INDIAN STATISTICAL INSTITUTE, KOLKATA

"An Attempt to Design a Neural Network Exploiting Biological Neurons"

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
FOR THE AWARD OF THE DEGREE

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE

BY

Neeraj Kumar Singh (CS1931)

UNDER THE GUIDANCE OF
Nikhil R. Pal
Professor
Electronics and Communication Sciences Unit (ECSU)
Computer and Communication Sciences Division
Kolkata

CERTIFICATE

This is to certify that the dissertation entitled **"An Attempt to Design a Neural Network Exploiting Biological Neurons"** submitted by **Neeraj Kumar Singh** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

15/07/21

Nikhil R. Pal
Professor,
Electronics and Communication Sciences Unit (ECSU), Kolkata,
Indian Statistical Institute,
Kolkata-700108, India.

Acknowledgements

I would like to show my highest gratitude to my advisor, *Prof. Nikhil R. Pal*, Electronics and Communication Sciences Unit (ECSU), Kolkata, for his guidance and continuous support and encouragement. He has literally taught me how to do good research, and motivated me with great insights and innovative ideas.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Neeraj Kumar Singh
Indian Statistical Institute
Kolkata - 700108 , India.

ABSTRACT

A multilayer perceptron network is a very effective tool for both classification and regression type problems, which has been successfully used in many areas. In this era of artificial intelligence, deep neural networks such as Convolutional Neural Networks (CNNs) have been found to be extremely successful in solving many difficult problems, often defeating human performance. Often deep networks are viewed as "all-cure" solutions. Unfortunately, most of these networks are generally of "black-box" nature and their functioning usually is not related to the way biological neural network works. Some of these networks have millions of free parameters! Moreover, training deep networks often demands a huge volume of training data.

In this study we intend to incorporate knowledge of biological neurons in some of the layers of convolutional neural networks. In particular, we study the computational models of some of the cells like Lateral Geniculate Nucleus (LGN) cells and Retinal Ganglion Cells and make use of such models to extract features from images to be used as input with the intention that if such features help in improving performance, such computational models will be built into the deep neural network. We also hope that this will enable us to reduce the size of the network because instead of blindly extracting features, it will try to mimic, to some extent, the way the brain extracts features. In this context, first we use the Combination of Receptive Fields (CORF) model. But our experiments do not exhibit the expected results. Then we propose another CNN model that uses the Difference of Gaussians (DoG) filters in some of the layers of the network because the CORF model makes all computations on DoG. This has resulted in noticeable improvement in performance with fewer trainable parameters than the ResNet-18 (we use ResNet-18 as the base network due to limited computing resources).

Contents

List of Figures

Chapter 1

Introduction

1.0.1 Deep Learning

Artificial neural networks (ANNs) were inspired by the biological brains. An ANN may have a large collection of neurons. These neurons are analogous to the neurons in the brain. The connections between neurons transmit the learnt patterns/information to other neurons. Similarly, the receiving neuron processes/integrates these learnt patterns into more complex patterns. Each connection is associated with a weight indicating the strength of the connection. If a neuron, for example, has p input connections then the weights of all p connections will determine how much importance the inputs, received by the p neurons, should get. We note that a connection weight could be excitatory or inhibitory.

Deep learning is a family of machine learning algorithms. It is a subset of machine learning and it is primarily based on the representation learning capability of the ANNs. The word deep means more than one hidden layers in the neural network. The Perceptron model was limited to linear classification problem. It was unable to handle complex decision boundaries, nor even the exclusive-OR type data. Later it was found that this issue can be addressed by an artificial neural network with one hidden layer of neurons with sigmoidal activation functions. Convolutional neural networks can model complex decision boundaries. They are typically feed forward neural networks without any feedback loop, in which data only flow in the forward direction.

In deep learning, each layer learns some patterns and transfers it to next layer. Deep learning based architectures can be constructed using a layer by layer training [7] as well as using end-to-end training [8]. There are many deep learning frameworks that provides built-in functions, methods, and classes to implement the architecture.

Some of the factors that possibly contributed to the huge interest in deep learning are:

- Availability of large corpus of labelled data for training and evaluating the deep learning models: With increase in the amount of data, typically the performance of deep learning models becomes more accurate. On the other hand, usually for the traditional networks, the performance does not continue to improve with the size of the dataset after it reaches a limit.

- Availability of Parallel computing using GPUs: It enabled us the transition from CPU-based training to GPU-based training which significantly accelerated the training process of the deep learning models. Nowadays, GPU has become the integral part of training any deep learning model.

- Representation ability of deep motels: Deep models are found to have excellent ability to extract salient features from the data, which enabled us to solve very large scale decision making problems that were impossible to address using the traditional learning systems.

1.0.2 Some Issues with Deep Learning

While the deep learning algorithms are found to outperform other learning systems on large corpus of data, there are some issues with it. Deep neural networks are very powerful in many areas such as image classification, image captioning, image recognition, and natural language processing. The performance of deep networks are very good for the image recognition task but the semantics of the feature maps that are learnt by the hidden layers are often unknown to us. This lack of interpretability of the network, often limits its use particularly in critical areas like healthcare and defence.

Explainability is another major issue of deep neural networks i.e., it failure to explain the results. We note that this is also a problem with MLP, but when the system has millions of free parameters, then the problem becomes more severe. Let us consider an example where we use a deep learning system to classify the images of dogs and cats into their respective classes. It may be the case that the performance of this system is excellent (almost equals to human performance) but it fails to reveal why it has led to good performance. We can find out location of nodes/neurons in each layer which are activated by a given image, but we are unable to figure out what these activated neurons are supposed to model/represent, i.e., how they are extracting the feature map in each layer, and what these neurons are doing collectively, and how the extracted feature maps are combined layer-after-layer to compute the final feature map. Thus, the network behaves like a black-box. Yet, DNNs (deep neural networks) are often viewed as "all-cure" solutions. Usually, the functioning of these deep networks are not related to the way biological neural networks function.

Another major issue is that most of these networks cannot say "Don't know" when given an input from a class that it was not trained on. This makes such networks untrustworthy as the network assumes a closed-world scenario while in realty for almost all problems, the situation is the "open world". Moreover, training such deep neural networks demands huge training data as well as huge computing power.

So a few question arise: Can we reduce the number of parameters of the network exploiting some knowledge of the brain? Can it also improve the performance compared to deep networks with similar architecture? We note that affirmative answers to these questions may not change the black-box nature of the CNN. But the reduction of number of free parameters and the exploitation of the computational models of information processing in the brain may be small step ahead in the right direction.

In this study, we intend to incorporate functioning of biological neurons in some of the layers of the artificial neural networks. In particular, we studied the CORF (combination of receptive fields) computational model of the Lateral Geniculate Nucleus (LGN) cells and Retinal Ganglion Cells [1] and made use of such models to extract features from images to be used as input to a CNN. We hope that this will enable us to reduce the size of the network because instead of blindly extracting features, it will use features extracted in a way that mimic the way the brain extracts features (at least to some extent). If this helps, then we can incorporate the brain inspired feature extraction models using a few layers of a CNN.

Chapter 2

Combination of Receptive Fields (CORF) Model

2.0.1 LGN Cell

We used model LGN (lateral geniculate nucleus) to implement the CORF model [1]. CORF stands for the "combination of receptive fields". In ganglion cells, there are two types of receptive fields :

1. On-center receptive field
2. Off-center receptive field

The CORF model is based on the ganglion cells of the human vision system. Whenever there is a contrast change, these ganglion cells generate responses. Contrast stimuli are the most fundamental components in the images (visual scenes). Our retina is composed of concentric RFs (receptive fields). The firing activity in the ganglion cells are controlled by the these receptive fields. If the generated response in a receptive field is greater than the action potential, then only the information will be passed to the brain.

On-center receptive fields respond strongly to light hitting on their center. Their responses start decreasing when we go towards the surroundings of the receptive field, while Off-center receptive fields respond strongly to light hitting on their surroundings. Their responses start decreasing when we go towards the center of the receptive field.

Therefore, sensitivities of the ganglion cells is modeled by using the difference of Gaussian (DoG) functions with the greatest sensitivity at the center. This sensitivity starts decreasing when we move in the surroundings (move away from the center). In our case, we use a 2-D Gaussian function to model this sensitivity. The CORF model considers model LGN cell in which half of the region is black and remaining half region is white. If we consider these black and white regions on the left and right sides from the center of the LGN cell, then such LGN cell will able to detect the contrast change in the vertical direction. So, we can detect this contrast change in all directions of a 2-D plane, i.e., from 0° to 360°. To get the combined response, all such responses obtained in different directions are combined. This is the basic idea of the CORF model.

In the CORF model [1], white disks represent the center-on LGN cells and black disks represent the center-off LGN cells. The center-on cell is represented by "+" and center-off cell by "-". This model LGN cell detects the contrast changes. The response of a sub-unit is calculated as the weighted sum of the responses of its inputs. The polarity (whether a cell is off or on), receptive field size, and the neighbouring receptive fields of these model LGN cells are the same. The receptive field of a sub-unit is the union of the involved model cells. Polarity of these sub-units will be the same as the polarity of the model LGN cell.

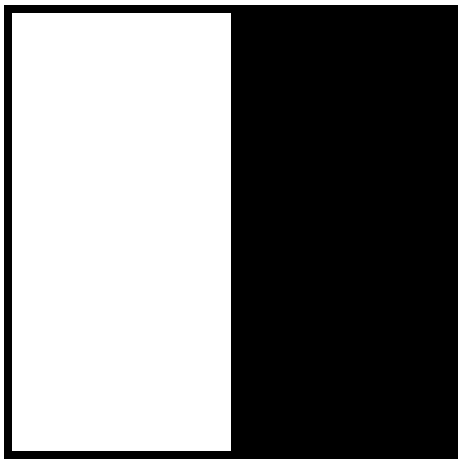

Figure 2.1: Synthetic stimulus image

2.0.2 Center-surround receptive field

To model an LGN cell, usually the difference of 2-D gaussian functions is used and that is what we do here.

$$DOG_{\sigma}^{+} = \frac{1}{2\pi\sigma^2} \exp \frac{-(x^2+y^2)}{2\sigma^2} - \frac{1}{2\pi(0.5\sigma)^2} \exp \frac{-(x^2+y^2)}{2(0.5\sigma)^2} \tag{2.1}$$

Here, standard deviations of the inner and outer Gaussian functions are taken as $\sigma/2$ and σ respectively. The standard deviation of inner-Gaussian function is set according to the electrophysiological findings of the LGN cells in mammals [5].

For center-on receptive field, its central region is excitatory while surrounding is inhibitory. It is represented by $DOG_{\sigma}^{+}(x, y)$. Similarly, for the center-off receptive fields, its central region is excitatory while surrounding is inhibitory. It is represented by $-DOG_{\sigma}^{+}(x, y)$, opposite to that of center-on receptive field.

In CORF model, the size of the stimulus is determined by the value of σ. Let us consider a square stimulus. For a given σ value, we will consider a set of circles. All circles must be within this stimulus. So, we have considered the side of the square as $2 \times (max\_radius + 1)$. In our case, we have chosen the value of σ as 2.5.

For this choice, we consider 4 circles of radii (3, 6, 13, 25). So, the maximum radius is 25 and each side of the squared stimulus is $2 \times (25 + 1) = 52$. Consequently, our stimulus is of size 52×52 pixels as shown in Fig. 2.1. All the sub-units location will be computed in the union image of DOG-plus and DOG-minus responses of this synthetic stimulus.

$$DOG_{\sigma}^{-}(x, y) = -DOG_{\sigma}^{+}(x, y) \tag{2.2}$$

The response of an LGN (Lateral Geniculate Nucleus) cell is calculated by the spatial summation of the intensity distribution in the input image which is weighted by the DOG response.

$$C_{\sigma}^{\delta}(x, y) = |I * DoG_{\sigma}^{\delta}|^{+} \tag{2.3}$$

Here, δ represents the polarity of the receptive field, $*$ denotes convolution operation and $|.|^{+}$ denotes half-wave rectification. In equation 2.3, we calculate the response of a receptive field which is centered at image coordinates (x, y) by the linear spatial summation of the intensity distribution $I(u, v)$, weighted with $DOG(x - u, y - v)$.

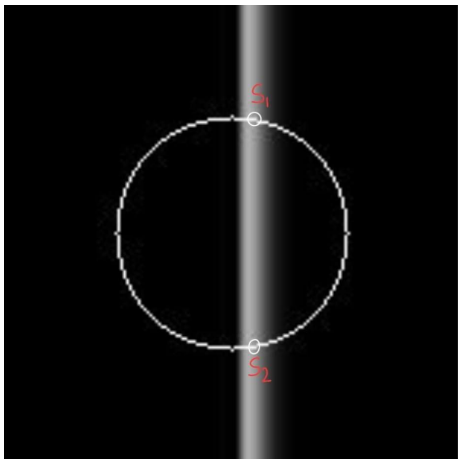

Figure 2.2: DOG plus response image of synthetic stimulus

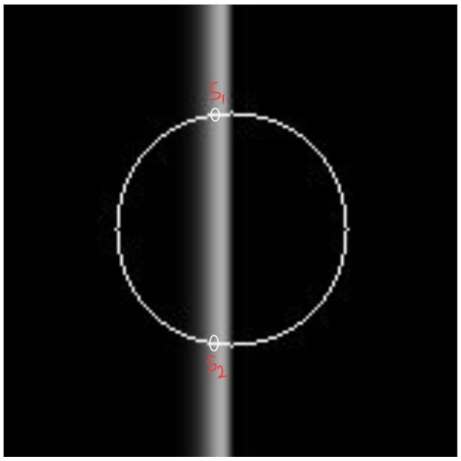

Figure 2.3: DOG minus response image of synthetic stimulus

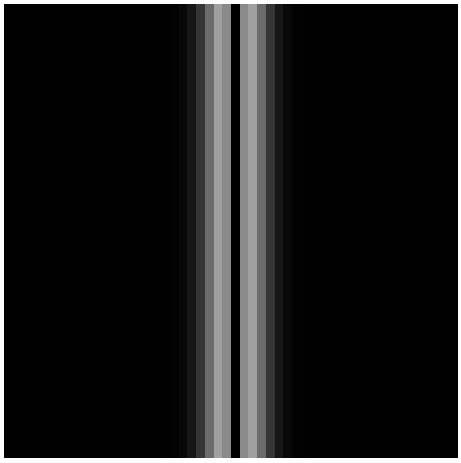

Figure 2.4: Union of DOG-plus and DOG-minus responses of synthetic stimulus

Fig. 2.2 and Fig. 2.3 show the response image produced by the DOG-plus and DOG-minus operators respectively, while Fig. 2.4 depicts the image obtained by taking the union of the DOG-plus and DOG-minus response images.

Let us now consider another input image shown in Fig. 2.5. For this input image, we have calculated the responses images produced by DOG-plus and DOG-minus operator as shown in Fig. 2.6 and Fig 2.7, respectively.

We apply CORF operator on the image obtained by union of DOG-plus response and DOG-minus response as shown in Fig. 2.8. Union image is obtained by calculating the pixel-wise maximum value in Fig 2.6 and Fig 2.7.

2.0.3 Sub-units and their parameters

In the CORF model cell [1] (Fig. 2.1), the center is located at the diagonals-bisector of the stimulus. In our case, it is located at position $(26, 26)$ on the stimulus image. Next, we calculate the maxima positions on each circle in both the images obtained by DOG-plus operator and DOG-minus operator.

For the given stimulus (Fig 2.1), union image of DOG-plus response and DOG minus

Figure 2.5: Input image

Figure 2.6: DOG plus response image of the input image

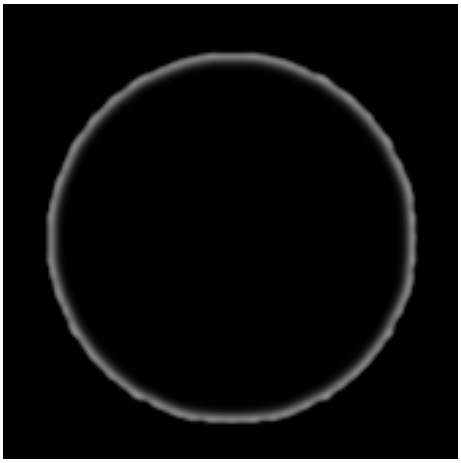

Figure 2.7: DOG minus response image of the input image

response will output four sub-unit on a circle. We get 2 sub-units for the image obtained by the DOG-plus operator (Fig. 2.2). Similarly, we get 2 sub-units for the image obtained by the DOG-minus operator (Fig. 2.3). So, for one circular path, we will get 4 sub-units location. We can also visualize it in Fig 2.2 and Fig 2.3. These are marked as S1 and S2 respectively. If we consider any circular path in the image, we will get high response in the bright response region. For any circular path, we get two such positions of local-maximum.

So, if we consider n concentric circles in the model LGN cell, there will be 4n sub-units locations, out of which 2n sub-units will be obtained by the response image of DOG-plus operator (Fig. 2.2) and the rest of 2n sub-units will be obtained by response image of DOG-minus operator (Fig. 2.3).

How many concentric circles should we consider?

A natural question comes: how many concentric circles should we consider? As we know, we are considering these sub-units in model LGN cells. In [1], authors have experimentally determined this value based on the value of σ. There is no evidence of how they have chosen these values.

For different values of σ, we considered concentric circles of different radius:

Figure 2.8: Union of DOG-plus and DOG-minus responses of input image

```
In [11]: operator_tuple

Out[11]: array([[ 0.   , 2.5  , 3.   , 0.8203],
                [ 0.   , 2.5  , 3.   , 5.4629],
                [ 1.   , 2.5  , 3.   , 2.3213],
                [ 1.   , 2.5  , 3.   , 3.9619],
                [ 0.   , 2.5  , 6.   , 1.2305],
                [ 0.   , 2.5  , 6.   , 5.0527],
                [ 1.   , 2.5  , 6.   , 1.9111],
                [ 1.   , 2.5  , 6.   , 4.372 ],
                [ 0.   , 2.5  , 13.  , 1.4137],
                [ 0.   , 2.5  , 13.  , 4.8695],
                [ 1.   , 2.5  , 13.  , 1.7279],
                [ 1.   , 2.5  , 13.  , 4.5553],
                [ 0.   , 2.5  , 25.  , 1.4923],
                [ 0.   , 2.5  , 25.  , 4.7909],
                [ 1.   , 2.5  , 25.  , 1.6493],
                [ 1.   , 2.5  , 25.  , 4.6338]])
```

Figure 2.9: 4-tuple value of 16 sub-units

- For $\sigma \in \{1, 1.5, 2\}$, we consider three concentric circles of radii $\rho \in \{3, 7, 14\}$.
- for $\sigma \in \{2.5, 3, 3.5\}$, we consider four radii $\rho \in \{3, 6, 13, 25\}$
- for $\sigma \in \{4, 4.5, 5\}$, we will use five radii $\rho \in \{3, 5, 9, 18, 34\}$

The intuitive idea behind such choices is that for higher the values of σ we consider more and wider circles because higher σ will give stronger responses over a wider area.

Representation of a Sub-unit

As suggested in [1], each sub-unit included in the LGN model is represented by a 4-tuples $(\delta, \sigma, \rho, \phi)$. Here, δ represents the polarity of the sub-unit, σ involved in the building of model LGN cells, ρ represents the radius and ϕ represents the polar angle of the center of the sub-unit relative to the center of the CORF model cell. So, if there are n such sub-units, then the set S of 4-tuples of sub-unit locations is: $S = \{(\delta_i, \sigma_i, \rho_i, \phi_i) | i = 1...n\}$. The set S represents all sub-units of the model LGN cell.

In our experiment, we have considered the value of σ equals to 2.5. So, for this σ value, we get four concentric radii as discussed above. Now, for these four radii, we get 16 sub-units as shown in Fig. 2.9.

2.0.4 Calculation of Sub-Unit Responses

The response of a sub-unit is calculated by the linear spatial summation of the calculated half-wave rectified responses of the model LGN cells with respect to the receptive field center of the CORF model cell using polarity δ, scaling σ and around position (ρ, ϕ) which is weighted by a two-dimensional Gaussian function G_σ.

$$s_{\delta\sigma\rho,\phi}(x,y) = \sum_{x'} \sum_{y'} c_\sigma^\delta(x - \Delta x - x', y - \Delta y - y') G_{\sigma'}(x', y') \tag{2.4}$$

Here, $\Delta x = -\rho\cos\phi$ and $\Delta y = -\rho\sin\phi$. The values taken by x' and y' lie inclusively between $-3\sigma'$ to $3\sigma'$. The standard deviation σ' is a linear function of the ρ which is determined by the relationship between the average receptive field diameter of the LGN (lateral geniculate nucleus) cell and the eccentricity.

In equation (2.4), the response $c_\sigma^\delta(x,y)$ is shifted by Δx and Δy. We determine this shift vector based on ρ and ϕ at location (x', y'). The value of σ' is determined by d_0, α and ρ which is computed as $\sigma' = (d_0 + \alpha\rho)/6$ where d_θ and α are constants. In [1], value of d_0 and α are suggested as 2 and 0.9, respectively .

2.0.5 Combinining Sub-Unit responses

The response of CORF model cell is calculated as the weighted geometric mean of all the sub-unit responses that belong to the specific selection determined by the set S [1]. The weight is inversely proportional to the distance of sub-unit from the CORF model cell. Thus, it will be more if the sub-unit is located close to the centre of CORF model and it will decline as the location of the subunit go away from the centre of CORF model cell.

Here, the weights are determined using an exponential function of ρ and σ' as: $w_i = \exp\{\frac{-\rho_i^2}{2\sigma'^2}\}$. The σ' is chosen as 1/3 of the maximum radius, i.e., $\sigma' = \frac{1}{3}\max_{i\in\{1....|S|\}}\{\rho_i\}$.

The response of the CORF model cell computed as:

$$r_S(x,y) = \prod (s_{\rho,\phi}(x,y))^w)^{(\frac{1}{\sum_{i=1}^{|S|} w_i})} \tag{2.5}$$

The response of a CORF model cell is maximum along the orientation for which it is configured and it declines with the deviation of the orientation of the input stimulus from the optimal one. The response becomes almost zero for a deviation greater than $\pi/4$ radian.

2.0.6 Calculating responses along different orientations

For a CORF model cell, its orientation preference depends on the orientation of the edge of the configured stimulus. For getting orientation preference in different directions, we have two options:

1. Present different edges to create models with different orientation preferences
2. Create a single model and manipulate its parameters to get another model as $\Re_\psi(S) = \{(\delta, \sigma, \rho, \phi + \psi) | \forall(\delta, \sigma, \rho, \phi) \in S\}$

In our case, we used the second option. We have considered 12 such orientation at an interval of 30° beginning with 0°. For each orientation, we have calculated the output response as shown in Fig.2.10-Fig.2.21.

Figure 2.10: Output response at orientation 0°

Figure 2.11: Output response at orientation 30°

Figure 2.12: Output response at orientation 60°

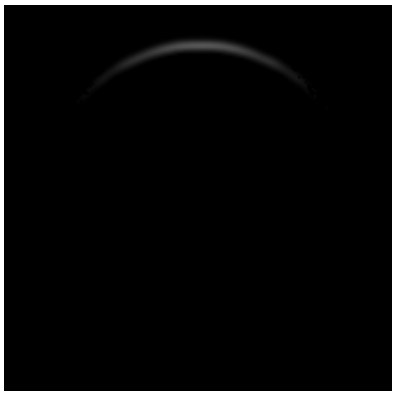

Figure 2.13: Output response at orientation 90°

Figure 2.14: Output response at orientation 120°

Figure 2.15: Output response at orientation 150°

Figure 2.16: Output response at orientation 180°

Figure 2.17: Output response at orientation 210°

Figure 2.18: Output response at orientation 240°

We merge the responses of the CORF models with these 12 orientations using the maximum superposition to calculate the final response as shown in Fig. 2.22.

Figure 2.19: Output response at orientation 270°

Figure 2.20: Output response at orientation 300°

Figure 2.21: Output response at orientation 330°

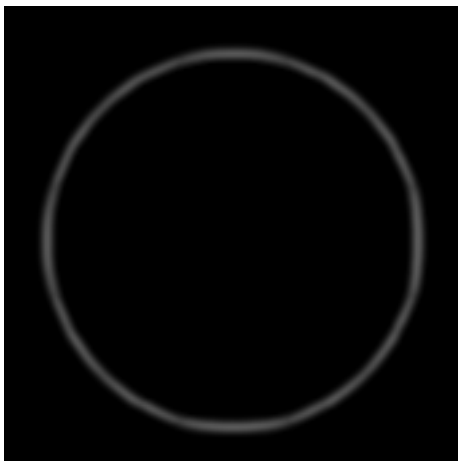

Figure 2.22: Image obtained by maximum superposition of images (Fig. 2.10 to Fig. 2.21)

To get the binary contour of the input image, thinning and hysteresis thresholding are applied on the calculated final response image. In hysteresis thresholding, we have used two parameters: low threshold and high threshold. We have set the value of high threshold equals to 0.3 times of the maximum pixel value of thinned output. The value of low threshold is 0.5 times of the high threshold. So, our final output image of CORF model is as shown in Fig. 2.23 for the input image (Fig. 2.5).

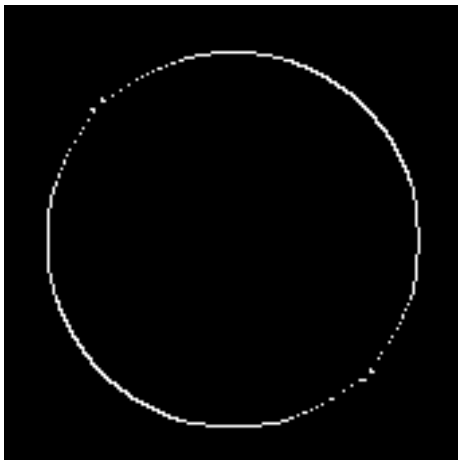

Figure 2.23: Output image after Thinning and Thresholding

Chapter 3

Implementation and Experiments

3.1 Implementation

3.1.1 Dataset

We have used the CIFAR-10 dataset[6]. This dataset consists of 60,000 RGB images of size 32x32. So, the size of each input images is $32 \times 32 \times 3$ which is changed to $32 \times 32 \times 1$ when we processed these images with the CORF computational model.

The images belong to objects of 10 classes such as frogs, horses, ships, and trucks. This dataset is divided into training images and testing images. The training set contains 50,000 images while the testing set contains 10,000 images. Among the training images, we used 45,000 images for training purpose and 5000 images for validating the model. Last 5000 images from the 50,000 training data is chosen for the validation.

3.1.2 Feature Extraction using CORF model

Relevant features are extracted using the CORF model [1]. Following are the steps used in CORF model:

1. First, we created a stimulus and calculated the DOG response of it for ON-center cells and OFF-center cells
2. Depending on the value of σ, we considered few concentric circles around the LGN cells and calculated the position of the maximum response.
3. For each position of the maximum response, we calculated the CORF operator, a four-tuple (polarity, σ, ρ, ϕ).
4. For the given input image, we calculated its DOG-response and applied the CORF operator on it in twelve equidistant orientations.
5. Merged the output of all orientations by using the maximum superposition
6. Thinning and hysteresis thresholding are performed on the output image to get the binarized contour

As mentioned eralier, the CIFAR-10 dataset contains images of the 10 classes (airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships and trucks) where each class contains 5000 training data and 1000 test data.

To increase the diversity in the training dataset, we perform data-augmentation on it. For this, we have used only three options:

- horizontal_flip
- width_shift_range
- height_shift_range

3.2 Network Architectures

We implemented all networks using the following packages: Tensorflow 2.5.x; Keras 2.1.5; Pickle (Python's built-in package); NumPy 1.14.2; and Matplolib 2.2.2

We have trained three different CNN architectures:

- CORF based CNN
- DOG based CNN
- ResNet-18

CORF based CNN Model:
In this architecture, we use the CORF model as a feature extractor for the first layer. In other words, this network receives the CORF output as the input in the first layer- at present the CORF computation is not built into the first layer, although that is our ultimate intention. After the first layer, we have stacked twelve convolution layers with small 3x3 filters. The number of filters from the second layer to the thirteenth layer are 32, 32, 64, 64, 128, 128, 256, 256, 512, 512, 1024, and 2048. We have used the same padding in each convolution layer except the first one to ensure that the size of the output feature map remains the same as the input feature map. After each convolution layer, we have used batch-normalization [4] and activation function. We have used ELU (exponential linear unit) activation function in all convolution layers. We initialized the weights using He-weight initialization. We performed max-pooling using stride equals to 2 in both the direction after the third, fifth, seventh, ninth, eleventh, twelfth, and thirteenth layer which is shown as **"/2"** in the architecture (Fig. 3.1).

After the operation of max-pooling in the last convolution layer, we flattened it. There are 2048 neurons in the flattened layer. We used a hidden layer with 96 neurons using ELU activation function and He-normal kernel initializer. We added a dropout equals to 0.5 in this hidden layer to prevent over-fitting. After the hidden layer, we used the output layer to classify the input into one of the ten possible classes. For this, we used softmax activation function with He-normal kernel initializer. Fig. 3.2 and Fig. 3.3 show the model's training and validation accuracies. Due to limited computational power, we have considered ResNet-18 as our baseline network. The proposed architecture is designed keeping in view that it uses less number of layers than ResNet-18, has less free parameters, and enjoys a simpler structure. We have used the Keras library for building CNN models.

However, as we shall see later the performance of the CORF-based CNN is not as good as we expected. One of the possible reasons may be that our implementation of the CORF model failed to produce outputs similar to what are reported in the CORF paper [1]. This could be due to some issues in understanding the implementation protocol followed in the CORF paper [1] because of the non-availability of details or a problem in our codes.

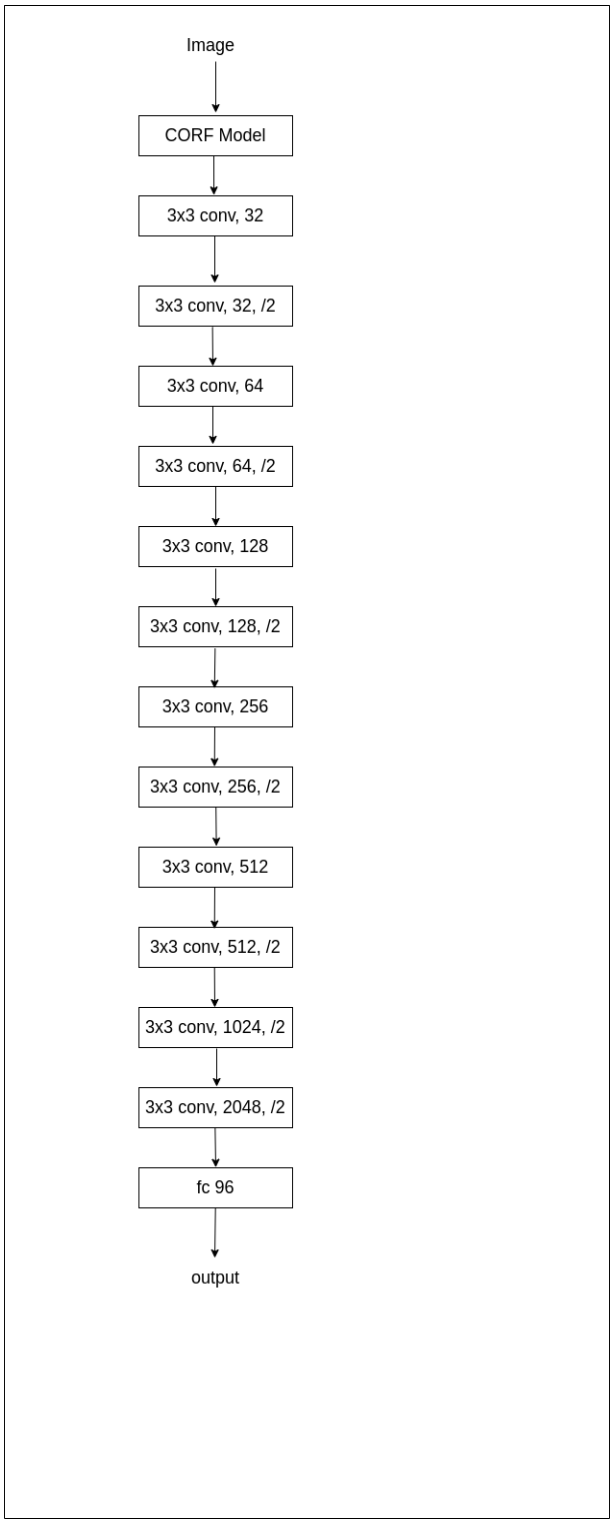

Figure 3.1: Architecture of CORF based CNN

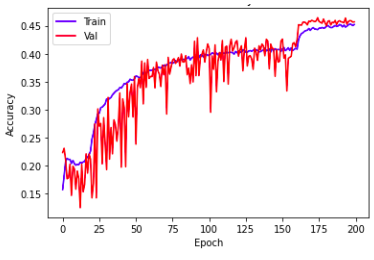

Figure 3.2: CORF based CNN accuracy curve

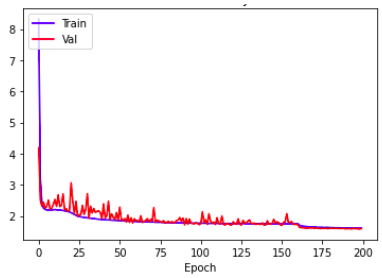

Figure 3.3: CORF based CNN loss curve

Since, the CORF model does all computations on the DoG outputs, so we propose another network, DoG based simple CNN, as described next.

DOG based CNN Model:
The proposed network has twelve two-dimensional layers as shown in Figure 3.4. In the first layer, we perform convolution using 25 DOG (difference of Gaussians) filters on the gray-scale images of CIFAR-10 using different σ-values ranging from 1.5 to 4 and stacked the result along the last axis.

After the first layer, we have stacked the ten convolution layers with small 3x3 filters. The number of filters from the second layer to the eleventh layer are 64, 64, 128, 128, 256, 256, 512, 512, 1024, and 2048. We have used the same padding in each convolution layer except the first one to ensure that the size of output feature map remain the same as the input feature map. After each convolution layer, we have used batch-normalization [4] and activation function. We have used ELU (exponential linear unit) activation function in all convolution layers. We initialized the weights using He-weight initialization. We performed max-pooling using stride equals to 2 in both the direction after third, fifth, seventh, ninth, tenth and eleventh layer which is shown as **"/2"** in the architecture (Fig. 3.4).

After the operation of max-pooling in the last convolution layer, we have flattened it. There are 2048 neurons in the flattened layer. We have used a hidden layer with 96 neurons using ELU activation function and He-normal kernel initializer. Here also we have used dropout equals 0.5 in this hidden layer to prevent over-fitting. After the hidden layer, we have used the output layer to classify the input into one of the ten possible classes. For this, we used the softmax activation function.

To train the network, we have used Adam optimizer with an initial learning rate equals 0.001. We have used a learning rate scheduler which helps in optimizing the network. Learning rate will decrease by a factor of $\frac{1}{10}th$ after 160, 250, and 300 epochs. These numbers are determined by a trial and error method, which is, of course, not the best way to do it.

In callbacks, we can pass instances of learning rate scheduler and early-stopping. At the end of each epoch, The validation score is calculated. We note that, although we have used a validation set, we did not use it to decide early stopping or so.

Fig. 3.5 and Fig. 3.6 show the model's training and validation accuracies. Up to 300 epochs, both errors are very consistent. This is the reason the validation performance has not played any role in monitoring the training. After the training, the model is evaluated on the test data to obtain the test accuracy.

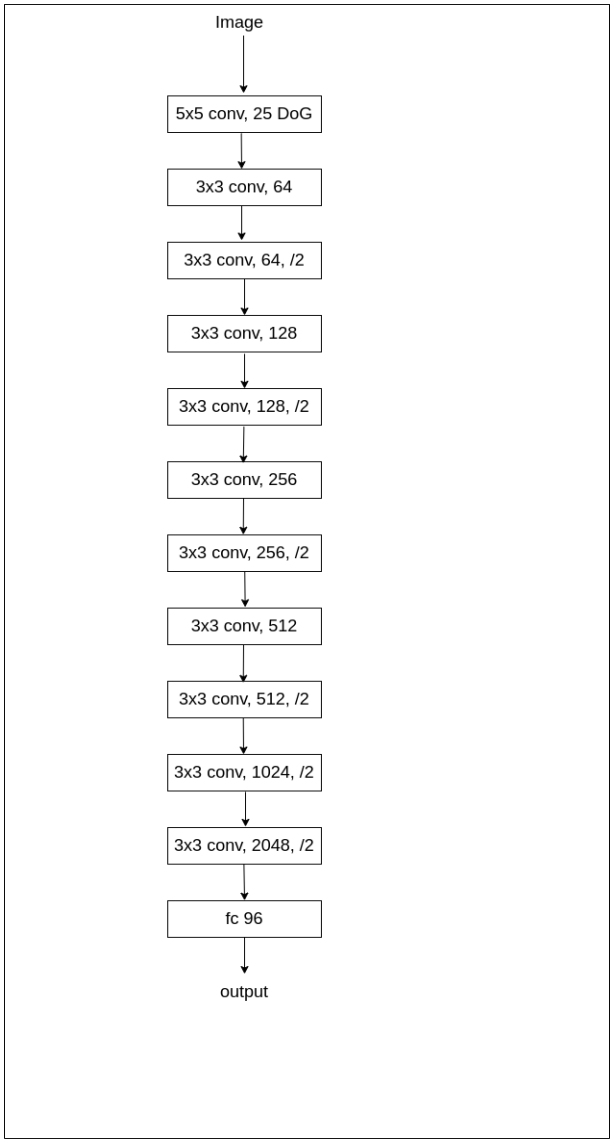

Figure 3.4: Architecture of DOG based CNN

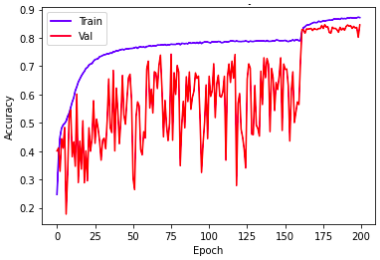

Figure 3.5: DOG based CNN accuracy curve

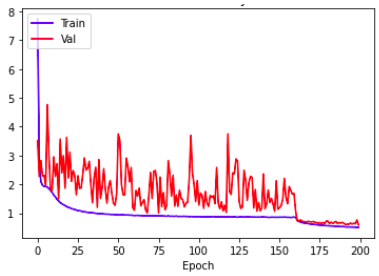

Figure 3.6: DOG based CNN loss curve

ResNet-18 Model:
The ResNet-18 [3] architecture is depicted in Fig. 3.7. This network has 11 million free parameters, while our proposed CORF-based network has 9 million free parameters.

The architecture uses eighteen layers with a different number of convolution filters. In the end, the model uses the operation of two-dimensional global average pooling. Max-pooling operation is shown as **"/2"** in the architecture (Fig. 3.7). We flattened this output feature map and performed the classification task in the output layer using the softmax activation function. Fig. 3.8 and Fig. 3.9 show the model's training and validation accuracies.

3.3 Comparison of Networks Output

Table 3.1: Accuracy of the CNN Models

Classification Model	Accuracy
CORF based CNN	45.27%
DOG based CNN	83.32%
ResNet-18	77.17%

The accuracy of all the tested models is presented in Table 3.1. Here, we are comparing the output of DOG based CNN, CORF based CNN, and ResNet-18. The DOG based CNN has been run five times and the average accuracy is reported in Table 3.1. The result indicates that the accuracy of DOG based CNN is about $\approx 6\%$ more than the accuracy of ResNet-18.

As we mentioned earlier, the Accuracy of CORF based CNN is quite low. We think this is because of some implementation issues of the CORF model. The output of our implementation of the CORF model on the RuG dataset [2] is not of the same quality as that given by the authors in [1]. We can see the differences in outputs between the two implementations in Fig. 3.10 -Fig. 3.15. Fig. 3.10 is the Bear_3 image. The output reported in [1] is shown in Fig. 3.11 and the output generated by our implementation is shown in Fig. 3.12. Similarly, for the Bear2_Thumb image in Fig. 3.13, the outputs generated by the two implementations are given in Fig. 3.14 and Fig. 3.15, respectively.

The comparison of the degrees of freedom (number of trainable parameters) among the three networks is presented in Table 3.2.

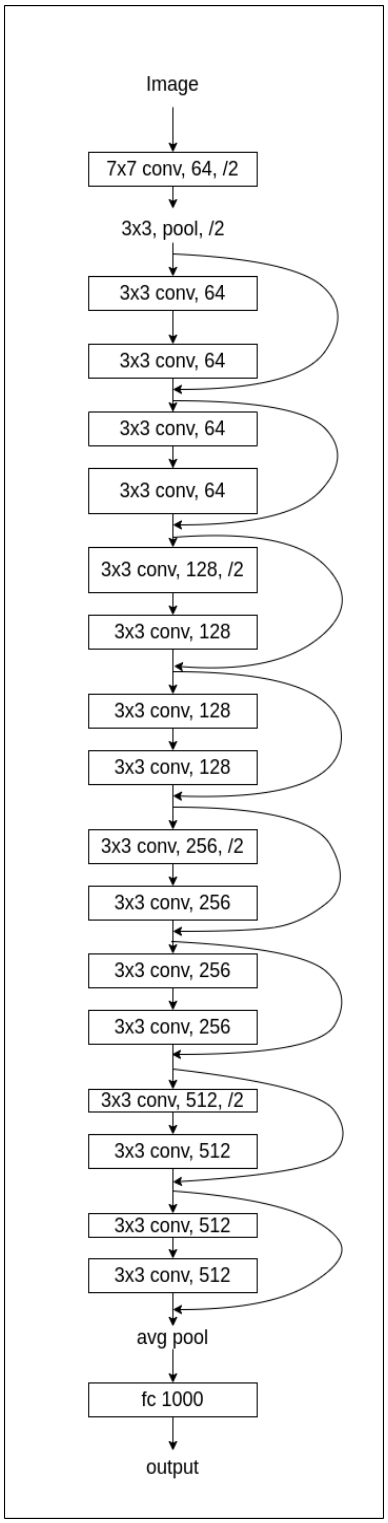

Figure 3.7: Architecture of ResNet-18

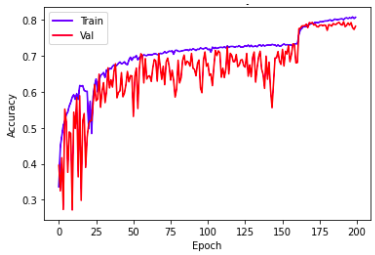

Figure 3.8: ResNet-18 accuracy curve

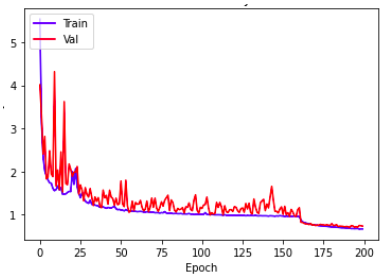

Figure 3.9: ResNet-18 loss curve

Figure 3.10: Bear_3

Figure 3.11: Desired CORF output of Bear_3 [1]

Figure 3.12: CORF output of Bear_3 by my code

Figure 3.13: Bear2_Thumb

Figure 3.14: Desired CORF output of Bear2_Thumb [1]

Figure 3.15: CORF output of Bear2_Thumb by my code

We have also performed another experiment using our DOG based CNN architecture (Fig. 3.4), where instead of using 25 filters of size 5×5 in the first layer, we used 8 filters of size 3×3, 8 filters of size 5×5 and 9 filters of size 7×7. The accuracy of this model on the CIFAR-10 dataset is shown in Table-3.3. It is interesting to observe that the performance of the DoG based CNN remains very consistent. The corresponding accuracy curve and loss curve are shown in Fig. 3.16 and Fig. 3.17.

Table 3.2: Parameters in the CNN Models

	DoG based CNN	ResNet-18	CORF based CNN
Total parameters	9,113,066	11,008,962	9,126,922
Trainable parameters	9,103,082	10,995,394	9,116,810
Non-trainable parameters	9,984	13,568	10,112

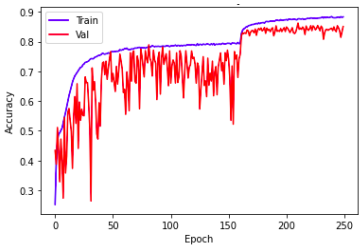

Figure 3.16: Accuracy curve of DOG(3x3, 5x5 and 7x7) based CNN

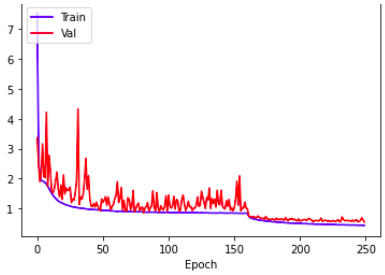

Figure 3.17: Loss curve of DOG (3x3, 5x5 and 7x7) based CNN

Table 3.3: Performance of DOG based CNN using three different types of DOG filters in the first layer

Train accuracy	88.32%
Validation accuracy	85.08%
Test accuracy	84.94 %

Chapter 4

Conclusion

Our objective in this study was to incorporate some knowledge from computational models of LGN cells into CNNs so that simpler networks can produce better performance. Also, we wanted the basic features to be extracted in a manner similar to those by biological neurons. For this, we have considered the CORF (combination of receptive fields) model. As a base network, we have considered ResNet-18 due to our limited computational resources. In this context, we first implemented a CNN with a simpler architecture which took the CORF model output as the input. Our ultimate intention was to incorporate the CORF computation in the first few layers of the CNN. However, the performance of our CORF-based CNN was not satisfactory. One possible reason for this could be some issues with our implementation of the CORF model because, on some benchmark data, our implementation could not produce the same outputs as reported by the authors in [1].

Since the CORF does its main computations on the difference of Gaussians (DoGs), we proposed a simple CNN, named DoG based CNN where the first layer of the CNN uses a number of DoG filters. This architecture has been found to produce better performance with a simpler architecture than ResNet-18.

There are few things, that we like to do in the future to complete the study, include the following: (i) Since the validation error on the chosen validation set was consistent with the training error, we did not use it for early stopping. However, random partitioning of the data into training and validation and use of the same for early stopping might be required if we had run the algorithm for more epochs. This could also improve the performance. (ii) The CORF implementation needs to be relooked at. We strongly believe that this would help realize a better network with simpler architecture. (iii) There are some parameters that we had to choose. Use of the validation set to find better choices of these parameters may further improve the performance. (iv) The use of a two-tower model, one using the ResNet-18 (or a similar) architecture and the other using the DoG based CNN could enhance the performance. This two-tower network will do the classification using the features extracted by both towers.

Bibliography

[1] George Azzopardi and Nicolai Petkov. A corf computational model of a simple cell with application to contour detection. *Biological Cybernetics*, 106:177–189, 2012.

[2] C. Grigorescu, N. Petkov, and M. A. Westenberg. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing*, 12(7):729–739, 2003.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 37:448–456, 2015.

[5] Norton TT Irvin GE, Casagrande VA. Center surround relationships of magnocellular, parvocellular, and koniocellular relay cells in primate lateral geniculate nucleus. *Vis Neurosci*, 10(2):363–373, 1993.

[6] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[7] Mandar Kulkarni and Shirish Subhash Karande. Layer-wise training of deep networks using kernel similarity. *CoRR*, abs/1703.07115, 2017.

[8] Lili Mou, Rui Men, Ge Li, Lu Zhang, and Zhi Jin. On end-to-end program generation from user intention by deep neural networks. *CoRR*, abs/1510.07211, 2015.