Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics

Master in Data Science

Deep learning technique for image classification by segmentation

---

THESIS to obtain the **DEGREE** of
MASTER IN DATA SCIENCE

A thesis presented by: **Abinaya Jayakumar**.

Thesis Advisor: **Dr. Fernando I. Becerra López**.

Tlaquepaque, Jalisco, May 2021

Contents

List of Figures

I want to dedicate this work to my father who encouraged me to pursue Master's and I also want to dedicate it to my husband who has been my pillar of support for completing my Master's.

1 Introduction

Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods that would be impossible by analogue means. In particular, digital image processing is a concrete application of, and a practical technology based on classification, localization, feature extraction and segmentation.

The main objective is to understand the following challenges and identify a solution. Without knowing the number of classes in the image it is difficult to find the number of segments in real world applications. Feature selection is a challenging process, as every image is represented by many features. Applying the exact segmentation method to find which will be suitable for the image data.

To understand the existing techniques, let us get some overview about how image segmentation techniques work. Suppose you want to know where an object is located in the image, the shape of that object, which pixel belongs to which object, etc. In this case you will want to segment the image, i.e., each pixel of the image is given a label. Thus, the task of image segmentation is to train a neural network to output a pixel-wise mask of the image. This helps in understanding the image at a much lower level, i.e., the pixel level. Image segmentation has many applications in medical imaging, self-driving cars and satellite imaging to name a few. Image Segmentation is the process by which a digital image is partitioned into various subgroups (of pixels) called Image Objects, which can reduce the complexity of the image, and thus analyzing the image becomes simpler.

We use various image segmentation algorithms to split and group a certain set of pixels together from the image. By doing so, we are actually assigning labels to pixels and the pixels with the same label fall under a category where they have some or the other thing common in them. Using these labels, we can specify boundaries, draw lines, and separate the most required objects in an image from the rest of the not-so-important ones. Image segmentation[1] is one of the hot spots in

[1] Jan Erik Solem. *Programming Computer Vision with Python*. O'Reilly, 2019

image processing and computer vision. It is also an important basis for image recognition. It is based on certain criteria to divide an input image into a number of the same nature of the category in order to extract the area which people are interested in. And it is the basis for image analysis and understanding of image feature extraction and recognition image segmentation is an essential but critical component in low–level vision, image analysis, pattern recognition, and now in robotic systems. Besides, it is one of the most difficult and challenging tasks in image processing and determines the quality of the results of the image analysis. Intuitively, image segmentation is the process of dividing an image into different regions such that each region is homogeneous while not the union of any two adjacent regions. An additional requirement would be that these regions had some correspondence to real homogeneous regions belonging to objects in the scene.

2 Theoretical framework

Contents

The chapter describes the different algorithms and their associated parameters that were used for image feature extraction and segmentation. Both machine learning and deep learning approaches are discussed in detail.

2.1 Image Localization

Image localization helps us to identify the location of a single object in the given image. In case we have multiple objects present, we then rely on the concept of object detection (OD). We can predict the location along with the class for each object using OD. We can divide or partition the image into various parts called segments. It is not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By

dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works. An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation.

A subset of artificial intelligence involved with the creation of algorithms which can modify itself without human intervention to produce desired output- by feeding itself through structured data. Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned.

Object detection builds a bounding box corresponding to each class in the image. But it tells us nothing about the shape of the object. We only get the set of bounding box coordinates. We want to get more information – this is too vague for our purposes. Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a far more granular understanding of the object(s) in the image.

2.2 *Machine Learning*

Machine learning[1] and computer vision are two fields that have become closely related. Machine learning has improved computer vision about recognition and tracking. It offers effective methods for acquisition, image processing, and object focus which are used in computer vision. In turn, computer vision has broadened the scope of machine learning. It involves a digital image or video, a sensing device, an interpreting device, and the interpretation stage. Machine learning is used in computer vision in the interpreting device and interpretation stage.

[1] Chapman and Hall. *Machine Learning 2nd edition*. Packt, 2014

2.2.1 *Thresholding*

Thresholding divides an image into a foreground and background. A specified threshold value separates pixels into one of two levels to isolate objects. Thresholding converts grayscale images into binary images or distinguishes the lighter and darker pixels of a color image.

2.2.2 *K-means clustering*

K-means clustering is an algorithm identifies groups in the data, with the variable K representing the number of groups. The algorithm assigns each data point (or pixel) to one of the groups based on feature similarity. Rather than analyzing predefined groups, clustering works iteratively to organically form groups.

2.2.3 Histogram-based image segmentation

Uses a histogram to group pixels based on gray levels[2]. Simple images consist of an object and a background. The background is usually one gray level and is the larger entity. Thus, a large peak represents the background gray level in the histogram. A smaller peak represents the object, which is another gray level.

[2] Oge Marques. *Practical Image and Video Processing Using MATLAB*. Wiley-IEEE Press, 2011

2.2.4 Edge detection

Identifies sharp changes or discontinuities in brightness. Edge detection usually involves arranging points of discontinuity into curved line segments, or edges. For example, the border between a block of red and a block of blue.

2.3 Deep Learning

Deep learning is a subfield of machine learning, and neural networks make up the backbone of deep learning algorithms. In fact, it is the number of node layers, or depth, of neural networks that distinguishes a single neural network from a deep learning algorithm, which must have more than three.

Deep learning structures algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own. Deep learning is a sub-field of machine learning. While both fall under the broad category of artificial intelligence, deep learning is what powers the most human-like artificial intelligence. Neural network are the building blocks of deep learning. Object recognition using deep learning does not need specifically defined features. The common approaches that use deep learning are based on convolutional neural networks (CNN).

A convolutional neural network is a type of deep neural network which is an artificial neural network with multiple layers between the input and output. Image segmentation with CNN involves feeding segments of an image as input to a convolutional neural network, which labels the pixels. It cannot process the whole image at once. It scans the image, looking at a small "filter" of several pixels each time until it has mapped the entire image.

2.3.1 Layers in CNN

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), pooling, fully connected layers (FC) and apply Softmax

function to classify an object with probabilistic values between 0 and 1[3].

[3] Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning

1. **Convolution layer** The convolution layer (CONV) uses filters that perform convolution operations as it is scanning the input II with respect to its dimensions. Its hyperparameters include the filter size FF and stride SS. The resulting output OO is called feature map or activation map. The convolutional layers classify every pixel to determine the context of the image, including the location of objects.

2. **Pooling layer** The pooling layer (POOL) is a down sampling operation, typically applied after a convolution layer, which does some spatial invariance. In particular, max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.

3. **Fully Convolutional Networks (FCN)** Traditional CNNs have fully-connected layers, which cannot manage different input sizes. FCNs use convolutional layers to process varying input sizes and can work faster. The final output layer has a large receptive field and corresponds to the height and width of the image, while the number of channels corresponds to the number of classes.

2.3.2 *Filters*

Filters are the core parameter for feature extraction of images in CNN. The filters are the matrix operators when convoluted around the image, extract the features[4]. Some concepts to know about filters are the following

[4] Afshine Amidi and Shervine Amidi. *cheatsheet-convolutional-neural-network*. https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

1. **Dimensions of a filter:** A filter of size F×F applied to an input containing C channels is a F×F×C volume that performs convolutions on an input of size I×I×C and produces an output feature map (also called activation map) of size O×O×1.

2. **Stride**: For a convolutional or a pooling operation, the stride SS denotes the number of pixels by which the window moves after each operation.

3. **Zero-padding**: Zero-padding denotes the process of adding P zeroes to each side of the boundaries of the input. This value can either be manually specified or automatically set.

2.3.3 *Activation Functions*

The choice of activation function in the hidden layer will control how well the network model learns the training dataset. Aside, choice of

activation function in the output layer will define the type of predictions the model can make. An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.[5]

Sometimes the activation function is called a "transfer function". If the output range of the activation function is limited, then it may be called a "squashing function". Many activation functions are nonlinear and may be referred to as the "nonlinearity" in the layer or the network design.

The choice of activation function has a large impact on the capability and performance of the neural network, and different activation functions may be used in different parts of the model.

Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.

A network may have three types of layers: input layers that take raw input from the domain, hidden layers that take input from another layer and pass output to another layer, and output layers that make a prediction.

All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model.

Activation functions are also typically differentiable, meaning the first-order derivative can be calculated for a given input value. This is required given that neural networks are typically trained using the backpropagation of error algorithm that requires the derivative of prediction error in order to update the weights of the model.

There are, perhaps three activation functions you may want to consider for use in hidden layers, they are:

- Rectified Linear Activation (ReLU)
- Logistic (Sigmoid)
- Hyperbolic Tangent (Tanh)

The output layer is the layer in a neural network model that directly outputs a prediction. All feed-forward neural network models have an output layer. There are perhaps three activation functions you may want to consider for use in the output layer, they are:

- Linear
- Logistic (Sigmoid)
- Softmax

[5] Afshine Amidi and Shervine Amidi. *cheatsheet-convolutional-neural-network*. https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

2.3.4 *Optimizers*

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses[6]. Some of the most used optimizers include the next:

[6] Richard Burton. *Hands on convolution neural network with tensorflow*. Packt, 2019

1. **Gradient descent:** It is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minimum. Through backpropagation, the loss is transferred from one layer to another, and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized. algorithm: $\theta - = \alpha \nabla J(\theta)$

2. **Stochastic gradient descent:** It is a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example.

3. **Momentum:** In momentum, what we are going to do is essentially try to capture some information regarding the previous updates a weight has gone through before performing the current update. Essentially, if a weight is constantly moving in a particular direction (increasing or decreasing), it will slowly accumulate some "momentum" in that direction. Hence when it faces some resistance and actually has to go the opposite way, it will continue going in the original direction for a while because of the accumulated momentum.

4. **Adagrad:** It is short for adaptive gradients. In this we try to change the learning rate (alpha) for each update. The learning rate changes during each update in such a way that it will decrease if a weight is being updated too much in a short amount of time and it will increase if a weight is not being updated much.

5. **RMSProp:** It is another adaptive learning rate that is an improvement of AdaGrad. Instead of taking cumulative sum of squared gradients like in AdaGrad, we take the exponential moving average of these gradients.

6. **Adam:** is kind of combining RMSProp with Momentum. First, we calculate our m value, which represents the momentum at the current point. Then, we calculate the accumulated cache, which is the same as it is in RMSProp.

.

2.3.5 *Residual Network*

A residual neural network (ResNet)[7] is an artificial neural network

[7] Kaiming He and Xiangyu Zhang. *Residual Network*. https://arxiv.org/pdf/1512.03385.pdf

(ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network. The Figure 2.1 shows the architecture of the ResNet model.

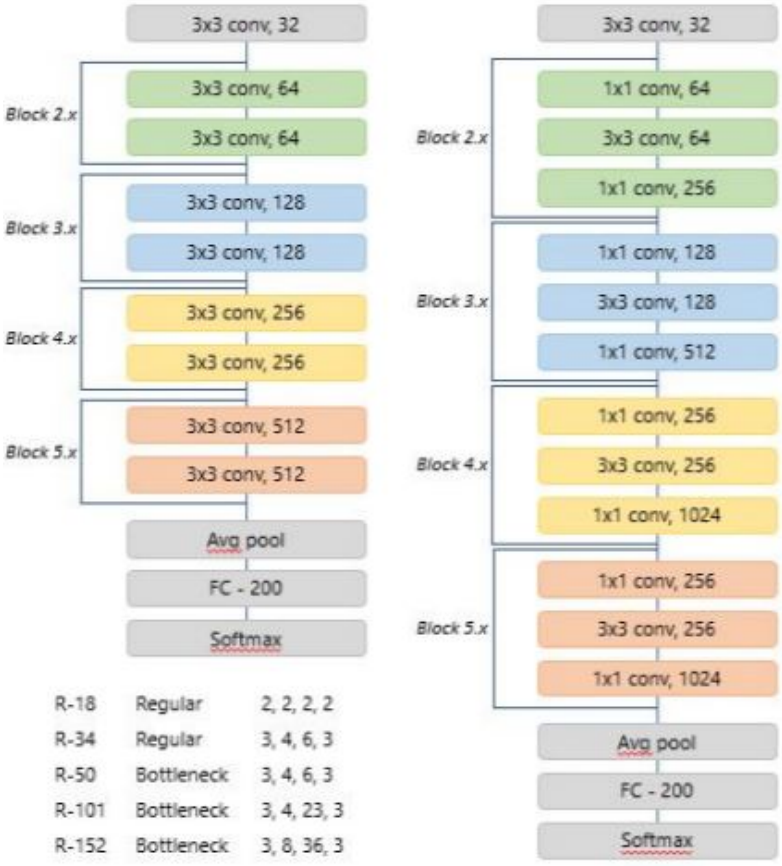

Figure 2.1: Resnet Architecture

2.3.6 *Regional Proposal*

Many pre-trained models are developed to directly use them without going through the pain of training models due to computational limitation. Many models got popular as well like VGG-16, ResNet 50,

DeepNet, AlexNet by ImageNet to generate these so called "proposals" for the region where the object lies, a small **network** is slide over a convolutional feature map that is the output by the last convolutional layer.

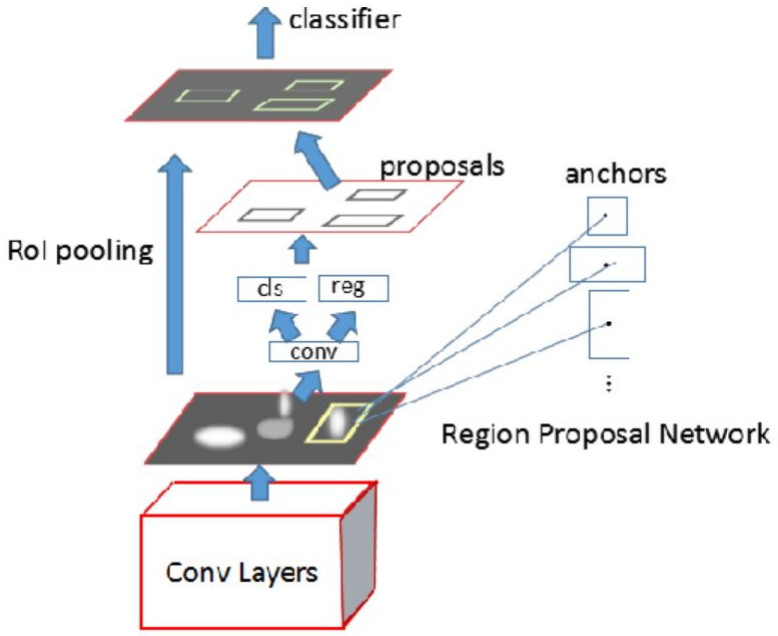

Figure 2.2: RPN architecture

The architecture of Region Proposal Network is shown in Figure 2.2. RPN generate the proposal for the objects. RPN has a specialized and unique architecture in itself. It is wanted to further breakdown the RPN architecture[8].

In the Figure 2.3 RPN has a classifier and a regressor. Anchor is the central point of the sliding window. For ZF Model, which was an extension of AlexNet, the dimensions are 256-d and for VGG-16, it was 512-d. Classifier determines the probability of a proposal having the target object. Regression regresses the coordinates of the proposals. For any image, scale and aspect-ratio are two important parameters. For those who do not know, aspect ratio = width of image/height of image, scale is the size of the image. The developers chose 3 scale and 3 aspect-ratio. So, total of 9 proposals are possible for each pixel, this is how the value of k is decided, K=9 for this case, k being the number of anchors. For the whole image, number of anchors is W*H*K. Presence of multi-scale anchors in the algorithm results in "Pyramid of Anchors" instead of "Pyramid of Filters" which makes it less time consuming and more cost efficient than previously proposed algorithms like Multi-Box. These anchors are assigned label based on two factors:

[8] Jan Erik Solem. *Programming Computer Vision with Python*. O'Reilly, 2019

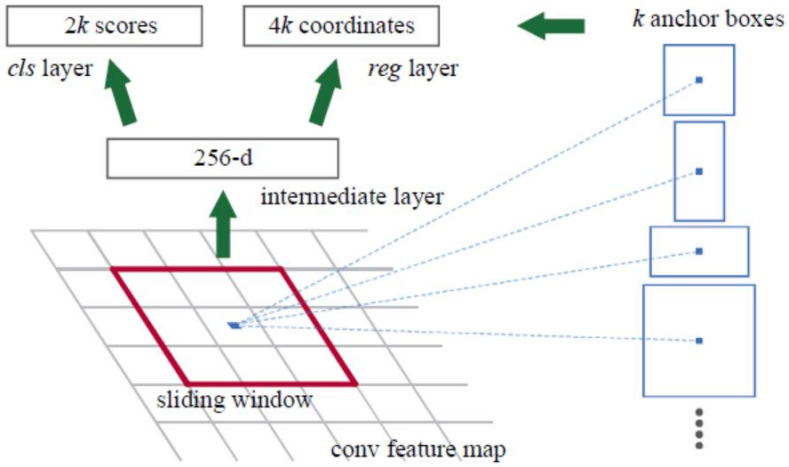

Figure 2.3: Classifier and Regressor

1. The anchors with highest Intersection-over-union overlap with a ground truth box.
2. The anchors with Intersection-Over-Union Overlap higher than 0.7.

Ultimately, RPN is an algorithm that needs to be trained. So, we definitely have our Loss Function p* with regression term in the loss function ensures that if and only if object is identified as yes, then only regression will count, otherwise p* will be zero, so the regression term will become zero in the loss function. In the equation 2.1 we see that Ncls and Nreg are the normalization. Default λ is 10 by default and is done to scale classifier and regressor on the same level. The equation 2.2 shows the final equation of the residuals that is obtained.

$$L(p_i, t_i) = (1/N_{cls})\Sigma L_{cls}(p_i, p_i^*) + (\lambda/N_{reg})\Sigma p_i^* L_{reg}(t_i, t_i^*) \tag{2.1}$$

$$L_{reg}(t_i, t_i^*) = R(t_i, t_i^*) \tag{2.2}$$

Thus, we were able to understand the various concepts of machine learning and deep learning algorithms like region proposal network, Residual Network, optimizer and activation functions.

2.4 *Dimensionality Reduction*

Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension. Working in high-dimensional spaces can be undesirable for

many reasons; raw data are often sparse as a consequence of the curse of dimensionality, and analyzing the data is usually computationally intractable. Dimensionality reduction is common in fields that deal with large numbers of observations and/or large numbers of variables, such as signal processing, speech recognition, neuroinformatics, and bioinformatics. Methods are commonly divided into linear and non-linear approaches. Approaches can also be divided into feature selection and feature extraction. Dimensionality reduction can be used for noise reduction, data visualization, cluster analysis, or as an intermediate step to facilitate other analyses.

2.4.1 *Principal Component Analysis*

Large datasets are increasingly common and are often difficult to interpret. Principal component analysis (PCA)[9] is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Finding such new variables, the principal components, reduces to solving an eigenvalue/eigenvector problem, and the new variables are defined by the dataset at hand, not a priori, hence making PCA an adaptive data analysis technique. It is adaptive in another sense too, since variants of the technique have been developed that are tailored to different data types and structures PCA focuses on Feature extraction as opposed to Feature elimination. Well, the dimensionality reduction occurs by introducing new independent variables and ignoring "least important" variables. In a nutshell, PCA combines the input variables in a specific way, drops the least important variables and finally retains the valuable parts of all the variables to ensure the new variables are independent of each other.

[9] Chapman and Hall. *Machine Learning 2nd edition*. Packt, 2014

Identifying the principal components is directly related to the variables in the data. The internals of PCA at a high level would look like:

- finds out the measure of association among variables using covariance matrix,
- obtaining the directions of data dispersion using Eigenvectors,
- calculating the relative importance of the directions by Eigenvalues.

2.4.2 *Autoencoder*

Autoencoders[10] are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Specifically, we will design a neural network architecture such that

[10] Ankur A Patel. *Hands on unsupervised learning using python*. O'Reilly, 2019

we impose a bottleneck in the network which forces a compressed knowledge representation of the original input. If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task. However, if some sort of structure exists in the data (i.e., correlations between input features), this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck.

2.4.3 *T-distributed Stochastic Neighbor Embedding*

T-SNE (T-distributed Stochastic Neighbor Embedding)[11] is an algorithm designed for the visualization of high-dimensional data sets. T-SNE is executed in two steps: first, it builds a probability distribution on pairs of samples in the original space, in such a way that similar samples receive a high probability of being chosen, while vastly different samples receive a low probability of being chosen. be chosen. The concept of "similarity" is based on the distance between points and density in the vicinity of a point. Second, T-SNE takes the points from the high-dimensional space to the low-dimensional space in a random way, defines a probability distribution similar to that seen in the destination space (the low-dimensional space), and minimizes the so-called divergence Kullback-Leibler between the two distributions with respect to the positions of the points on the map (the Kullback-Leibler divergence measures the similarity or difference between two probability distribution functions). In other words, T-SNE tries to reproduce the distribution that existed in the original space in the final space. In the next chapter the details about the data described. Breaking into parts:

- Stochastic: Not definite with random probability
- Neighbor: Retains the variance of neighbor points
- Embedding - Transforming data into lower dimensions

[11] Giussepe Boorcoso. *Mastering Machine Learning algorithms*. Packt, 2018

3 Data and Design

Contents

3.1 Data

The dataset involves images from different animals obtained from public datasets repositories and each image is of different shape and size. There are about 10 different classes of animals and birds and overall, around 26000 images used for this model building.

The idea is to have images of different quality as well, so that the model understands images from different standards. As part of making the model more diverse, images with one or more objects are included as part of training.

We can see data covers different scenarios like the Figure 3.1 where majority of the picture is background. When a normal CNN model is used for this, it will contain more features about the background rather than the features of the elephant. But since we use Region Proposal Network it is able to concentrate more on the elephants and the features obtained are closer to the objects in the image rather than the background.

Also, we see the Figure 3.2 where the background is very dense and has more noise. Here some of the features of the elephant are hidden by the dense background but still our model is able to capture all the features associated.

Likewise, the Figure 3.3 shows a distinct outline for the elephant and hence the model will be able to easily capture the features of the elephant. Since this model has a very distinct outline the noise is much reduced here and this will be a representative feature of the elephant when compared with any other elephant images

The other scenario is the Figure 3.4 where the person and the horse are together in the picture but still the model is able to capture the

Figure 3.1: More background rather than the image objects

Figure 3.2: Dense background with noise the features of the elephant are overlapped by the background adding noise

Figure 3.3: Picture with distinct outline easy for the model to extract the features of the elephant

features of the horse. Though we have two objects in this picture the model gives more importance to the features of the horse as it captures the features more effectively.

The Figure 3.5 has a butterfly which is very small as compared to the bark of the tree but still the model identifies the butterfly and its features.

Figure 3.4: Image consisting of more than one class

Figure 3.5: Picture showing a small sized class object

3.2 *Design*

The model is built as a Pytorch model and the specifications are as follows:

- Programming language: Python - 3.7.10
- Pytorch - 1.8.0+cu101: PyTorch is an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab. It is free and open-source software.
- scikit-learn - 0.22.2.post1: Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn.
- matplotlib - 3.2.2: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.
- opencv-python - 4.1.2.30: OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.
- Pillow - 7.0.0: Python Imaging Library is a free and open-source additional library for the Python programming language that adds

support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

The IDE is Google colab and the code is maintained in G-drive.

4 Approaches

Contents

The different approaches that were implemented and there corresponding results are discussed here. There are few widely used approaches and strategies that usually perform out of the box and solve the problem. However, that is not the case always.

The following approaches are attempted as part of the thesis and we have summarized the same. Please note that this section is not a deep dive documentation about the approaches, rather a high-level introduction.

4.1 Principal Components Analysis

Principal component analysis (PCA) is a dimensionality reduction technique or method that is used to reduce the dimensions of large data sets. This is achieved by transforming large variables into smaller ones, but still containing most of the information. In other words, PCA helps to reduce the dimensions of feature space to have fewer relationships between variables.

In Figure 4.1 we see that the classes dog, chicken, cat and cow are all being grouped together. Also horse, elephant and few samples of spider are segmented together. Since the images have nonlinear features and PCA is a linear dimensionality reduction mechanism it is not able to capture the segments properly. The feature reduction strategy is linear and completely based on the measurement of variance. For image processing, such linear transformations might not perform well, and non-linear reduction techniques help to extract complex features.

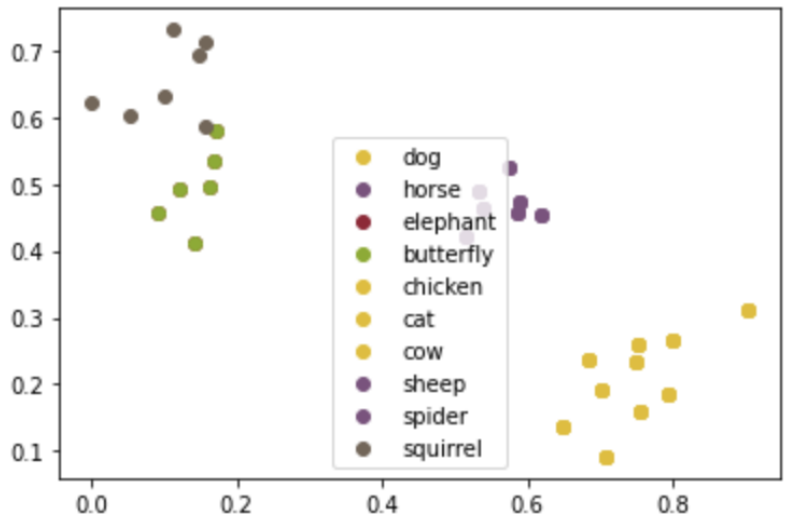

Figure 4.1: PCA image segmentation

4.2 *Autoencoder*

Autoencoder[1] falls into the ocean of unsupervised learning which takes any input, breaks into compression chunks and uses that to reconstruct the input to its original state. Ideally the reconstruction phase heavily depends on correlations between input features. Most of the images have noisy data and if such data is removed, we might end up in the important features of the image. Autoencoder by its definition the compression phase can help in achieving the same. The compressed data lies in the hidden layer, from which the original image can be reconstructed. This is out of scope as we want to focus on extracting important features. There are other use cases that can be derived from the hidden layer such as denoising the image data. However, such salient features can be obtained by using variant of autoencoder called as undercomplete autoencoder.

We desperately wanted autoencoder to perform better than PCA and hoping we get the best of the features in the hidden layer. Well, it did perform better but not to the expectation. The image segmentation overall is evaluated on the distance between the clusters and it was subsequently higher. As we see in Figure 4.2 though the model can give 10 different classes, the distance among different cluster points is not separated well and many of the clusters are overlapped.

[1] Ankur A Patel. *Hands on unsupervised learning using python*. O'Reilly, 2019

4.3 *T-Distributed Stochastic Neighbor Embedding*

The T-Distributed Stochastic Neighbor Embedding (T-SNE)[2] falls into non-linear technique of dimensionality reduction for high dimensional

[2] Giussepe Boorcoso. *Mastering Machine Learning algorithms*. Packt, 2018

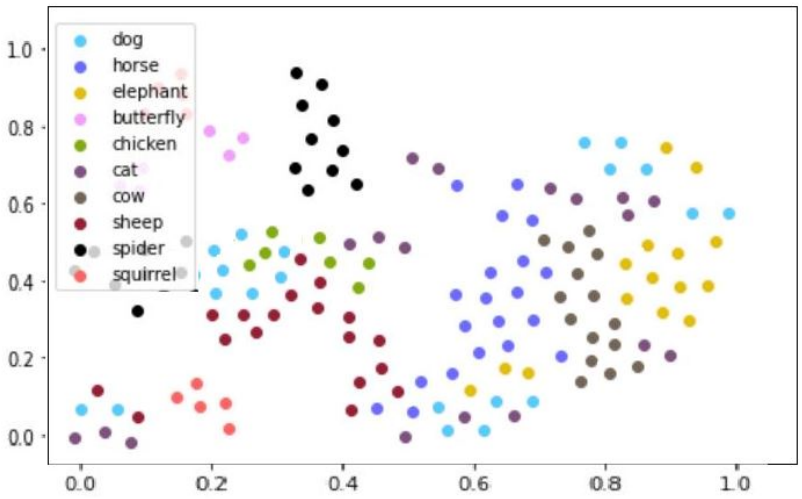

Figure 4.2: Autoencoder segmentation

images. By book, T-SNE maps multi-dimensional data into lower dimensional space and attempts to find patterns based on similar data points.

In the Figure 4.3 the similar images are being grouped together and the distance among different segments are more. T-SNE Calculates the probability of similar points in higher dimensions and calculates the probability of similar of points in lower dimensions. Similar objects are assigned a higher probability and dissimilar objects with a lower probability. Tries to minimize the difference between the probabilities to represent the data points in lower dimensional space. It minimizes the sum of KL Divergence to achieve the same. Upon training and testing using T-SNE, we got better results in segmentation similar images. The distance between clusters is considerably higher than autoencoder. Also, T-SNE has reduced the features into just two features which resulted in distinctive clusters.

4.4 *RESNET model*

The RESNET (Residual Neural Network)[3] model was used for identifying the different classes, by using the layers of this model the image features are extracted. The extracted features are given as an input to the T-SNE algorithm which is able to segment the images based on the distinct features. In Figure 4.3 we see the segmentation of the images. Hence most of the similar points are placed together in a single segment. The solution obtained is such that the distance among the points in the same segment are less when compared to the distance between the points in different segments. The Figure 4.4 gives

[3] Kaiming He and Xiangyu Zhang. *Residual Network*. https://arxiv.org/pdf/1512.03385.pdf

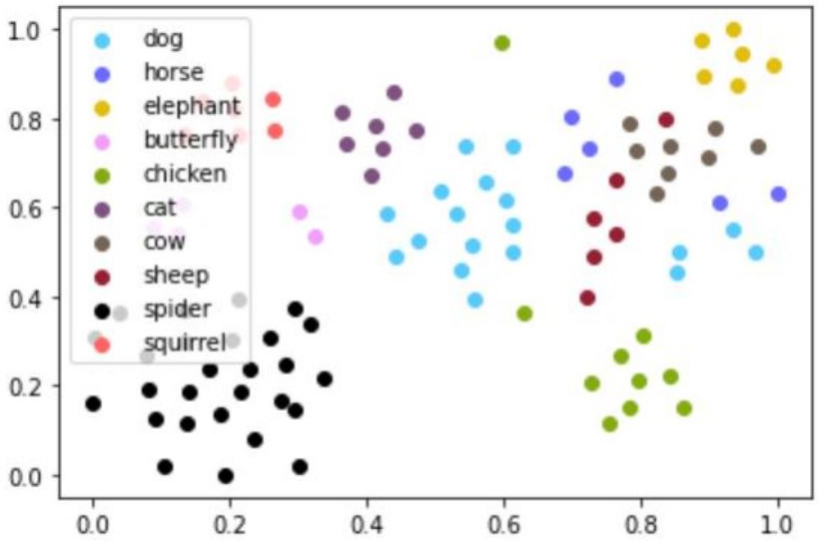

Figure 4.3: TSNE image segmentation

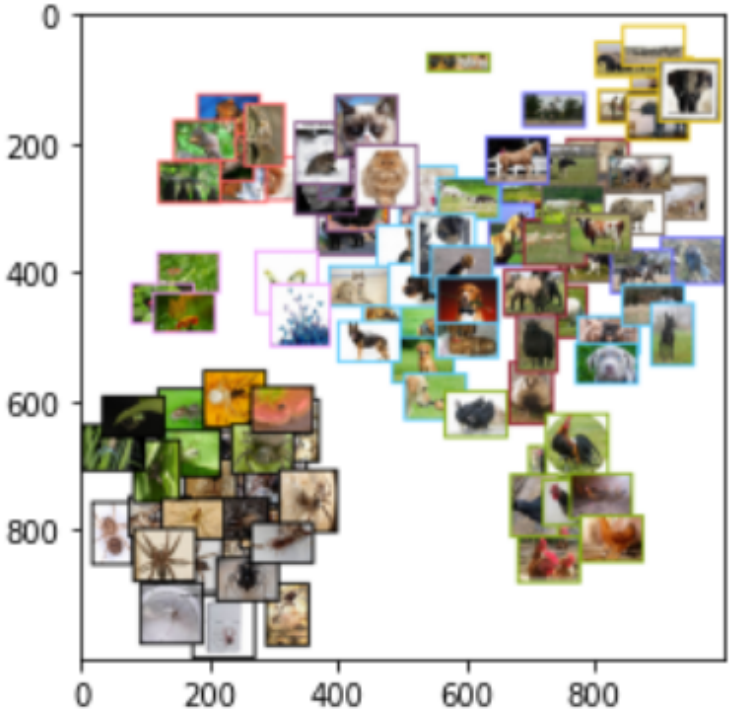

Figure 4.4: Visualization with the input images

us a better visualization of the output. We can note that here only few images of dog are overlapped with other segments of chicken, horse and sheep. In the upcoming details the behavior of this unusual pattern is described.

Here we can note that some of the dog images are placed near a different cluster which has more horse images the behavior for this is due to the image Figure 4.5 which is a dog but more similar like a horse. Though it is a dog it is very giant and if we consider just the outline of this image its features will resemble more like a horse feature. Due to this the model output has placed this image in between the dog segment and the horse segment.

Also, we note that some of the dog image is related to the cluster that has chicken images. The Figure 4.6 is more resembling like a cock so the convolution neural network has learned the features of a cock and identified it has a cock, hence the model places this image such that it is closer to both cock segment and dog segment.

The Figure 4.7 is more similar to sheep so the convolution neural network learns its features as a sheep and the segmentation is done such that the image is closer to the sheep segment. Thus, we can see that RESNET model is able to capture the features correctly but, in some cases, when the features overlap the T-SNE algorithm places those points in between the overlapping segments.

Figure 4.5: Tall giant dog with its features similar to a horse

Figure 4.6: Dog similar to the shape of a cock

Figure 4.7: Dog similar to the sheep features

5 Conclusions

We implemented Image segmentation such that similar images are being grouped in same segment and the distance between non similar images are more. The Region proposal architecture seems to be better in extracting the image features when compared with other models since the regions proposals reduce the noise in the image and concentrate on the actual objects in the images.

On feature reduction Principal component analysis was a linear approach and was not suitable for nonlinear features like the image features. Auto encoder decoder created more noise in the obtained features. Finally, T-SNE was a better approach and grouped similar objects together.

In the future we need to extend the model to do clustering of other kinds of images like medical images. Currently this project scope is based on semantic segmentation. Instance based segmentation may not be obtained using T-SNE.

Bibliography

Afshine Amidi and Shervine Amidi. *cheatsheet-convolutional-neural-network*. https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

Giussepe Boorcoso. *Mastering Machine Learning algorithms*. Packt, 2018.

Richard Burton. *Hands on convolution neural network with tensorflow*. Packt, 2019.

Chapman and Hall. *Machine Learning 2nd edition*. Packt, 2014.

Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning.

Kaiming He and Xiangyu Zhang. *Residual Network*. https://arxiv.org/pdf/1512.03385.pdf.

Oge Marques. *Practical Image and Video Processing Using MATLAB*. Wiley-IEEE Press, 2011.

Ankur A Patel. *Hands on unsupervised learning using python*. O'Reilly, 2019.

Jan Erik Solem. *Programming Computer Vision with Python*. O'Reilly, 2019.