part of it (e.g. monitor frames)

    #collect object annotations that match current category
    object_pos=[]
    for obj in objects:
        if obj[0][0]==image_class:
            object_pos.append(obj[2])

    #create matrix with object pixels that are not inside any of their parts
    outeroverlap=np.zeros((333,500))
    for pos_field in object_pos:
        if pos_field.shape[0]>pos_field.shape[1]:
            pos_field=np.rot90(pos_field,-1)
        #as images are cropped to the fixed size, annotations have to be, too
        location=pos_field[:333,:]
        #annotation of complete object only useful for pixels that are not in a part
        remaining=np.maximum(np.zeros((333,500)),location-overlap)
        #when object of interest is inside current annotation matrix compute
        #mean color of object pixels and put those at the remaining pixels of the object
        if np.sum(location)!=0:
            midval = np.sum(np.where(np.expand_dims(location,2),imarray,np.zeros((333,500,3))),
                                                                    axis=(0,1))/np.sum(location)
```

```
            outlinesa.append(remaining*midval[0])
            outlinesb.append(remaining*midval[1])
            outlinesc.append(remaining*midval[2])
            outeroverlap+=remaining

    #total pixels that are annotated by either parts or objects (of interesting class)
    overlap+=outeroverlap

    #map of background pixels
    backposition=1-(np.minimum(overlap,np.ones((333,500))))
    #number of background pixels
    backcount=np.sum(backposition)
    if backcount!=0:
        #compute colorchannels for the background pixels (where no interesting object was annotated)
        backgrounda=np.where(overlap==0,imarray[:,:,0],np.zeros((333,500)))
        backgroundb=np.where(overlap==0,imarray[:,:,1],np.zeros((333,500)))
        backgroundc=np.where(overlap==0,imarray[:,:,2],np.zeros((333,500)))

        #compute mean color of background
        backcolora=np.sum(backgrounda)/backcount
        backcolorb=np.sum(backgroundb)/backcount
        backcolorc=np.sum(backgroundc)/backcount
    else:
        backcolora=0
        backcolorb=0
        backcolorc=0

    #number of objects that correspond to one pixels with their mean color
    #replace 0 with -1 so that division by 0 is avoided,
    #even though values on those positions are never used
    divisor=np.where(overlap==0,np.ones((333,500))*-1,overlap)
    #compute mean color for pixels where objects are
    coloredobjectsa =np.where(overlap>0,sum(outlinesa)/divisor,np.zeros((333,500)))
    coloredobjectsb =np.where(overlap>0,sum(outlinesb)/divisor,np.zeros((333,500)))
    coloredobjectsc =np.where(overlap>0,sum(outlinesc)/divisor,np.zeros((333,500)))

    #combine background and object pixels to image
    channela=np.where(overlap==0,np.ones((333,500))*backcolora,coloredobjectsa)
    channelb=np.where(overlap==0,np.ones((333,500))*backcolorb,coloredobjectsb)
    channelc=np.where(overlap==0,np.ones((333,500))*backcolorc,coloredobjectsc)

    #combine color channels to image
    imarray =np.stack((channela,channelb,channelc),axis=-1)

    #operations require casting to uint8
    return Image.fromarray(imarray.astype('uint8'))
```

A.2 Receptive Field and Ganglion Spike Computation

```
import numpy as np
from PIL import Image
import os
```

A.2.1 Receptive Field Generation

```
#number of DoG fields to create, size of input matrix to consider, sigma of DoG
#and mahalanobis distance from RF center for which values shall be computed
def generate_receptive_fields(count,size,sigma,mahalanobis=5):
    x_size=size[0]
    y_size=size[1]
    rec_fields=[]
    #needed to determine which points still contribute to RF
    deviationfactor=np.max(sigma)
    radius=int(np.rint(np.sqrt(deviationfactor)*mahalanobis))
    #values that are the same for all positions and don't have to be calculated for each RF
    determinant_sqrt = np.sqrt(np.linalg.det(sigma))
    inverse = np.linalg.inv(sigma)
    constant_factor= np.divide(1.0,np.multiply(2*np.pi,determinant_sqrt))
    #constant values for the outer gaussian,
    #the covariance matrix is set to double of the center gaussian
    surround_sigma = sigma*2
    surround_determinant_sqrt = np.sqrt(np.linalg.det(surround_sigma))
    surround_inverse=np.linalg.inv(surround_sigma)
    constant_surround =np.divide(1.0,np.multiply(2*np.pi,surround_determinant_sqrt))
    #loop for each RF
    for i in range(0,count):
        #random sampling for a center in the image/'retina' area
        y_pos =np.random.randint(0,y_size)
        x_pos =np.random.randint(0,x_size)
        field=np.zeros((x_size,y_size))
        #compute contribution to RF activation for each point in a the radius around the center of
        #the RF the radius depends on the covariance matrix and the mahalanobis cutoff distance
        for x in range(x_pos-radius,x_pos+radius):
            for y in range(y_pos-radius,y_pos+radius):
                if((x-x_pos)**2+(y-y_pos)**2)<deviationfactor*mahalanobis**2:
                    #Difference of Gaussians value for the point
                    activationvalue=np.multiply(constant_factor,np.exp(np.multiply(-(1.0/2),
                                    [x-x_pos,y-y_pos])@inverse@
                                    (np.array([x-x_pos,y-y_pos]).transpose())))
                                    -np.multiply(constant_surround,
                                    np.exp(np.multiply(-(1.0/2),[x-x_pos,y-y_pos])@
                                    surround_inverse@(np.array([x-x_pos,y-y_pos]).transpose())))
                    #put contribution value to the pixel or add it
                    #to the edge of the 'retina' if it is outside
                    field[min(max(0,x),x_size-1),min(max(y,0),y_size-1)]+=activationvalue
        #add receptive field to the list
        rec_fields.append(np.array(field))
    return rec_fields
```

A.2.2 Spike Computation

```
#computes activation of given ganglion list given an input image with values between 0 and 1
#the percentage of resulting activations can be set.
#a learning algorithm thus cannot simply learn on the number of activations for a category
def compute_normalized_output(ganglions,image,percentactivated=10):
    spikes=[]
    activations=[]
    #compute activation value for each RF
    for receptive_field in ganglions:
```

```
        activations.append(np.sum(receptive_field*image))
    #spike threshold is dependend on the whished activated percentage
    threshold=np.percentile(activations,100-percentactivated)
    #0 or 1 is appended for each RF so that the given percentage of spikes is achieved
    for activation in activations:
        spikes.append(np.where(activation>threshold,1,0))
    return spikes
```

A.3 Multinomial Logistic Regression Classifier

```
import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import os

condition="segment-object"
settype="output"
repeats=5
batch_size=5
epochs=200

class MultiRegression(nn.Module):
    def __init__(self):
        super(MultiRegression, self).__init__()
        self.l1 = nn.Linear(5000, 10)

    def forward(self, x):
        x = self.l1(x)
        x = F.softmax(x,dim=1).float()
        return x

def prediction_accuracy(nn,batch_size):
    correct = 0
    total = 0
    for i in range(int((len(testspikes))/batch_size)):
        spikes = Variable(torch.from_numpy(testspikes[i:i+batch_size]).float())
        truth=(testtruth[i:i+batch_size])
        outputs = nn.forward(spikes)
        prediction = outputs.data.numpy().argmax(axis=1)
        total += len(truth)
        correct += (prediction == truth).sum()
    return correct / total

def train(batch_size,epochs,learnr=0.3,feedback_every_epoch=10):
    #Learning properties
    neuralNet =[]
    neuralNet = MultiRegression()
    lossFunc =nn.NLLLoss(weight=weights)
    parameters=neuralNet.parameters()
    optimizer = torch.optim.Adadelta(parameters,lr=learnr)
    results=[]
    for epoch in range(epochs):
        running_loss = 0.0
        running_correct=0.0
        batches=int(len(trainspikes)/batch_size)
```

```
        for i in range(batches):
            spikes=trainspikes[i:i+batch_size]
            categories=traintruth[i:i+batch_size]
            spikes=Variable(torch.from_numpy(spikes).float())
            categories=Variable(torch.from_numpy(categories))

            optimizer.zero_grad()

            outputs = neuralNet(spikes)
            error = lossFunc(outputs, categories)
            error.backward()
            optimizer.step()

            running_loss += error.data[0]
            running_correct +=np.sum(categories.data.numpy()==outputs.data.numpy().argmax(axis=1))
        if feedback_every_epoch!=0 and epoch%feedback_every_epoch==feedback_every_epoch-1:
            print("epoch: "+str(epoch+1)+", running loss: "+str(running_loss)+",
                                    running correct: "+str(int(running_correct)))
        results.append(prediction_accuracy(neuralNet,1))

    print('fine')
    classpercentages=[]
    for i in range(len(output_class_map)):
        iindices=(testtruth==i)
        stimuli=testspikes[iindices]
        classresult=(neuralNet(Variable(torch.from_numpy(stimuli).float())))
                                                .data.numpy().argmax(axis=1)
        classpercentages.append(np.sum(classresult==i)/len(classresult)*100)
    return [results,running_correct/len(trainspikes)*100,classpercentages]

#train test split indices generation
def split_data(parts=11):
    testelements=int(len(truth)/parts)
    testlist=[]
    trainlist=[]
    indices=np.arange(len(truth))
    np.random.shuffle(indices)
    for i in range(0,parts):
        testind=indices[testelements*i:testelements*(i+1)]
        trainind1=indices[:testelements*i]
        trainind2=indices[testelements*(i+1):]
        trainind=np.concatenate((trainind1,trainind2))
        testlist.append(testind)
        trainlist.append(trainind)
    return [trainlist,testlist]

#location of computed ganglion activations with class information
data_folder=".../ganglion-activation/"
#load output (all) or selected (easy10+difficult100) file from condition folder
data=np.load(data_folder+condition+"/"+settype+".npz")
activations=data['stimuli']
truth=data['truth']
#order of classes like encoded as ints in the output
output_class_map=
        ['chair','dog','bird','bottle','boat','tvmonitor','horse','aeroplane','person','car']

#weights to account for different frequency of classes in dataset
weights=np.zeros(10)
```

```
types=['output','selected']
for datatype in types:
    for condition in os.listdir(data_folder):
        print(condition)
        data=np.load(data_folder+condition+"/"+datatype+".npz")
        activations=data['stimuli']
        truth=data['truth']
        weights=np.zeros(10)

        for i in range(10):
            weights[i]=1-np.sum(truth==i)/len(truth)
        weights=torch.from_numpy(weights).float()

        #multiple dataset configuration averaging
        testaccuracies=[]
        trainaccuracies=[]
        classpercentages=[]
        for k in range(repeats):
            print('split ',k)
            indices=split_data()
            trainindices=indices[0]
            testindices=indices[1]
            for j in range(len(trainindices)):
                trainspikes=activations[trainindices[j]]
                testspikes=activations[testindices[j]]
                traintruth=truth[trainindices[j]]
                testtruth=truth[testindices[j]]
                results=train(batch_size,epochs,feedback_every_epoch=200)
                testaccuracies.append(results[0])
                trainaccuracies.append(results[1])
                classpercentages.append(results[2])
        np.savez(data_folder+condition+"/learning"+datatype,testaccuracy=testaccuracies,
                        trainaccuracy=trainaccuracies,classpercentage=classpercentages)
```