Can I Make My Deep Network Somewhat Explainable?

A dissertation
submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Technology
in
Computer Science

Submitted by

Mohan Kumar Dangi
(Roll No: CS1716)

Under the Guidance of

Prof. Nikhil R. Pal
Electronics and Communication Sciences Unit

Indian Statistical Institute
Kolkata-700108, India
July, 2019

M.Tech(cs) Thesis Completion Certificate

Title: Can I Make My Deep Network Somewhat Explainable?
Student: Mohan Kumar Dangi
Roll No: CS1716
Supervisor: Prof. Nikhil R. Pal

This is to certify that the thesis entitled **"Can I Make My Deep Network Somewhat Explainable?"** submitted by **Mohan Kumar Dangi** in partial fulfillment for the award of the degree of **Master of Technology (Computer Science)** is a bonafide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and has reached the standard needed for submission. The results contained in this thesis have not been submitted to any other university for the award of any degree or diploma.

Prof. Nikhil R. Pal
ECS Unit
ISI Kolkata

Place: Kolkata
Date:

Dedicated to my parents, my brother, and my well wishers...

Acknowledgements

I am deeply indebted to many people who have contributed in making the completion of this thesis work possible. I would like to express my earnest gratitude and respect to my supervisor, **Prof. Nikhil R. Pal**, for his guidance, constant support and monitoring throughout the course of this work. I do wish to gratefully acknowledge the continuous cooperation and support provided by Ankit Varshney and all other research fellows and project linked personnel's of ECSU, Indian Statistical Institute, Kolkata. I convey my gratitude to the teachers of Indian Statistical Institute for there support.

Mohan Kumar Dangi

Abstract

Deep neural networks (DNNs), as well as shallow networks, are usually black boxes due to their nested non-linear structure. In other words, they provide no information about what exactly makes them arrive at their predictions/decisions. This lack of transparency can be a major drawback, particularly in critical applications, such as medicine, judiciary, and defense. Apart from this, almost all DNNs make a decision even when the test input is not from one of the classes for which they were trained or even when the test point is far from the training data used to design the system. In other words, such systems cannot say "don't know" when they should. In this work, we develop systems that can provide some explanations for their decisions and also can indicate when they should not make a decision. For this, we design DNNs for classification, which can classify an object and provide us with some explanation. For instance, if the network classifies an image, say a bird of kind Albatross, the network should provide some explanatory notes on why it has classified the image as an instance of Albatross. The explanation could be pieces of information that are distinguishing characteristics of Albatross. The system also detects situations when the inputs are not from the trained classes. To realize all these, we use four networks in an integrated manner: a pre-trained convolutional neural network (we use it as we do not have an adequate computing power to train from the scratch), two multilayer perceptron networks, and a self-organizing (feature) map. Each of these networks serves a distinctive purpose.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Though deep neural networks (DNNs) are extremely popular due to their classification abilities, they lack transparency, and hence, act as *black boxes*. A *black box* cannot describe why it makes a decision. Hence, they are not trustworthy. It is, therefore, important to create models that are trustworthy, responsible, and capable of providing us with an explanation [1]. Explainable artificial intelligence (XAI) aims to produce more explainable models while maintaining a high level of learning performance (prediction accuracy); and enable human users to understand, trust, and effectively manage the emerging generation of artificially intelligent partners [2]. Note that, in this work, we do not make CNN transparent but we augment it with explainability.

1.1 What is Explainable Artificial Intelligence (XAI)?

The concept of explainability sits at the intersection of several areas of active research in AI, with a focus on the following [3]:

Transparency: We have a right to have decisions affecting us explained to us in the language we understand.

Causality: When we learn a model from data, can this model provide us with not only the correct inferences but also some explanation for the underlying phenomena or the causes?

Bias: How can we ensure that the AI system has not learned a biased view of the world based on the limitations of the training data or objective function?

Fairness: If decisions are made based on an AI system, can we verify that they were made fairly?. This issue is somehow related to the bias.

Trustworthiness: Can we trust our AI system without an explanation of how it reaches the conclusions?. Can it decline to make a decision when it should?

An XAI or transparent AI or interpretable AI is an AI in which the actions of the system can be easily understood and analyzed by humans.

1.2 Objective and Organization of This Thesis

The objective of this work is to design a classifier using DNNs, which can provide us with an explanation behind the decision it makes, has a decent performance in terms of accuracy and can refuse to make decisions when required.

This thesis has seven chapters. Chapter 2 provides the preliminary knowledge required for understanding the proposed work. Chapter 3 presents related works. Chapter 4 gives an overview of the dataset. In Chapter 5, we discuss the proposed work. Chapter 6 presents the experiments and their results. Finally, we conclude in Chapter 7.

Chapter 2

Preliminaries

2.1 Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) has an input layer to receive the input and an output layer that makes a decision or prediction about the input. Moreover, in between those two layers, it can have an arbitrary number of hidden layers that are the true computational engine of the MLP. Here, we consider an MLP with a single hidden layer. Figure 2.1 shows the architecture of the MLP that we use here. An MLP with a single hidden layer is a function $f : \mathbb{R}^D \to \mathbb{R}^L$, where D is the length of each input vector $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$, and L is the size of the output vector, $\mathbf{f}(\mathbf{x})$, such that, in a matrix notation:

$$f(\mathbf{x}) = \mathcal{S}^{(2)}(\mathbf{b}^{(2)} + W^{(2)}(\mathcal{S}^{(1)}(\mathbf{b}^{(1)} + W^{(1)}\mathbf{x}))), \tag{2.1.1}$$

where $\mathbf{b}^{(i)}$, $W^{(i)}$, and $\mathcal{S}^{(i)}(\cdot)$ denote the bias vector, weight matrix, and an array of activation functions of the ith layer, respectively. In this work, we use ReLU as the activation functions of the hidden layer neurons, sigmoid function as the activation functions of the penultimate layer neurons and softmax function as the activation functions of the output layer neurons. In particular,

$$\mathcal{S}^{(1)}_{(j)}(\xi) = \begin{cases} 0 & \text{for} \quad \xi < 0 \\ \xi & \text{for} \quad \xi \geq 0 \end{cases}, j = 1, 2, \cdots, H \tag{2.1.2}$$

$$\mathcal{S}^{(2)}_{(j)}(\xi) = \frac{1}{1 + e^{-\xi}}, j = 1, 2, \cdots, L. \tag{2.1.3}$$

Here, H is the number of nodes in the hidden layer.

For an input $\mathbf{x}$, let $\mathbf{o} = (o_1, o_2, \ldots, o_L)^T \in \{0, 1\}^L$ be the desired output. After feeding the input $\mathbf{x}$ to the network, at the output layer we get $\mathbf{y} = (y_1, y_2, \ldots, y_L)^T \in \mathbb{R}^L$. Our objective is to minimize the difference between the predicted output $\mathbf{y}$ and the desired output $\mathbf{o}$. So we want to minimize the instantaneous squared error

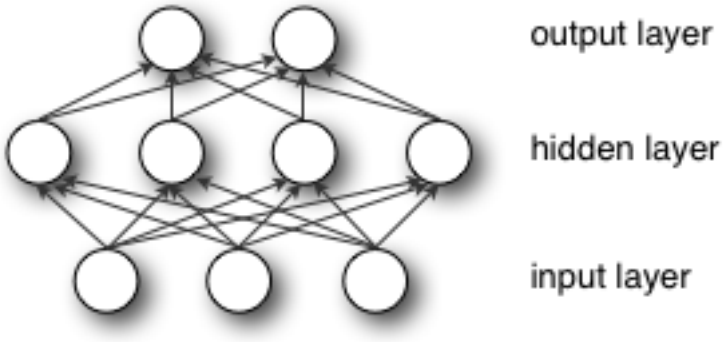


Figure 2.1: MLP with one hidden layer

(ISE), $E(\mathbf{x})$:

$$E(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{L}(o_i - y_i)^2 \tag{2.1.4}$$

Thus, for a given set of training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{o}_i)\}_{i=1}^{n}$ with n samples, we minimize the mean squared error, $\mathcal{E}$:

$$\mathcal{E} = \frac{1}{n}\sum_{\mathbf{x}} E(\mathbf{x}) \tag{2.1.5}$$

To train an MLP, we learn all parameters of the model, $\boldsymbol{\theta} = \{W^{(2)}, \mathbf{b}^{(2)}, W^{(1)}, \mathbf{b}^{(1)}\}$, using stochastic gradient descent.

2.2 Self Organizing Map (SOM)

Kohonen's self-organizing map (SOM) [4, 5], a type of artificial neural network (ANN), can be used to represent important topological information. A SOM is trained in an unsupervised fashion that generates a lower (usually two or three) dimensional discrete representation of the training samples, which is called a map. SOMs apply a competitive learning rule, where the output nodes compete among themselves to represent distinct training samples by preserving the relationships, i.e., the topology in the data. Figure 2.2 shows the architecture of a SOM. A SOM works in two modes: training and mapping. For the input data, "training" mode builds the map, whereas in the "mapping" mode it classifies a new input example. A finite lower-dimensional map space is defined before it goes to the "training" mode. In this work, we use a two-dimensional map.

A SOM is a two-layered network. The first layer is the input layer. If the training data are in an D-dimensional space, the input layer will have D input nodes $(x_1, x_2, \ldots, x_D)$. The second layer is the competitive layer (or output layer) of nodes $(N_1, N_2, \ldots, N_k)$. There is a complete connection between the two layers. The sec-

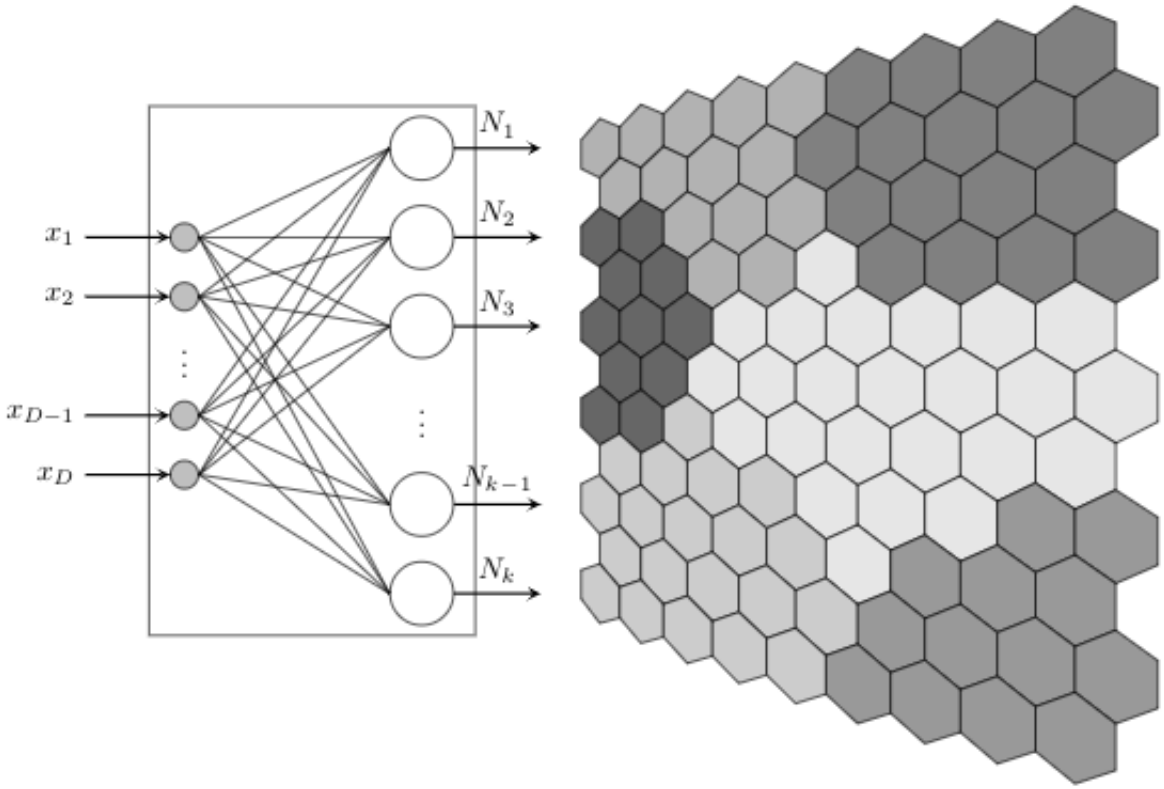


Figure 2.2: The Architecture of a Self Organizing Map (SOM)

ond layer nodes are arranged on a two-dimensional lattice. Thus each node is associated with a D-dimensional weight vector. The task of "training" is to move the weight vectors towards input data, but without damaging the topology induced from the map space. A SOM involves the following three characteristics [6]:

Competition: The output nodes in a SOM compete with each other for a better representation of the particular input sample. For each output node, we compute a value by comparing an input vector to the associated weight vector using a discriminant function (usually Euclidean distance). The output node with the minimum Euclidean distance associated with the particular input is declared as the best-matching unit (BMU) or the winner of the competition.

Let there be k nodes in the map space. So for the j^{th} node the weight vector is, $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jD})^T$ where $j = 1, 2, \ldots, k$. Now, to find the best matching unit, we find the Euclidean distances between input $\mathbf{x}$ and the weight vector $\mathbf{w}_j, \forall j$. For input $\mathbf{x}$, we select the $i_\mathbf{x}$th node as the winner (best matching) node, such that, the following hold:

$$i_\mathbf{x} = \arg\min_j \left\{ ||\mathbf{x} - \mathbf{w}_j||^2 \right\} \tag{2.2.1}$$

Cooperation: A SOM performs topological preservation of input data in which

nearby locations in the output space represent inputs with similar properties. For, an input $\mathbf{x}$, a SOM finds a neighborhood around the winning node, i.e., the $i_{\mathbf{x}}th$ neuron. Instead of defining an explicit neighborhood, we use an implicit neighborhood function. For this, we define $h_{ji_{\mathbf{x}}}$ as the topological neighborhood centered around the $i_{\mathbf{x}}^{th}$ node, encompassing node j. $h_{ji_{\mathbf{x}}}$ decreases with the lateral distance on the 2D-lattice(grid), i.e., the distance between the winning node $i_{\mathbf{x}}$ and node j. The lateral distance between node $i_{\mathbf{x}}$ and node j is denoted by $d_{ji_{\mathbf{x}}}$, where node $i_{\mathbf{x}}$ is the winning node and node j is the excited node as an effect of the winning node. Here, a typical choice is to use Gaussian type neighborhood function, i.e.,

$$h_{ji_{\mathbf{x}}} = \exp\left(-\frac{d_{ji_{\mathbf{x}}}^2}{2\sigma^2}\right). \tag{2.2.2}$$

This neighbourhood does not depend on the position of the winning node. Therefore, it is translation invariant. In case of two dimensional output space $d_{ji_{\mathbf{x}}} = ||\mathbf{r}_j - \mathbf{r}_{i_{\mathbf{x}}}||^2$, where $\mathbf{r}_j$ is the coordinate of the excited node j and $\mathbf{r}_{i_{\mathbf{x}}}$ is the coordinate of the winning node $i_{\mathbf{x}}$. With iterations, we decrease σ as follows:

$$\sigma(t) = \sigma_{init} - (\sigma_{init} - \sigma_{final})\frac{t}{t_{max}}, \tag{2.2.3}$$

where t is the iteration counter, σ_{init} and σ_{final} are parameters to control neighbourhood, and t_{max} is the maximum number of iterations. Consequently, for iteration t, (2.2.2) can be rewritten as follows:

$$h_{ji_{\mathbf{x}}}(t) = \exp\left(-\frac{d_{ji_{\mathbf{x}}}^2}{2\sigma^2(t)}\right). \tag{2.2.4}$$

We note that there could be other ways of shrinking the neighborhood.

Synaptic Weight Adaptation: We adjust the weight vectors of the winner and its neighboring units in the map in favor of higher values of their discriminant functions. Here, we use Hebbian learning. It means that when the presynaptic and post-synaptic features are correlated then the synaptic connection is strengthened. If they are not correlated then the synaptic connection is weakened.

The usual way of finding the change in weight with the Hebbian hypothesis is as follows:

$$\Delta\mathbf{w}_j = \eta y_j \mathbf{x}, \tag{2.2.5}$$

where $\Delta \mathbf{w}_j$ is the change in the weight vector $\mathbf{w}_j$, and $\eta \in \mathbb{R}_+$ is the learning rate. Now to forget, considering $y_j = h_{ji_\mathbf{x}}$ we modify (2.2.5) as follows:

$$\begin{aligned} \Delta \mathbf{w}_j &= \eta y_j \mathbf{x} - \eta y_j \mathbf{w}_j \\ &= \eta h_{ji_\mathbf{x}} \mathbf{x} - \eta h_{ji_\mathbf{x}} \mathbf{w}_j \\ &= \eta h_{ji_\mathbf{x}} (\mathbf{x} - \mathbf{w}_j) \end{aligned} \tag{2.2.6}$$

Using discrete time formulation we obtain the following weight updation formula:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{ji_\mathbf{x}}(t)(\mathbf{x} - \mathbf{w}_j(t)), \tag{2.2.7}$$

where $\eta(t)$ is dependant on t and it is defined as follows:

$$\eta(t) = \eta_{init} \left(1 - \frac{t}{t_{max}}\right). \tag{2.2.8}$$

Here, η_{init} is the initial value of the learning rate. Note that, with this update rule, the weight vector associated with the winning node achieves the maximum degree of update and it moves closer to $\mathbf{x}$. All other weight vectors also move closer to the input. However, for them, the strength of the update reduces with an increase in the distance of the node from the winner node.

We use a two-phase learning for SOM. In the first phase, the weights of the winner and its neighbors are updated in the way mentioned above. In the second phase, we update only the weight vector associated with the winner node for some iterations to refine the weights.

2.2.1 Learning of a SOM

We assume that the initial value of the learning rate (η_{init}); parameters to control neighbourhood (σ_{init}) and (σ_{final}); and the maximum number of iterations (t_{max}); and the size and the dimension of the output space are given. We perform the following steps to train a SOM [7]:

i. We randomly initialize the weight vectors $w_j(0)$, normalize the training data, and set iteration index $t = 1$.

ii. We randomly select an input vector $\mathbf{x}$ from the training data set.

iii. We compute the Euclidean distances between $\mathbf{x}$ and each weight vector of the output node to find the best matching node $i_\mathbf{x}$ using (2.2.1).

iv. We update the weights of the winning node and its neighborhood using (2.2.7).

v. We set $t \leftarrow t + 1$. We then update the neighborhood size and the learning rate using (2.2.4) and (2.2.8), respectively.

vi. We repeat Steps (ii)-(v) until t reaches $0.8 \times t_{max}$.

vii. We repeat Steps (ii)-(v) for $0.2 \times t_{max}$ steps while updating only the winner.

Chapter 3

Related Works

Usually, deep neural networks (DNNs) are large networks, in terms of the number of layers, the number of learnable parameters, and the number of nodes per layer. Consequently, they require a large number of elementary operations to make a decision. When we attempt to explain a decision made by a DNN, we need to reduce the complexity of these operations. In [8], the authors have proposed LIME, a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction.

Recently, both the hardware quality and data availability have improved significantly. Consequently, high-performance predictive systems, which are complex and opaque, have emerged. However, these models do not consider the interpretability and explainability of these systems. Explainable models consider important features while predicting the output, such that, these important features can be used to generate an explanation for a given decision. In [9], researchers have proposed a layer-wise relevance propagation and sensitivity analysis to explain prediction for DNNs in terms of input variables.

Hendricks et al. [10] proposed a model that focuses on discriminating properties of a visual object. It jointly predicts a class label and explains why the predicted label is appropriate for the image. Their explanation model differs from the caption model. This is so because it includes the object category as an additional input and uses a reinforcement learning-based discriminative loss to learn the same.

Based on fine-grained visual categorization, Berg et al. [11] proposed a fully automatic method for choosing the best part-based features from a large set of features. These features identify the difference between two similar classes. This method annotates the images to show these distinguishing features.

Chapter 4

Dataset

We use Caltech-UCSD Birds-200-2011 (CUB-200-2011) [12] dataset. It is an image dataset with 11788 samples that is divided into a training dataset with 5994 samples and a test dataset with the remaining 5794 samples. There are 200 North American bird species (classes) in the data, and hence, there are approximately 60 samples per class. Figure 4.1 shows 200 samples from the dataset, such that, each sample corresponds to a distinct class [13]. Each bird is characterised by 28 attributes, where depending on the type of the birds, different attributes take different values. When these attributes are represented using orthogonal coding, the length of the attribute vector for each bird becomes 312. These attributes are: (1) bill shape, (2) wing color, (3) upper-parts color, (4) underparts color, (5) breast pattern, (6) back color, (7) tail shape, (8) upper tail color, (9) head pattern, (10) breast color , (11) throat color, (12) eye color, (13) bill length, (14) forehead color, (15) under tail color, (16) nape color, (17) belly color, (18) wing shape, (19) size, (20) shape, (21) back pattern, (22) tail pattern, (23) belly pattern, (24) primary color, (25) leg color, (26) bill color, (27) crown color, and (28) wing pattern. Description of some bird attributes are given in Table 4.1. From this dataset, we consider 10 distinct classes of birds that contain 600 images (60 samples per class) along with their 312 binary attributes. These 10 species are (1) Laysan Albatross, (2) Red winged Blackbird, (3) Bobolink, (4) Indigo Bunting, (5) Eastern Towhee, (6) Pelagic Cormorant, (7) Shiny Cowbird, (8) American Crow, (9) Black billed Cuckoo, and (10) Purple Finch.

Figure 4.1: Illustrative examples from the Caltech-UCSD Birds-200-2011 dataset

Attributes	Description
bill shape	curved, dagger, hooked, needle, hooked seabird etc.
wing color	blue, brown, iridescent, purple, rufous, grey, yellow, olive etc.
breast pattern	solid, spotted, striped, multi colored
tail shape	forked, rounded, notched, fan shaped, pointed, squared
head pattern	spotted, crested, masked, unique pattern, plain etc.
bill length	about the same as head, longer than head, shorter than head
wing shape	rounded, pointed, broad, tapered, long
size	large, small, very large, medium, very small
shape	like upright perching water, like chicken marsh, like long legged, like duck, like owl, like gull etc.

Table 4.1: Attribute Description for the Caltech-UCSD Birds-200-2011 dataset

Chapter 5

Proposed Work

In this Chapter, we first discuss the training of the system used in this work. Then, we discuss the testing of the system.

5.1 Training of the System

5.1.1 Extracting Features With A Pre-trained CNN

We use a pre-trained convolutional neural network (CNN) called VGG16 [14] which is present in `keras` library (version 2.2.4). It is trained on a dataset of roughly 1.2 million images belonging to 1000 classes. We remove the last three fully connected layers of VGG16, and use the features extracted by the remaining network. We illustrate the modified architecture of the remaining CNN in Fig. 5.1. In Fig. 5.1, f_i represents the ith feature extracted from the modified VGG16. When we provide a 224×224 dimensional RGB images in the input layer, the modified CNN produces a $7 \times 7 \times 512$ dimensional representation of the images that on flattening produces a feature vector of dimension 25088.

5.1.2 Training of the First MLP - Classification MLP

Now, we train an MLP with the 25088 dimensional features extracted by the modified VGG16 to predict the class labels. Consequently, there are 10 nodes in the output layer and we use a one-hot encoding of dimension 10. We use three hidden layers in this MLP, such that, these layers have 1000, 500, and 150 numbers of nodes, respectively. We choose this architecture based on some ad-hoc experiments. No claim is made that this is the best architecture for this problem. Additionally, each of the input layer and the three hidden layers also contains a single bias node. The input layer is a fanout layer. We use ReLu activation functions in the first hidden layer, sigmoidal activation functions in the second and third hidden layers, and softmax activation function in the output layer. The architecture of this MLP is shown in Fig. 5.2. Here f_i represents the i^{th} feature extracted from the modified VGG16. We use `keras` (Python version 3.7.2) based implementation for this MLP along with the `adam` optimizer and the `categorical_crossentropy` loss function. We use only 150 epochs for training and a batch size of 30.

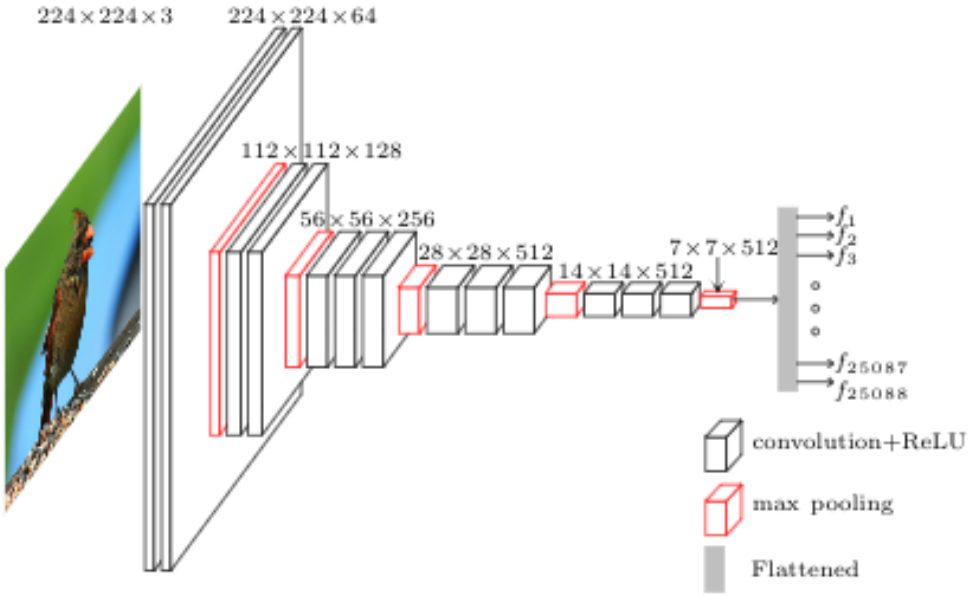


Figure 5.1: Modified VGG16 network for feature extraction

5.1.3 Training of the Second MLP - Attribute MLP

We remove the last layer of the Classification MLP and use the features extracted by the remaining network. When we provide a 25088 dimensional feature vector (extracted from modified VGG16) in the input layer of the Classification MLP, it produces a feature vector of dimension 150 at the penultimate layer. We use these 150 dimensional features to train the second MLP, where we use the 312 dimensional binary vectors as the target vectors. We call this MLP as Attribute MLP. This Attribute MLP has two hidden layers with 500 and 600 nodes, respectively. We consider 150 features extracted from the Classification MLP to train an Attribute MLP to predict the 312 binary attributes of the birds. The input layer is a fanout layer and uses ReLu activation functions in the first hidden layer. The second hidden layer and output layer uses sigmoidal activation functions. Additionally, each of the input layer and the two hidden layers also contains a single bias node. The architecture of this MLP is shown in Fig. 5.3. In Fig. 5.3, p_j represents the jth feature extracted from the penultimate layer of the Classification MLP and $\hat{a}_k$ represents the kth predicted attribute of the associated bird. We use `keras` (Python version 3.7.2) based implementation for this MLP along with the `adam` optimizer and the `mean_square_error` function. Here we use 1500 epochs and a batch size of 30 for training. The intention here is to use attributes that are distinguishing characteristics of the birds i.e., the attribute that defines the category of the birds.

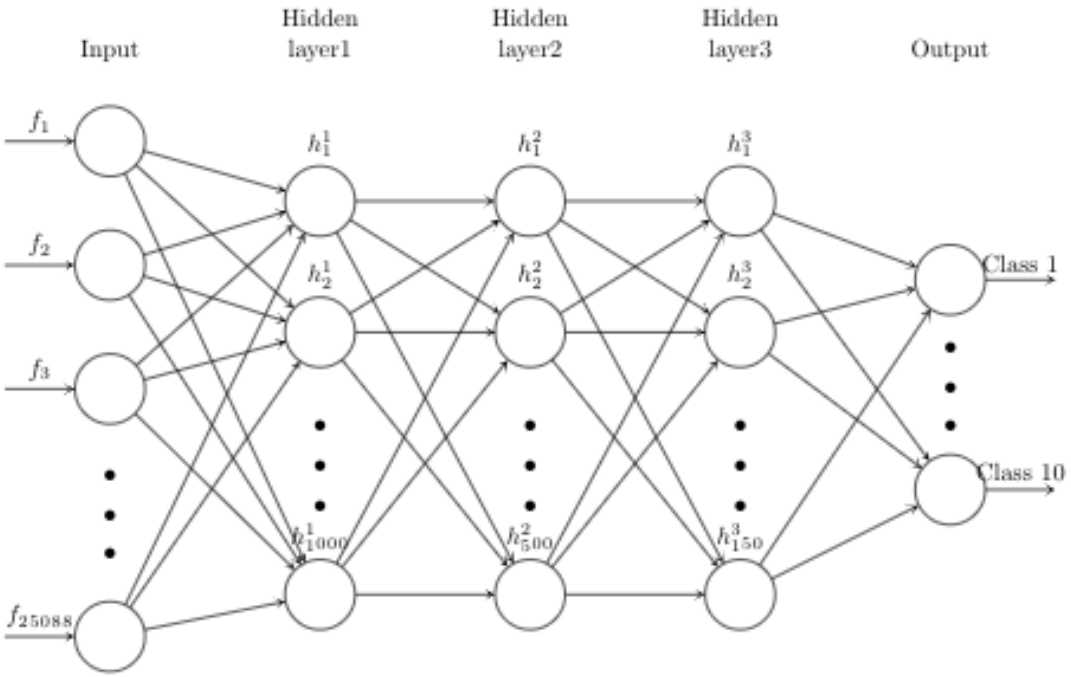


Figure 5.2: The Architecture of the Classification Multilayer Perceptron (MLP)

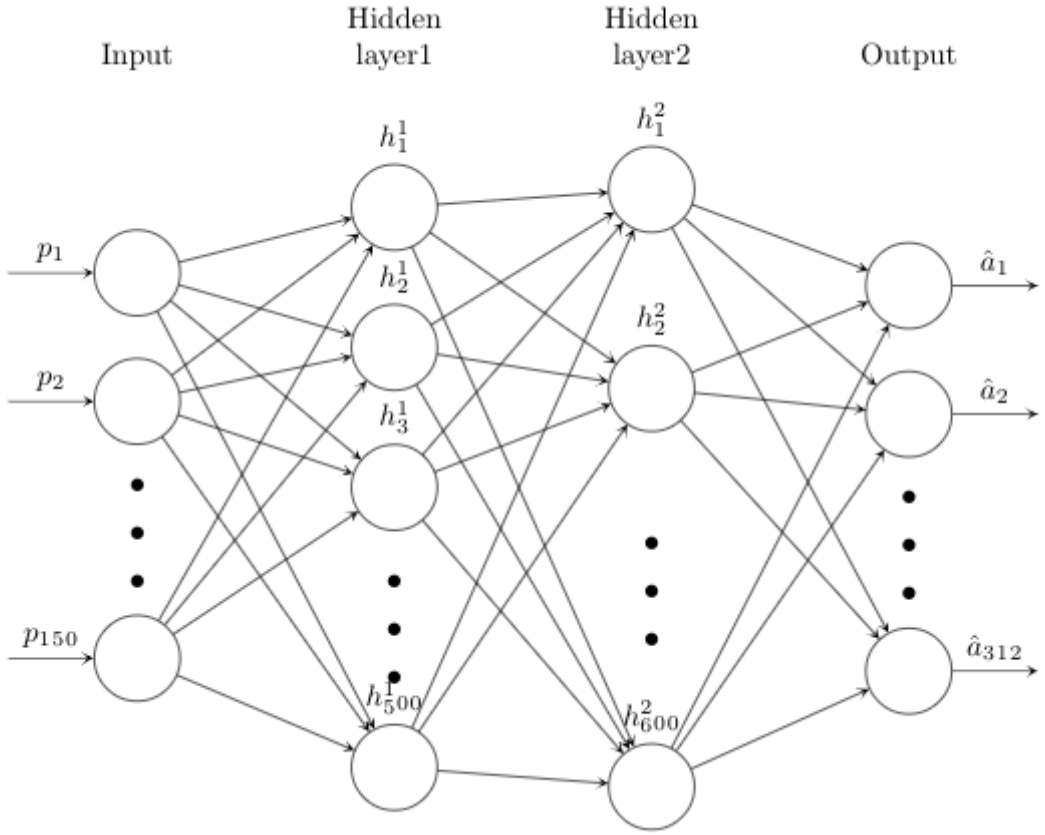
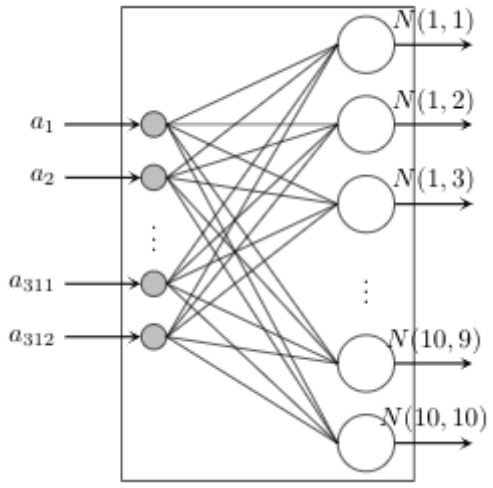


Figure 5.3: The Architecture of the Attribute Multilayer Pereceptron (MLP)

5.1.4 Training a Self Organizing Map (SOM)

We use the original 312-dimensional binary attributes to train a SOM. As explained earlier, there are two layers in the SOM: an input layer and an output layer. The



8	8	8	7	2	2	2	3	3	3
8	0	7	7	2	2	2	3	3	3
8	8	8	7	5	5	5	5	3	3
6	7	2	2	5	5	5	5	5	10
6	6	2	7	3	5	10	10	10	10
6	6	2	7	3	3	10	10	10	10
6	6	6	4	4	10	10	10	10	10
1	1	6	7	9	3	5	9	4	4
1	1	9	9	9	0	5	10	4	4
1	1	9	9	9	9	9	4	4	4

Figure 5.4: SOM Training with attributes of birds

single dimensional input layer has 312 nodes and the two dimensional output layer has 10×10 nodes. The architecture of the SOM is shown in Fig. 5.4. Here a_k represents the kth binary attribute of the bird and $N(i,j)$ represents the output node present at the ith row and the jth column. The output nodes in this SOM compete with each other to best represent the particular bird attributes. To train the SOM, we use the following parameters: (1) number of iterations $t_{max} = 80000$, (2) the range of the (uniformly distributed) initial weights is $[0,1]$, (3) learning rate is $\eta = 0.01$, and (4) the parameters to control neighbourhood are $\sigma_{init} = 11$ and $\sigma_f = 0.2$

The number in the rectangular grid represents the class label of bird having maximum frequency at that node, such that, a zero indicates that no class has been assigned to that node and a value $v \in \{1, 2, \ldots, 10\}$ indicates that the maximum number of samples assigned to the cell belongs to class v.

After Training the SOM, for each sample $\mathbf{x}$ we compute its similarity with the associated winner node:

$$S_{i_\mathbf{x}} = \exp\left(-||\mathbf{x}, W_{\text{SOM}}(i_\mathbf{x})||^2\right); \tag{5.1.1}$$

where $S_{i_\mathbf{x}}$ is the similarity of the $i_\mathbf{x}$th node on mapping $\mathbf{x}$ for which $i_\mathbf{x}$th node is the winner node, and W_{SOM} is the weight matrix of the trained SOM. Now, we calculate the threshold T_j for the jth output node as follows:

$$T_j = \min_{\substack{\mathbf{x} \\ i_\mathbf{x}=j}} \{S_{i_\mathbf{x}}\}. \tag{5.1.2}$$

Thus T_j is the minimum similarity of all inputs whose best matching unit was the jth SOM node. While testing, if the similarity of the winner for a given attribute

vector is less than kT_j, $k > 0$ than the match can be questioned to be good enough to make a decision. Here we use k=1.

5.2 Testing of the System

5.2.1 Extracting Features with Modified VGG16

We provide a 224×224 dimensional RGB image in the input layer of the modified VGG16 to produce a 25088 dimensional representation of the image.

5.2.2 Predicting the class label of the image

The 25088 dimensional feature extracted from the Modified VGG16 is used as an input to the Classification MLP to predict the class label of the image.

5.2.3 Predicting bird attributes

The 150 dimensional feature extracted by the penultimate layer of the Classification MLP is used as an input to the Attribute MLP to produce the prediction of a 312 dimensional attributes of the bird. For any test input, this predicted attribute vector will be used to match the SOM nodes.

5.2.4 Passing Predicted attributes to the pretrained SOM

We use the 312-dimensional predicted attributes as an input to a SOM. The output of SOM is the Best Matching Unit (BMU) or Winner node corresponding to the predicted attribute of the test input. In Fig. 5.5, we illustrate what happens to a SOM when we use the predicted attributes of the bird as an input to the SOM. Here $\hat{a}_k$ represents the kth predicted attribute of the bird and $N(i, j)$ represents the output node present at the ith row and the jth column. After finding the BMU, we consider the four neighbours of the winner node: the neighbours at the left, right, top, and bottom. Among these four neighbours, we find the Second Best Matching Unit (SBMU) or the second winner node corresponding to the predicted attribute of the bird. We do this as SOM preserves the topology of the inputs. Thus birds with similar attributes will be mapped to nearby nodes on the lattice.

5.2.5 Generating Explanation

We first describe how we generate an explanation from the SOM weight vector. For each attribute, there are multiple possible values. While generating the explanation, first we find the highest attribute value. If the highest attribute value

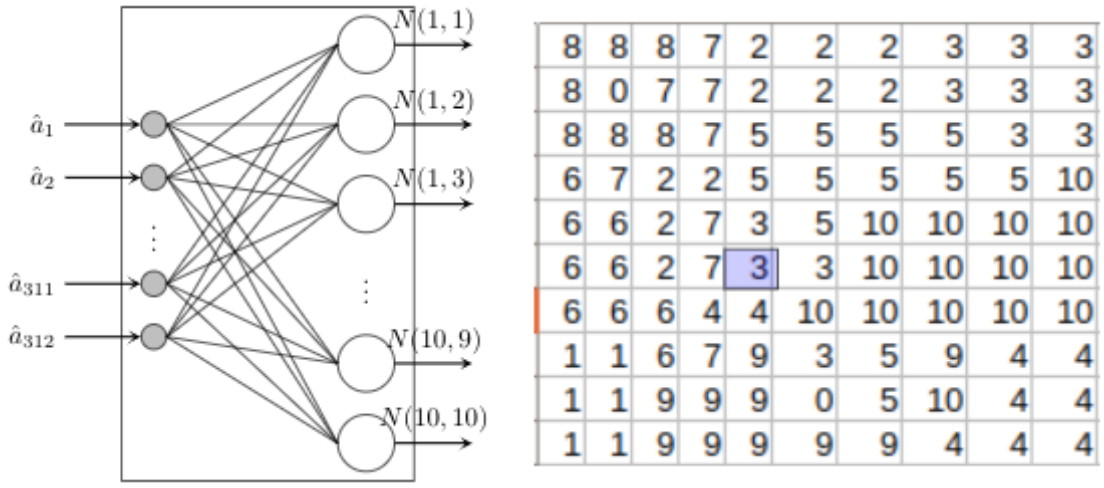


Figure 5.5: Passing predicted attributes to the pretrained SOM

is greater than equal to 0.7 we declare that attribute is present. Else if it lies in $[0.5, 0.7)$ we say that the attribute is probably present. Otherwise, it is absent.

Let the class label of the first MLP, best matching unit and second best matching unit of SOM be $Pred_{class}$, W_{class} and SW_{class}, respectively. Now, to generate an explanation, we first find the similarity between the input sample and the winner node of the SOM. If the similarity of the sample is greater than the threshold (T_j), then we can have any of the following three cases. First, if the $pred_{class}$ is equal to the W_{class} then we generate an explanation using the weight vector of the BMU. Otherwise we compare the $pred_{class}$ and SW_{class}. If they are equal then we generate the explanation using the weight vector of the SBMU. Otherwise, we use the weight vectors corresponding to both BMU and SBMU(using the max of the two) to produce an explanation for the image.

If the similarity between the input sample and winner node of the SOM is less than the threshold (T_j) then we may have any of the following three doubtful cases. First, if the $pred_{class}$ is equal to the W_{class} than we generate a probable explanation using the weight vector of the BMU. Otherwise, we compare the $pred_{class}$ and SW_{class}, if they are equal then we also generate a probable explanation using the weight vector of the SBMU. Otherwise, we reject the image and say "don't know" to indicate that the test sample may correspond to some unknown distribution. We use the algorithm shown in Algorithm 1 to generate the textual description. We note here that this is one possibility that we have considered. However one can design different algorithms using different use of SOM score(winner + neighborhood) and Classification MLP output. A natural question may come, why are we not considering the second best output of the Classification MLP. Yes, this is plausible but we do not use it as it is well known that MLP can produce strange generalization [15, 16], while SOM is a more transparent model.

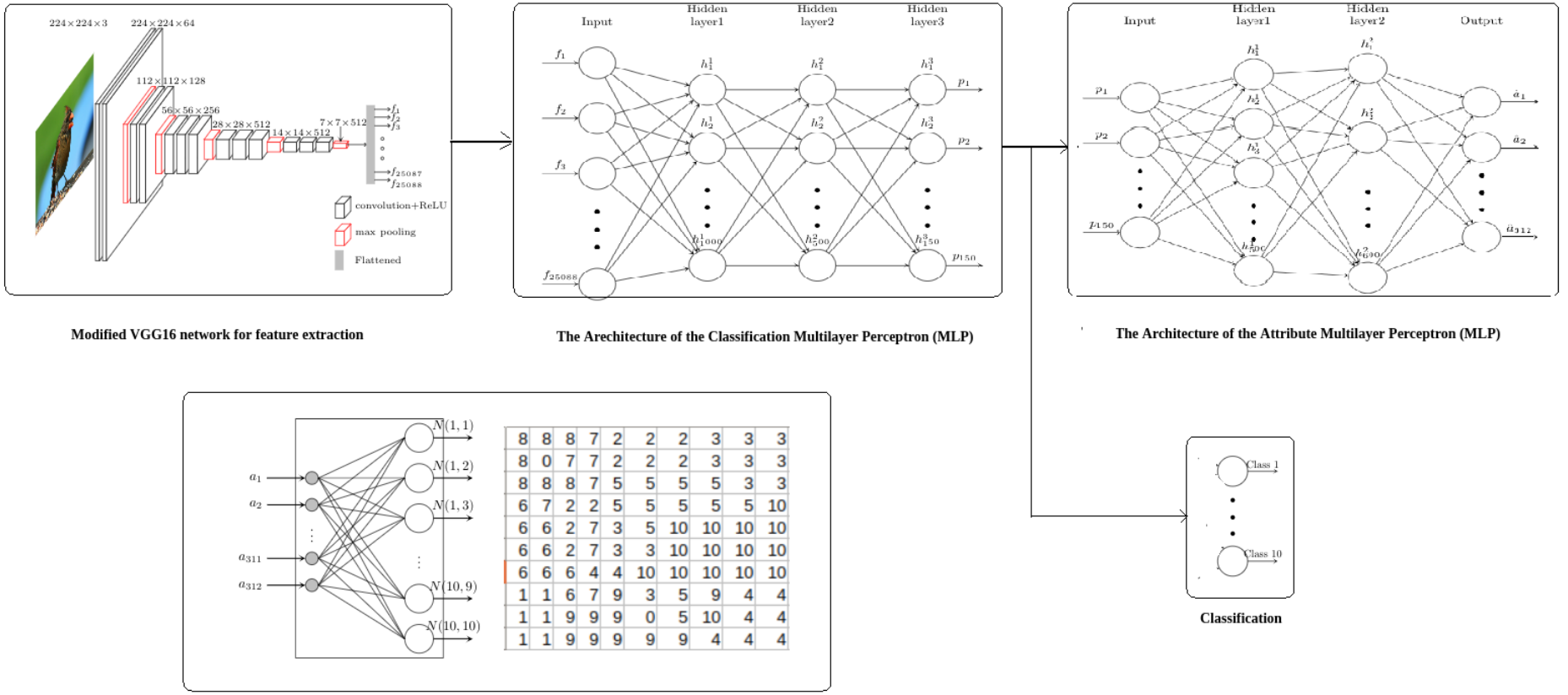
Algorithm 1 Generating Explanation

Result: Explanation of the bird.

```
if (S_{i_x} > T_j) then
    if (pred_class = W_class) then
        Generate the explanation using the weights of BMU
    else
        if (pred_class = SW_class) then
            Generate the expalnation using the weights of SBMU
        else
            Generate the explanation using weights of both BMU and SBMU
        end
    end
else
    if (pred_class = W_class) then
        Generate explanation as a doubtful case using weights of BMU
    else
        if (pred_class = SW_class) then
            Generate explanation as a doubtful case using weights of SBMU
        else
            Reject the image and say it does not belong to the set of training classes
        end
    end
end
```

5.3 Summary

We illustrate the complete training process of the proposed system in figure 5.6. Figure 5.7 shows the process of generating an explanation given an input image.



8	8	8	7	2	2	2	3	3	3
8	0	7	7	2	2	2	3	3	3
8	8	8	7	5	5	5	5	3	3
6	7	2	2	5	5	5	5	5	10
6	6	2	7	3	5	10	10	10	10
6	6	2	7	3	3	10	10	10	10
6	6	6	4	4	10	10	10	10	10
1	1	6	7	9	3	5	9	4	4
1	1	9	9	9	0	5	10	4	4
1	1	9	9	9	9	9	4	4	4

Figure 5.6: Training of a system

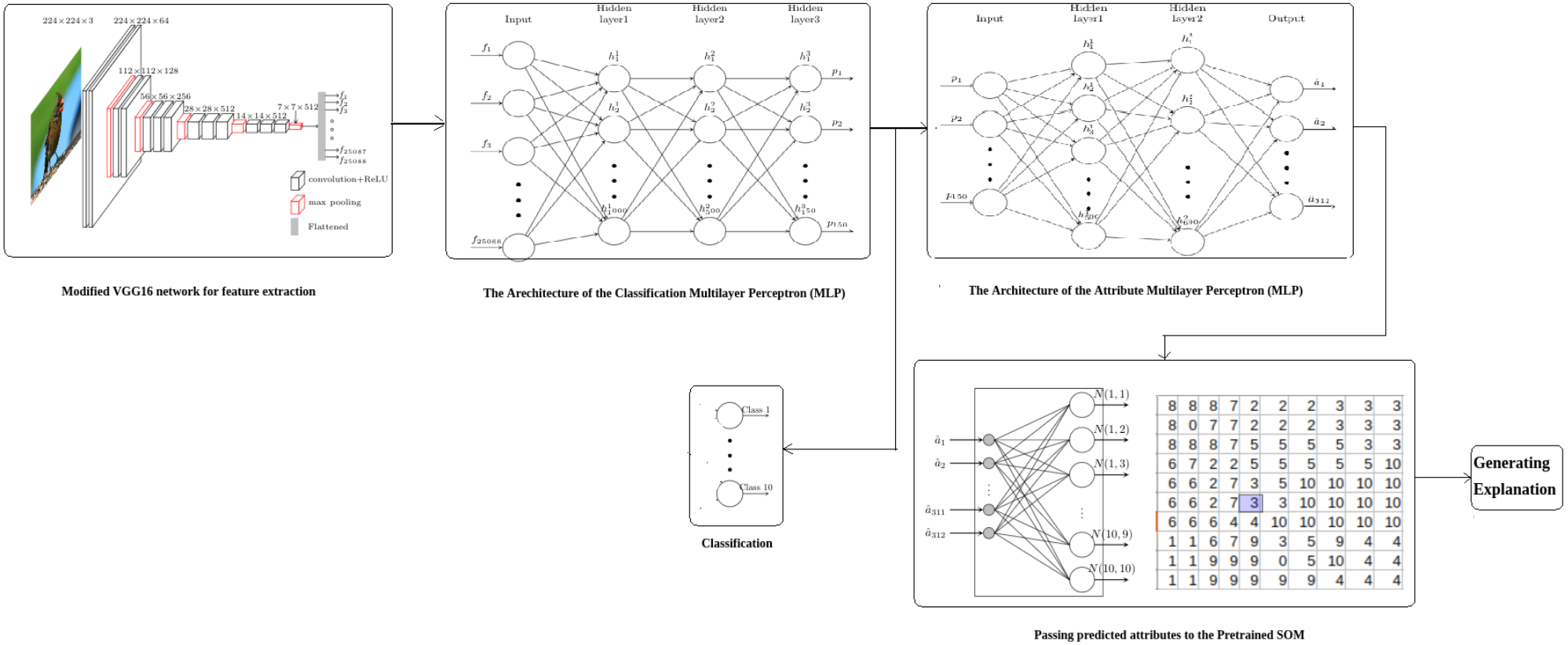


Figure 5.7: Testing of a system

Chapter 6

Results

In this Chapter, we discuss the results obtained by providing the training images, the test images, and the images that do not belong to the training classes.

6.1 Train Data

We obtain 100% training accuracy with the first MLP as the classifier. In Fig. 6.1 we show two images that belong to the training data along with the predicted class labels and the descriptions generated using the proposed method. In Fig. 6.1, the best matching unit (BMU) of the SOM predicts the same class label as that of the Classification MLP. The results shown in Fig. 6.2 are the examples of birds in which predicted class label by the Classification MLP is the same as the second best matching unit (SBMU) of the SOM. The explanation generated by the weights of the SBMU shows the visible attributes of the bird present in the image. The results shown in Fig. 6.3 are the examples of birds in which the class labels predicted by the first MLP is different from the class labels predicted by both the BMU and the SBMU of the SOM. In this case, though the SOM failed to predict the correct class label, the explanation generated matches the attributes of the bird. Here, we would like to emphasize that we do not consider the class label generated by the SOM as the predicted class label of the proposed system. Instead, we consider the class label predicted by the Classification MLP as the predicted class label of the system.

6.2 Test Data

The accuracy of the proposed system on the test data is 90%. For none of the test images (from the 10 classes), the system said "don't know". The results shown in Fig. 6.4 correspond to some cases, where the predicted class by the Classification MLP is the same as the predicted class by the BMU of the SOM. Here also, the generated explanations show the visible attributes of the birds present in the images. The results shown in Fig. 6.5 are some of the examples of birds in which the predicted class by the Classification MLP is the same as the class predicted by the SBMU of the SOM. In this case, too, the explanations generated by weights of the

The Bird is of type Purple Finch because it is small bird, has cone bill, and looks like perching.It also has bill length shorter than head, black eyes.The Bird *Probably* has multi colored wings, plain head, buff belly.

The Bird is of type Laysan Albatross because it is has hooked seabird bill, white crown, white nape.It also has bill length about the same as head, black eyes, white head, white throat, white solid breast, white solid belly.The Bird *Probably* is medium bird, has buff bill, black solid long wings, and looks like gull.

Figure 6.1: $Pred_{class}$ is the same as W_{class} of SOM

SBMUs correctly illustrates the visible attributes of the birds present in the image. The result shown in Fig. 6.6 are the examples of birds in which class labels predicted by the Classification MLP is different from the class labels predicted by both the BMU and the SBMU of the SOM. In this case, the bird class may not be correctly classified but the explanation generated matches with the visible description of the bird.

6.3 What happens when the test images are not from the training classes?

First, we randomly select 20 samples corresponding to the other 10 classes of the CUB-200-2011 dataset, such that, these classes have not been used in the training of the proposed system. We find that in 55% cases the SOM and the Classification MLP produce the same prediction for these 20 samples. In Fig. 6.7, we depict two randomly chosen images among them. The descriptions generated for these birds are sufficiently successful to capture the visible attributes of birds. This limited experiment confirms that the proposed system can also generate an acceptable textual description of types of birds that are not used for training, at least for some cases. In Fiq. 6.8 we show two birds among these 20 birds for which the proposed system said "don't know". This is what we expect the system to do when it finds patterns from unknown distributions.

To inspect further, we randomly select 20 images of animals from the Internet.

The Bird is of type Shiny Cowbird because it is small black bird, has black bill, black crown, black nape, black-blue solid tail, and looks like-perching.It also has black eyes, black plain head, black throat, black solid breast, black solid belly, black legs, black solid back.The Bird *Probably* has cone bill, black multi-colored rounded-wings, bill length shorter than head.

The Bird is of type Eastern Towhee because it has cone bill.It also has bill length shorter than head.The Bird *Probably* is small bird, has black head and looks like-perching.

Figure 6.2: $Pred_{class}$ is the same as SW_{class} of SOM

However, the system said "don't know" for only four cases. Fig. 6.9 shows two such images. For the remaining 16 images, the system provided us with a textual description of the image. Figure 6.10 depicts two such images with their descriptions. We observe that at least for these two cases the descriptions are somewhat meaningful, atleast for some attributes. For instance, the system identified that horse (though it classified it as a bird) "probably" has "buff legs".

The Bird *may* be Red winged Blackbird because it is small black bird, has black all-purpose bill, black crown, black nape, black solid wings, black-blue solid tail.It also has bill length about the same as head, black eyes, black plain head, black throat, black solid breast, black solid belly, black legs, black solid back.

The Bird *may* be Laysan Albatross because it is small brown bird, has brown crown, brown nape, brown solid tail, and looks like perching.It also has bill length shorter than head, black eyes, brown head, brown solid back.The Bird *Probably* has cone bill, brown rounded-wings, brown notched-tail, brown throat, solid breast, solid belly.

Figure 6.3: $Pred_{class}$ is different from both W_{class} and SW_{class} of SOM

The Bird is of type Black billed Cuckoo because it is small bird, has all-purpose bill, solid wings.It also has black eyes, white solid breast, white solid belly.The Bird *Probably* has black bill, brown wings, and looks like-perching, bill length shorter than head, eyering head, white throat, solid back.

The Bird is of type Eastern Towhee because it is small bird, has black cone bill, black crown, black nape, and looks like-perching.It also has bill length shorter than head, black plain head, black throat, white multi-colored breast, white belly.The Bird *Probably* is black bird, has white multi-colored wings, black eyes, multi-colored belly, buff legs, black back.

Figure 6.4: $Pred_{class}$ is the same as W_{class} of SOM

The Bird is of type Shiny Cowbird because it is small black bird, has black all-purpose bill, black crown, black nape, black solid wings, black-blue solid tail.It also has bill length shorter than head, black eyes, black plain head, black throat, black solid breast, black solid belly, black legs, black solid back.The Bird *Probably* has rounded-wings, and looks like-perching.

The Bird is of type Purple Finch because it is very small red bird, has cone bill, red crown, and looks like-perching.It also has bill length shorter than head, black eyes, red head, red throat, red breast, multi-colored belly.The Bird *Probably* has red nape, multi-colored wings, white belly, black legs.

Figure 6.5: $Pred_{class}$ is the same as SW_{class} of SOM

The Bird *may* be Purple Finch because it is medium bird.It also has black eyes.The Bird *Probably* has solid wings, solid tail, bill length about the same as head, solid breast, solid belly.

The Bird *may* be Bobolink because it has cone bill, and looks like-perching.It also has bill length shorter than head, black eyes, buff belly.The Bird *Probably* is small bird, has buff striped wings, striped breast, striped belly.

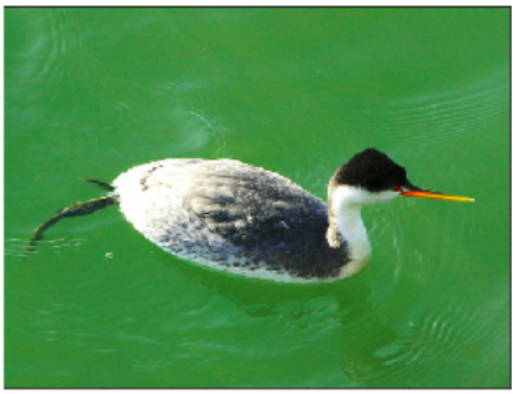
Figure 6.6: $Pred_{class}$ is different from both W_{class} and SW_{class} of SOM

The Bird is of type Black billed Cuckoo because it is small bird, has brown crown, brown nape, brown solid wings, solid tail, and looks like-perching.It also has black eyes, white throat, white solid breast, white solid belly, brown solid back.The Bird *Probably* is brown bird, has black bill, brown-buff tail, bill length about the same as head, brown head, grey legs.

The Bird is of type Laysan Albatross because it has buff bill, white crown, white nape, solid wings.It also has black eyes, white head, white throat, white solid breast, white solid belly.The Bird *Probably* is medium bird, has black wings, bill length about the same as head, eyeline head, solid back.

Figure 6.7: $Pred_{class}$ is the same as W_{class} or SW_{class} of SOM

We are not able to assign a class label because the image *may not* belong to the set of training classes.



We are not able to assign a class label because the image *may not* belong to the set of training classes.

Figure 6.8: Similarity is less than the threshold

We are not able to assign a class label because the image *may not* belong to the set of training classes.

We are not able to assign a class label because the image *may not* belong to the set of training classes.

Figure 6.9: Similarity is less than the threshold

The Bird *may* be Purple Finch because it is small bird, has cone bill. It also has bill length shorter than head.The Bird *Probably* has brown crown, brown striped wings, black eyes, brown eyering head, striped breast, solid belly, and looks like perching.

The Bird *may* be Laysan Albatross because it is small bird, has black cone bill, black crown, black nape, and looks like perching.It also has bill length shorter than head, black plain head, black throat, white multi-colored breast, white solid belly.The Bird *Probably* is white bird, has white multi-colored wings, solid tail, black eyes, black breast, multi-colored belly, buff legs, black back.

Figure 6.10: $Pred_{class}$ is same as W_{class} or SW_{class} of SOM

Chapter 7

Conclusions and Future Scopes

7.1 Conclusions

The work discussed in this thesis has addressed the problem of "explainablity" while classifying image dataset with neural networks. Using four neural networks together, we proposed a system that augments a CNN with explainability. The proposed system either says "don't know" or provides us with a predicted class label and a textual explanation about its prediction. For this, we use a pre-trained convolutional neural network (CNN), two multilayer perceptrons (MLPs), and a self-organizing map (SOM). The first MLP (Classification MLP) predicts the class label and we use the output from the SOM to generate an explanation.

7.2 Future Scopes

The predicted output depends on the classification model so a good classification model will provide a better result. We can give confidence to the predicted output using SOM. A fuzzy label can also be attached to the SOM class label and can be used in generating a class label and description. New methods can be used to find the Second Winner of SOM. Highly impure nodes may be removed from the SOM. The explanation can be improved by considering the neighboring nodes of the BMU. The threshold of each output node of SOM can be improved which can reject unknown classes more accurately. In other words, we used k=1, but other choices can be used and learned using validation data.

Bibliography

[1] Hani Hagras. "Toward Human-Understandable,Explainable AI". *IEEE Computer Society*, September 2018.

[2] D. Gunning. "Explainable artificial intelligence (XAI)." Defense Advanced Research Projects Agency (DARPA). *http://www.darpa.mil/program/explainable-artificial-intelligence*, November 2017.

[3] C. Wierzynski. "The Challenges and Opportunities of Explainable AI". *https://ai.intel.com/the-challenges-and-opportunities-of-explainable-ai/*, January 2018.

[4] T. Kohonen. "The Self-organizing Map". *Proceedings of the IEEE*, 78:1464–1480, September 1990.

[5] S. Kumar. "Neural networks: A classroom approach". 2004.

[6] S. Haykin. "Neural Networks: A Comprehensive Foundation". *2nd ed. Prentice-Hall*, 78, September 1999.

[7] Umut Asan and Secil Ercan. *"An Introduction to Self-Organizing Maps"*, pages 307–325. 01 2012.

[8] S. Singh M. T. Ribeiro and C. Guestrin. "Why should i trust you?":explaining the predictions of any classifier. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 78:1135–1144, 2016.

[9] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models". *CoRR*, abs/1708.08296, 2017.

[10] LA. Hendricks J. Donahue B. Schiele T. Darrell Z. Akata, M. Rohrbach. "Generating Visual Explanations". March 2016.

[11] P. Belhumeur T. Berg. "How Do You Tell a Blackbird from a Crow?" *2013 IEEE International Conference on Computer Vision*, December 2013.

[12] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. "The Caltech-UCSD Birds-200-2011 Dataset". Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[13] The caltech-ucsd birds-200-2011 dataset. `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`. Accessed: 2019-06-20.

[14] A. Zisserman K. Simonyan. "Very Deep Convolutional Networks for Large-Scale Image Recognition". Technical Report arXiv:1409.1556, 2015.

[15] Bikram Karmakar and Nikhil R. Pal. How to make a neural network say dont know. *Information Sciences*, 430-431:444 – 466, 2018.

[16] D. Chakraborty and N. R. Pal. A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning. *IEEE Transactions on Neural Networks*, 14(1):1–14, Jan 2003.