A Project Report

on

Automatically Identifying Animals Using Deep Learning

by

1. Parinita Badre
2. Siddhant Bandiwadekar
3. Prachi Chandanshive
4. Aakanksha Chaudhari

under the guidance of

Prof. Sonali Jadhav

Department of Computer Engineering

University of Mumbai

2017-2018

MANJARA CHARITABLE TRUST

RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

Off Juhu-Versova Link Road, Versova, Andheri(West), Mumbai-400053

Certificate

Department of Computer Engineering

This is to certify that

1. Parinita Badre
2. Siddhant Bandiwadekar
3. Prachi Chandanshive
4. Aakanksha Chaudhari

Have satisfactory completed this project entitled

Automatically Identifying Animals Using Deep Learning

Towards the complete fulfillment of the

BACHELOR OF ENGINEERING
IN
(COMPUTER ENGINEERING)

as laid by University of Mumbai.

Project Guide	**Head of Department**
Prof. Sonali Jadhav	**Dr. Satish Y. Ket**

Principal

Dr. Udhav Bhosle

Internal Examiner	**External Examiner**

Dissertation Approval for B. E.

This dissertation entitled ***"Automatically Identifying Animals Using Deep Learning"*** by ***Parinita Badre, Siddhant Bandiwadekar, Prachi Chandanshive, Aakanksha Chaudhari*** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Engineering*** from ***University of Mumbai***.

Internal Examiner

_____

External Examiner

_____

Head of Department

_____

Principal

_____

Date:

Place: Mumbai

Declaration

We wish to state that the work embodied in this synopsis titled "Automatically Identifying Animals Using Deep Learning" forms our own contribution to the work carried out under the guidance of "Prof. Sonali Jadhav" at the Rajiv Gandhi Institute of Technology.

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. we understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Parinita Badre **Roll no:** A-802

Siddhant Bandiwadekar **Roll no:** A-804

Prachi Chandanshive **Roll no:** A-811

Aakanksha Chaudhari **Roll no:** A-845

Acknowledgement

We wish to express our sincere gratitude to Dr. Udhav. V. Bhosle, Principal and Dr. Satish. Y. Ket , H.O.D of Computer Department of Rajiv Gandhi Institute of Technology for providing us an opportunity to do our project work on "Automatically Identifying Animals Using Deep Learning".

This project bears on imprint of many people. We sincerely thank our project guide Mrs. Sonali Jadhav for her guidance and encouragement in carrying out this synopsis work.

Finally, we would like to thank our colleagues and friends who helped us in completing the Project work successfully.

1. Parinita Badre
2. Siddhant Bandiwadekar
3. Prachi Chandanshive
4. Aakanksha Chaudhari

Abstract

Efficient and reliable monitoring of wild animals in their natural habitats is essential to inform conservation and management decisions. Thus having accurate, detailed, and up-to-date information about wildlife location and behavior across broad geographic areas would revolutionize our ability to study, conserve, and manage species and ecosystems. Currently, such data are mostly gathered manually at great expense, and thus are sparsely and infrequently collected.

In areas like an airport or the agricultural areas placed near the forest many animals destroy the crops or even attack on people therefore there is a need of system which detects the animal presence and gives warning about that in the view of safety purpose. Here, the ability to automatically collect such data, which could transform many fields of biology, ecology, and zoology into big data sciences is investigated.

Currently, remote wildlife cameras capture images based on passive infrared sensing or similar simple motion detecting sensors. This may capture intended wildlife, but also captures many unintended activities such as wind, etc. False positive rates are very high, as well as true negatives. A task is relegated to the researcher to filter through large amounts of images with search requirements including species identification and count. This information is often used to track migration patterns and population dynamics. Using humans to perform these tasks is the current practice, and is extremely inefficient.

Leveraging on recent advances in deep learning techniques in computer vision, we propose a framework to build automated animal recognition in the wild, aiming at an automatically identifying wildlife system. In particular, we use a single-labeled dataset and the state-of-the-art deep convolutional neural network architectures, to train a computational system capable of filtering animal images and identifying species automatically.

In this project, we demonstrate that such data can be automatically extracted by deep neural networks (deep learning), which is a cutting-edge type of artificial intelligence. Thus, the aim is to train neural networks that automatically identifies animals.

The approach for the project is using Deep Convolutional Neural Networks (CNN) to learn features of images and trained Backpropagation(BP) Neural Networks for training. The dataset consists of images which were grouped in to number of training images and test images. The experimental results show that the proposed method has a positive effect on overall animal recognition performance.

Contents

List of Figures **i**

List of Tables **ii**

1 Introduction **1**
1.1 Introduction Description 1
1.2 Task Definition 2
1.3 Motivation 2
1.4 Organization of Report 3

2 Literature Review **4**
2.1 Technical papers review 4
2.2 Methodology 5
2.3 Problem Formulation 6
2.4 Problem Statement 6
2.5 Proposed System 6

3 Design & Implementation **7**
3.1 Design 7
3.1.1 System Architecture 7
3.1.2 Convolutional Neural Networks 7
3.2 Implementation 21
3.2.1 System Requirements 21
3.2.2 Hardware Requirements 21
3.2.3 Software Requirements 21
3.2.4 TensorFlow 22

4 Results & Analysis **24**
4.1 Results 24
4.2 Analysis 26

5 Conclusions and Future Enhancement **28**
5.1 Future Enhancement 28

References **29**

Publication **30**

Annexure **31**

List of Figures

2.1 Proposed CNN Architecture 6

3.1 System Architecture 7
3.2 Procedure of Convolutional Neural Networks 8
3.3 Convolutional Neural Networks 8
3.4 Convolutional Neural Networks with multiple layers 9
3.5 Principle of Convolutional Neural Network 10
3.6 Convolving the filter with image matrix to obtain final value in feature map 10
3.7 Convolutional Neural Networks 11
3.8 The ReLU operation 12
3.9 ReLU operation working 13
3.10 Max Pooling Technique 14
3.11 Max Pooling Technique applied to Rectified Feature Maps 14
3.12 Fully connected layer 15
3.13 Inductive Learning vs. Inductive Transfer 17
3.14 Variation in Training 18
3.15 TensorFlow Toolkit Hierarchy 22

4.1 Training Dataset 24
4.2 Training Images For Cat Class 25
4.3 Input Image From Camera 25
4.4 Probabilistic output for a sample image 26
4.5 Classification Output 27

List of Tables

4.1 Accuracy for each iteration 26

Chapter 1

Introduction

1.1 Introduction Description

Observing wild animals in their natural environments is a central task in ecology. The fast growth of human population and the endless pursuit of economic development are making over-exploitation of natural resources, causing rapid, novel and substantial changes to Earth's ecosystems. Monitoring biodiversity especially the effects of climate and land use change on wild populations is a critical challenge for our society. Having an updated knowledge about wildlife behaviour would impact our ability to study and manage species and our ecosystem. Researches regarding animals in machine learning have been an important field to numerous applications.

Currently, the animal detection and recognition is still a difficult challenge and there is no unique method that provides a robust and efficient solution to all situations. Generally, the animal detection algorithms implement animal detection as a binary pattern classification task. That means, that given an input image, it is divided in blocks and each block is transformed into a feature. Features from the animal that belongs to a certain class are used to train a certain classifier. Then, when given a new input image, the classifier will be able to decide if the sample is the animal or not. The animal recognition system can be divided into the following basic applications:
Identification - compares the given animal image to all the other animals in the database and gives a ranked list of matches (one-to-N matching).
Verification (authentication) - compares the given animal image and involves confirming or denying the identity of found animal (one-to-one matching).

While verification and identification often share the same classification algorithms, both modes target distinct applications. In order to better understand the animal detection and recognition task and its difficulties, the following factors must be taken into account, because they can cause serious performance degradation in animal detection and recognition systems:
Illumination and other image acquisition conditions - the input animal image can be affected by factors such as illumination variations, in its source distribution and intensity or camera features such as sensor response and lenses.
Occlusions - the animal images can be partially occluded by other objects and by other animals.

Many algorithms and methods have been developed by human being in order to have a better understanding on animal behaviour. Understanding the neural mechanisms that control animal behavior often requires monitoring animals during behavioral tasks. To better understand the complexities of natural ecosystems and better manage and protect them, it would be helpful to have detailed, large-scale knowledge about the number, location, and behaviors of animal in natural ecosystems. Human analysis is sometimes both highly labor-intensive and possibly error-prone, making automation very desirable and often critical to achieve acceptable throughput.

Automatic animal identification and counting would improve all biology missions that require identifying species and counting individuals. Here, we harness deep learning, a state-of-the-art machine learning technology that has led to dramatic improvements in artificial intelligence in recent years, especially in computer vision. Identifying animal attributes, analysing their behaviour in the pictures remains an expensive time consuming manual task performed by various researchers. Thus, we demonstrate that such detection of animal can be done by deep convolution neural network. We proposed a novel deep convolutional neural network based species recognition algorithm for wild animal classification on an imagery data.

Deep learning only works well with vast amounts of labeled data, significant computational resources, and modern neural network architectures. Here the combination of dataset with labeled data, modern computing, and stateof-the-art deep neural network architectures is used to test whether deep learning can automate animal identification. The net result is a system that dramatically improves our ability to automatically extract valuable knowledge from nature.

1.2 Task Definition

The basic task is to create an algorithm to classify whether an image contains an animal. The input for this task is images of animals from training dataset, while the output is the classification on test dataset. The classification will show all the classes and the maximum value to a particular class will therefore exhibit that the image belongs to that particular class.

We proposed a novel DCNN based animal recognition algorithm. On the very challenging imagery dataset, our DCNN based species recognition algorithm will be useful for feature extraction as well as classification.

1.3 Motivation

Although the camera traps should only capture animal images, the method generates a lot of false positive captures (images without animals). For instance, the camera trapping study performed by Diaz-Pulido et al. [1] where only 1% of that information was valuable, or the Snapshot Serengeti database [2] where 26.8% of the images contain animals.As a result, wildlife scientists must analyze thousands of photographs, of which a high percentage does not show wildlife. This problem, albeit very well known in the camera-trap community, is

far from being solved. Furthermore, biologists must classify tens of animal species or genera from thousands of images.

Many people has worked or are working on constructing machine learning classifiers to address this problem. In [3], a classifier based on color features got 56.9% accuracy on the Asirra dataset [4]. In [5], an accuracy of 82.7% was achieved from a SVM classifier based on a combination of color and texture features. In our project, the aim is to consume less time for classification and achieve higher performance.

1.4 Organization of Report

Describe every chapter (what every chapter contains)

- **Chp. 1 Introduction:** An overview of deep learning and it's use for animal classification and also description of the task and motivation behind developing the project.
- **Chp. 2 Review of Literature:** Different image classification techniques used in various papers and the methodology for the proposed system.
- **Chp. 3 Design and Implementation:** It introduces the method used for identification of animals and steps taken in each layer of neural network for implementing animal classification.
- **Chp. 4 Simulation, Result and Analysis:** Determining the experimental results and accordingly analyzing the results.
- **Chp. 5 Conclusion and Scope:** It shows how deep learning can thus be useful in identifying wild animals and the future scope.

Chapter 2

Literature Review

Research has been done to find out different methods of reducing the human efforts required for animal recognition in an image.The framework which has been proposed consists of exorbitant camera traps and wavering accuracies. Hence research is still being done in order to recognize the most efficient algorithm that can be used in this system.

2.1 Technical papers review

A number of methods employing different algorithms have previously been implemented in wildlife classification systems. This section reviews previous approaches to identify multiple species in camera-trap images. To the best of our knowledge there are only two previous approaches to identify animal species in camera-trap images.

One of these is the Sparse coding spatial pyramid matching (ScSPM) which was used by Yu et.al [6] to recognize 18 species of animals, reaching 82% of accuracy on their own dataset (composed of 7196 images).

There is use of dense SIFT descriptors extracted by ScSPM and cell-structured local binary patterns as local features; subsequently global features are generated using global weighted sparse coding and max pooling thought multi-scale pyramid kernel. The images are classified using a linear support vector machine. As input to the ScSPM photo-trap images were preprocessed: Removing empty frames (images without animals), manually cropping all the animals from the images, and selecting only those images that captures the animals whole body.

A deep convolutional neural network (ConvNet) was used by Chen et.al [7] to classify 20 animal species in their own dataset. An important difference from [6] is that they use an automatic segmentation method (Ensemble Video Object Cut) for cropping the animals from the images and use this crops to train and test their system. The ConvNet used only has 6 layers (3 convolutional layers and 3 max pooling layers) which give them a 38.31% of accuracy.

Several recent works harnessed deep learning to classify images. Chen et. al. [7] harnessed convolutional deep neural networks (DNNs) to fully automate animal identification. However, they demonstrated the techniques on a dataset of around 20,000 images and 20

classes, which is of much smaller scale than the SS [7]. In addition, they only obtain an accuracy of 38%, which leaves much room for improvement. Interestingly, Chen et al. found that DNNs outperforms a traditional Bag of Words technique if provided sufficient training data [7]. Similarly, Gomez et al. [10] also had success harnessing DNNs to distinguish birds vs. mammals in a small dataset of 1,572 images and distinguish two mammal sets in a dataset of 2,597 images.

In addition, there has been some recent work on re-ranking Google search results using only images [8, 9] and on re-ranking search results using text plus images. However, by focusing on animal categories we are working with much richer, more difficult data, and can show unequivocal benefits from a visual representation. Animals are demonstrably among the most difficult classes to recognize. This is because animals often take on a wide variety of appearances, depictions and aspects. Animals also come with the added challenges of articulated limbs and the fact that multiple species while looking quite different in appearance have the same semantic category label, e.g. African leopards, black leopards, and clouded leopards. There has been some work on recognizing animal categories using deformable models of shape. However, they concentrate on building a single model for appearance and would not be able to handle the large changes in aspect or multiple species that we find in our data.

2.2 Methodology

Our classifier comprises of two stages, training and testing. In the training stage, a set of images are provided as visual examples. In the testing stage, a newly captured image i.e. the test image is given as input to the classifier. With the help of the knowledge gained from training, the test image is accordingly classified into the most favourable class.

A. Receiving the Input Image: In the proposed system, an image is captured using the camera connected to the system. This test image is fed as input which is then converted into a binary pattern. A set of previously labelled images are present in the dataset whose features are matched with those of the test image, in order to determine the species of the animal present.

B. Feature Extraction: The test image which is received as input can be transformed into a reduced set of features. The selected features may contain the significant information from the input data, due to which the desired task can be performed by using this reduced amount of data instead of the initial unaltered data. Human crafted features are a form of fixed features directly extracted from images. Divergent from these, deep neural networks recognize features from images, and determine multiple levels of representation, with higher-level features depicting more abstract characteristics of the data.

C. Classifying the Species present in a Picture: For the task of classification of species, the corresponding output layer generates the probabilities of the animal detected in the image belonging to one of the possible classes. Even though providing such a result would save a huge amount of human effort that will be needed in recognizing the correct species, the testing of this hypothesis will require human knowledge.

2.3 Problem Formulation

Keeping a track of the whereabouts of animals in a wildlife sanctuary currently requires manpower in order to recognize the animal captured in the trap cameras. Human observation is not reliable in several cases such as when the image might be disrupted due to physical factors. This gap in evidence might lead to decrease in the integrity of the result. The proposed system diminishes the need for such manpower and assures the veracious solution to this problem.

2.4 Problem Statement

One main task for extracting information from images is to classify which species of animals are in the image. Here, we focus on this animal species classification task, which is challenging even for humans. Images taken from camera may or may not be perfect, and many images may contain animals that are far away, too close, or only partially visible. In addition, different lighting conditions, shadows, and weather can make the identification task even harder. Taking such factors in to consideration, we demonstrate that wild animals detection can be done using deep convolutional neural networks.

2.5 Proposed System

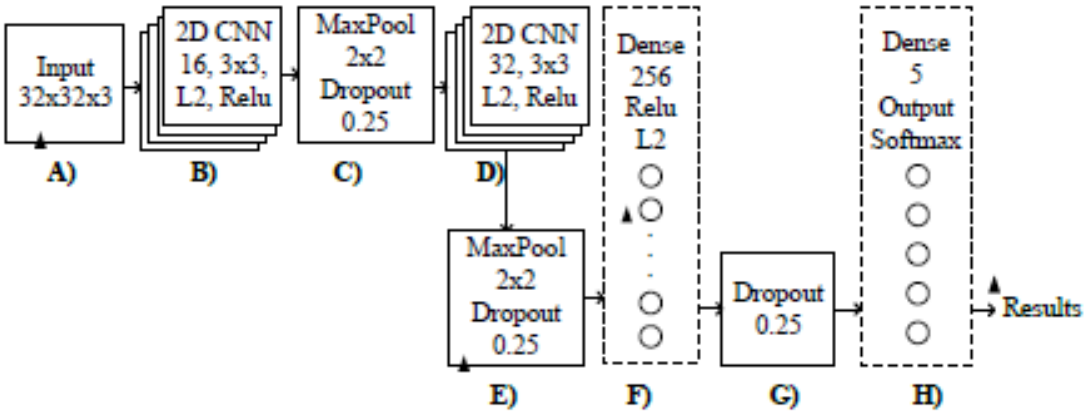


Figure 2.1: Proposed CNN Architecture

Chapter 3

Design & Implementation

In previous chapter when you finished literature to understand methods.Their may be many methods to develop your proposed system.But one which you chose to implement that need to be design in this chapter.

3.1 Design

3.1.1 System Architecture

The system will consist of a camera which will be connected to a computer system and the scenes captured by the camera will be monitored through it. The image captured by the camera will be fed as input to the system. The system will first detect if an animal is present in the region of interest, and then recognition of the species will be done with the help of a Deep Convolutional Neural Network (DCNN) based image classification algorithm and a large dataset, consisting of the training and testing images of various animals. The system will be trained with the help of the dataset as well as the new images being fed to the system for recognition.

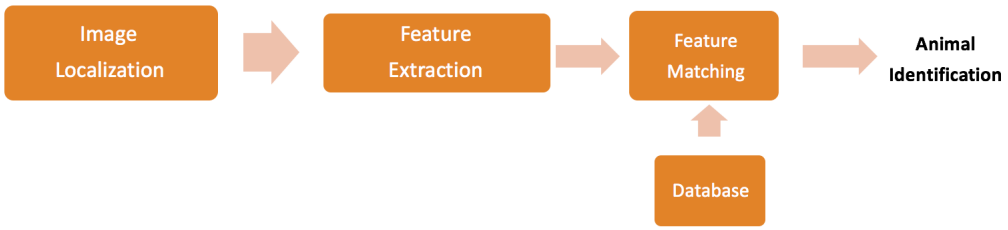


Figure 3.1: System Architecture

3.1.2 Convolutional Neural Networks

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal

preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

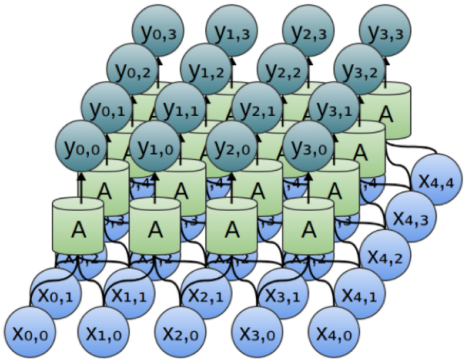


Figure 3.2: Procedure of Convolutional Neural Networks

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

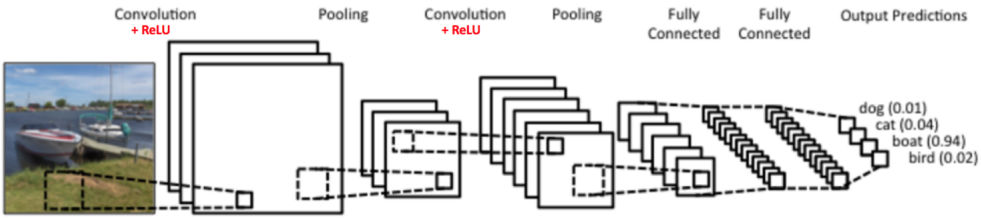


Figure 3.3: Convolutional Neural Networks

The Convolutional Neural Network in Figure is similar in architecture to the original LeNet and classifies an input image into four categories: dog, cat, boat or bird (the original LeNet was used mainly for character recognition tasks). As evident from the figure above, on receiving a boat image as input, the network correctly assigns the highest probability for boat (0.94) among all four categories. The sum of all probabilities in the output layer should be one.

CNN compares any image piece by piece and the pieces that it looks for in an image while detection are called as features. CNN gets trained by finding approximate feature matches in

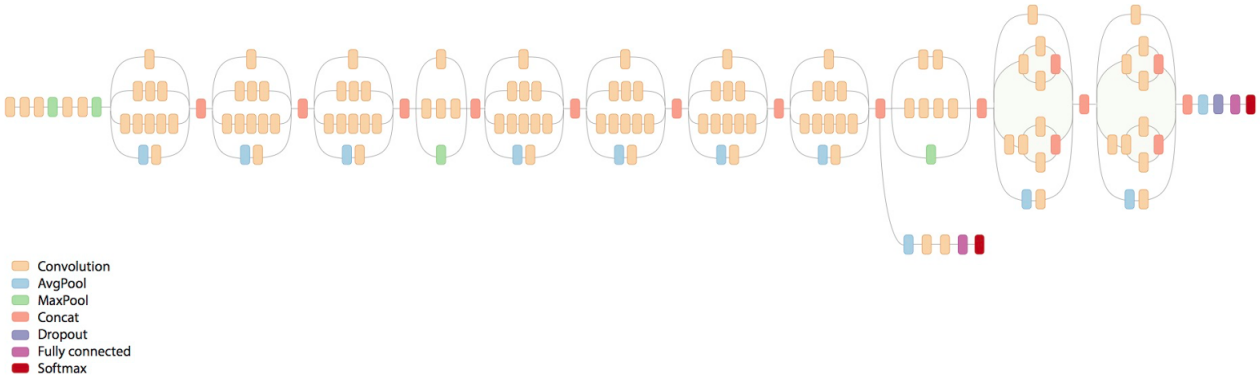


Figure 3.4: Convolutional Neural Networks with multiple layers

the same position in two different images. Every neuron in CNN will be connected to small region of neurons below it, this would allow handling less amount of weights and number of neurons required will also be less.

There are four main operations in the ConvNet shown in Figure above:

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

These operations are the basic building blocks of every Convolutional Neural Network, so understanding how these work is an important step to developing a sound understanding of ConvNets.

Convolution Layer

The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.

Overview and intuition: The CONV layers parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position.

As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position.

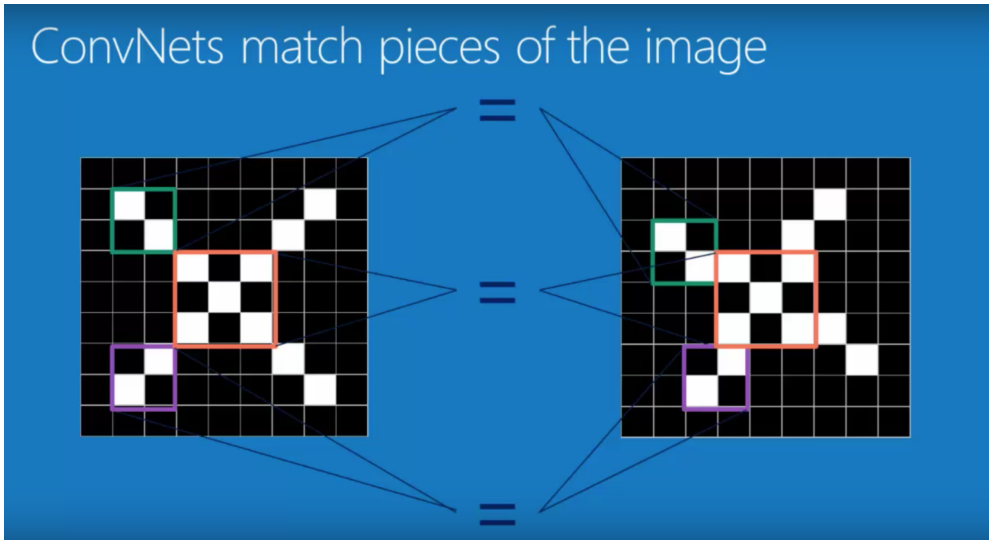


Figure 3.5: Principle of Convolutional Neural Network

Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

Every entry in the 3D output volume can also be interpreted as an output of a neuron that looks at only a small region in the input and shares parameters with all neurons to the left and right spatially (since these numbers all result from applying the same filter).

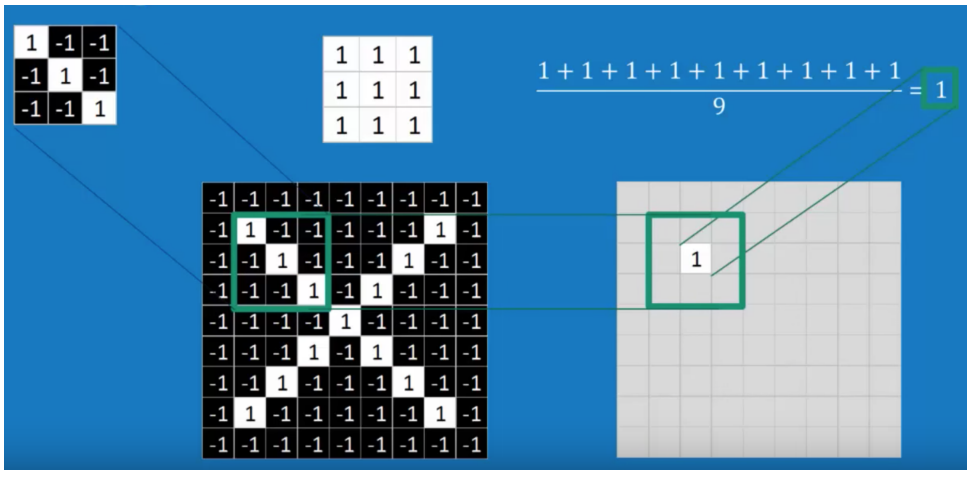


Figure 3.6: Convolving the filter with image matrix to obtain final value in feature map

Local Connectivity When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent

of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size).

The extent of the connectivity along the depth axis is always equal to the depth of the input volume. It is important to emphasize again this asymmetry in how we treat the spatial dimensions (width and height) and the depth dimension: The connections are local in space (along width and height), but always full along the entire depth of the input volume.

The size of the Feature Map (Convolved Feature) is controlled by three parameters [4] that we need to decide before the convolution step is performed:

Depth: Depth corresponds to the number of filters we use for the convolution operation. In the network shown in Figure , we are performing convolution of the original boat image using three distinct filters, thus producing three different feature maps as shown. You can think of these three feature maps as stacked 2d matrices, so, the depth of the feature map would be three.

Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution.

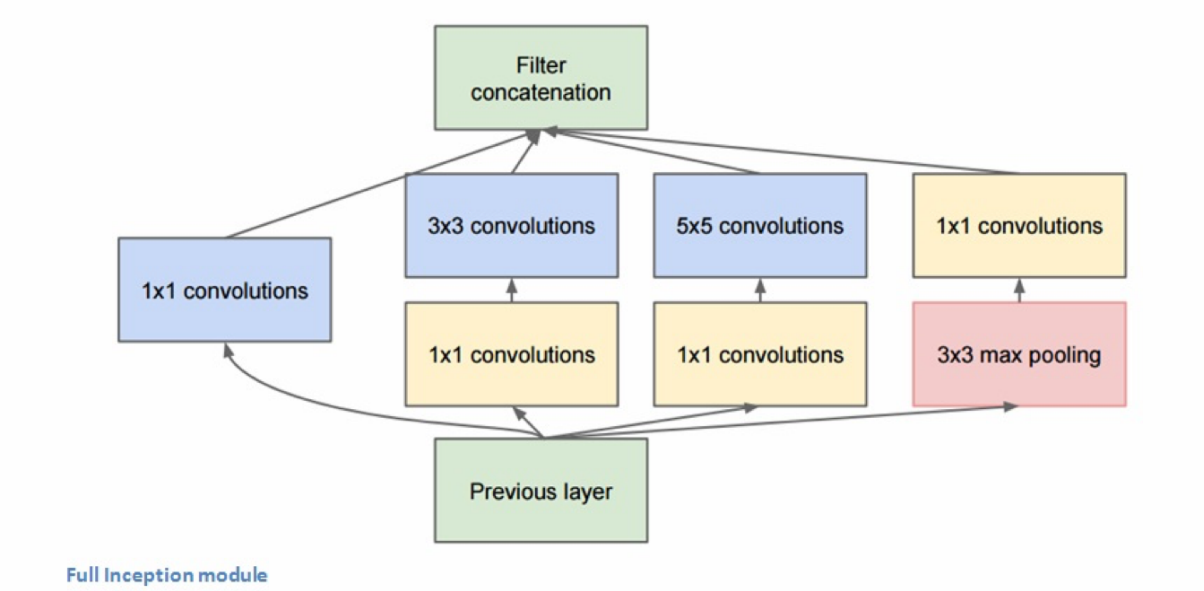


Figure 3.7: Convolutional Neural Networks

A filter slides over the input image (convolution operation) to produce a feature map. The convolution of another filter, over the same image gives a different feature map. It is important to note that the Convolution operation captures the local dependencies in the original image. Two different filters generate different feature maps from the same original image. The image and the two filters above are just numeric matrices.

In practice, a CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process). The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

Introducing Non Linearity (Rectified Linear Unit)

The rectifier function is an activation function f(x) = Max(0, x) which can be used by neurons just like any other activation function, a node using the rectifier activation function is called a ReLu node. The main reason that it is used is because of how efficiently it can be computed compared to more conventional activation functions like the sigmoid and hyperbolic tangent, without making a significant difference to generalisation accuracy. The rectifier activation function is used instead of a linear activation function to add non linearity to the network, otherwise the network would only ever be able to compute a linear function. An additional operation called ReLU has been used after every Convolution operation. ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

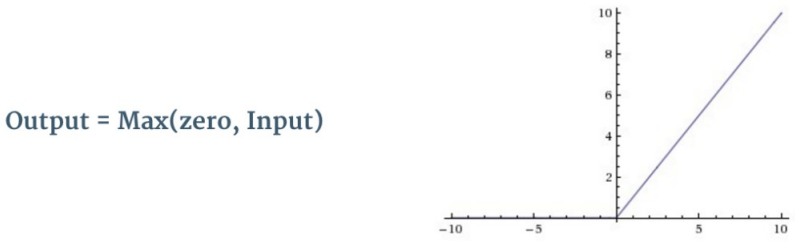


Figure 3.8: The ReLU operation

We can compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border. You can convince yourself that the correct formula for calculating how many neurons fit is given by (WF+2P)/S+1. For example for a 7x7 input and a 3x3 filter with stride 1 and pad 0 we would get a 5x5 output. With stride 2 we would get a 3x3 output.

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

The ReLU operation can be understood clearly from Figure below. It shows the ReLU operation applied to one of the feature maps obtained in Figure 6 above. The output feature map here is also referred to as the Rectified feature map.

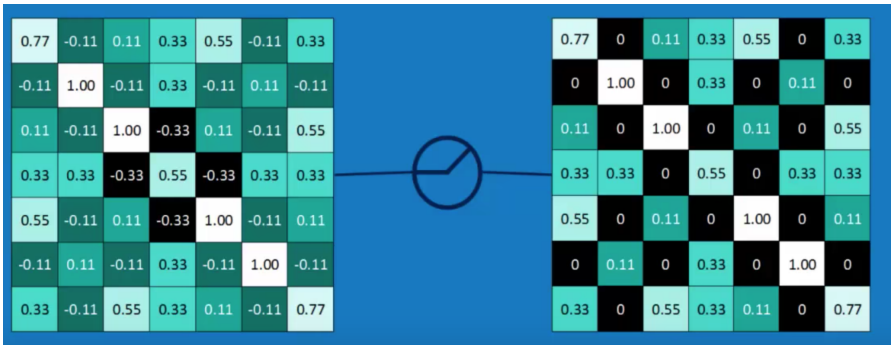


Figure 3.9: ReLU operation working

Other non linear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Pooling Layer

Sometimes when the images are too large, we would need to reduce the number of trainable parameters. It is then desired to periodically introduce pooling layers between subsequent convolution layers. Pooling is done for the sole purpose of reducing the spatial size of the image. Pooling is done independently on each depth dimension, therefore the depth of the image remains unchanged. The most common form of pooling layer generally applied is the max pooling.

In case of Max Pooling, we define a spatial neighborhood (for example, a 22 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

Accepts a volume of size W1H1D1 Requires two hyperparameters: their spatial extent F, the stride S, Produces a volume of size W2H2D2 where: W2=(W1F)/S+1 H2=(H1F)/S+1 D2=D1

Introduces zero parameters since it computes a fixed function of the input It is not common to use zero-padding for Pooling layers. It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: A pooling layer with F=3,S=2 (also called overlapping pooling), and more commonly F=2,S=2. Pooling sizes with larger receptive fields are too destructive.

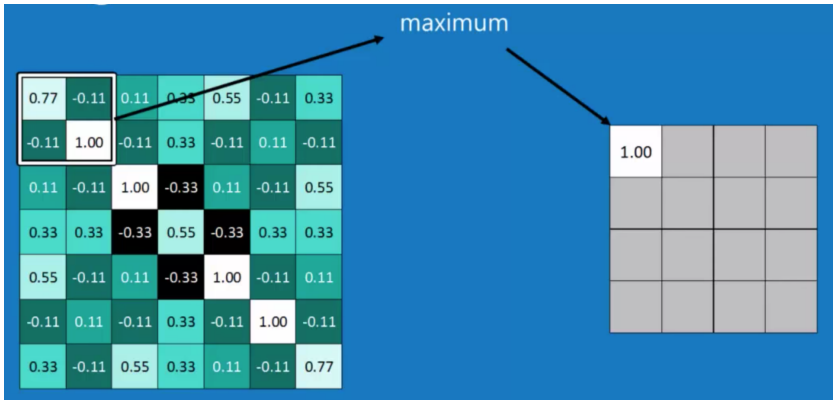


Figure 3.10: Max Pooling Technique

General pooling

In addition to max pooling, the pooling units can also perform other functions, such as average pooling or even L2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice.

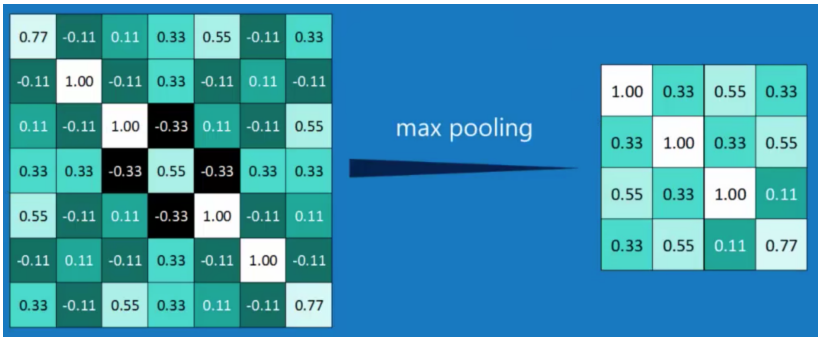


Figure 3.11: Max Pooling Technique applied to Rectified Feature Maps

The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling

1. Makes the input representations (feature dimension) smaller and more manageable
2. Reduces the number of parameters and computations in the network, therefore, controlling overfitting
3. Makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling since we take the maximum / average value in a local neighborhood).

4. Helps us arrive at an almost scale invariant representation of our image (the exact term is equivariant). This is very powerful since we can detect objects in an image no matter where they are located.

Fully Connected Layer

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term Fully Connected implies that every neuron in the previous layer is connected to every neuron on the next layer.

After multiple layers of convolution and padding, we would need the output in the form of a class. The convolution and pooling layers would only be able to extract features and reduce the number of parameters from the original images. However, to generate the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need. It becomes tough to reach that number with just the convolution layers. Convolution layers generate 3D activation maps while we just need the output as whether or not an image belongs to a particular class. The output layer has a loss function like categorical cross-entropy, to compute the error in prediction. Once the forward pass is complete the backpropagation begins to update the weight and biases for error and loss reduction.

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

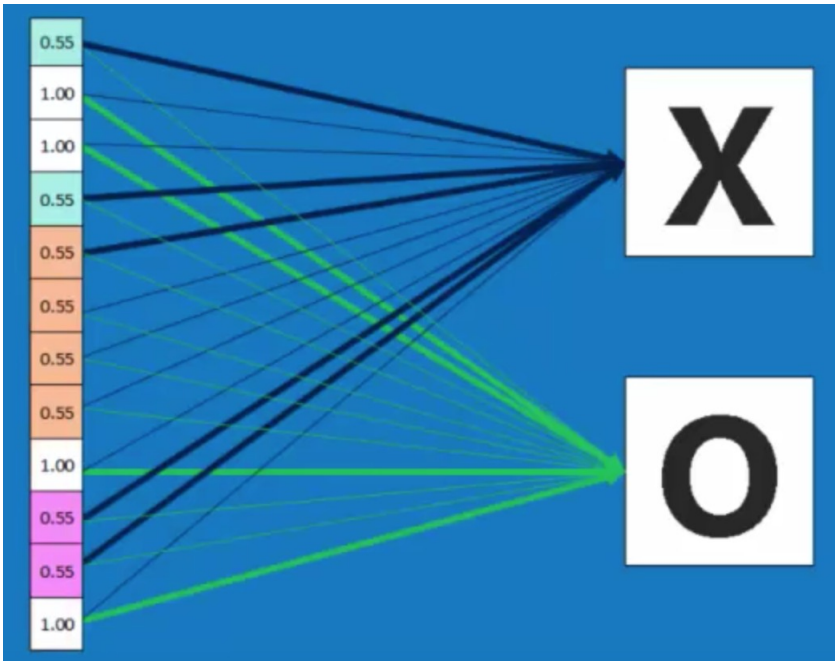


Figure 3.12: Fully connected layer

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. For example, the image classification task we set out to perform has four possible outputs

Apart from classification, adding a fully-connected layer is also a (usually) cheap way of learning non-linear combinations of these features. Most of the features from convolutional and pooling layers may be good for the classification task, but combinations of those features might be even better.

The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one.
The overall training process of the Convolution Network may be summarized as below:

- **Step 1**: We initialize all filters and parameters / weights with random values
- **Step 2**: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class. Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3] Since weights are randomly assigned for the first training example, output probabilities are also random.
- **Step 3**: Calculate the total error at the output layer (summation over all 4 classes)
  Total Error = (target probability output probability)
- **Step 4**: Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error. The weights are adjusted in proportion to their contribution to the total error. When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0]. This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced. Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process only the values of the filter matrix and connection weights get updated.
- **Step 5**: Repeat steps 2-4 with all images in the training set. The above steps train the ConvNet this essentially means that all the weights and parameters of the ConvNet have now been optimized to correctly classify images from the training set.

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples). If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.

Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

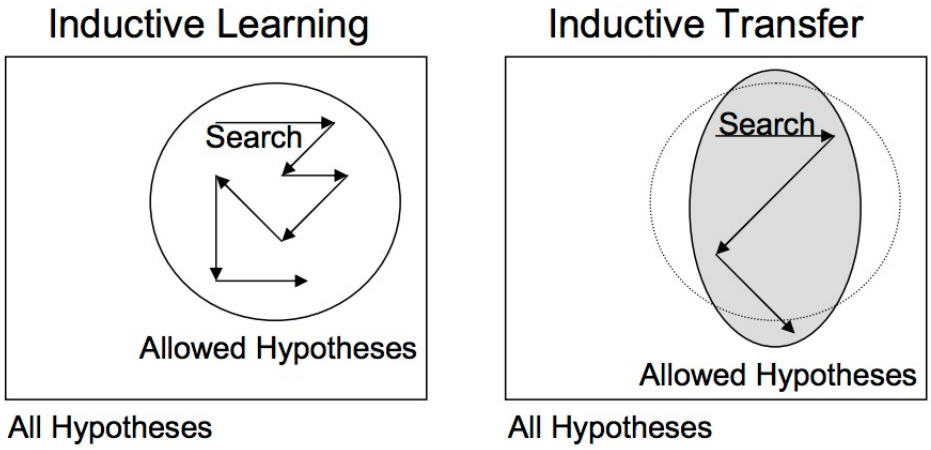


Figure 3.13: Inductive Learning vs. Inductive Transfer

Pre-trained Model Approach

1. **Select Source Model:** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

2. **Reuse Model:** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

3. **Tune Model:** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

Transfer Learning with Image Data

It is common to perform transfer learning with predictive modeling problems that use image data as input. This may be a prediction task that takes photographs or video data as input.

For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.

The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware.

These models can be downloaded and incorporated directly into new models that expect image data as input.

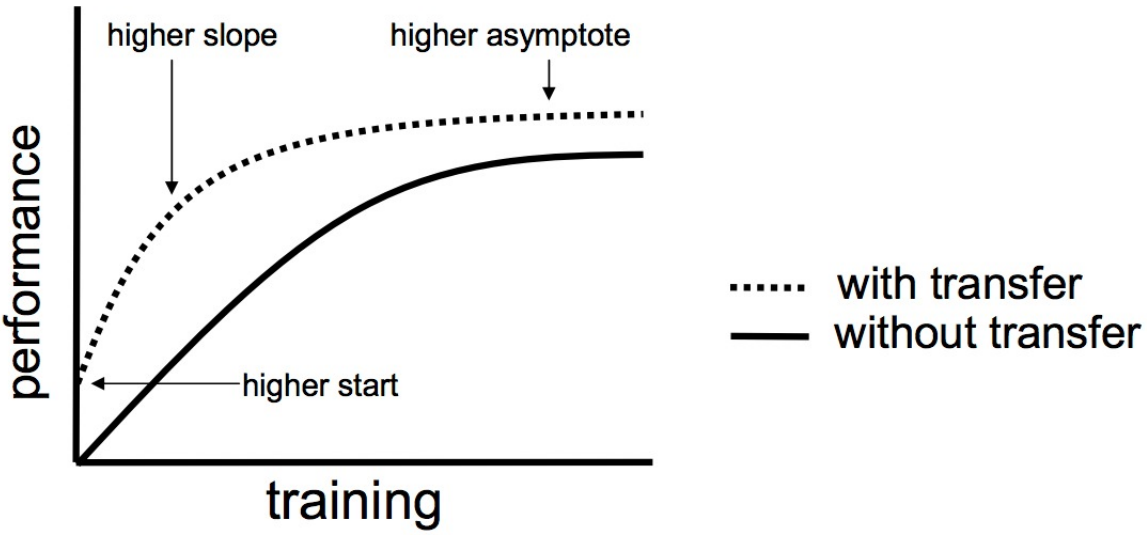


Figure 3.14: Variation in Training

Inception model

This model achieves 5.64% top-5 error while an ensemble of four of these models achieves 3.58% top-5 error on the validation set of the ImageNet whole image ILSVRC 2012 classification task.

A system for training an Inception-v3 model provides many opportunities, including: Exploration of different variants of this model architecture in order to improve the image classification system.

Comparison of optimization algorithms and hardware setups for training this model faster or to a higher degree of predictive performance.

Retraining/fine-tuning the Inception-v3 model on a distinct image classification task or as a component of a larger network tasked with object detection or multi-modal learning. The last topic is often referred to as transfer learning, and has been an area of particular excitement in the field of deep networks in the context of vision. A common prescription to a computer vision problem is to first train an image classification model with the ImageNet Challenge data set, and then transfer this model's knowledge to a distinct task. This has been done for object detection, zero-shot learning, image captioning, video analysis and multitudes of other applications.

RCNN

Region Based CNNs

Some may argue that the advent of R-CNNs has been more impactful that any of the previous papers on new network architectures. With the first R-CNN paper being cited over 1600 times, Ross Girshick and his group at UC Berkeley created one of the most impactful advancements in computer vision. As evident by their titles, Fast R-CNN and Faster R-CNN worked to make the model faster and better suited for modern object detection tasks.

The purpose of R-CNNs is to solve the problem of object detection. Given a certain image, we want to be able to draw bounding boxes over all of the objects. The process can be split into two general components, the region proposal step and the classification step.

The authors note that any class agnostic region proposal method should fit. Selective Search is used in particular for RCNN. Selective Search performs the function of generating 2000 different regions that have the highest probability of containing an object. After we've come up with a set of region proposals, these proposals are then resolved into an image size that can be fed into a trained CNN (AlexNet in this case) that extracts a feature vector for each region. This vector is then used as the input to a set of linear SVMs that are trained for each class and output a classification. The vector also gets fed into a bounding box regressor to obtain the most accurate coordinates.

Faster R-CNN

Faster R-CNN works to combat the somewhat complex training pipeline that both R-CNN and Fast R-CNN exhibited. The authors insert a region proposal network (RPN) after

the last convolutional layer. This network is able to just look at the last convolutional feature map and produce region proposals from that. From that stage, the same pipeline as R-CNN is used (ROI pooling, FC, and then classification and regression heads).

3.2 Implementation

3.2.1 System Requirements

These prerequisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements.

3.2.2 Hardware Requirements

- **Processing device (Computer with GPU GTX NVIDIA 970):** Deep learning is a field with intense computational requirements and the choice of the GPU will fundamentally determine the deep learning experience. Powered by NVIDIA Maxwell architecture, these graphics cards delivers incredible performance, unmatched power efficiency, and cutting-edge features.

- **Canon Digital Camera SX510 HS:** The Canon PowerShot SX510 HS is a superzoom digital compact camera featuring a 30x optical zoom, 12.1 megapixel CMOS sensor, DIGIC 4 processor, optical image stabiliser, FullHD video, wifi connectivity and GPS location tagging.

3.2.3 Software Requirements

- **Windows 10, Windows 8:** It is a personal computer operating system that was produced by Microsoft as a part of Windows NT family of operating system. Its updated methods are Windows Update, Windows server update services, system center configuration.

- **Python (Recognition)**: Top deep learning libraries are available on the Python ecosystem like Theano and TensorFlow. Python is a fully featured general purpose programming language, unlike R and Matlab. It is also quick and easy to write and understand, unlike C++ and Java. Theano and TensorFlow are two top numerical libraries for developing deep learning models, but are too technical and complex for the average practitioner. They are intended more for research and development teams and academics interested in developing wholly new deep learning algorithms.

- **TensorFlow Libraries**: TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

- **Kaggle Dataset For Training Images**: Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know beforehand which technique or analyst will be most effective.

3.2.4 TensorFlow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

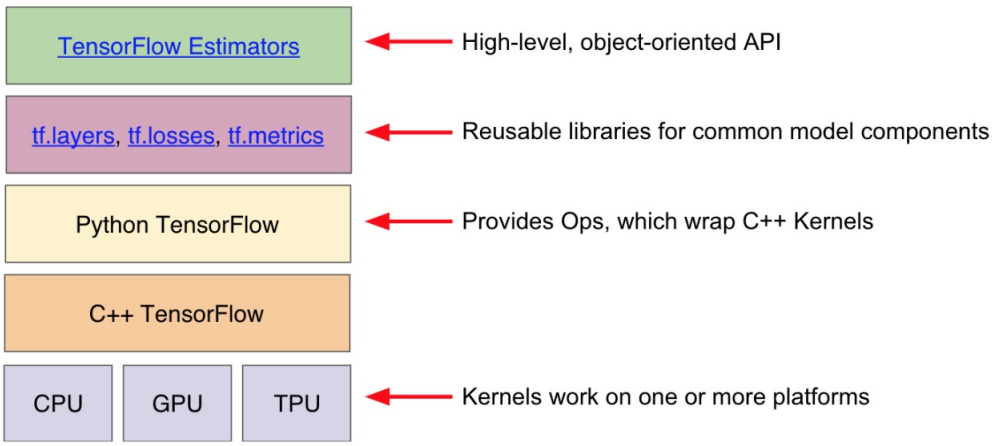


Figure 3.15: TensorFlow Toolkit Hierarchy

Tensorflow is a computational framework for building machine learning models. TensorFlow provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. You can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, you can use higher-level APIs (like tf.estimator) to specify predefined architectures, such as linear regressors or neural networks.

TensorFlow consists of the following two components:

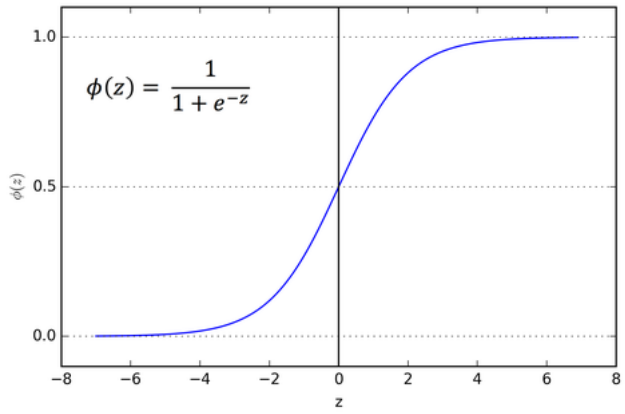
- **A graph protocol buffer**
- **A runtime that executes the (distributed) graph**

These two components are analogous to Python code and the Python interpreter. Just as the Python interpreter is implemented on multiple hardware platforms to run Python code, TensorFlow can run the graph on multiple hardware platforms, including CPU, GPU, and TPU.

One should use the highest level of abstraction that solves the problem. The higher levels of abstraction are easier to use, but are also (by design) less flexible. We recommend you start with the highest-level API first and get everything working. If you need additional flexibility for some special modeling concerns, move one level lower. Note that each level is built using the APIs in lower levels, so dropping down the hierarchy should be reasonably straightforward.

Softmax Function

In probability theory, the output of the softmax function can be used to represent a categorical distribution that is, a probability distribution over K different possible outcomes. In fact, it is the gradient-log-normalizer of the categorical probability distribution.



The softmax function is used in various multiclass classification methods, such as multinomial logistic regression (also known as softmax regression):206209 , multiclass linear discriminant analysis, naive Bayes classifiers, and artificial neural networks. Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability.

Chapter 4

Results & Analysis

4.1 Results

The results in this system are returned to the user in the form of probabilities. The animal which has high similitude with the animal in input image will exhibit highest probability.

- Capturing the input image: The image that is captured on the camera by the user will be uploaded to the cloud storage via an active internet connection. This image is analogized to the huge number of images stored in the system. Figure 4.1 depicts the same.

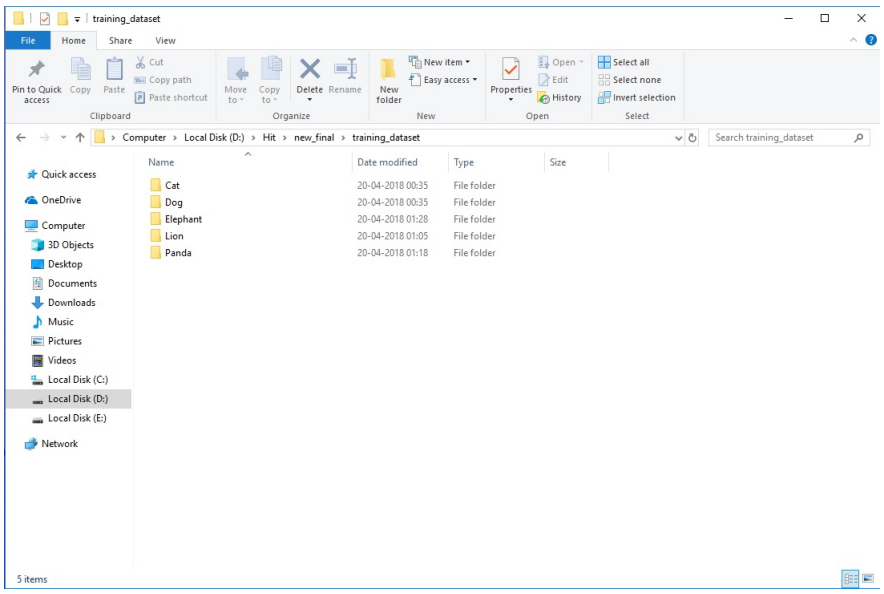


Figure 4.1: Training Dataset

- When the image classification algorithm is run, a number of features are extracted from the image and compared with the features of images belonging to each class. Figure 4.2 represents the images belonging to the class "cat".

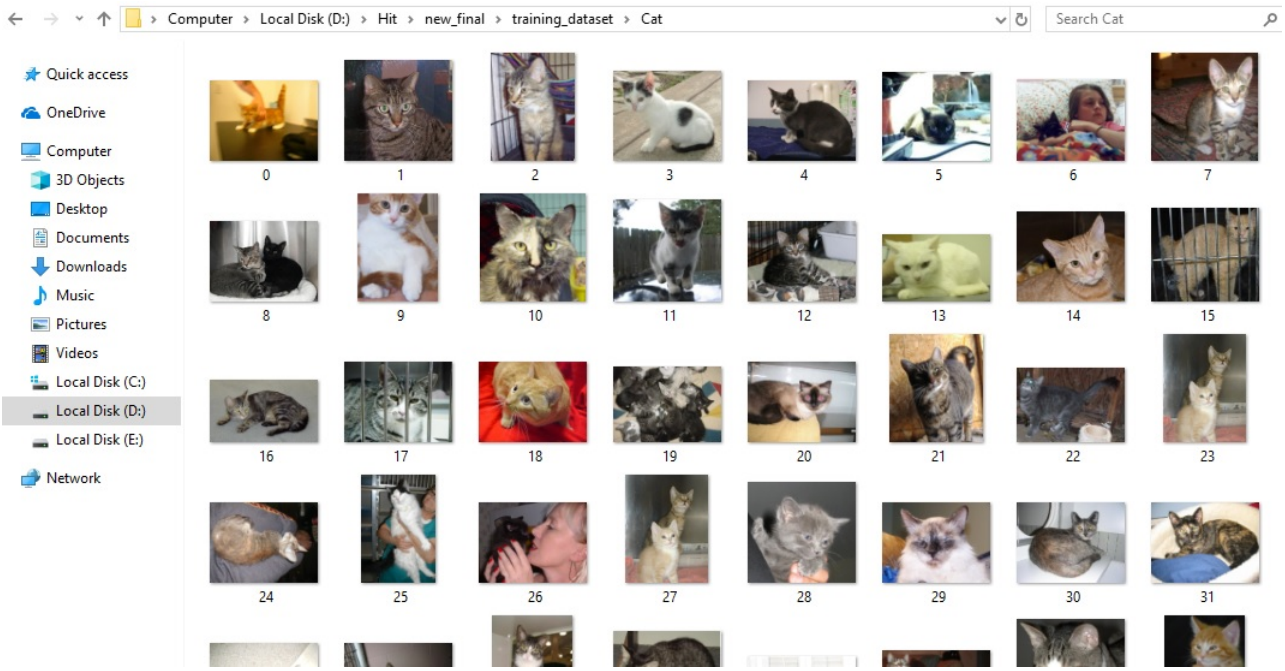


Figure 4.2: Training Images For Cat Class

Figure 4.3: Input Image From Camera

- Agent captures an input image using the camera and uploads it to the system.The output of the classification appears in the form of probabilities as shown below:

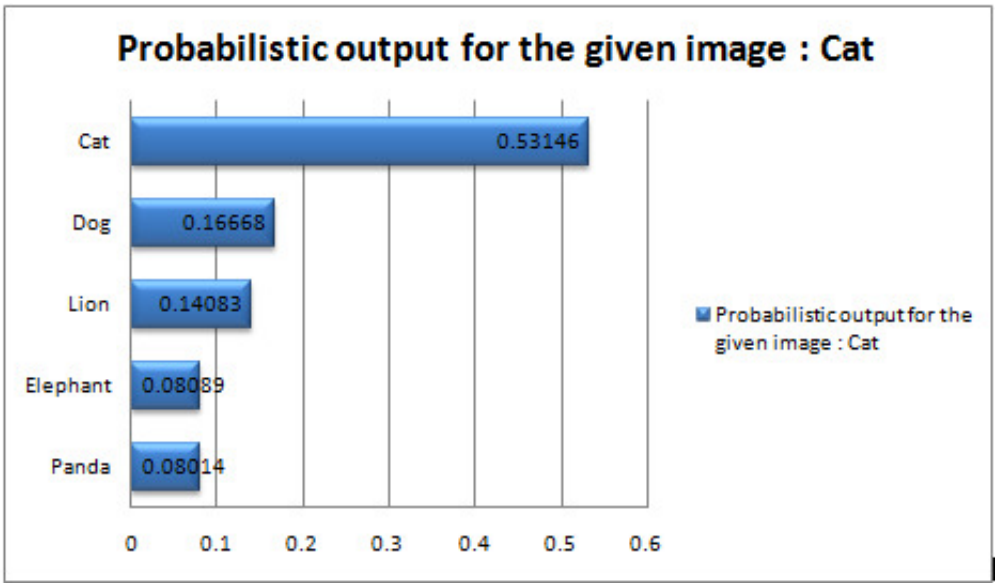


Figure 4.4: Probabilistic output for a sample image

4.2 Analysis

The training accuracy will grow as we increase the number of iterations while training the system with the training dataset.The number of iterations(n) can have any numeric value.On increasing the value of n, more features will be extracted from the dataset and thus, accuracy will increase.

No. of Iterations	Train Accuracy (%)	Validation Accuracy (%)	Final Test Accuracy (%)
1	53.0	37.0	59.0
2	80.0	64.0	69.0
3	53.0	37.0	59.0
4	82.0	79.0	82.3
5	92.8	82.0	83.9
6	97.0	83.0	82.3
7	66.8	91.8	89.5
8	75.0	72.0	86.2
9	85.0	65.0	93.0
10	97.0	90.0	95.2

Table 4.1: Accuracy for each iteration

We have set the number of iterations to 15 due to the limitation of gpu memory as it is dependent completely on the graphics processor (GTX 960).The statistical data observed

from the result is mentioned below.

Statistics obtained after the 1st iteration:

- Train accuracy = 53.0%
- Cross entropy = 1.475672
- Validation accuracy = 37.0% (N=100)

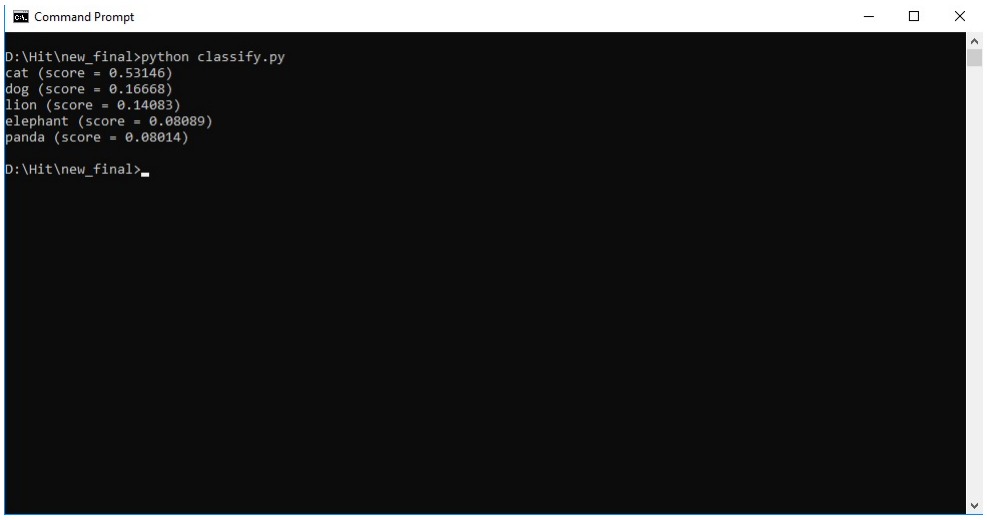


Figure 4.5: Classification Output

Statistics obtained after the 15th iteration:

- Train accuracy = 97.0%
- Cross entropy = 0.812473
- Validation accuracy = 99.0% (N=100)

We can clearly see that when the number of iterations is increased we get a better accuracy for training as well as validation.

Chapter 5

Conclusions and Future Enhancement

Thus, a proposed CNN for the image recognition, feature extraction and image classification was presented . The proposed CNN was evaluated on the created animal database. The overall performances were obtained using different number of training images and test images. The training was done with the help of Back-propagation algorithm. The best experimental results of animal recognition were obtained using the proposed CNN.

The obtained experimental results of the performed experiments show that the proposed CNN gives the best recognition rate for a greater number of input training images. Perhaps most importantly, our results show that employing deep learning technology can save a tremendous amount of time for researchers in biology and the human volunteers that help them by labeling images.

When the image is divided into more windows the classification results should be better. On the other hand, the computation complexity will increase. Automating animal identification can thus dramatically reduce the cost to extract informative and actionable information from wild habitats, potentially revolutionizing studies of animal behavior, ecosystem dynamics, and wildlife conservation.

5.1 Future Enhancement

There are quite a few things that can be polished or add in the future work.

- In the future work, we plan to investigate reliability of the presented methods by involving larger databases of animal images.
- We would expand the dataset to double the number of classes iteratively, watching the performance at each level, until reaching the entire 50 animal class breadth of the dataset.
- Future works can also include experiments with this method on other animal databases.

Bibliography

[1] Anglica Diaz-Pulido and E Payan. Densidad de ocelotes (leopardus pardalis) en losllanos colombianos. Mastozoologa neotropical, 18(1):6371, 2011.

[2] Alexandra Swanson, Margaret Kosmala, Chris Lintott, Robert Simpson, Arfon Smith,and Craig Packer. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. Scientific data, 2:150026, 2015.

[3] J. Elson, J. Douceur, J. Howell and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. Proc. of ACM CCS 2007, pp. 366-374.

[4] MSR Asirra: http://research.microsoft.com/en-us/um/redmond/projects/asirra/

[5] Golle, P. (2008, October). Machine learning attacks against the Asirra CAPTCHA. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 535-542). ACM.

[6] Yu, X., Wang, J., Kays, R., Jansen, P.A., Wang, T., Huang, T.: Automated identification of animal species in camera trap images. EURASIP Journal on Image and Video Processing 2013 (1) (2013) 110

[7] Chen, G., Han, T.X., He, Z., Kays, R., Forrester, T.: Deep convolutional neural network based species recognition for wild animal monitoring. In: 2014 IEEE International Conference on Image Processing (ICIP), IEEE (2014)

[8] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In ECCV, May 2004.

[9] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In ICCV, Oct. 2005.

[10] Gomez A, Salazar A (2016) Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks.

Publication

Paper entitled "Automatically Identifying Animals Using Deep Learning" is presented at *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE - 2018)* in Volume 6, Issue 3, March 2018.

Annexure

1. Paper entitled "Automatically Identifying Animals Using Deep Learning" published in the journal *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE - 2018)*, Volume 6, Issue 3, March 2018.

=210mm =297mm

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

Automatically Identifying Animals Using Deep Learning

Parinita Badre [1], Siddhant Bandiwadekar [2], Prachi Chandanshive[3], Aakanksha Chaudhari[4], Mrs. Sonali Jadhav [5]

Bachelor of Engineering, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[1-4]

Assistant Professor, Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India[5]

ABSTRACT: Having accurate, detailed, and up-to-date information about wildlife location and behavior across broad geographic areas would revolutionize our ability to study, conserve, and manage species and ecosystems. Currently, such data are mostly gathered manually at great expense, and thus are sparsely and infrequently collected. Here we investigate the ability to automatically collect such data, which could transform many fields of biology, ecology, and zoology into big data sciences. In areas like an airport or the agricultural areas placed near the forest many animals destroy the crops or even attack on people therefore there is a need of system which detects the animal presence and gives warning about that in the view of safety purpose. In this project, we demonstrate that such data can be automatically extracted by deep neural networks (deep learning), which is a cutting-edge type of artificial intelligence. Thus, the aim is to train neural networks that automatically identifies animals.

KEYWORDS: Deep learning; Back-Propagation algorithm; Inception model

I. INTRODUCTION

Monitoring biodiversity especially the effects of climate and land use change on wild populations is a critical challenge for our society. Having an updated knowledge about wildlife behaviour would impact our ability to study and manage species and our ecosystem. Researches regarding animals in image processing have been an important field to numerous applications. Many algorithms and methods have been developed by human being in order to have a better understanding on animal behaviour. Identifying animal attributes, analysing their behaviour in the pictures remains an expensive time consuming manual task performed by various researchers. Thus, we demonstrate that such detection of animal can be done by deep convolution neural network.

II. RELATED WORK

Authors in [8], [5] designed a system, which uses web cameras which are to be placed in the detecting areas from where the animal may cross their boundary. The videos are sent to the processing unit and then uses image mining algorithm, which identifies the change in set reference background. If there is a change in the newly acquired image, then authors applied content-based retrieval algorithm (CBIR) to identify the animal. The proposed method in [8] based on CBIR algorithm suffers from many issues like unsatisfactory querying performance-CBIR systems uses distance functions to calculate the dissimilarity between a search image and database images resulting in low-quality recovery results. This approach is very slow and response times in the range of minutes may take place if size of database is too large. To find the accurate location of fishes in the marine, researchers [9] aimed a technique using LIDAR (light detection and ranging). Some of the above-specified methods have been discussed in [10] also.

Researchers in [11] tried to discover an animal's presence in the scene (image) affecting the power spectrum of the image. This method of animal detection was also considered not appropriate since quicker results with this approach would involve the massive amount of image processing in a short period.

 DOI: 10.15680/IJIRCCE.2018. 0603001

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

III. METHODOLOGY

Our classifier comprises of two stages, training and testing. In the training stage, a set of images are provided as visual examples. In the testing stage, a newly captured image i.e. the test image is given as input to the classifier. With the help of the knowledge gained from training, the test image is accordingly classified into the most favorable class.

A. *Receiving the Input Image*

In the proposed system, an image is captured using the camera connected to the system. This test image is fed as input which is then converted into a binary pattern. A set of previously labelled images are present in the dataset whose features are matched with those of the test image, in order to determine the species of the animal present.

B. *Feature Extraction*

The test image which is received as input can be transformed into a reduced set of features. The selected features may contain the significant information from the input data, due to which the desired task can be performed by using this reduced amount of data instead of the initial unaltered data. Human crafted features are a form of fixed features directly extracted from images. Divergent from these, deep neural networks recognize features from images, and determine multiple levels of representation, with higher-level features depicting more abstract characteristics of the data.

C. *Classifying the Species present in a Picture*

For the task of classification of species, the corresponding output layer generates the probabilities of the animal detected in the image belonging to one of the possible classes. Even though providing such a result would save a huge amount of human effort that will be needed in recognizing the correct species, the testing of this hypothesis will require human knowledge.

IV. PROPOSED ALGORITHM

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. [1] CNN compares any image piece by piece and the pieces that it looks for in an image while detection are called as features. CNN gets trained by finding approximate feature matches in the same position in two different images. Every neuron in CNN will be connected to small region of neurons below it, this would allow handling less amount of weights and number of neurons required will also be less.

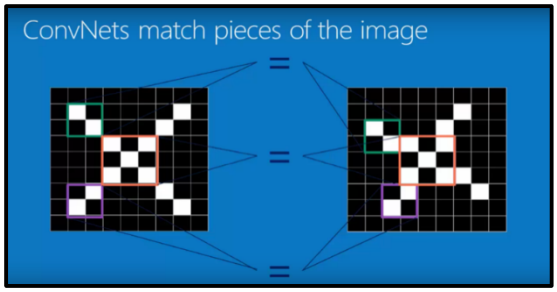


Fig. 1: Principle of Convolutional Neural Network

A. *Convolution Layer*

The Convolution layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. [2] It preserves spatial relationship between pixels thereby extracting and learning features out of them. It captures the local dependencies in the original image. The image is represented as a matrix and a filter, which is also a matrix is used to obtain the convolved feature map or activation map by sliding the feature matrix over the image matrix. We can perform numerous operations just by changing values in the filter matrix.

 DOI: 10.15680/IJIRCCE.2018. 0603001

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

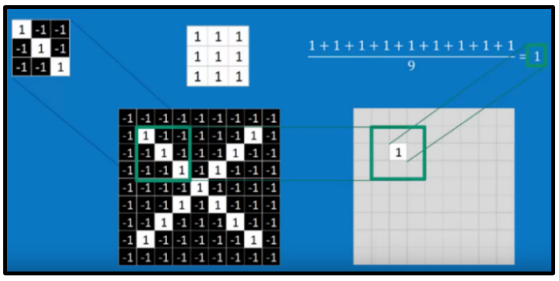


Fig. 2: Convolving the filter with image matrix to obtain final value in feature map

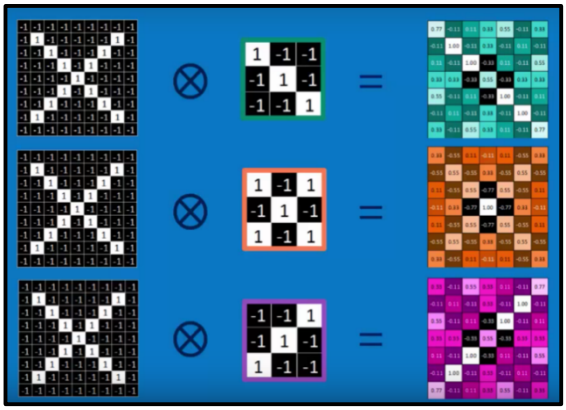


Fig. 3: Convolving each filter with all image matrices to obtain various feature maps

B. *Rectified Linear Unit Layer*

It is called as rectified linear unit(ReLU), which is an activation function that activates a node if an input is above a certain quantity(threshold); while if the input is 0 then the output will be 0. But, if the input is above certain threshold it has a linear relationship with the dependent variable.

Fig. 4: ReLU Operation to obtain rectified feature maps

It replaces all the negative values in the feature map by zero and generates a rectified feature map as shown in Fig. 4. ReLU introduces non-linearity in the ConvNet since most of the image data are non-linear in nature in the real world.

C. *Pooling Layer*

In this layer, we reduce the dimensionality of the feature map to get smaller or shrinked maps that would reduce the parameters and computations. Pooling can be Max, Average or Sum Pooling from the rectified and downsized feature map. Number of filters in convolution layer is same as the number of output maps from pooling as shown in Fig. 3. It also makes the network invariant to small transformations, distortions and translations in the input image.

 DOI: 10.15680/IJIRCCE.2018. 0603001

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

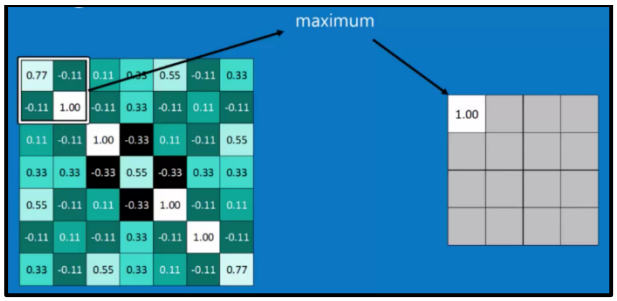


Fig. 5: Max Pooling technique

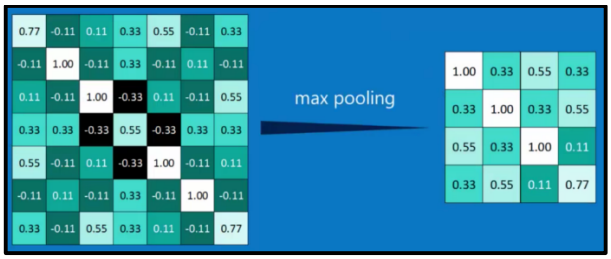


Fig. 6: Max Pooling applied to Rectified Feature Maps

D. *Fully Connected Layer*

This is the final layer where the actual classification occurs where we take our downsized or shrinked images obtained after processing through convolution, ReLU and pooling layer and put them into single list or a vector.

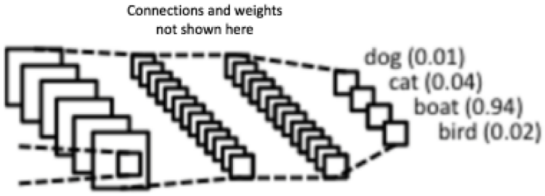


Fig. 7: Fully Connected Layer

It is a traditional Multi-layer perceptron uses a softmax activation function. Convolution and pooling layers generate high-level features. The purpose of the fully connected layers is to use these features to classify the data into various classes based on the dataset.

E. *Training of CNN using Back-Propagation*

Initialize all filters and parameters with random variables

Take input images for training, go through the forward propagation and find output probability for each class.

Calculate total error by the formula:

Total Error = $\sum$ ½ (target probability – output probability) ² (1)

Calculate gradients of the error with respect to the weights and use gradient descent to update the filter values and parameters to minimize the output error.

Repeat above steps for all images in the training set.

F. *Inception Model*

Very deep convolutional networks have been central to the largest advances in image recognition performance in recent years. One example is the Inception architecture that has been shown to achieve very good performance at relatively low computational cost

 DOI: 10.15680/IJIRCCE.2018. 0603001

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

The Inception deep convolutional architecture was introduced as GoogLeNet in (Szegedy et al. 2015a), here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Ioffe and Szegedy 2015) (Inception-v2). Later by additional factorization ideas in the third iteration which is referred to as Inception-v3.

We can train a model from scratch to its best performance on a desktop with 8 NVIDIA Tesla K40s in about 2 weeks. In order to make research progress faster, we are additionally supplying a new version of a pre-trained Inception-v3 model that is ready to be fine-tuned or adapted to a new task.

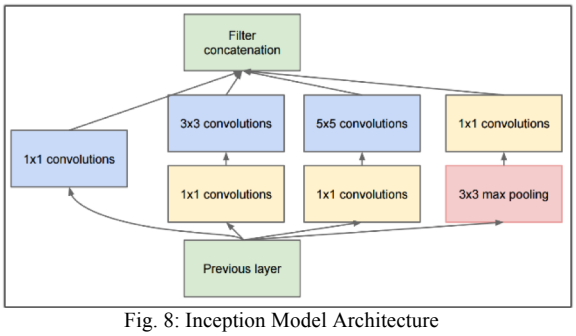


Fig. 8: Inception Model Architecture

V. SIMULATION RESULTS

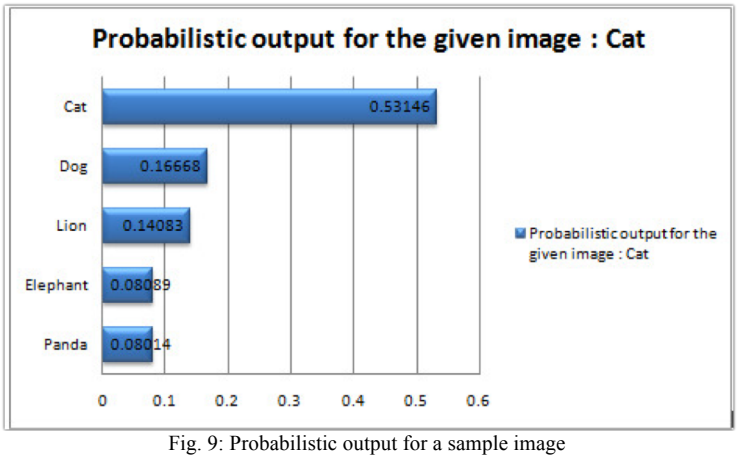


Fig. 9: Probabilistic output for a sample image

The obtained results after experimentation have been shown in the figure 8. The dataset through which features got extracted helped in creating various classes. For a test image to be correctly classified the output showed various classes of animals. Each class showed the probabilistic value of whether the image of the animal belongs to that particular class of animal. The class with the maximum probabilistic value thus showed that the features of the animal resemble to that specific class on large scale and hence the image will be classified to the class with large probabilistic value.

No. of Iterations	Train Accuracy (%)	Validation Accuracy (%)	Final Test Accuracy (%)
1	**53.0**	**37.0**	**59.0**
2	80.0	64.0	69.0
3	87.0	82.0	79.0
4	82.0	79.0	82.3

 DOI: 10.15680/IJIRCCE.2018. 0603001

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

5	92.8	82.0	83.9
6	97.0	83.0	82.3
7	66.8	91.8	89.5
8	75.0	72.0	86.2
9	85.0	65.0	93.0
10	**97.0**	**90.0**	**95.2**

Table 1: Accuracy for each iteration

For each iteration batch of dataset was passed through the neural network. The training accuracy, validation accuracy and final test accuracy was calculated during every iteration as shown in table 1. Thus the accuracy gradually increased according to the iteration giving 95.2% final accuracy with training and test accuracy of 97.0% and 90.0% respectively.

VI. CONCLUSION

In this report, we first briefly explained our motivation of this project and showed some background materials. Then, we precisely illustrated our task of demonstrating that wild animals' detection can be done using deep convolutional neural networks. Features learned by Deep Neural Networks are hierarchical. With more hidden layers, the learned features become more high level and specific. In this paper, we tested the ability of state-of-the-art computer vision methods called deep neural networks to automatically identify animals. Automating animal identification can thus dramatically reduce the cost to extract informative and actionable information from wild habitats, potentially revolutionizing studies of animal behavior, ecosystem dynamics, and wildlife conservation.

VII. FUTURE SCOPE

In the system proposed above, we have used a fixed number of training images of each species. Increasing the number of images for each species will result in increase in the prediction of the class to which the animal in the captured image belongs. Furthermore, a newly captured image can be added to the dataset and in the next iteration, the system will consider this image for training along with the previous images. Also, the number of features that can be extracted from a single image can be increased in order to make the result unambiguous. The challenge in some cases is how to train a model without access to a large number of labeled images. Here, transfer learning can help, wherein a deep neural network is trained on a huge, labeled dataset at first and then the knowledge learned is remodeled to classify a different dataset with fewer labeled images. Given the rapid pace of progress in this field, all of the performance measures should be considered just a preview of what is possible and it is likely that they will be substantially improved on each year for the foreseeable future. Therefore in future we want to extract regions for addressing the location of objects and extract other features as well to get better results.

REFERENCES

1. Leila Mansourian, Muhamad Taufik Abdullah, Lilli Nurliyana Abdullah and Azreen Azman ,Evaluating classification strategies in Bag of SIFT Feature method for Animal Recognition, Research Journal of Applied Sciences, Engineering and Technology, pg 1266-1272, August 2015
2. Bang Liu, Yan Liu and Kai Zhou ,Image Classification for Dogs and Cats
3. Kaggle DogVCat Competation: http://www.kaggle.com/c/dogs-vs-cats
4. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
5. Mohammed Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Ali Swanson, MeredithPalmer, CraigPacker, and JeffClune1, Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning , ArXiv , 1703.05830v5, November 2017
6. Chattopadhyay P, Vedantam R, Selvaraju RR, Batra D, Parikh D (2017) Counting everyday objects in everyday scenes, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1604.03505v3 [cs.CV], 9 May 2017.
7. Krizhevsky, A., Sutskever, I., & Hinton, G. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 , pg. 1106-1114, 2012.
8. Shaikh S, Jadhav M, Nehe N and Verma U, 2015, Automatic animal detection and warning system, International Journal of Advance Foundation and Research in Computer, vol. 2, pg: 405-410, 2015.
9. Mitra V, Wang C and Edwards G, "Neural network for LIDAR detection of fish", Proceedings of the International Joint Conference on Neural Networks, pg. 1001-1006, 2003.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

10. Sharma S and Shah D, 2016, Real-Time automatic obstacle detection and alert system for driver assistance on Indian roads, Indonesian Journal of Electrical Engineering and Computer Science, vol.1, pg:635-646, 2016
11. Ragheb H, Velastin S, Remagnino P and Ellis T, "Human Action Recognition using Robust Power Spectrum Features", Proceedings of the 15th International Conference on Image Processing, (ICIPO'08), San Diego, CA, pg.753-756, 2008

 DOI: 10.15680/IJIRCCE.2018. 0603001