UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Escuela Técnica Superior de Ingeniería Industrial

Deep Learning models

for

Retinal Neural Encoding and Decoding

of

Light Patterns

Industrial Engineering Master Thesis

Author: Antonio Manuel Lozano Ortega
Director: José Manuel Cano Izquierdo
Codirectors: Eduardo Fernández Jover, Javier Garrigós Guerrero
Elche,

October 9, 2017
To my parents, who inspire me with their humanity since I remember,

To my brother, who will make the world a better place,

To Momoko, the light on my sky.
«Todo hombre puede ser, si se lo propone, escultor de su propio cerebro».

Santiago Ramón y Cajal
INDEX

Acknowledgements ................................................................................................. 1
Abstract .................................................................................................................. 2
Introduction ............................................................................................................ 4
Motivation and goals ............................................................................................... 6
An initial approach to retina modeling ..................................................................... 7
A Machine/Deep Learning Paradigm ....................................................................... 9
Chapter: Matherials and Methods and Experiment designing ................................. 12
  Ethical approval ................................................................................................ 12
  Materials and methods: Retinal recordings and pre-processing ............................ 12
  Stimuli .............................................................................................................. 14
Data processing ....................................................................................................... 16
  Image processing: ............................................................................................ 16
  Neural Activity ................................................................................................. 16
  Poisson spike generation .................................................................................... 17
A note on Cloud Computing .................................................................................... 18
Modeling the Neural Coding ................................................................................... 20
  The main tools ................................................................................................. 21
  Input ............................................................................................................... 21
  Output ............................................................................................................. 22
  Architectures .................................................................................................. 23
  Parameters, optimizers, training strategy ........................................................ 25
  Grayscale Moving Bars .................................................................................... 30
  Neural Coding of Color Flashes Patterns .......................................................... 32
Neural decoding...................................................................................................... 36
  A brief introduction .......................................................................................... 37
  Decoding retinal neural responses to moving bars ............................................ 39
  Decoding retinal neural responses color to homogeneous color images .......... 40
  Decoding retinal neural responses to natural images ........................................ 43
Discussion and future work .................................................................................... 45
Bibliography .......................................................................................................... 46
Acknowledgements

First of all, I would like to thank professors Javier Garrigós and Javier Martínez for all this years of intense discussions and ininterruped support, as well as José Manuel Cano. This work owes so much to them. Jose Manuel Ferrández introduced me to the amazing neuroengineering world, as well as Joaquín Roca did with biomedical engineering, along with advice and guidance, the same guidance and support that I am very grateful to had from Ana Nieto.

Thanks to Eduardo Fernández for his trust, for sharing his vision of a better world for the blind people and for accepting me as part of the Biomedical Neuroengineering Research Group at UMH, with a team full of intelligent talented people as Sebastián Peña, with whom discussing about the mysteries of the mind and universe it’s a daily activity, and Cristina Soto, I consider her an example of professionalism and a great person. Thanks to Mario too for the beautiful model’s image and for that contagious entrepreneur spirit.

This work is the last stage of six years of continuous efforts, won and lost battles, six years of life which I shared with so many colleagues and friends in Cartagena, too much to mention everyone but I want to thank Checo, Ismael, Jan and Rubén and so many more great people who I am proud about and that are part of me.

I would never have reached any far without the incondicional love and support of my parents and my brother, and I will never have gratitude enough to them, as well as to Inma, Andrés, Daniel and Jesús, who are marvelous people that I am lucky to have around.

I want to thank to some of the great professors of Politecnico di Milano, including Pietro Cerveri and Riccardo Barbieri, it is been a pleasure to learn with them, as well as to work with Mohtashem and the neuroengineering team colleagues.

I will never forget the great moments in Milán with my people from seven nations, who showed me how much amazing people live around the world, and I keep them in my heart.

Finally, I want to show my profound love and gratitude to Momoko, who makes me the luckiest human on this earth and whose love lights my heart no matter the time or distance.
Abstract

**Title:** Deep Learning models for Retinal Neural Encoding and Decoding of Light Patterns

**Purpose:** In this work, state of the art deep learning techniques are applied to build models able to imitate biological Retinal Ganglion Cell’s encoding of light patterns, through a data driven methodology. In addition, decoding systems will be also developed. Our purpose is doublefold: to contribute on the scientific understanding of the visual system functioning and to give a step further on developing technologies that some day will restore vision to the blind.

**Methods:**

The first stage of this work includes creating visual stimuli with python libraries and Matlab, and processing neural recordings in collaboration with the Biomedical Neuroengineering Research Group at Miguel Hernández University (UMH) using open source spike sorting software and Matlab. The models will be build in a Cloud Computing service using the most actual Deep Learning libraries used by the AI community, and will be based on the use of dense feed forward, recurrent and convolutional layers mainly. A Poisson point process will be used to generate spike trains that will be compared to the biological retina, among its spike firing rates.

**Results:**

A data-driven methodology for modeling retinal ganglion cell responses to different kinds of light patterns has been applied to a biological mouse retina with positive results, showing a high correlation with the spiking firing rates obtained from the biological spike train responses. Two different approaches were successfully tested: fitting single trial responses and PSTH fitting, having the second one a better spiking simulation results. In addition, the Peri-Stimulus Time Histogram built from Poisson-simulated spike trains showed a similar behavior in both the model and the real retina. On the way to achieve these results, several structural and parametric decisions were taken for the model, resulting in a 3D CNN model that showed high sensitivity to the activity and parametric regularizers on the dense layers on one side, and to the variance of the gaussian with which the spike trains were filtered on the other hand. These facts revealed that the retinal spiking variability handling will play an important role in future developments of the model.

On the other hand, deep learning systems for neural decoding have been deployed, showing the feasibility of neural networks to reconstruct grayscale images and color patterns, providing better results with Long Short Term Memory units. This opens the door to a more extensive research on this area, that will be explored on the future.
Conclusions:

The model deployed here implies a new tool to step further on not only the neural encoding of spatio-temporal light patterns but also in the effect of luminance and color coding of the retina, given that is able to mimic the different characteristic response shapes, and we will explore this feature in depth in future works. For this, new experiments will be set up in a way that our model is able to mimic a wider and representative range of retinal behaviors and reveal new insights on it.

In the case of neural decoding, we proved experimentally that an LSTM-Deconvolutional neural network is able to map a temporal series of neural activity to homogeneous color patterns, grayscale moving targets and certain number of arbitrary complex natural images, thus more extensive research on this area needs to be done to build a more generalizing deconstructing model, that may be useful, when applied to both biological and artificial retinas as way to compare retina models and give insight on how much information are this models able to convey, with decisive implications on prosthesis development, along with the neural coding model.

Keywords: Retina modeling, Neural coding, Neural Decoding, Deep learning, Convolutional neural networks
Introduction

Neurosensoric systems still remain as a big challenge for scientists, although great achievements have been done so far. Combined efforts from several biological and engineering disciplines have contributed to the knowledge and technology that allows us to prevent, treat and hopefully overcome some of the human diseases and limitations by building the knowledge corpus, diagnostic systems, rehabilitation treatments and prosthesis for the disabled among others.

In this work, a deep learning data-driven approach is proposed to address one of this early neurosensoric pathways challenges: modeling the neural activity of a mammal retina in response to the stimulation with light patterns. In this approach, retinal neural recordings in response to the stimulation with homogeneous and checkerboard combinations of color flash patterns were processed and fed into a supervised machine learning system—a 3D Convolutional Neural Network (3D CNN)—able to process spatio-temporal visual stimuli and reproduce the retinal behavior.

This kind of 3D CNN has been used to learn spatiotemporal features[1] and convolutional neural networks in general are being applied successfully to model several ways of information processing of the visual system[2][3].

In the past, several retinal ganglion cell models have been proposed. Some of them are used as general models of early neurosensoric pathways, like Linear-Nonlinear[4]Generalized Linear Models[5]or Integrate and Fire[6] and their application provided different results in either explained variance, average number of spikes or correlation as it is shown on[7]. Recent works have proposed hybrid systems as fine tuning physiological models with Genetic Algorithms[8] and the use of Deep Learning techniques[7][9]. These new models were derived from recent advances in the field of machine learning that witnessed the rise of new deep artificial neural networks, layers and architectures—Convolutional Neural Networks, Long Short Term Memory layers, Gated Recurrent Units, Generative Adversarial Networks—with a series of features that helped to handle the vanishing gradient problem and performed better in some situations[10]. Some of these features use techniques like dropout[11] for regularization purposes, parameter sharing that decreased the computational cost per layer, and the wide use of alternatives to the sigmoid and hyperbolic tangent activation functions, like ReLU, LeakyRELU, PReLU or ELU functions.

In addition, the high computational charge that these machine learning systems entail has been notably tackled by the use of GPUs, simplifying the model’s training and tuning procedures.
Recently, Convolutional Neural Networks have been proved to be a great tool for visual recognition problems, outperforming other traditional machine learning techniques[12] as Support Vector Machines, with the significant advantage of being able to perform end-to-end learning, i.e., the absence of a necessity for handcraft features. In addition, machine learning techniques in general are being showed to outperform traditional techniques also in neural decoding tasks[13].

This, and the structural analogy between CNNs and the visual LGN-V1-V2-V4-IT pathway[14] following the way of Fukushima’s Neocognitron[15], makes CNNs to be a likely suitable solution for the bio-inspired vision encoding task.

This kind of CNN, data-driven methodology has recently been used for modeling tiger salamander retina ganglion cells[7] in response to both white noise and grayscale natural scenes, and it is also the kind of methodology on which our modeling is driven. In[7], convolutional neural networks were proved to be able to model more accurately retinal responses to both natural and artificial stimuli than the previous techniques and generalize better across stimuli types. In the same kind of approach, Recurrent Neural Networks have recently been used to predict primate retinal responses on grayscale natural images[16].

In our approach, a 3D Convolutional Neural Network is trained to mimic the fire rates obtained after stimulating real mouse retinas with simple color patterns, with the aim to advance in building a system able to capture not only grayscale but also the possible color coding of a biological retina.
Motivation and goals

This work aims to be both a step further on the basic research and understanding of the retina and on the development of biomedical technologies.

Deep Learning offers flexible, powerful tools that are being proved useful in the advance of our understanding of the neural coding of neurosensory systems. In this work, a 3D Convolutional Neural Network is used in order to mimic the behavior of a population of mouse retinal ganglion cells in response to color light patterns, with the aim to advance in building a system able to capture not only grayscale but also color coding of a biological retina, in an effort to step towards the development of bioinspired systems that could help us understand how their biological counterparts works and mimic them as a potential way to enhance millions of people's lives that suffer from visual degeneration or impairment.

Our main concrete goals will be:

- To design and perform visual stimuli experiments stimuli with a mammal’s biological retina in collaboration with the Biomedical Neuroengineering Research Group at Miguel Hernández University.

- To analyze the data and process it to be convenient for the modeling task.

- To select a proper Cloud Computing with Graphical Processing Units suitable to develop the deep learning systems in a fast an efficient way.

- To build deep learning models able to perform the necessary computations to mimic the retinal encoding of light patterns with both accuracy and a qualitative similar behavior.

- To create a related decoder that maps the ganglion cells activity to the stimulus that the retina was given.
An initial approach to retina modeling

Since long ago, the first stage of vision processing has been a target of scientific research. At the beginning the retina was believed to be a simple edges filter [18] but through years of research it was proved to be a highly dynamic system that performs several spatio-temporal nonlinear processing steps [19].

![Image](image.png)

*Figure 1. The retina performs a series of spatiotemporal nonlinear computations within a highly elaborated circuitry. Image from [https://sites.duke.edu/kaplab/](https://sites.duke.edu/kaplab/)*

To give insight to how the retina works, several models of its functioning have been proposed, from purely mathematic physiological descriptions and hybrid approaches[8] to close to fully data-driven models [7]. Our approach is based on a purely data driven methods, while some similarities had been established between Convolutional Neural Networks and the Visual System information processing [14,15].

Here we will present briefly two of the most successfully models used to model the retinal behaviour in response to light.
**Linear-Nonlinear model:**

This model consists of a set of predefined or fitted by an optimization process linear filters applied to the incoming image, and which output is weighted and passed through a nonlinear function which will provide an estimation of the spiking firing rate [5], which then allows to generate spikes by means of a Poisson point process—that will be commented afterwards on this work. This is a standard model used widely to model neurosensory systems responses.

![Diagram of a Linear-Nonlinear model for spike generation.](http://pillowlab.princeton.edu/teaching/mathtools16/slides/lec17_GLMs_LogisticRegression.pdf)

**Integrate and Fire**[6]

In a first approximation, the neural behavior could be described as an integration process of an input (in this kind of models the input is usually current I(t) induced by another neurons) which drives the evolution of a membrane potential u(t). When this potential rises above a defined threshold, a spike event occurs, and the membrane potential is reset to a lower value. Several variations of this model have been developed in order to describe different phenomena, as the Leaky-Integrate and Fire, that solves the “memory” problem—temporal dynamics—by adding a leak term to the membrane potential, modeling the diffusion process of the ions that occurs when some equilibrium is not reached on a biological cell.

\[
I(t) - \frac{V_m(t)}{R_m} = C_m \frac{dV_m(t)}{dt}
\]

In the next chapter, we will present our actual approach to this task.
A Machine/Deep Learning Paradigm

As it already has been presented, our aim is to create a model that will map images or sequences of images to spike trains –this is, mimicking the retinal neural code-, and go the opposite way: mapping neural activity to the stimuli that originated it. This can be seen as a traditional supervised fashion machine learning problem, when a n-dimensional input is mapped to a n’-dimensional output –n and n’ can be equal or different-. In our application, n will be the number of pixels –multiplied by three in the case of rgb images- and n’ will be the number of neurons. The cited dimensions will be exchanged when talking about neural decoding –mapping the neural responses to the stimuli that produced it-.

Given the impressive advances in computer vision by means of deep learning, this becomes a natural option when thinking about working with images.

Figure 3. Deep Learning systems have the advantage to be able to work with raw data and learn internal representations automatically. Image from http://cs231n.github.io/convolutional-networks/

Deep Learning[14] it’s a set of techniques within the Machine Learning subfield that have been proved to work well in several tasks, like computer vision, natural language processing, robot automation, self driving cars, automated medical diagnosis [20]. This set of tools has its base in Artificial Neural Networks, since this particular algorithm its able to create successively more abstract “internal representations” as far as it goes “deep” and this way is able to process raw signals and extract automatically its most revelant features. This feature of deep learning algorithms suppose a huge advantage with respect to “traditional” Machine Learning techniques, where the ability of the engineer that extracts the features of the object of study to feed them into the algorithm. Deep Learning are composed of systems that are able to learn internal representations by itself.
Several factors contributed to the rise of Deep Learning. First of all, the main algorithms were developed several years ago, as Backpropagation [21], Autoencoders [22], Deep Belief Networks [23], Convolutional Neural Networks [12], Recurrent Neural Networks [24], but this had to wait until both a huge boost on the computers parallel computing power—the use of Graphical Processing Units, GPUs, to perform thousands of computations in parallel each cycle—and the start of the Big Data era, where it’s possible to collect enough data samples to train the algorithm and, if properly done, achieve human or super-human level performance [25]. To give an example, the most known Computer Vision (CV) challenge, ImageNet has 14,197,122 millions of images at the present moment which the CV algorithms have to classify in one of a thousand categories.
Deep Learning algorithms based mainly in Convolutional Neural Networks are exceeding the rest in this challenge, including Support Vector Machines by a significative margin since 2012 [12]. In addition, severe problems like the “Vanishing/Exploding Gradient” [26] have been partially or totally solved, in part by the use of Rectified Linear Units and similar as activation functions, allowing the machine learning engineers to “go deep” in creating this kind of systems.

In the rest of this work, we will rely mainly on Convolutional Neural Networks to process or reconstruct images and in fully-connected Feed Forward layers to build our models.
Chapter: Materials and Methods

To obtain the training data that we will feed to our models with the aim of imitate a biological retina, several experiments were designed and carried on in the Visual rehabilitation and Neuroprosthesis Group laboratory in the Miguel Hernández University.

In this chapter, the main details, protocols and tools used are described.

Ethical approval

All experimental procedures conformed to directive 2010/63/EU of the European Parliament and of the Council, and the RD/53/2013 Spanish regulation on the protection of animals use for scientific purposes and approved by the Miguel Hernández University Committee for Animal use in Laboratory.

Materials and methods: Retinal recordings and pre-processing

Retinas were extracted from adult C57BL/6J mice following the same preparation setup as in [27]. Extracellular recordings were obtained from the retinal ganglion cell layer from the isolated mouse retinas with an Utah Electrode Array consisting of 10x10 platinum electrodes with a 400μm electrode interspacing [28].

The data recorded from each electrode were digitized with 16-bit resolution and 30 kHz sampling rate and stored for further analysis. The recorded spike events were sorted using Neural Sorter [29] an open source tool application for offline spike sorting analysis.

Two different patterns of visual stimulation were applied to the retinas. The first one consisted in a set of six color flashes of 500ms duration interspersed with a dark uniform stimulus for 1500ms repeated 25 times, for a total duration of 300s. The second visual pattern consisted of a sequence of 24 binary color checkerboard combinations (each square spanning 8x8 pixels) of 500ms duration followed by a dark uniform stimulus for 1500ms, for a total stimulus length of 48s.
The recorded spike trains from the retinal ganglion cells that provided insightful data were used in the analysis to develop our retina model. The training input of the model consisted on sets of 50 frames of 140x140 pixels, each frame corresponding to a 10ms of stimulus with a total duration of 500ms.

The output is a spiking probability function for each of the neurons recorded that provided insightful data. This function comes from the convolution of spike trains with a Gaussian function which transforms these discrete spike events into a virtually continuous objective to fit, so it transforms a classification problem -discrete classes corresponding to number of spikes in a bin- into a regression one -analog firing function-.

Given that the darkness duration lasted 1500ms, to allow the retina recover from consecutive light stimulations, a relevant subset of training data consisted of spontaneous firing corresponding to the recovering time of the retina. Therefore only the initial 500ms of darkness after each flash were included in the model, reducing the training time with a more balanced dataset.
Stimuli

To create a reliable model of a retina, several experiments have to be carefully designed. To create and project the visual stimuli to the retina, both Matlab and python’s VisionEgg library were used. Here we show the main details of the stimuli.

The temporal scheme for the images presentation was coded using VisionEgg[30]. This python library allows to use inexpensive computers to present visual stimuli for vision research experiments, and to program a trigger each time that a new frame is presented, this way it is possible to accurately relate the neural activity with each image.

![Figure 7. A demonstration of different visual stimuli from the VisionEgg library](image)

**Bars**: Crossing bars in a homogeneous black or white background will allow us to model a basic behavior of the ganglion cells: ON, OF and ON/OFF cell types will rice or cease firing differently when a bar is crossing its receptive field. 80 seconds of bars stimuli each experiment with 10 different combination of angles and directions were used.
**Color flashes:** Homogeneous light flashes of 6 different colors interspersed with no stimuli (darkness) were showed to the retina. Creating a model able to mimic the responses may help in the retinal color coding study.

**Color checkerboards:** In addition to the homogeneous color flashes, different color combinations were shown in a checkerboard fashion in order to create a model able to encode both color and spatiotemporal features.

**Natural images:** A set of 4890 grayscale natural images were selected for one of the decoding tasks.
Data processing

Image processing:

In order to create a training/testing dataset, the stimuli presented to the retina was converted to python’s numpy[31]arrays with float 32 precision, in batches of D frames of 140x140 pixels, each frame representing a 10 miliseconds bin of stimuli. This way, our image matrix tensor will have a dimension of NxBxDxC pixels, with N the width/height of the image, D the depth in time (D can be 50 in the case of the 3D Convolutional Neural Network model that we will present afterwards, or 1 in the case of the grayscale image reconstruction).

In the checkerboard case, when the 5 dimensional arrays were too heavy to fit –in the case of the models that take time in account by having several image as inputs-, the byte precision of the image matrix was changed to int8, with values ranging from 0 to 255. To handle this high values, a Batch Normalization layer was added in the beginning of the models when necessary (this layer normalizes the data in values close to the range -1,1-).

Neural Activity

The recorded neural activity was sorted using the open-source NeuralSorter[29] software for handling neural events, that uses different methods for spike sorting based on PCA and different clustering algorithms.

After this stage, a .NEV file is generated, which contains all the relevant information of the recording, including the timestamp of all neural events with the electrode and the unit on which the event was registered, and its corresponding waveform.

![Figure 8. Neural Sorting Workflow](image)

16
Matlab code was generated to transform this .NEV structure in a spike matrix, with the purpose of creating a matrix containing the spike trains for each neuron that registered enough Signal to Noise ratio in a bin of 10 milliseconds.

This spike trains were smoothed afterwards with a Gaussian function in order to transform discrete spike events into probability of spikes, and this way take in account the single-trial variability of the retina.

**Poisson spike generation**

Given that our encoding model will fit firing rates or firing probability functions, we need a tool to generate the actual spikes. An stochastic process was selected, in the line of several neural models on the field [32], retina Stanford), giving that the retinal ganglion cells present certain variability[33].

A Poisson process is a kind of stochastic binary process defined by one variable r(t) that in our case will be the firing rate. This rate is dependent on time so our process will be an inhomogeneous Poisson process[34]. This means, for a time interval with a firing rate r1, spikes will be generated randomly with that overall firing rate, following a Poisson probability function.

\[
P\{n \text{ spikes during } \Delta t\} = e^{-r \Delta t} \frac{(r \Delta t)^n}{n!}.
\]

The simplest way of generating spikes based on the firing rate its calculating the probability of one single spike in a sufficiently small time interval (the order of milisecond can be enough) and give a spike if the probability surpasses 0.5 of 1.

\[
P\{1 \text{ spike during } \delta t\} \approx r \delta t
\]

*Figure 9. Spike count histogram calculated from many Poisson generated spike trains with a firing rate of 100 Hz in a bin on 1 msec. The continuous line represents the theoretical result for a perfect Poisson spike distribution. Figure and equations reproduced from http://www.cns.nyu.edu/~david/handouts/poisson.pdf*
A note on Cloud Computing

The first experiments in the development of our bioinspired deep learning models were performed in a local machine, although for an efficient development of the final models, a Cloud Computing platform was chosen following a criteria of specificity (it must be a machine learning/deep learning platform with NVIDIA (CITA NVIDIA) GPU), simplicity, flexibility and an enough computing power. This last requirement is covered by all the options considered, where we can count Google Cloud, Amazon Web services and Microsoft Azure and Floydhub among the most popular, offering a huge range of computing power and services.

Among all the options, FloydHub Cloud GPU was selected based on a few of key features:

- **Zero Setup**: as a Deep Learning service, FloydHub has already prepared totally configured environments that include CUDA(cita cuda), a compiler and set of tools for parallel computing that python libraries will use underneath, cuDNN(cita cudnn), a GPU accelerated library for deep learning and any of the main deep learning libraries used in the present as Tensorflow, Keras, Theano, Caffe, PyTorch, MxNet as well as the most usual Python libraries for arrays manipulation and data science in general (numpy, PIL, scipy...), which make the models development much less painful, without compatibility complications.

- **GPU instances offering a Nvidia Tesla K80 GPU with 12 GB RAM and 2496 CUDA cores, and 60GB of memory with 4 CPU cores**

- **Interactive Jupyter Notebooks**: thanks to this feature, it is much intuitive to work online and “see” what’s going on in your job meanwhile it is executing, or change it interactively which gives flexibility to the development.
• Per second billing at affordable prices, which makes it competitive in front of the big Cloud Computing competitors.

Figure 10. Floydhub Deep Learning Platform was selected for its GPU capability and its painless environment setup

To deploy a job, it is necessary, after creating an account, to log in from the Windows or Linux console, create a Dataset configuration and upload that Dataset to the Cloud, where it will be ready to be used in any Project that calls it. After this, it is possible to create a Project with a Python or Ipython notebook that can be run directly in the server or interactively within the Jupyter Notebook provided by the server, indicating a folder which will contain the possible output of the project.
Modeling the Neural Coding
The main tools

To build, train and test our deep learning systems we will rely on the Keras, a python library developed exclusively for deep learning applications that allows to develop models in a flexible, modular way. Keras acts as a wrapper of one backend to actually perform the computations – at the present moment, Theano, Tensorflow and Microsoft’s CNTK-. In addition, several python libraries for array manipulation, image processing and plotting, and general data science tools, among them are numpy, PIL, scipy, h5py and matplotlib.

Keras makes the development of deep learning systems higher-level, clear and efficient, and the backend chosen was Google’s Tensorflow, that allows automatic multiple GPU computing. In addition, the compatibility is full since Keras was chosen as an official interface to Tensorflow.

![Keras + TensorFlow](image)

*Figure 11. Keras offers a flexible, high level, modular interface to Tensorflow computations, without reducing the possibility to go deep in the computations background when necessary.*

Input

The training input of the model consisted on batches in a range of 20-50 frames (140x140 pixels) with a variable total duration (500ms in the case of the color flashes, 200ms in the case of the moving bars), and the output is the spiking probability function for each of the neurons recorded that provided insightful data. This function comes from the convolution of spike trains with a Gaussian function which transforms this discrete spike events into a virtually continuous objective to fit, so it transforms a classification problem -discrete classes corresponding to number of spikes in a bin- into a regression one -analog firing function-. 
Given that the darkness duration lasted 1500ms, to allow the retina recover from consecutive light stimulations, a relevant subset of training data consisted of spontaneous firing corresponding to the recovering time of the retina. Therefore only the initial 500ms of darkness after each flash were included in the model, reducing the training time with a more balanced dataset.

Output

Two different approaches were tested (both in a supervised learning way) in our application: in the first one, the smoothed ganglion cell responses were fed to the CNN in a single-trial way, i.e., each training example consisted on the firing probability in a 10 milliseconds bin during the whole recording leading to a multiple-batch experiment training, each batch corresponding to one entire pass throughout the stimulus pattern repeated through time. As in a biological neural system, the retinal responses to each trial presented certain degree of variability, thus we lead our model to learn a general response shape of the fire probability distribution among all the trials. For the homogeneous color flash patterns, convergence between training and validation loss was achieved around the 17th of 24 batch fed to the network in the first training epoch, meaning that the model achieved its best fitting of the retinal behavior before the whole recording from the responses to the pattern was shown to the neural network.

In a second training approach, the output for every neuron's model consisted in a smoothed version of each biological neuron's Peri-Stimulus Time Histogram (PSTH) in a single batch. This way, we feed the CNN with the most characteristic way of response for each ganglion cell, with a better signal-to-noise ratio. Four epochs through the overall responses of the cells were enough to achieve convergence, proven a more efficient performance with the PSTH approach.
This results will be useful in the design of future stimuli experiments, where the search for a tradeoff between number of different stimuli and number of repetitions will be considered.

![Graph](image)

*Figure 13. An example of the smoothed PSTH which the model has to learn*

**Architectures**

The architecture of the proposed model consists on a series of 3-dimensional convolutions of volumes of data coming from the stimulus video frames followed by a fully connected -dense- layer that will provide the desired output. In the case of the checkerboard patterns fitting, 4 convolutional layers followed by a fully connected layer worked best. Each convolutional layer is followed by a nonlinear activation function (ReLU, PReLU), a Batch Normalization layer (BN) and a Maxpooling layer to reduce the dimensionality of the output volume of the convolution layers, extracting the most relevant descriptive features by compressing the spatiotemporal information. An empirical, trial-and-error process was carried out to adjust the size and several parameters of the different layers.

The selected values represented a trade off between several tendencies. Thus, applying pooling to the temporal axis would mean more robustness to pattern shifts on this dimension and less computational charge on the consequent layers, but temporary dependence is a key to model the behavior of the retina, so an excess of temporal pooling would undermine time resolution and therefore, model’s prediction ability of the temporal dynamics.
Figure 14. Illustration of one of the models used. The input consisted in sets of 50 frames, and the output are the actual spiking firing rates of the biological neurons.

After these stages the resultant tensor is flattened, and it inputs a fully connected layer followed with another nonlinear activation function - ReLu - that will output the firing probability or firing count prediction for each neuron in the next ten milliseconds.

The actual spikes were then generated by means of a Poisson process with a millisecond resolution for each firing probability estimate preceded by a firing rate normalization in the case of the PSTH curve fitting.

The number and size of the filters of the convolutional layers were varied to achieve the best results. As expected, the temporal dimension of the kernels was the most relevant set of parameters, and the first layer’s parameters had a key role in the results given that this first convolution addresses the three color channels of the image batches (RGB values of the pixels).

For the weight initialization, a Lecun uniform function was used, which takes samples from an uniform distribution parametrized in relation to the number of inputs to that layer. Both L1 and L2, weight regularizers and eventually activity regularizers were also included among the parameters for every layer. The network building and training was performed using the widely adopted deep learning frameworks Tensorflow[35] and Keras[36].
Parameter selection, optimizers and training strategy

The main parameters taken in account are shown below. A brief description of the layers and several concepts are given. In this work, a minimal deep learning knowledge is presupposed, but informative links are given for each element:

Type of layer:

**Dense** [37] This layer consists on a 1 dimensional set of neurons which take input from the previous layer. It is also called “fully connected” because each neuron of this layer receives the input from all the neurons of the previous one. Thus, the output of the neuron will be result of the weighted sum of the activations from the previous layer (including a bias term) passed through a nonlinearity (usually a sigmoid, hyperbolic tangent or Rectified Linear Unit function).

\[
s_j = g_j \left( \sum_{i=1}^{I} w_{ji} s_i - b_j \right)
\]

![Diagram of a multilayer perceptron](image)

*Figure 15. Multilayer perceptron. The output of each neuron consists of a weighted sum of the activations from the previous layer, passed through a nonlinear function.*

**Convolutional** [38]: The output of this layer will be a collection of feature maps that are a result of the convolution of a learned nonlinear filter with the input of the previous layer.
Flatten: This is a simple operation that consist of transforming any arbitrary dimensional array into a 1D array by ordering its elements.

Reshape: Similar as Flatten, reshape transforms any arbitrary dimensional array into an array with the specified dimensions, with the restriction that the new and the old arrays have the same number of elements.

Batch Normalization\cite{39}: This operation has been proved to accelerate the training of deep neural networks by normalizing each layer’s output by its mean an variance.

Max Pooling: Pooling layers reduces the dimensionality of the input while preserving the most relevant information –we must be careful with this because a too aggressive pooling can throw away too much information-, thus reducing the computational complexity of the models. Specifically, a Max pooling layer outputs the maximum value of the matrix part on which it is applied. Most common max pooling operations used in deep learning are 2x2 (heigh, width, on image applications).
**Dropout:** this is a training procedure based on randomly turning off the connections between neurons during the forward pass, forcing this way the model to perform its task being robust to perturbations and noise. Dropout is used as a regularization technique in Deep Learning[11].

**Number of layers:** a higher number of layers usually implies an augmentation of the network’s capacity to generate higher level –more abstract- features of the input

**Number of neurons on each layer:** a wider layer means more possible combinations of the inputs from the previous layer, and therefore a richer set of features learned. In some cases, a bottleneck layer its used: this is, a layer with a significantly less number of neurons than the previous in order to forced the network to compress the information. This is an efficient way to compress images, for example, and it is the main principle of neural autoencoders[22].
Activity regularizers: it takes in account the magnitude of the output activations of the computing element, and it forces it to be active to only the relevant features on its input.

Weight regularizers: this feature helps to maintain a low magnitude on the weights and thus prevent overfitting by adding a penalization to the high weight values on the cost function.

$$C = -\frac{1}{n} \sum_{x_j} \left[ y_j \ln a_j^L + (1 - y_j) \ln (1 - a_j^L) \right] + \frac{\lambda}{2n} \sum_w w^2.$$ 

Dropout percentage: the higher the dropout, the higher the number of neuron’s output that will be set to zero on the forward pass. It usually helps to build a model more robust to noise, and it is used widely as another regularizer technique.

Optimizers: Adam, Stochastic Gradient Descent, RmsProp[40] all of them based on the Gradient Descent method to train neural networks [21], with the addition of momentum and second order momentum terms, gradient history terms, exponentially decaying average of the gradients.
Figure 21: Training a Neural Network in a supervised way implies solving an optimization problem. On the image: several steps of gradient descent towards a global optima on the loss function.

**Loss functions**[41]: Mean Squared Error, and Mean Squared Logarithmic Error, Binary Cross-entropy, and Poisson loss were the main ones used here. The Poisson loss was a main option given that it is related with the log-likelihood of two variables under Poisson distribution assumption, and the spikes of the encoding model were actually generated by a temporal varying Poisson process.

For the training schedule, a Keras Early Stopping callback was created. This has the purpose of measuring both training and validation loss and stop the training when the validation loss cease to decrease for a predefined number of steps (called “patience”), and therefore prevent overfitting, that occurs when the model fits perfectly the training set but fails to generalize to new examples.
Grayscale Moving Bars

This first model will fit the retinal responses to spatiotemporal patterns consisting in white bars crossing in different angles—ranging from 0 to 270 degrees—with a black background. For this, a 3D Convolutional Neural Network will be built. This CNN consists in three convolutional layers that accepts 10 frames of grayscale video (and thus, it process a 100ms history of the input), followed by a hidden dense layer connected to a final output layer that predicted the spiking firing rate. Weight regularizers were used in the convolutional part and activity regularizers in the outpour layer, thus promoting a low firing rate baseline. Only three epochs with a batch size of 50 were needed to achieve a good fitting, which results are shown below.

*Figure 22. Up: Poisson-simulated raster plots from the CNN model versus real responses. Down: modeled firing rates after training*
For this particular application, the correlation between firing rates, in despite of being high, is not the most relevant parameter, but the firing timing and the ON/OFF/ON-OFF behavior, that our model was able to present after the training.

In addition, a simpler 2D convolutional model was created (with only one convolutional layer that takes just a frame of the video as an input and a dense layer that produces the desired firing rate) in order to fit the neurons response and visualize the filter learned, showed below:

![Convolutional layer](image)

*Figure 23. One convolutional layer with eight filters was enough to provide a reasonable fitting. On the image, raster plot of one of our artificial neurons responding during 10 trials to a crossing bar for one specific orientation.*

With this model, we can visualize which are the activations that produce the maximum response to each output neuro, since we set the convolutional layer to not produce any change.
in the input images dimension. This may be a way to find receptive fields while using white noise or natural images, and it will be explored in the future.

![Image](image.png)

*Figure 24. The input images that can activate a neuron the most can be visualized from the model. On the left, an ambiguous function was learned. On the right it can be observed that this neuron had a specific selectivity to horizontal and vertical bars.*

**Neural Coding of Color Flashes Patterns**

Training was performed over two different datasets: homogeneous color flashes and a checkerboard dataset with binary color combinations. The flashes training set consisted of 30000 data samples (obtained from 3000 seconds of stimulus-response recording of sliding volumes of frames) and the correspondent probability of spike firing for 37 neurons, with a standard 70–30\% split for training/validation. The same ratio was used with the checkerboards datasets, having these a total duration of 8 minutes (a total of 48000 data points).

Different dropout percentages were used in the layers of the network to prevent overfitting and several optimizers like ADAM, Stochastic Gradient Descend and RMSprop were tested. The loss functions that achieved better performance were Poisson and Mean Squared Logarithmic Error (MSLE), which lead to slightly different characteristic shapings of the objective function.

The figure below shows the evolution of Poisson loss objective function for training and validation sets as training was performed in the flashes dataset. In addition, other indicators as Mean Absolute Error (MAE) and Mean Squared Error (MSE) were used for monitoring. Finally, the training stopped when the validation set error started to increase, meaning that the model started to overfit.
To measure the goodness of the fitting, three metrics were used, Poisson loss between model predictions and unseen data (related with the log-likelihood of two variables under Poisson distribution assumption), Pearson's correlation coefficient between spiking probability functions and Pearson's coefficient between PTH (Peri-Stimulus Time Histogram) generated from 25 trials of the same stimulus (6 color flashes) for each of the 37 neurons on the network's output (Fig.~\ref{fig:correlations}).
The results showed high correlations. Fig.\ref{fig:PSTH} illustrates the PSTHs from both model and biological neuron responses measured within 10 milliseconds bins, showing a qualitatively coherent behavior.

![PSTH comparison](image)

*Figure 27. Peri-stimulus Time Histogram (PSTH) of both real (blue) and model (black) neuron 17.*

As the figure above this paragraph shows, the model responds with a different characteristic firing probability waveform to each stimuli, with a rising fire rate with a slightly different delay for each neuron when the light was showed and with a depression or rising of the spiking probability when the light faded out or after some milliseconds of of darkness. In brief, ON, OFF and ON/OFF behaviors were observed both in the real retinal recordings and the model predictions, and the CNN was able to model at the same time a variety of neuronal behaviors to the same stimulus, with different qualitative responses and different firing rate baselines. This kind of adaptation can be observed below, where a comparison between a biological neurons firing probability function and the CNN model’s prediction is shown, for two neurons with significant different behavior, having the maximum firing rate at the beginning and the ceasing of the flash correspondingly.

![Dynamic firing probability function](image)

*Figure 28. Dynamic firing probability function of a real neuron (blue) versus model neuron (black). The model is able to fit different types of neurons at once, with different base firing rate and characteristic shapes. On the figure, neurons 30 -left- and 1 -right*
The CNN was also very sensitive to the gaussian smoothing of the actual spike train, changing the shape of the fitting curve depending on the standard deviation used on the Gaussian, predicting poorly a mean fire rate when the standard deviation was too low and therefore, unable to compensate the neuron's variability on the response (data not shown).

Here we show the raster plots for both biological and model ganglion cell's responses to 10 repeated trials of 7 of the checkerboard patterns (neuron 2 in this case). Its noticeable that the CNN's neuron is predicting a qualitatively similar response than the real ganglion cell, with a concentrated firing activity after the light flashes and similar temporal dynamics.

Figure 29. Comparison between the biological retina spikes to 25 stimulus repetitions of checkerboard patterns and the Poisson-generated spikes from out model's predictions
Neural decoding
A brief introduction

Neural decoding is a subfield of neuroscience that attempts to establish ways to map neural activity (brain EEG signals, fMRI images, neurosensory recorded spike trains etc.) to the stimuli that elicited that response. The kind of techniques has been proved useful in decoding retinal ganglion cells responses in the past, both with traditional linear methods [27] and artificial neural networks [42].

![Diagram of neural decoding](image)

*Figure 30. Retinal neural decoding scheme as proposed in [42].*

Recently, machine learning algorithms have been proved very useful and able to outperform traditional ways of decoding – including Kalman filter-[13].

The goal that we will pursue on this chapter is to reconstruct the image that a retina is seeing from the spiking firing rate or directly from its spikes, and generate a reconstruction video. A similar attempt was done in our previous work [43], where the color that the retina was seeing was classified with classical machine learning algorithms.
For the decoding task, a vector with the firing rates is fed as an input to a series of dense layers and deconvolutions –transpose convolutions- or convolutions + upsampling until we get the predicted image.
Decoding retinal neural responses to moving bars

The neural network trained to decode the spike trains of a population of ganglion cells during the bars experiments consisted on a stack of two dense layers with a ReLu activation function and 10% dropout followed by three deconvolution/batch normalization stages with 3x3 kernel – filter- size and each of the two first convolutional layers consisting of 8 filter maps, when the last one has only one map –1 channel grayscale image-. The input of the model that gave lead to better results consisted on a sum of the response of each cell to each set of 10 repetitions of bars crossing on the same angle and direction (PSTH for each 10 repetitions of every stimulus).

![Diagram](image)

Figure 6. Illustration of the model used for decoding. A dense layer received spike firing rates of each population’s neuron and produced an image as an output

The results were accurate where the bars crossed specific spatial regions, presumably when they were inside a receptive field region, hypotheses that has still to be demonstrated still for this ganglion cell recordings.

![Images](image)

Figure 7. Up: Sample of frames accurately reconstructed. Down: samples were the neural network provides incomplete or ambiguous reconstructions.
Decoding retinal neural responses color to homogeneous color images

On the case of reconstructing 3 dimensional color images, the network architecture consisted in two dense layers followed by a reshaping layer that makes the previous activations suitable to enter a three stage deconvolutional layers with interspersed batch normalization. The main difference with the bars deconvolution will be in this case the output layer: one filter with three channels that has to be learned. As in the case of the reconstructing of moving bars, binary crossentropy between the pixels color level and the real ones was used as a loss function to minimize, with an ADAM optimizer, and several batch sizes were used, ranging from 10 to 30.

The model was trained over 600 data points with the average activity of each one of the 37 neurons responding to the light patterns in a 10 miliseconds bin (the other 600 data points were discarded to balance the dataset, since most of the training examples consisted into dark images).

\[
\begin{align*}
  &\text{Figure 8. Long Short Term Memory unit, which takes input not only from the previous layers but also from its past states.} \\
  &\text{An input, output and forgetting gate controls the flow of information, allowing the network to learn temporal dependencies over arbitrary time intervals CITA http://colah.github.io/posts/2015-08-Understanding-LSTMs/}
\end{align*}
\]

In order to achieve best results, a change in the model was done, replacing the first dense layers with Long Short Memory layers (recurrent neural networks capable of learning long-term dependencies). In this case, the input of the model consists not only of the firing rate of each neuron but, in addition, a 20 bin activation history (the 20 previous firing rate levels of the neuron), allowing the network to learn over the temporal dynamics of the biological neurons for a more accurate image reconstruction.
Figure 9. Example of the training set: 3 smoothed Peri Stimulus Time Histograms that were part of the input to the decoding model.

Here we show a sample of the results achieved:

Figure 10. Reconstruction of homogeneous color images with a Deconvolutional Neural Network. Adding Long Short Term Memory layers at the beginning improved the color obtained from the neural responses.
Since here we are reconstructing a series of frames during the time interval that it was shown to the retina and the ganglion cells have a highly dynamic response, the addition of the LSTM layers supposed a boost of the performance of the network, since it can learn not only instant firing rates but temporal patterns, and therefore offer a more time-consistent reconstruction.

**Figure 11. Image reconstruction on different time instants.**
Decoding retinal neural responses to natural images

On this last attempt to decode neuronal responses from 60 retinal ganglion cells, a set of grayscale natural images was shown to the retina, each image presented during 50 milliseconds, and its spike responses were sorted, neuron units identified and spike trains binned into 10 milliseconds intervals which were smoothed with a 10 milliseconds Gaussian. This means that for each image presentation we will have 5 frames.

On this case, each image over a total of 4890 was presented only once to the retina, so the input to the network won’t be the sum of the spikes over the repetitions, but single trials responses. The model that performed better had two LSTM layers with 20 timesteps history followed by a Dense and two deconvolution stages with ReLu activation function followed by a last deconvolution layer with Sigmoid activation function. Results are shown below.

As a first approach, we trained the network over one the responses to 5 temporal frames of the same image, with a high detailed result, proving this way that the network has enough computational complexity to map accurately arbitrary elaborated images.

![Original](image1) ![Reconstructed with LSTM - Deconvolutional NN](image2)

*Figure 12: Reconstruction of a tenshi from one single trial retinal ganglion cell responses using a LSTM-Deconvolutional Neural Network*
In a next training phase, the number of images to reconstruct was augmented to 50, 200 and 500, successively. As the number of training examples augment, the capacity of the network to reconstruct images accurately decays, fact that leads us to the following hypothesis: there is not a straight way to reconstruct directly an image from the activity of only 60 retinal ganglion cells, and our LSTM-Deconvolutional system is learning suboptimal internal representations from the dynamic neural spiking activity in the first stages of the system, and a mapping between that internal representations and the images to reconstruct, in an overfitting way. This is possible because the high number of parameters and the relative complexity of the network (that performs out spatiotemporal computations) and, in essence, the capacity of neural networks to be Universal Approximators [44]. We propose to demonstrate this in future works and investigate a direct way to reconstruct images developing models able to reconstruct only the receptive field of every single neuron, possibly with another deep neural network that integrates all that information in a posterior stage.
Discussion and future work

In this work, a data-driven methodology for modeling retinal ganglion cell responses to different kind of light patterns has been applied to a biological mouse retina with positive results, showing a high correlation with the spiking firing rates obtained from the biological spike train responses. Two different approaches were successfully tested: fitting single trial responses and PSTH fitting, having the second one a better spiking simulation results. In addition, the Peri-Stimulus Time Histogram built from Poisson-simulated spike trains showed a similar behavior in both the model and the real retina. On the way to achieve these results, several structural and parametric decisions were taken for the model, resulting in a 3D CNN model that showed high sensitivity to the activity and parametric regularizers on the dense layers on one side, and to the variance of the gaussian with which the spike trains were filtered on the other hand. These facts revealed that the retinal spiking variability handling will play an important role in future developments of the model.

The model deployed here implies a new tool to step further on not only the neural encoding of spatio-temporal light patterns but also in the effect of luminance and color coding of the retina, given that is able to mimic the different characteristic response shapes, and we will explore this feature in depth in future works. For this, new experiments will be set up in a way that our model is able to mimic a wider and representative range of retinal behaviors and reveal new insights on it.

On the other hand, deep learning systems for neural deconding have been deployed, showing the feasibility of neural networks to reconstruct grayscale images and color patterns, providing better results with Long Short Term Memory units. This opens the door to a more extensive research on this area, that will be tackled on the future.

Among our proposals for future works on the modeling part are the use of recurrent layers that take into account the temporal states of each neuron, the change in the spike generation to more advanced and flexible models like inhomogeneous Gamma and inverse Gaussian proposed in [45], the visualization of the inner activations and learned filters as done in[7] that may help in the understanding of retinal computations, building more computationally optimized networks as done in [46], efficient image reconstruction from both real and simulated retinal ganglion cells and, finally, the use of highly-complex and realistic grayscale and color visual stimuli that allows us to build a powerful and more generalizing retinal model, with the hope that it means a step further in the development of realistic a bio-inspired retinal suitable for biomedical applications such as disease simulation and prosthesis development. In the decoding case, applying a reconstruction system to both biological and artificial retinas could be a way to compare retina models and give insight on how much information are this models able to convey.
Bibliography


[27] A. Garc, L. Humphreys, and E. Fern, “Artificial Computation in Biology and Medicine. IWINAC 2015, Towards the reconstruction of Moving Images by Populations of Retinal


[43] “Estudio estadístico y modelización mediante técnicas de aprendizaje máquina de registros de células ganglionares retinianas en ratones.”


[47] “Bionic Eye, General Electrics, created by Dr. Joseph Rizzo and Prof. John Wyatt.”