An Approach for Multi Label Image Classification Using Single Label Convolutional Neural Network Classifier with Objectness Measure and Selective Search

Setu Basak
Roll: 1107043
Shubhashis Karmakar
Roll: 1107001

Supervisor: Dr. Kazi Md. Rokibul Alam
Professor, CSE, KUET

April, 2016

Acknowledgement

With due homage and honor, we want to express our gratitude to Almighty Allah.

We express our indebtedness to our dedicated supervisor Dr. Kazi Md. Rokibul Alam (Professor, Department of Computer Science and Engineering, KUET) for his necessary guidance, suggestions and encouragement to all phases of our work. We would also like to thank all the teachers who've been much helpful and encouraging in completion of our thesis work.

Authors

Abstract

Single label image classification has been promisingly demonstrated using Convolutional Neural Network (CNN). However, how this CNN will fit for multi-label images is still difficult to solve. It is mainly difficult due to the complex underlying object layouts and insufficient multi-label training images. In this work, we propose an approach for classifying multi-label image by a trained single label classifier using CNN with objectness measure and selective search. We took two established image segmentation techniques for segmenting a single image which is a multi-label image into some segmented images. Then we forwarded the images to our trained CNN and predicted the labels of the segmented images by generalizing the result. Our single-label image classifier gives 87% accuracy on CIFAR-10 dataset. Using objectness measure with CNN gives us 51% accuracy on a multi-label dataset and gives upto 57% accuracy using selective search both considering top-4 labels which is significantly good for a simple approach rather than a complex approach of multi-label classifier using CNN.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Image classification containing a single-label, which has been extensively studied in the recent years [1], [2]. For image representation and classification, conventional approaches utilize carefully designed hand-crafted features, e.g., SIFT [3], along with the bag-of-words coding scheme, followed by the feature pooling [4], [5] and classic classifiers, such as Support Vector Machine (SVM) [6] and random forests [7]. Recently, in contrast to the hand-crafted features, learnt image features with deep network structures have shown their great potential in various vision recognition tasks [8], [9], [10], [11]. Among these architectures, one of the greatest breakthroughs in image classification is the deep convolutional neural network (CNN) [10], which has achieved the state-of-the-art performance (with 10% gain over the previous methods based on hand-crafted features) in the large-scale single-label object recognition task, i.e., ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [2] with more than one million images from 1,000 object categories.

Multi-label image classification is however a more general and practical problem, since the majority of real-world images are with more than one objects of different categories. Many methods [12], [13], [14] have been proposed to address this more challenging problem. The success of CNN on single-label image classification also sheds some light on the multi-label image classification problem. However, the CNN model cannot be trivially extended to cope with the multi-label image classification problem in an interpretable manner, mainly due to the following reasons. Firstly, the implicit assumption that foreground objects are roughly aligned, which is usually true for single-label images, does not always hold for multi-label images. Such alignment facilitates the design of the convolution and pooling infrastructure of CNN for single-label image classification. However, for a typical multi-label image, different categories of objects are located at various positions with different scales and poses. For example, as shown in Figure 1.1, for single-label images, the foreground objects are roughly aligned, while for multi-label images, even with the same label, i.e., cat and bird, the spatial arrangements of the cat and bird instances vary largely among different images. Secondly, the interaction between different objects in multi-label images, like partial visibility and occlusion, also poses a great challenge. Therefore, directly applying the original CNN structure for multi-label image classification is not feasible. Thirdly, due to the tremendous parameters to be learned for CNN, a large number of training images are required for the model training. Furthermore, from single-label to multi-label (with n category labels) image classification, the label space has been expanded from n to 2n, thus more training data is required to cover the whole label space. For single-label images, it is practically easy to collect and annotate the images. However, the burden of collection and annotation for a large scale multi-label image dataset is generally extremely high.

So we propose a simple technique using our trained single-label classifier of CNN with the objectness measure [15] and selective search [16]. First we segment a multi-label image into some segments or image windows using the two approaches and then test these images against our trained single label model and predict the multiple labels of the image. We have taken different approaches like taking top-1, top-2, total sum of the scores and cumulative percentage sum of the scores to predict the labels.

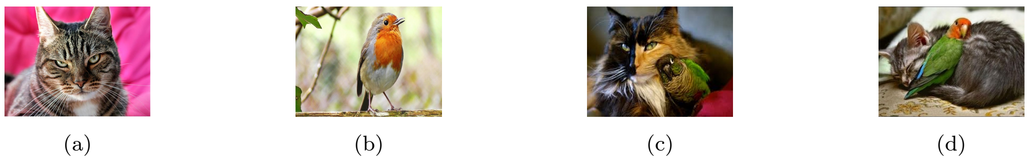


Figure 1.1: (a),(b)Some examples from CIFAR-10 [17]. The objects in single-label images are usually roughly aligned.(c),(d) However, the assumption of object alignment is not valid for multi-label images. Also note the partial visibility and occlusion between objects in the multi-label images.

1.2 Motivation

The major, recurrent theme throughout this work is our search for a good generative model for labeling of natural images. In addition, we seek a model with image segmentation based on objectness measure and selective search that is capable of extracting multiple labels from images. The hope is that we would find out a simple model that can classify multiple labels. Finally, the training time for the model would be small and segmenting the images would be easy enough.

1.3 Problem Statement

Being able to automatically detect the multiple labels of an image is a very challenging task, but it could have great impact, for instance by helping visually impaired people better identify the content of images on the web. This task is significantly harder, for example, than the well-studied image classification or object recognition tasks, which have been a main focus in the computer vision community. It is also very tough to get dataset of multiple images and train it for a model. That model will be huge and will take loads of time. So our task will be as follows:

1. To predict a single label image.
2. Segmentation of an image containing multiple objects into some segmented images using objectness measure [15] and selective search [16].

3. Using the segmented images to predict multiple labels in an image.

1.4 Objectives

- First objective is to maximize the accuracy of single-label image detection using Convolutional Neural Network(CNN).
- Next objective is to produce image windows from an image having multiple objects using objectness measure and selective search.
- Last objective is to predict labels of an image using top-1, top-2, total score sum and cumulative percentage sum of the score of every segmented images.

Chapter 2

Related Works

During the past few years, many works on various multi-label image classification models have been conducted. These models are generally based on two types of frameworks: bag-of-words (BoW) [18], [12], [13], [14] and deep learning [19].

2.1 Bag-of-Words Based Models

A traditional BoW model is composed of multiple modules, e.g., feature representation, classification and context modeling. For feature representation, the main components include hand-crafted feature extraction, feature coding and feature pooling, which generate global representations for images. Specifically, hand-crafted features, such as SIFT [3], Histogram of Oriented Gradients [20] and Local Binary Patterns [21] are firstly extracted on dense grids or sparse interest points and then quantized by different coding schemes, e.g., Vector Quantization [22], Sparse Coding and Gaussian Mixture Models [23]. These encoded features are finally pooled by feature aggregation methods, such as Spatial Pyramid Matching (SPM) [4], to form the image-level representation. For classification, conventional models, such as SVM [6] and random forests [7], are utilized. Beyond conventional modeling methods, many recent works [18], [13] have demonstrated that the usage of context information, e.g., spatial location of object and background scene from the global view, can considerably improve the performance of multi-label classification and object detection.

Although these works have made great progress in visual recognition tasks, the involved hand-crafted features are not always optimal for particular tasks. Recently, in contrast to hand-crafted features, learnt features with deep learning structures have shown great potential for various vision recognition tasks, which will be introduced in the following subsection.

2.2 Deep Learning Based Models

Deep learning tries to model the high-level abstractions of visual data by using architectures composed of multiple non-linear transformations. Specifically, deep convolutional neural network (CNN) [8] has demonstrated an extraordinary ability for image classification [9], [10] on single-label datasets such as CIFAR-10/100 [17] and ImageNet [2]. More recently, CNN architectures have been adopted to address multi-label problems. Gong et al. [19] studied and compared several multi-label loss functions for the multi-label annotation problem based on a similar network structure to [10].

However, due to the large number of parameters to be learned for CNN, an effective model requires lots of training samples. Therefore, training a task-specific convolutional neural network is not applicable on datasets with limited numbers of training samples.

Chapter 3

Initial Attempts

Before migrating to Convolutional Neural Network(CNN) we tried many simple approaches for single-label image classification which we would use as a trained model for multi-label image classification. Though the accuracy of the approaches were not that good and also had many difficulties.

3.1 Nearest Neighbor Classifier

The nearest neighbor classifier will take a test image, compare it to every single one of the training images, and predict the label of the closest training image. We compare the images pixel by pixel and add up all the differences. In other words, given two images and representing them as vectors $\left(I_1, I_2\right)$, a reasonable choice for comparing them might be the L1 distance:

$$d_1(I_1, I_2) = \sum_p \left(I_1^p - I_2^p\right) \tag{3.1}$$

There are many other ways of computing distances between vectors. We can use the L2 distance, which has the geometric interpretation of computing the euclidean distance between two vectors.

$$d_1(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)} \tag{3.2}$$

Accuracy
We got about 35% using the NN Classifier.

3.1.1 k - Nearest Neighbor Classifier

Instead of finding the single closest image in the training set, we will find the top k closest images, and have them vote on the label of the test image. In particular, when k = 1, we recover the Nearest Neighbor classifier. The k-nearest neighbor classifier requires a setting for k. But what number works best? We have tried out many different values of k and experimented what k value works best.

3.1.2 Cross-validation

In cases where the size of the training data (and therefore also the validation data) is small, we can use a hyperparameter tuning called cross-validation. Instead of arbitrarily picking datapoints to be the validation set and rest training set, we can get a better and less noisy estimate of how well a certain value of k works by iterating over different validation sets and averaging the performance across these. For example, in 5-fold cross-validation, we would split the training data into 5 equal folds, use 4 of them for training, and 1 for validation.

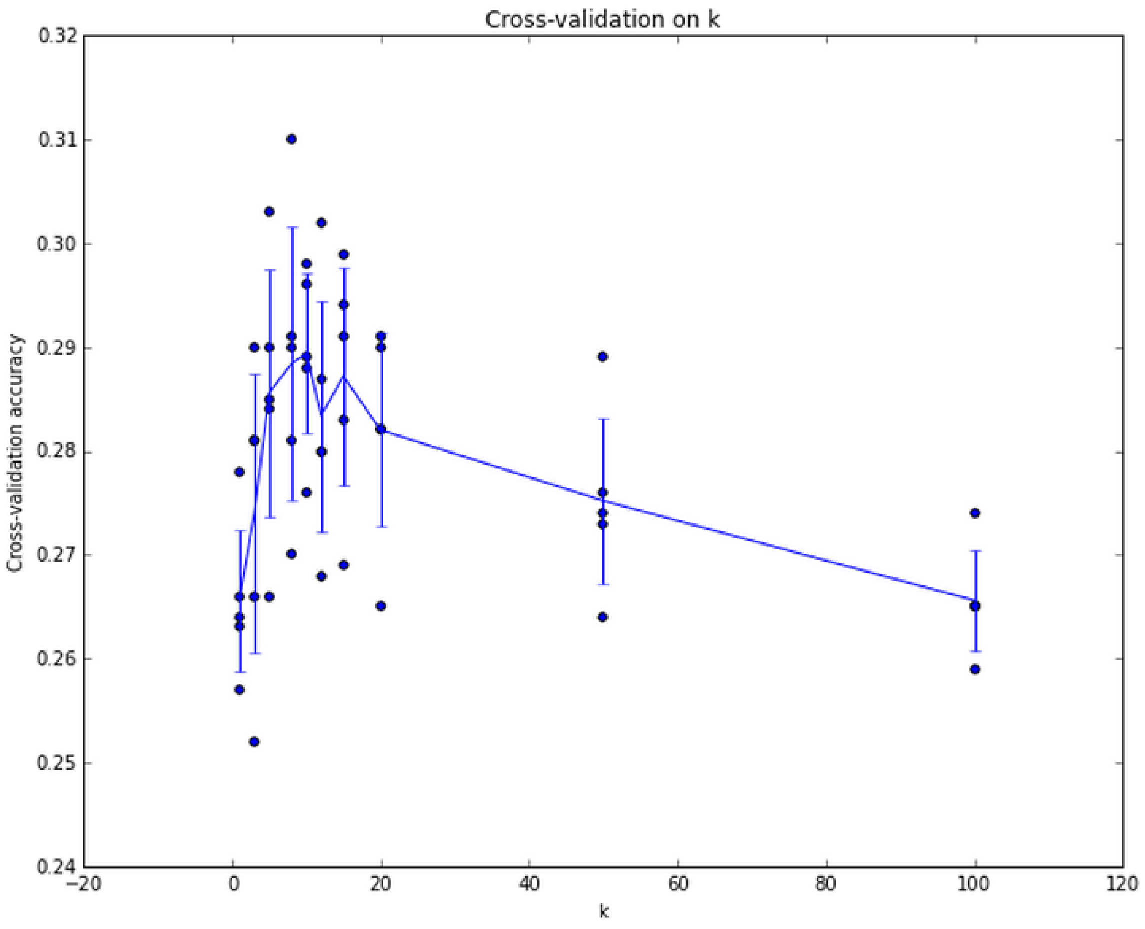


Figure 3.1: Example of a 5-fold cross-validation run for the parameter k.

Pros and cons of Nearest Neighbor Classifier

Pros
1. It is very simple to implement and understand.
2. The classifier takes no time to train.

Cons
1. Accuracy rate is bad.

2. The classifier must remember all of the training data and store it for future comparisons.
3. Using pixel differences to compare images is inadequate(Figure 3.2)

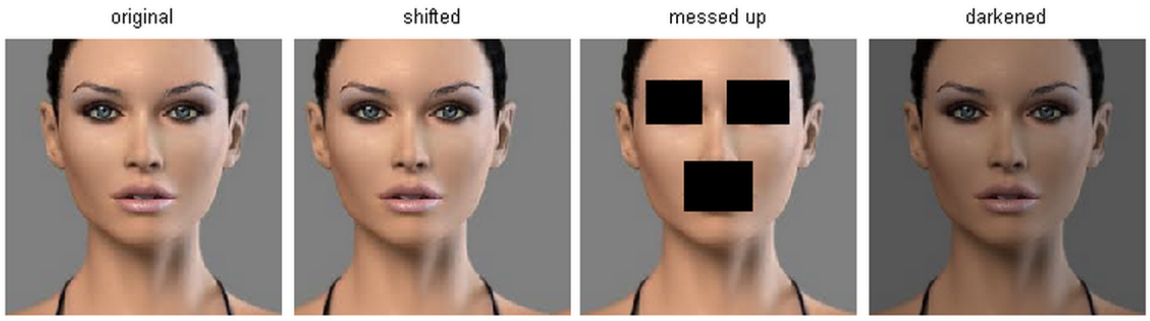


Figure 3.2: Pixel-based distances on high-dimensional data (and images especially) can be very unintuitive. An original image (left) and three other images next to it that are all equally far away from it based on L2 pixel distance. Clearly, the pixel-wise distance does not correspond at all to perceptual or semantic similarity.

3.2 Linear Classification

This approach will have two major components: a score function that maps the raw data to class scores, and a loss function that quantifies the agreement between the predicted scores and the ground truth labels.

Score Function
For the score function we will use this linear mapping function:

$$f(x_i, W, b) = Wx_i + b \tag{3.3}$$

In the above equation, we are assuming that the image x_i has all of its pixels flattened out to a single column vector of shape $[D * 1]$. The matrix W (of size $[K * D]$), and the vector b (of size $[K * 1]$) are the parameters of the function. In CIFAR-10 each image were of size $[32 * 32 * 3]$. Here x_i contains all pixels in the i-th image flattened into a single $[3072 * 1]$ column, W is $[10 * 3072]$ and b is $[10 * 1]$, so 3072 numbers come into the function (the raw pixel values) and 10 numbers come out (the class scores). The parameters in W are often called the weights, and b is called the bias vector because it influences the output scores.
An advantage of this approach is that the training data is used to learn the parameters W,b, but once the learning is complete we can discard the entire training set and only keep the learned parameters.

Cost Function
We will use a loss function called the Multiclass Support Vector Machine (SVM) loss. The SVM loss is set up so that the SVM wants the correct class for each image to a have a score higher than the incorrect classes by some fixed margin $\triangle$. So in each

iteration if we achieve lower loss, then it is better. The Multiclass SVM loss for the i-th sample is :

$$L_i = \sum_{i \neq j} max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \triangle) \tag{3.4}$$

Regularization There is an issue with loss function e.g. this set of W is not necessarily unique: there might be many similar W that correctly classify the examples. We can encode some preference for a certain set of weights W over others to remove this ambiguity. We can do so by extending the loss function with a regularization penalty $R(W)$.

$$R(W) = \sum_k \sum_l W_{k,l}^2 \tag{3.5}$$

We told earlier that Multiclass Support Vector Machine loss is made up of two components: the data loss (which is the average loss L_i over all examples) and the regularization loss. So the full Multiclass SVM loss becomes:

$$L = \frac{1}{N} \sum_i L_i + \lambda R(w) \tag{3.6}$$

3.3 Softmax Classifier

In the Softmax classifier, the function mapping is as we used before on linear classifier. But we now interpret these scores as the unnormalized log probabilities for each class. Here we have a cross-entropy loss that has the form:

$$L_i = -log(\frac{e^{f_{y_i}})}{\sum_j e^{f_j}} \tag{3.7}$$

3.4 Optimization : Gradient Descent

We need to improve our weight vector so that we can get lower loss. We need lower loss because loss function quantifies the agreement between the predicted scores and the ground truth labels. So obtaining lower loss means we are achieving better results. If we do a random search for best direction, then the computational cost will be higher. We can use gradient descent to compute the best direction along which we should change our weight vector. This direction will be related to the gradient of the loss function. This approach roughly corresponds to feeling the slope of the hill below our feet and stepping down the direction that feels steepest. The gradient is just a vector of slopes (more commonly referred to as derivatives) for each dimension in the input space.

By computing Gradient Descent, we are making an update of weight in the negative direction of the gradient since we wish our loss function to decrease, not increase.

Finally with the gradient descent we applied on SVM we get the decreasing loss in Figure 3.3. Also the accuracy was around 37%.

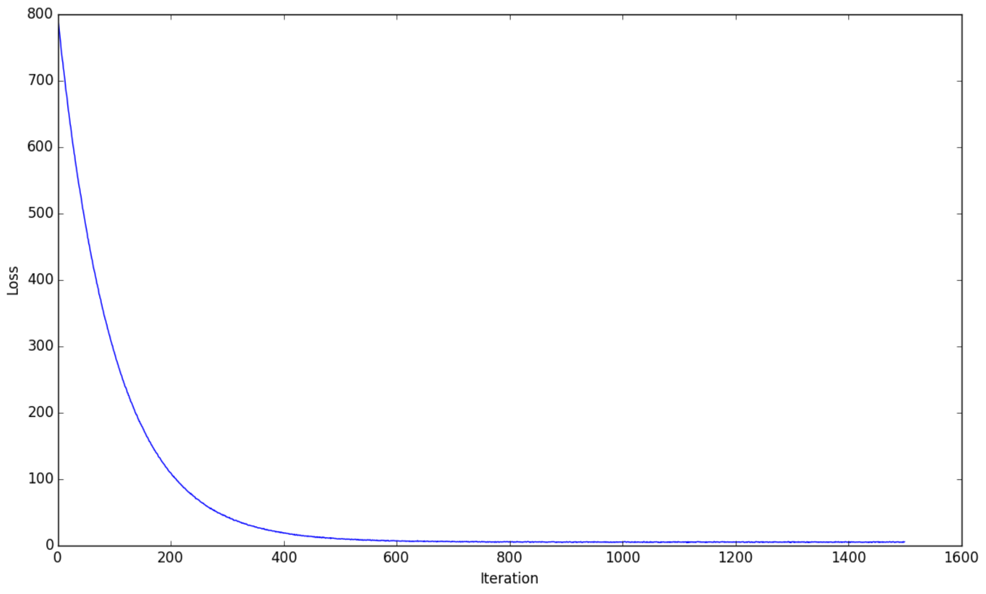


Figure 3.3: Decreasing loss as iteration increases

3.5 Neural Network in Object Detection

Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections.

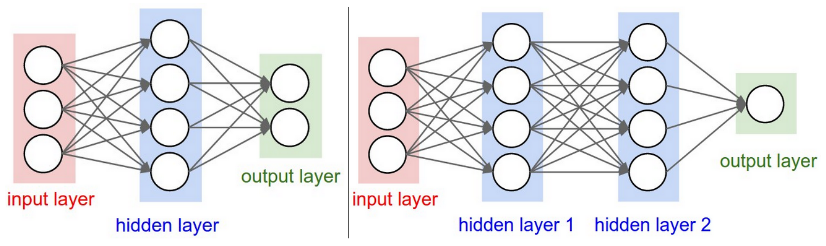


Figure 3.4: Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. Right: A 3-layer neural network with three inputs.

More Layers Better?

If we increase the size and number of layers in a Neural Network, the capacity of the network increases. That is, the space of representable functions grows since the neurons can collaborate to express many different functions. In the Figure 3.5 we see that higher hidden neuron means better representational power but it comes with a cost. The problem is overfitting.

Overfitting

Overfitting occurs when a model with high capacity fits the noise in the data instead of the underlying relationship. For example, the model with 20 hidden neurons fits all the training data but at the cost of segmenting the space into many disjoint red and green decision regions. A model which has been overfit will generally have poor performance on new data, as it can exaggerate minor fluctuations in the training set.

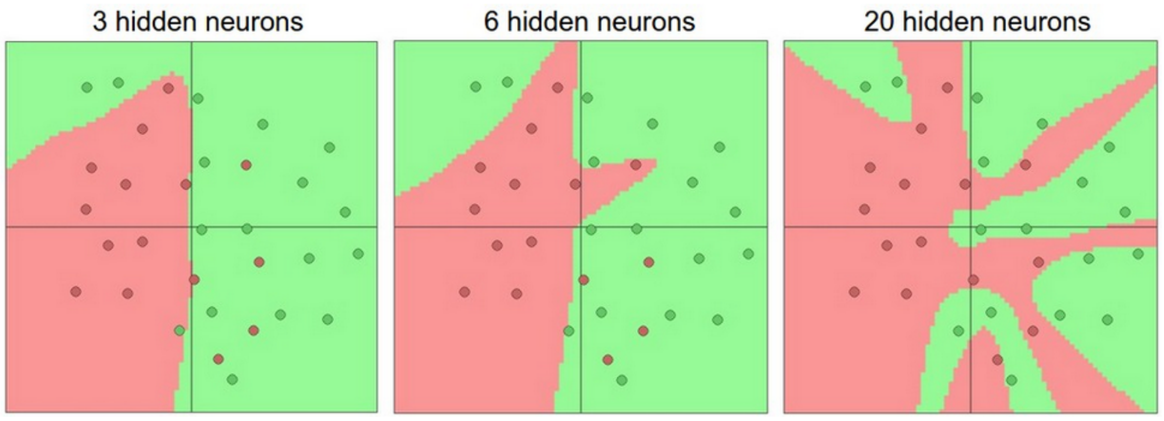


Figure 3.5: Larger Neural Networks can represent more complicated functions. But more layers can also cause overfitting(example with 20 hidden neurons)

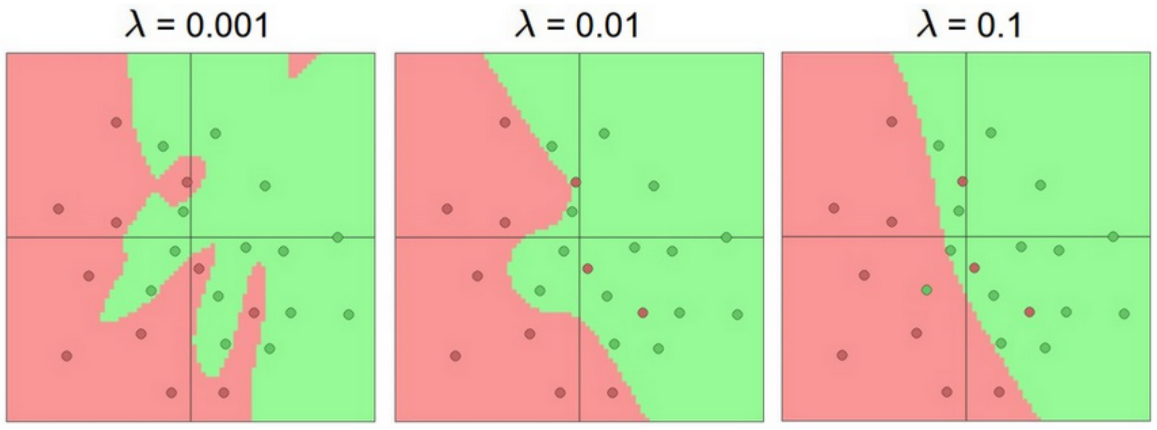


Figure 3.6: The effects of regularization strength

How to control the overfitting?

Regularization strength is the preferred way to control the overfitting of a neural network.

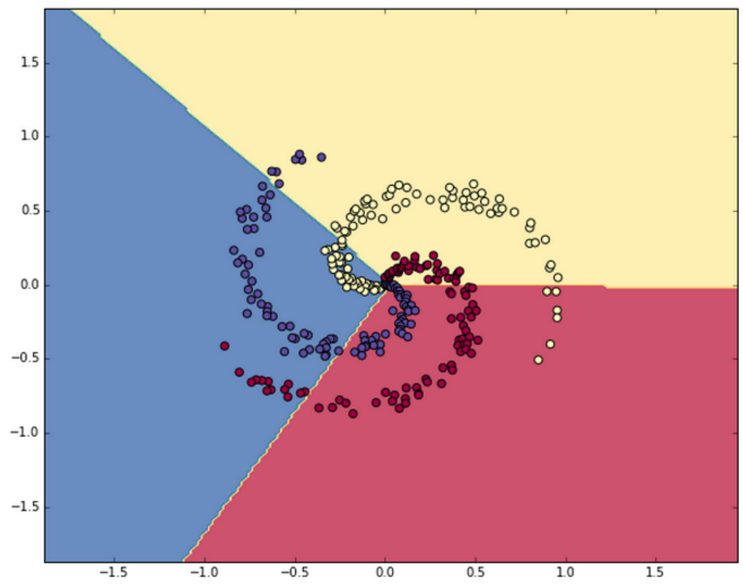


Figure 3.7: Linear classifier fails to learn the spiral dataset. Accuracy was around 49%

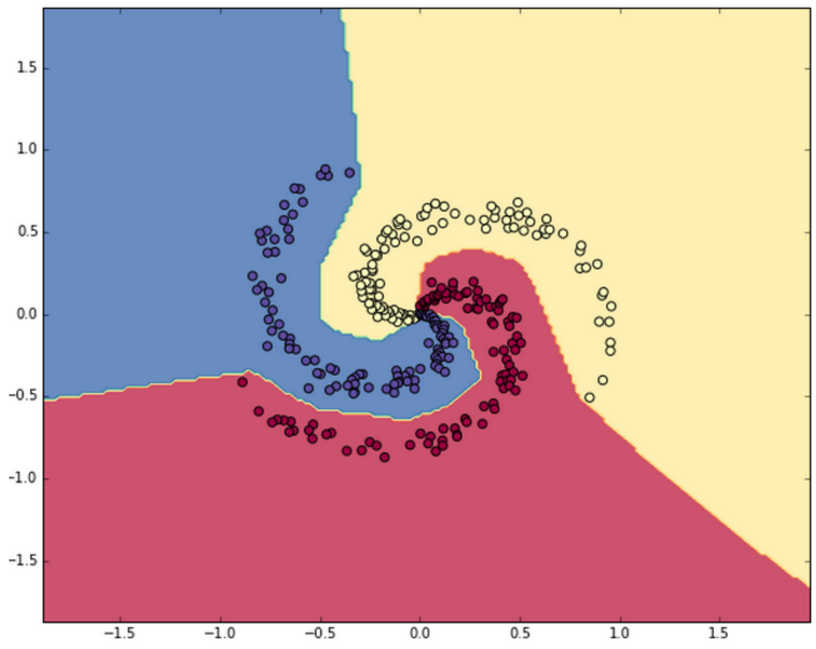


Figure 3.8: Neural Network classifier is significantly better achieving around 98% accuracy

Linear Classifier vs. Neural Network

If we implement a 2D dataset and tell any linear classifier to identify, it will fail to identify it correctly. Consider a spiral dataset containing 3 colors:yellow,blue and red.

Now if we implement a linear classifier to identify each color in the 2D space we get a result similar to Figure 3.7.

If we use one additional hidden layer then we can get a lot better result.
Here we see that adding one additional layer helped us to attain bigger and better functionality.

Chapter 4

Dataset

4.1 Single Label Dataset

4.1.1 CIFAR-10 Dataset

The CIFAR-10 dataset is a subset of the 80 million tiny images. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The sample from CIFAR-10 dataset is given in Figure 4.1.

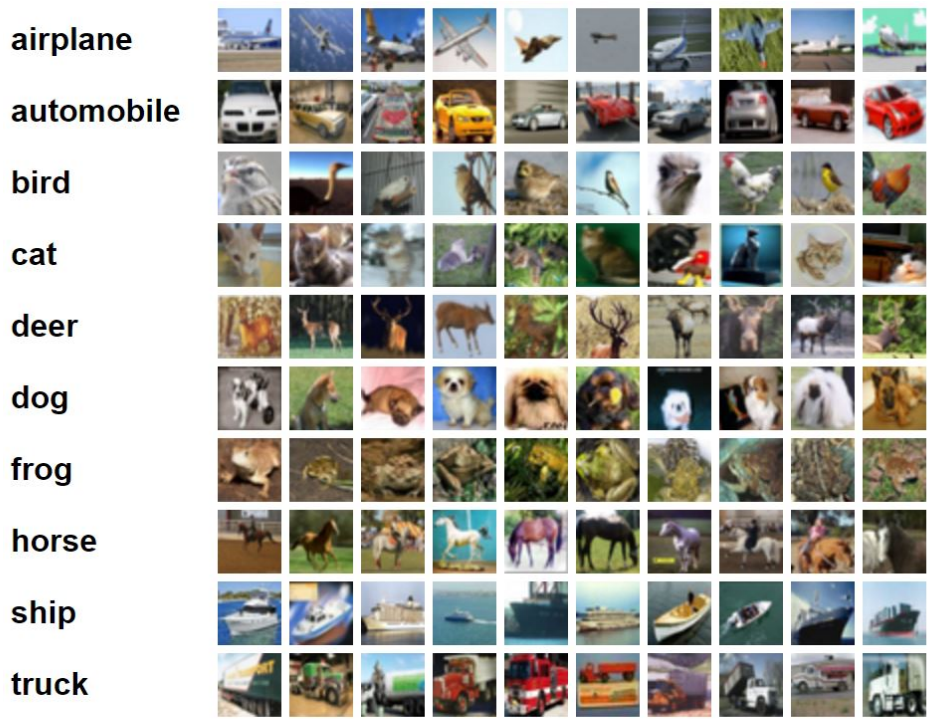


Figure 4.1: CIFAR-10 Dataset

4.2 Multi Label Dataset

As we are using single label dataset(CIFAR-10) for our single label Convolutional Neural Network, we have to use images that contain the identical contents from CIFAR-10. So, we picked random 100 images containing the identical contents as CIFAR-10 and tested it with our model. For an example, from Figure 4.2 we see that the image contains only a cat and a dog, which our Single Label Convolutional Network can classify. So, the images we chose only contains subset of airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

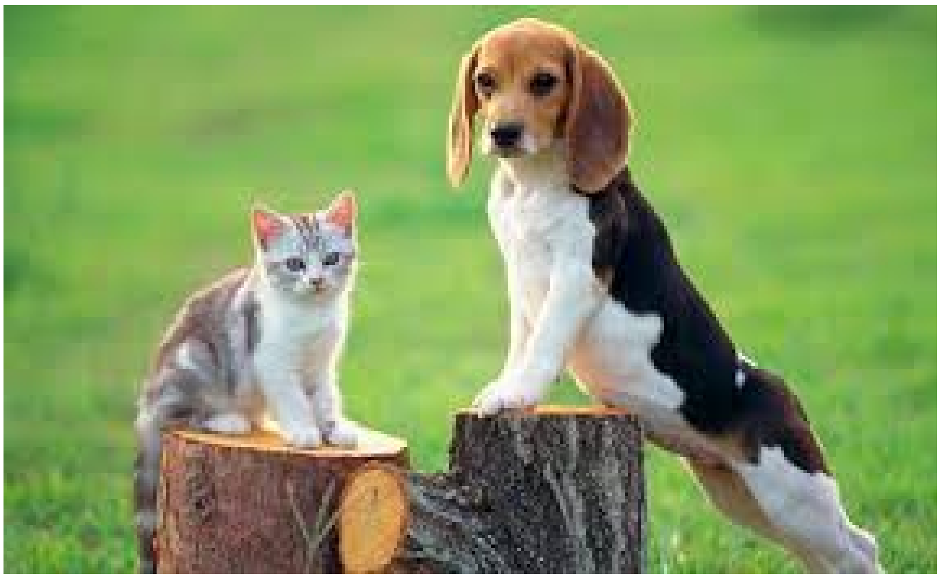


Figure 4.2: Multi Label Image

Chapter 5

The Architecture

5.1 The Architecture

5.1.1 Architecture for classifying single label images

In the figure 5.1 is a cropped down version of the Convolutional Neural Network that is used to classify single label images. This has been given to give an idea about the network.

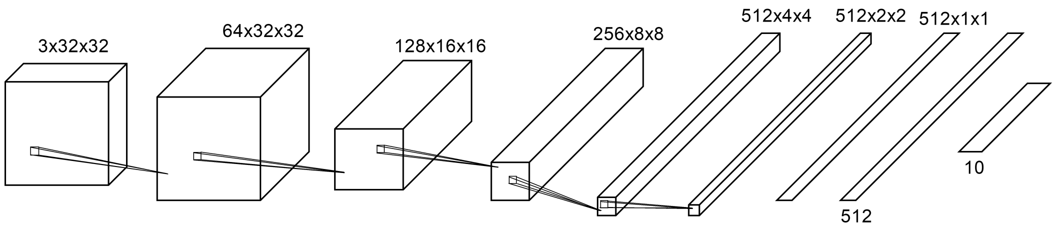


Figure 5.1: Convolutional Neural Network for classifying single label images(Cropped Down)

The full network is like the one in the figure. It has 59 layers in sequential manner. The input is the image and output is the label associated with that image.
input→1→2→3) → (4) → (5) → (6) → (7) → (8) → (9) → (10) → (11) → (12) → (13) → (14) → (15) → (16) → (17) → (18) → (19) → (20) → (21) → (22) → (23) → (24) → (25) → (26) → (27) → (28) → (29) → (30) → (31) → (32) → (33) → (34) → (35) → (36) → (37) → (38) → (39) → (40) → (41) → (42) → (43) → (44) → (45) → (46) → (47) → (48) → (49) → (50) → (51) → (52) → (53) → (54) → (55) → (56) → (57) → (58) → (59) → output
A short description about each layer is given below.
(1): Spatial Convolution(3 → 64, 3x3, 1,1, 1,1)
(2): Spatial Batch Normalization
(3): ReLU
(4): Dropout(0.3)
(5): Spatial Convolution(64 → 64, 3x3, 1,1, 1,1)
(6): Spatial Batch Normalization
(7): ReLU
(8): Spatial Max Pooling(2,2,2,2)
(9): Spatial Convolution(64 → 128, 3x3, 1,1, 1,1)
(10): Spatial Batch Normalization
(11): ReLU
(12): Dropout(0.4)

(13): Spatial Convolution(128 $\rightarrow$ 128, 3x3, 1,1, 1,1)
(14): Spatial Batch Normalization
(15): ReLU
(16): Spatial Max Pooling(2,2,2,2)
(17): Spatial Convolution(128 $\rightarrow$ 256, 3x3, 1,1, 1,1)
(18): Spatial Batch Normalization
(19): ReLU
(20): Dropout(0.4)
(21): Spatial Convolution(256 $\rightarrow$ 256, 3x3, 1,1, 1,1)
(22): Spatial Batch Normalization
(23): ReLU
(24): Dropout(0.4)
(25): Spatial Convolution(256 $\rightarrow$ 256, 3x3, 1,1, 1,1)
(26): Spatial Batch Normalization
(27): ReLU
(28): Spatial Max Pooling(2,2,2,2)
(29): Spatial Convolution(256 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(30): Spatial Batch Normalization
(31): ReLU
(32): Dropout(0.4)
(33): Spatial Convolution(512 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(34): Spatial Batch Normalization
(35): ReLU
(36): Dropout(0.4)
(37): Spatial Convolution(512 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(38): Spatial Batch Normalization
(39): ReLU
(40): Spatial Max Pooling(2,2,2,2)
(41): Spatial Convolution(512 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(42): Spatial Batch Normalization
(43): ReLU
(44): Dropout(0.4)
(45): Spatial Convolution(512 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(46): Spatial Batch Normalization
(47): ReLU
(48): Dropout(0.4)
(49): Spatial Convolution(512 $\rightarrow$ 512, 3x3, 1,1, 1,1)
(50): Spatial Batch Normalization
(51): ReLU
(52): Spatial Max Pooling(2,2,2,2)
(53): View
(54): Dropout(0.5)
(55): Linear(512 $\rightarrow$ 512)
(56): Batch Normalization
(57): ReLU
(58): Dropout(0.5)
(59): Linear(512 $\rightarrow$ 10)

Spatial Convolution: Applies a 2D convolution over an input image composed of several input planes. For an example, Spatial Convolution(3 $\rightarrow$ 64, 3x3, 1,1, 1,1) means that number of input channel is 3, and the number of output channel is 64. The kernel size is 3x3 and as there are 64 output channels, there will be 64 kernels, each having dimension 3x3. The step of the convolution is 1 step for height and width. The padding is 1 for height and width. Let's visualize this. In the figure 5.2 the input neurons are the ones representing each pixel of an image, the kernel size or the filter size is 5x5. This filter would iterate over every single pixel and for each pixel look at the 5x5 neighborhood and produce corresponding images. As the number of output channel is 64, this will be done 64 times.

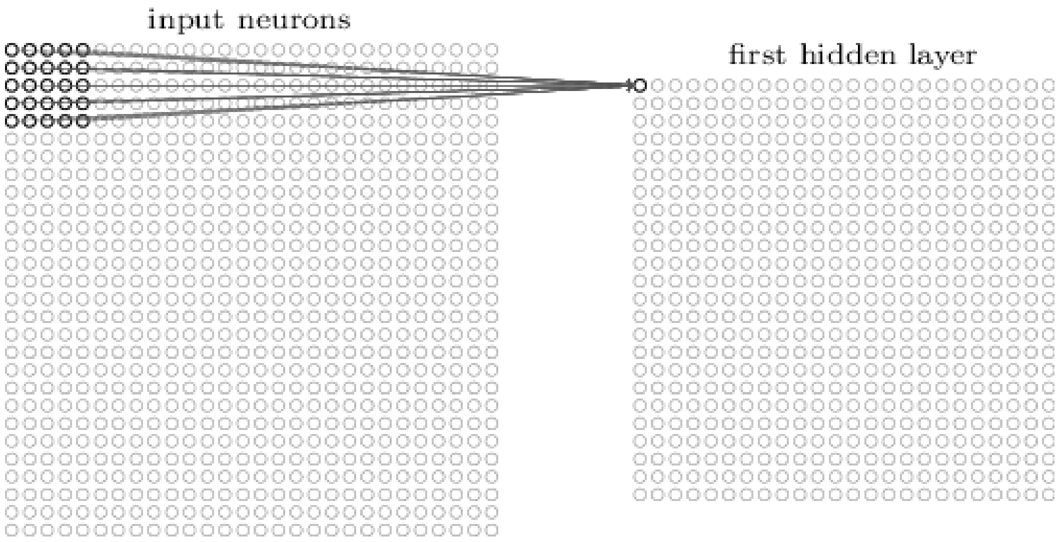


Figure 5.2: Spatial Convolution with kernel size 5x5

Spatial Batch Normalization: Implements Batch Normalization as described in the paper [26]. The operation implemented is:

$$y = \frac{x - mean(x)}{standard - deviation(x)} * gamma + beta$$

where the mean and standard-deviation are calculated per feature-map over the mini-batches and pixels and where gamma and beta are learnable parameter vectors of size N (where N = number of feature maps).

ReLU: It is the activation function defined as:

$$f(x) = max(0, x)$$

.

Spatial Max Pooling: Applies 2D max-pooling operation. For example, Spatial Max Pooling(2,2,2,2) means that the max pooling will be done with filter 2x2 with step 2 in height and step 2 in width direction. Max pooling means that we will select the maximum value from 2x2 filter or the input area.

Dropout: Dropout is used to prevent the neural network from overfitting. Dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter), or setting it to zero otherwise. The input neuron is scaled by $1/p$ if it is not deactivated.

Linear: Applies a linear transformation to the incoming data ($y = mx + c$). For example, Linear(512 $\rightarrow$ 10) means that there are 512 input channels and these 512 channels are converted to 10 channels. So, the weight matrix will be 10x512.

Loss Function: Cross-entropy loss function has been used which has the form:

$$L_i = -\log\left(\frac{e^{f_y}}{\sum_j e^{f_j}}\right)$$

5.1.2 Normalization on CIFAR-10

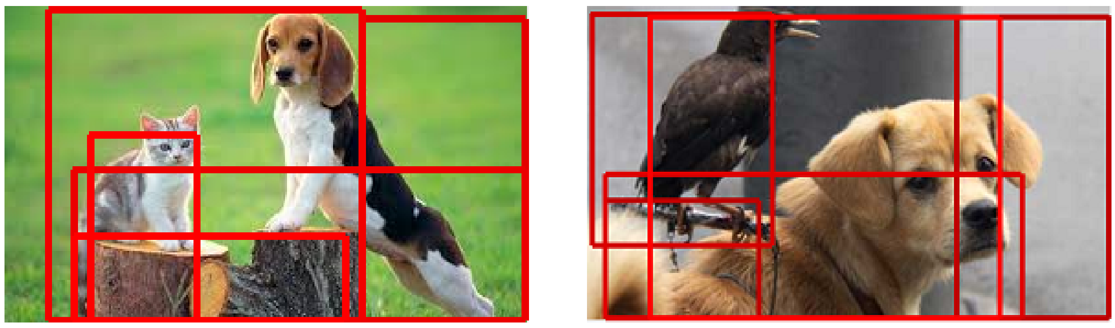
Normalization is required so that all the inputs are at a comparable range. This can be done to force the input values to a certain range. The images were converted from RGB channel to YUV channel. Then U and V channels were normalized globally with mean and standard deviation. The Y channel was normalized locally.

5.2 Object Recognition

There are many approaches for segmenting an image having multiple objects into many segmented images. Our target is to take any approach and pass it to our trained CNN model. We found out two established approaches and tried on that.

5.2.1 Measuring the objectness of image windows

B. Alexe et al. [15] presented a generic objectness measure, quantifying how likely it is for an image window to contain an object of any class. They explicitly trained it to distinguish objects with a well-defined boundary in space, such as cows and telephones, from amorphous background elements, such as grass and road. The measure combines in a Bayesian framework several image cues measuring characteristics of objects, such as appearing different from their surroundings and having a closed boundary. These include an innovative cue to measure the closed boundary characteristic. Finally, they presented two applications of objectness. In the first, they sample a small number windows according to their objectness probability and gave an algorithm to employ them as location priors for modern class-specific object detectors. As they showed



(a) (b)

Figure 5.3

experimentally, this greatly reduces the number of windows evaluated by the expensive class-specific model. In the second application, they used objectness as a complementary score in addition to the class-specific model, which leads to fewer false positives. As shown in several recent papers, objectness can act as a valuable focus of attention mechanism in many other applications operating on image windows, including weakly supervised learning of object categories, unsupervised pixelwise segmentation, and object tracking in video. Computing objectness is very efficient and takes only about 4 sec. per image. This technique finds out some image windows like Figure 5.3.

Algorithm 1: Using objectness for class-specific detectors.

```
Input: $F, D, C$
Ouput: $Det$
Step 1: $l = \{w_1, ..., w_F\}, w_i \rightarrow D, \forall_i$
Step 2: $l_s = \{(w_1, sw_1), ..., (w_F, sw_F)\}, sw_i = c(w_i), \forall_i$
Step 3: $\rho_s = NMS(l_s) = \{(w_{n1}, sw_{n1}), ..., (w_{np}, sw_{np})\}$
Step 4: $Ł = \{w_{n1}^{lm}, ..., w_{np}^{lm}\}, w_{nj}^{lm} = max(s_w)$
Step 5: $Det = NMS(Ł)$
```

The general scheme for using their objectness measure as a location prior for object detectors is algorithm 1. The algorithm inputs the class-specific confidence function c which the detector employs to score a window. They build an initial set I of $F = 1000$ windows multinomially sampled from the distribution D of windows scored by their objectness measure (Multi-scale Saliency)MS +(Color Contrast)CC + (Superpixels Straddling)SS (step 1). They use c to score each window in I (step 2). We then run the non-maxima suppression. This results in a set ρ_s of promising windows (step 3). For every window $w_p \epsilon \rho_s$, they iteratively move to the local maximum of c in its neighborhood V_{wp}, resulting in window w_p^{lm} (step 4). Finally, they run NMS on the local maxima windows Ł and obtain detections Det (step 5). In order to use this algorithm one has to specify a window scoring function c, which is specific to a particular detector and object class, and a window neighborhood.

5.2.2 Selective Search for Object Recognition

J.R.R. Uijlings et al. [16] took a hierarchical grouping algorithm to form the basis of their selective search. Bottom-up grouping is a popular approach to segmentation [24] [25], hence they adapted it for selective search. Because the process of grouping itself is hierarchical, they can naturally generate locations at all scales by continuing the grouping process until the whole image becomes a single region. This satisfies the condition of capturing all scales. As regions can yield richer information than pixels, they wanted to use region-based features whenever possible. To get a set of small starting regions which ideally do not span multiple objects, they used the fast method of Felzenszwalb and Huttenlocher [25], which found well-suited for such purpose. Their grouping procedure now works as follows. They first used [25] to create initial regions. Then they used a greedy algorithm to iteratively group regions together: First the similarities between all neighbouring regions are calculated. The two most similar regions are grouped together, and new similarities are calculated between the resulting region and its neighbours. The process of grouping the most similar regions is repeated until the whole image becomes a single region.

5.2.3 Architecture For Classifying Multi-Label Images

Our Final architecture for detecting multi-label images consists of (1) image segmentation and (2) convolutional neural network for detecting segmented images. The image segmentation techniques, objectness measures and selective search and the architecture for single label image detection has been described before. In the Figure 5.4 there is an overview of the architecture.

At first we split the image with objectness measures and selective search. Then we get multiple split images. Note that there will be many split images. In the figure only a few are shown. Then we pass the split images through the convolutional neural network that classifies the label associated with that image. As there are many split images, there may be some labels that will be wrongly classified. We inspect these labels and their associative scores and finally conclude which of the labels are likely to be associated with the images. We performed several techniques for inspecting the images. These are discussed in the experimental result section.

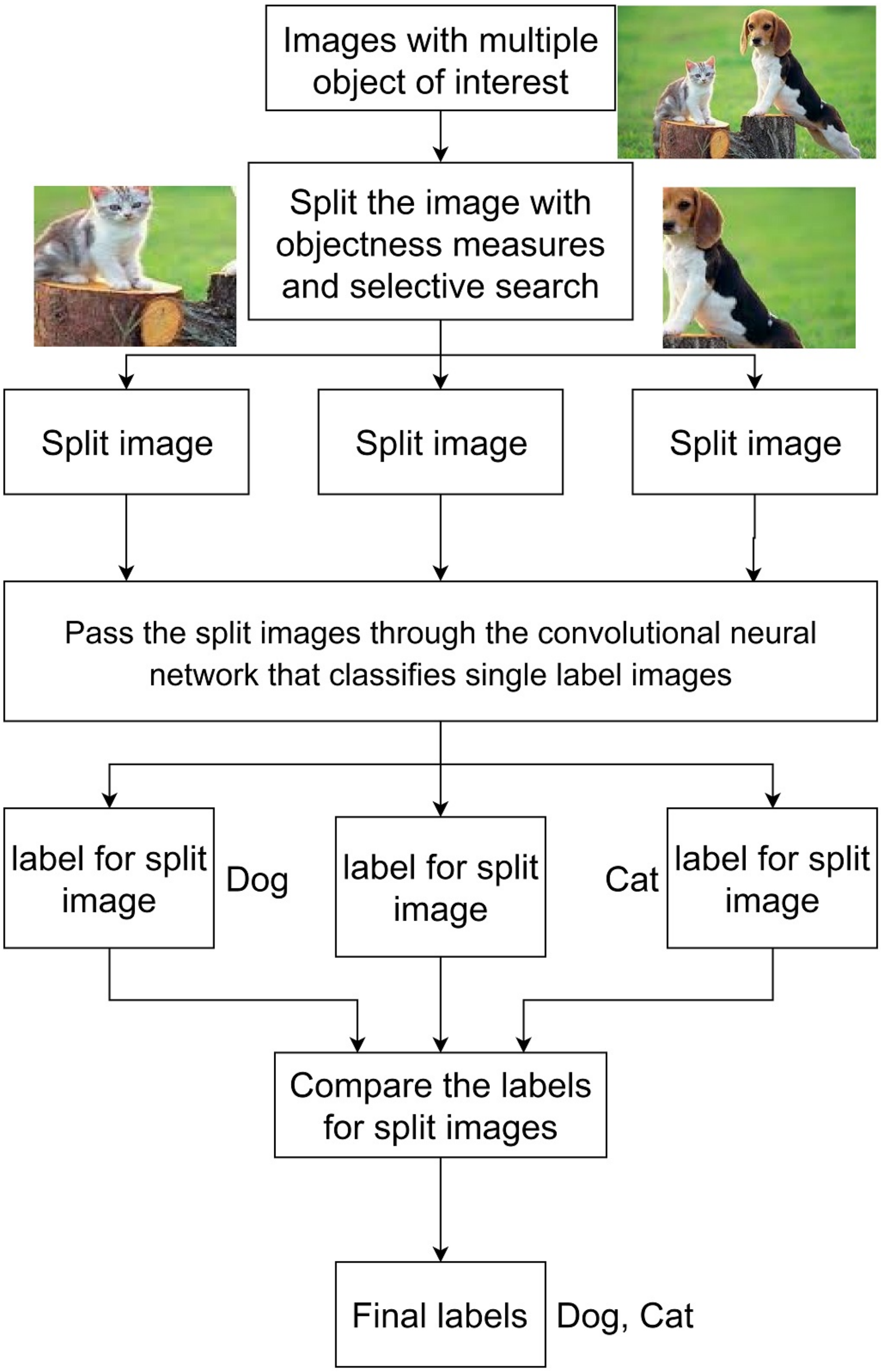

Images with multiple object of interest
Split the image with objectness measures and selective search
Split image
Split image
Split image
Pass the split images through the convolutional neural network that classifies single label images
label for split image
Dog
label for split image
Cat
label for split image
Compare the labels for split images
Final labels
Dog, Cat

Figure 5.4: Architecture of Convolutional neural network

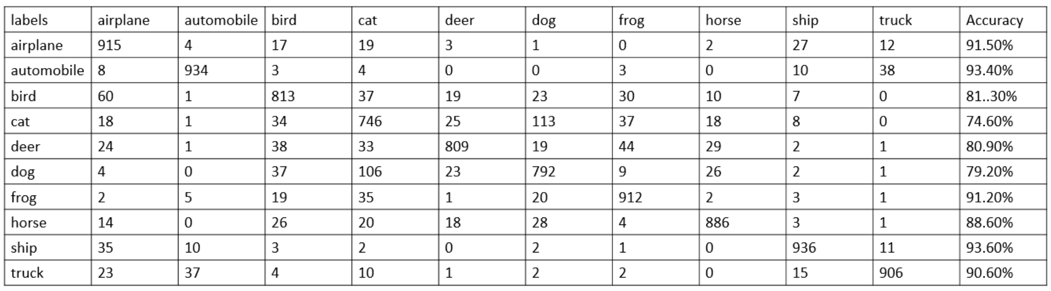
Chapter 6

Experimental Result

6.1 Experimental Results

6.1.1 Experimental Result for CIFAR-10

In the fig 6.1 there is the result for the testing images of CIFAR-10 dataset. Here, each row represents the testing label, each column represents the label produced by the network. For example the cell (1,1) represents that the testing label is airplane and the classified label is airplane, the cell (1,2) represents that the testing label is airplane and the classified label is automobile. So, each cell (i,i) where $[i = 1..10]$ represents the correctly classified labels. Each row consists of 1000 image labels. So the accuracy will be cell$(i, i)/1000$.



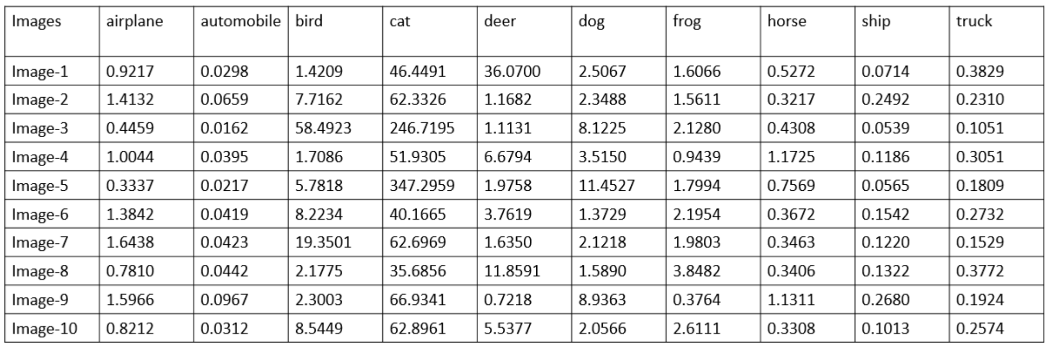
labels	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	Accuracy
airplane	915	4	17	19	3	1	0	2	27	12	91.50%
automobile	8	934	3	4	0	0	3	0	10	38	93.40%
bird	60	1	813	37	19	23	30	10	7	0	81..30%
cat	18	1	34	746	25	113	37	18	8	0	74.60%
deer	24	1	38	33	809	19	44	29	2	1	80.90%
dog	4	0	37	106	23	792	9	26	2	1	79.20%
frog	2	5	19	35	1	20	912	2	3	1	91.20%
horse	14	0	26	20	18	28	4	886	3	1	88.60%
ship	35	10	3	2	0	2	1	0	936	11	93.60%
truck	23	37	4	10	1	2	2	0	15	906	90.60%

Figure 6.1: CIFAR-10 Confusion Matrix

6.1.2 Experimental Result for Multi-Label Image Classification

To give an overview of the process of inspecting the result of the segmented images, we will need the help of Table from Figure 6.2. The table is for a single multi-label image. Each $image_i$ represents the segmented image from the multi-label image. Each row represents the class scores given by the network.

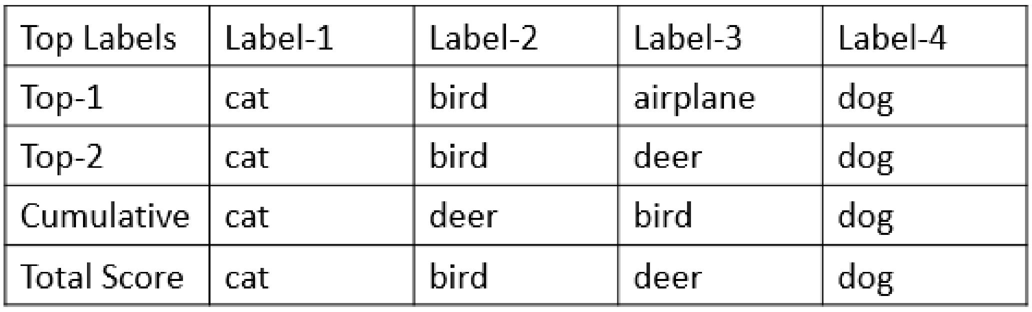
Selecting Top-1 Score: In this approach we select the top 1 score and its associating label from each segmented image $image_i$. Then we increase the frequency of labels for each top 1 score. After that we select the top 4 scoring labels. For example, in the Figure 6.2 the top 1 label for $image_1$ would be cat. So we increase the frequency of cat by 1. Similarly for $image_2$ again the score for label cat is highest. So frequency of cat will be increased by 1.



Images	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Image-1	0.9217	0.0298	1.4209	46.4491	36.0700	2.5067	1.6066	0.5272	0.0714	0.3829
Image-2	1.4132	0.0659	7.7162	62.3326	1.1682	2.3488	1.5611	0.3217	0.2492	0.2310
Image-3	0.4459	0.0162	58.4923	246.7195	1.1131	8.1225	2.1280	0.4308	0.0539	0.1051
Image-4	1.0044	0.0395	1.7086	51.9305	6.6794	3.5150	0.9439	1.1725	0.1186	0.3051
Image-5	0.3337	0.0217	5.7818	347.2959	1.9758	11.4527	1.7994	0.7569	0.0565	0.1809
Image-6	1.3842	0.0419	8.2234	40.1665	3.7619	1.3729	2.1954	0.3672	0.1542	0.2732
Image-7	1.6438	0.0423	19.3501	62.6969	1.6350	2.1218	1.9803	0.3463	0.1220	0.1529
Image-8	0.7810	0.0442	2.1775	35.6856	11.8591	1.5890	3.8482	0.3406	0.1322	0.3772
Image-9	1.5966	0.0967	2.3003	66.9341	0.7218	8.9363	0.3764	1.1311	0.2680	0.1924
Image-10	0.8212	0.0312	8.5449	62.8961	5.5377	2.0566	2.6111	0.3308	0.1013	0.2574

Figure 6.2: Scores for the sample image

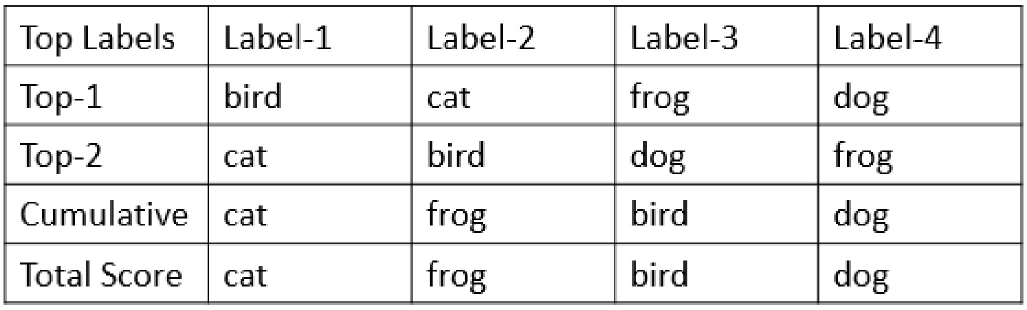
Selecting Top-2 Score: In this approach we select the top 2 scores and its associating label from each segmented image $image_i$. Then we increase the frequency of labels for each top 2 scores. After that we select the top 4 scoring labels. For example, in the Figure 6.2 the top 2 labels for $image_1$ would be cat and deer. So the frequency of cat and deer will be increased by 1.



Top Labels	Label-1	Label-2	Label-3	Label-4
Top-1	cat	bird	airplane	dog
Top-2	cat	bird	deer	dog
Cumulative	cat	deer	bird	dog
Total Score	cat	bird	deer	dog

Figure 6.3: Result for the sample image for Objectness Measures

Selecting Total Score: In this approach we just add all scores of each $label_i$ for each segmented image $image_i$. Then we select the top 4 scoring labels. For example, every class score for every label will be added. So, the score for cat label will be the total score in the column cat. Similarly the score for bird will be the total score in the column bird.



Top Labels	Label-1	Label-2	Label-3	Label-4
Top-1	bird	cat	frog	dog
Top-2	cat	bird	dog	frog
Cumulative	cat	frog	bird	dog
Total Score	cat	frog	bird	dog

Figure 6.4: Result for the sample image for Selective Search

Selecting Cumulative Percentage: Each $label_i$ of segmented image $image_i$ has the percentage $score_i / \sum_i score_i$. We select the top percentage from each segmented image $image_i$. We add these percentages for each associated labels and select the top 4 scoring labels. For example for $image_1$ the percentage of cat will be $score_{cat} / \sum_i score_i$.

The results for these four approaches for the sample image from Figure 6.5 is given at Figure 6.3 and Figure 6.4

These approaches were repeated in our selected multi label images. The results for Objectness Measure is given in Figure 6.6 and for Selective Search in Figure 6.7

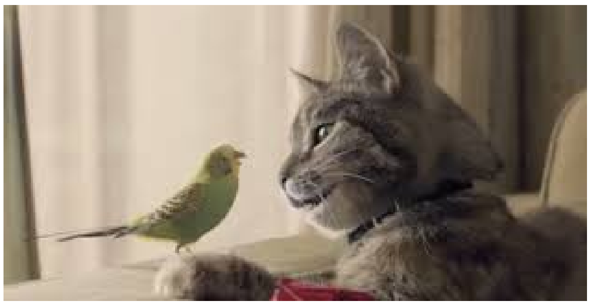


Figure 6.5: Sample Multi-Label Image

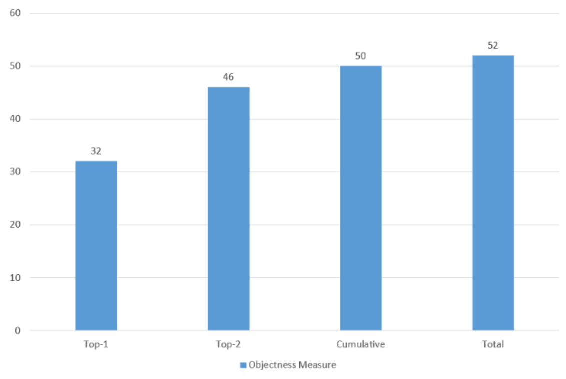


Figure 6.6: Result for multi-label images for Objectness Measures

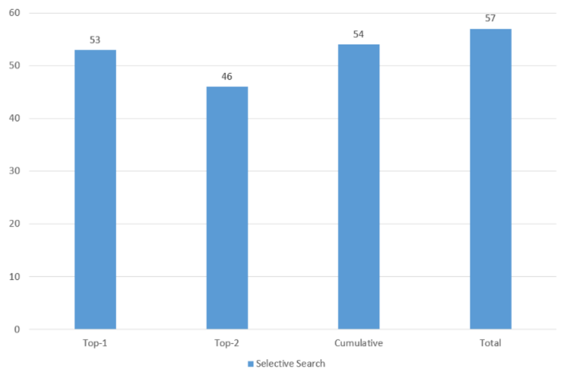


Figure 6.7: Result for multi-label images for Selective Search

Chapter 7

Tools and Recommendations for Future Works

7.1 Used Tools

Torch

Torch is a scientific computing framework with wide support for machine learning algorithms. It is easy to use and efficient. At the heart of Torch are the popular neural network and optimization libraries which are simple to use, while having maximum flexibility in implementing complex neural network topologies.

Our Convolutional Neural Network runs on top of Torch.

Waffle

Waffle is a fast, asynchronous, express-inspired web framework for Torch built on top of ASyNC.

We used this to build a server that classifies the single and multi-label images.

7.2 Future Works

The strength of our multi-label image classification architecture depends on single-label convolutional neural network classifier and the method for segmenting images.

So, be refining out convolutional neural network to include more labels by training CIFAR-100 and using better methods to segment the image, we can improve the performance of the network.

Chapter 8

Conclusion

We have presented an easy and simple approach to classify multi-label images using a trained single-label image classifier with objectness measure and selective search of an image. It gave us the result in quick time. Our training time for single-label network is fast. Segmenting the images is also fast. Our technique is a stepping stone to this approach. It is giving a good threshold result. In the future, this model can be enhanced and cope with larger dataset. So it will be interesting to see how advancement is made in this approach.

Bibliography

[1] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding, 106(1):59–70, 2007.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, pages 248–255, 2009.

[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91– 110, 2004.

[4] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition, volume 2, pages 2169–2178, 2006.

[5] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Localityconstrained linear coding for image classification. In Computer Vision and Pattern Recognition, pages 3360–3367, 2010.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. ACM Trans. Intelligent Systems and Technology, 2(3):27, 2011.

[7] L. Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[8] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In Neural Information Processing Systems, 1990.

[9] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In International Conference on Computer Vision, pages 2146–2153, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Neural Information Processing Systems, pages 1106–1114, 2012.

[11] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In International Conference on Computer Vision, pages 2056–2063, 2013.

[12] F. Perronnin, J. Sanchez, and T. Mensink. Improving the ´ fisher kernel for large-scale image classification. In European Conference on Computer Vision, pages 143–156, 2010.

[13] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. Hierarchical matching with side information for image classification. In Computer Vision and Pattern Recognition, pages 3426–3433, 2012.

[14] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan. Subcategory-aware object classification. In Computer Vision and Pattern Recognition, pages 827–834, 2013.

[15] Alexe, B., Deselares, T. and Ferrari. Measuring the objectness of image windows. V. PAMI 2012.

[16] R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, Arnold W. M. Smeulders. Selective Search for Object Recognition, Jasper International Journal of Computer Vision, Volume 104 (2), page 154-171, 2013.

[17] Dataset of CIFAR-10 https://www.cs.toronto.edu/ kriz/cifar.html.

[18] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In Computer Vision and Pattern Recognition, pages 237–244, 2009.

[19] Y. Gong, Y. Jia, T. K. leung, A. Toshev, and S. Ioffe. deep convolutional ranking for multi label image annotation. In International Conference on Learning Representations, 2014.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, volume 1, pages 886–893, 2005.

[21] T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. Pattern recognition, 29(1):51–59, 1996.

[22] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. IEEE Trans. Communications, 36(8):957– 971, 1988.

[23] P. Hedelin and J. Skoglund. Vector quantization based on gaussian mixture models. IEEE Trans. Speech and Audio Processing, 8(4):385–401, 2000.

[24] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. TPAMI, 24:603–619, 2002.

[25] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient GraphBased Image Segmentation. IJCV, 59:167–181, 2004.

[26] Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.