DISTRIBUTED COLLABORATIVE FRAMEWORK FOR DEEP LEARNING IN OBJECT DETECTION

A THESIS IN
Computer Science

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment
of the requirements for the degree

MASTER OF SCIENCE

By
SIRISHA RELLA

Vellore Institute of Technology, Vellore,632014, India

Kansas City, Missouri
2020

DISTRIBUTED COLLABORATIVE FRAMEWORK FOR DEEP LEARNING IN OBJECT DETECTION

Sirisha Rella, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2020

ABSTRACT

Object detection has gained much attention in recent years because of its ability to localize and classify the objects in videos and images that can be incorporated into many applications. Traditional object detection algorithms need substantial computational resources to build a model. There is an increasing demand for a practical approach to constructing object detection models adapted to the local context, limited computing resources, and application logics while supporting real-time inferencing without degrading accuracy and performance.

In this thesis, we proposed a distributed-collaborative framework to build practical object detection models. The framework is based on a novel approach for collaborative group inferencing that is designed with a single-class-single-model mechanism for multiple objects in a distributed manner. For useful grouping, we made use of the intraclass correlation from existing models during inferencing. Results from the case studies with Pascal VOC 2007 and our data collected through Google Street View showed that the proposed model significantly improved performance while making it potentially suitable for customized application building with limited computing resources. A prototype for the proposed model has been built, and a neighborhood application has been demonstrated to validate the proposed work.

.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled “Distributed Collaborative Framework for Deep Learning in Object Detection” presented by Sirisha Rella, candidate for the Master of Science degree, and certify that in their opinion, it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D. (Committee Chair)
Department of Computer Science Electrical Engineering

Zhu Li, Ph.D.
Department of Computer Science Electrical Engineering

Yusuf Sarwar Uddin, Ph.D.
Department of Computer Science Electrical Engineering

Table of Contents

LIST OF ILLUSTRATIONS

Figure Page

ACKNOWLEDGMENTS

I want to thank Dr.Yugyung Lee for her valuable and constructive suggestions throughout the planning and development of this research work. Her constant support, guidance, and feedback helped me a lot in finishing the research project. I appreciate all the learning opportunities and help provided by Gharib Gharibi, Rajaram Anantharam, Duy Ho throughout my master's program at UMKC.

I would also like to extend my thanks to Dr. Zhu Li and Dr. Yusuf Sarwar Uddin for taking out time to attend my thesis defense. It is an honor to present my thesis in front of two reputed faculties at the University of Missouri-Kansas City. I am grateful to be part of the National Science Foundation – Center for Big Learning (NSF-CBL), and UMKC Distributed Intelligent Computer Association (UDIC) lab as a research contributor that facilitated me to enhance my skills.

Finally, I would like to express my gratitude to my parents for always encouraging me to pursue my dreams regardless of all the difficulties. I appreciate my friends for endless support and always helping me to make the right decisions in my master's career.

CHAPTER. 1 INTRODUCTION

Deep Learning [1] has become popular since 2010 [2] because of the availability of massive datasets and computational resources. There are several domains of computer vision [3] in deep learning, such as image classification, object detection, and image segmentation. The availability of computational resources and advancements in object detection facilitated many things in today's world. It is widely used in real-time applications such as self-driving cars and NeighborNet [4]. Detecting objects in real-time helps to identify the obstacles and take the right direction in self-driving vehicles, whereas in NeighborNet helps to analyze neighborhood efficiently.

Popular deep learning frameworks such as Tensorflow [5], PyTorch [6], Gluon [7], and Keras [8] made building deep learning models easy. These models need to be trained hundreds and thousands of hours with a large corpus of data on high-end GPUs [9] before going to the production environment. The models thus trained are huge and should run on high-end GPUs for inference.

As the proposed framework is evaluated in computer vision [3] domain, the typical pipeline of any computer vision deep learning model has four stages, namely data pre-processing, training, deployment, and inference. Computer vision datasets [10] are available as open-source for training. Dataset pre-processing steps such as resizing the image, scaling, and data augmentation implemented, and the output of data-preprocessing fed to a neural network for training. However, the entire process is not as easy as it sounds. It requires high-end GPUs and takes much time to build such models.

1.1 Problem Statement

This section describes problems in current object detection algorithms. Conventional object detection aims at localizing and classifying the objects in an image, and label them with a rectangular bounding box along with the confidence score. A typical object detection framework follows two approaches. The traditional method involves generating a region-proposals at first and then classifies the region proposals. The second approach follows a unified plan of creating and organizing the objects in an image.

Object detection models [11] should be trained on a large amount of data and need high-computational GPU support for training and inference. Several thousands of hours of training is required to get an accurate model. The costly and inefficient training of models is because of the lack of a distributed collaborative environment for building and testing deep learning models. We also require efficient algorithms to optimize the building of these models. The models constructed thus should be evaluated based on several parameters to work efficiently in real-world applications. The proposed framework helps developers to build cost-efficient, high-performance, and customized object detection models.

1.2 Proposed Framework

In this paper, we proposed a solution, distributed, collaborative framework for object detection in deep learning. We aim to provide efficient methodologies to build and test object detection models in the distributed environment in a cost-effective way. Our framework also helps the user to customize their models in the future.

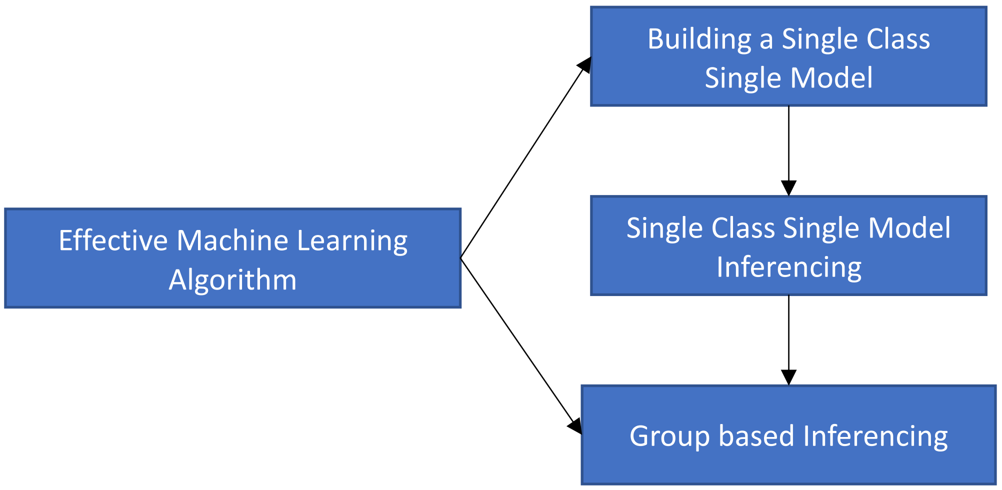

Building a Single Class Single Model
Effective Machine Learning Algorithm
Single Class Single Model Inferencing
Group based Inferencing

Figure 1 Proposed Framework Architecture

The proposed framework focuses on achieving two objectives. Firstly, to build a cost-effective Single Class Single Model (SCSM) in a distributed manner. Secondly, to perform distributed collaborative inferencing using SCSM-based and group-based approaches. The framework utilized an efficient machine learning algorithm approach to create cost-efficient and high-performance deep learning object detection models.

We evaluated all the proposed methodologies with Pascal VOC and neighborhood datasets. The neighborhood dataset consists of all the objects present in communities such as houses, apartments, and garage developed at UMKC Distributed Intelligent Computing group (UDIC). Several metrics, such as recall, precision, f1-score, and mean average precision (MAP), are taken into consideration to evaluate the deep learning models.

CHAPTER. 2 BACKGROUND & RELATED WORK

This chapter provides background information and related work about various essential items such as deep learning, different algorithms in the computer-vision domain, and algorithms in object-detection. We also talk about pre-trained models for an efficient transfer learning approach that helps to gain an understanding of the problem statement.

2.1 Background

2.1.1 Deep Learning

This component explains the internal architecture and functions of Deep Neural Networks (DNN). Deep neural networks or Deep Learning [1] is one of the things that changed the world. Deep Learning is a subset of machine learning which takes an input, draws a pattern, and produces the output, whereas machine learning [12] has statistical algorithms to produce an output. One of the reasons for deep learning became popular than machine learning is, it draws pattern and understand the features by itself and produces an output. For instance, to classify if an image is a dog or cat, for a machine learning model to train, we need to explicitly pass all the features, whereas, in deep learning, the features are learned by the model itself during the training process.

Deep learning is also called deep neural networks is a layered network that imitates the human brain. There are three crucial layers in deep learning neural networks, namely input layers, hidden layers, and output layers. Each layer comprises of a set of neurons, and these neurons are the core processing units of a deep learning model.

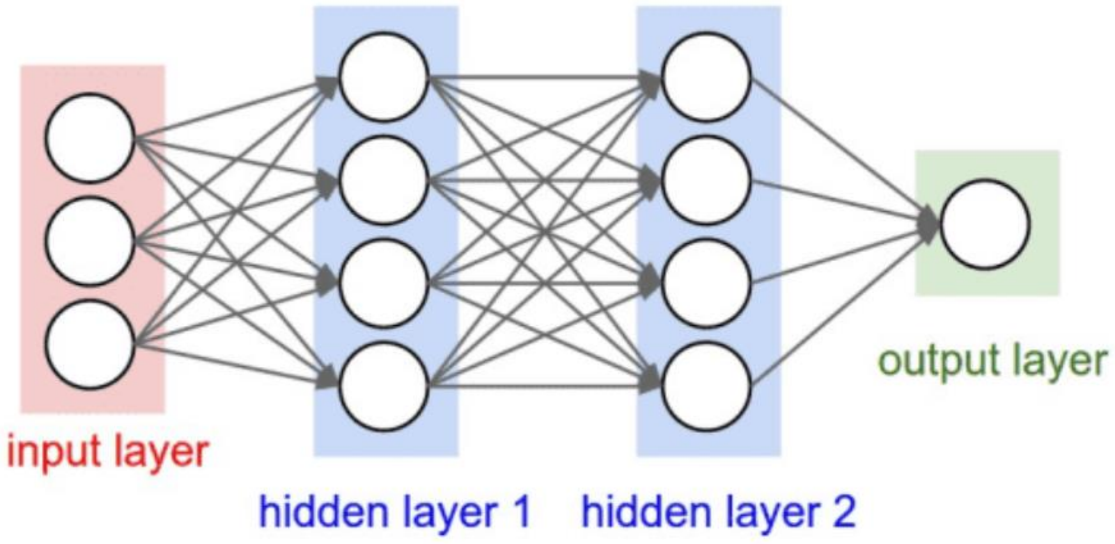


Figure 2 Feed Forward Neural Network Architecture [42]

The input layer receives the input, and the output layer predicts the output. In between exists the hidden layers which perform most of the computations required by our network. For instance, we have an image of a circle made of 28 * 28 pixels, which make up for 784 pixels. Each pixel is given as input to each neuron of the first layer of the neurons. Every neuron of the first layer connects to every other neuron in the next layer through channels—these channels associates with a numerical value known as weight. The inputs multiply with corresponding weights, and the sum passes as an input to the neuron in the hidden layer.

Each of these neurons in hidden layers is associated with bias, which adds to the input sum. This value transfers to a threshold function called the activation function [13]. The activation function defines the activated neuron. An activated neuron transmits the data to the neurons of the next layer over the channels. In this manner, the data propagates in the network. It is called feed-forward propagation. In the output layer, the neuron with the highest value fires and determines the predicted output. The values are the probability here.

The predicted output is compared against the original production to predict the error in the prediction. The magnitude of the error suggests if our predicted values are lower or higher than expected. This calculation is called the loss function [14]. The optimizer function calculates the magnitude and direction of the channels to reduce the error rate. This information is transformed backward through our network, which is called backpropagation [15]. In this backward propagation mechanism, weights and biases adjust accordingly. This forward and backward propagation performs iteratively with multiple inputs. This process continues until the network predicts most of the shapes correctly.

2.1.2 Convolution Neural Networks

We elaborate on a type of deep learning, which is a convolutional neural network in this section. Convolutional neural network (CNN) [16], aka Convnet, is an artificial neural network so far been most popularly used for analyzing images. Although image analysis has been the widespread use of CNN's, we can use them in data analysis or classification problems. Commonly, CNN is an artificial neural network that has some knowledge for being able to pick out or detect patterns and make the idea of them. These pattern detections are what makes CNN most useful for image analysis. The CNNs have hidden layers called convolutional layers, and these layers make CNN helpful in detecting the patterns. CNN's can also have non-convolutional layers as well in the architecture. However, convolutional layers are the building blocks of CNNs.

Like other hidden layers, the convolutional layer transforms the received input and then outputs the processed data to the next layer. This transformation is called a convolution operation. Convolution operations use for detecting the patterns in the images. Precisely, with each convolutional layer, we need to specify the number of filters the layers should have to identify the patterns. Patterns are what goes in an image such as edges, shapes, textures, and objects. One type of pattern that a filter could detect edges in images. So, this filter is called edge detection. Some other filters can detect corners, circles, squares. These are the geometric filters we see at the starting of the network. The deeper the neural network goes, the more complex these filters become. So, in later layers, rather than edges in simple shapes, filters may be able to detect specific objects like eyes, ears, hairs, and feathers. In deeper layers, the filters can discover even more sophisticated objects like full dogs and cats.

To understand the filters in more detail, let us see an example. For instance, we have a convolutional neural network that is accepting images of handwritten digits from the popular MNIST [17] digits. Our network is classifying them into respective categories of whether the images of 1,2 and 3. Assuming the first layer in our model is a convolutional layer when adding this layer to a model, we also must specify how many filters we want the layer to have. A filter can technically be a matrix where we decide the number of rows and columns that the matrix has. The values inside the matrix initialize with random numbers. In our example, let us specify we want our matrix to be of 3*3. When the convolutional layer receives the input, the filter slides over each 3*3 set of pixels from the input itself until it slides over every 3*3 block of pixels from the entire image. This sliding is referred to as convolving. We should say that the filter is going to convolve across each 3*3 block of pixels from the input.

Once the convolution operation performed, the next step in CNN's is pooling. Pooling adds to CNN's following each convolution layer. There are different types of pooling, such as max-pooling [18], min pooling, and average pooling. Max pooling reduces the dimensionality of the images by reducing the number of pixels in the convolution layers output.

.

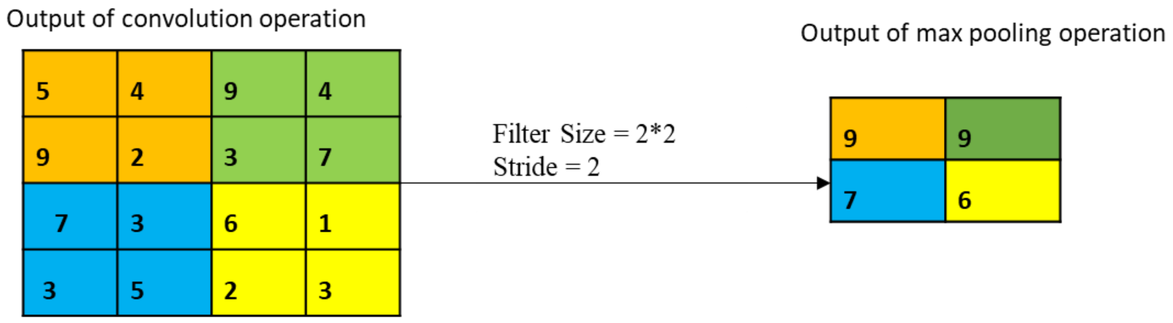

Output of convolution operation
5
4
9
4
9
2
3
7
7
3
6
1
3
5
2
3
Filter Size = 2*2
Stride = 2
Output of max pooling operation
9
9
7
6

Figure 3 Max Pooling Operation

The above diagram shows the calculation of max-pooling operation on the output of the convolution operation. Several domains in CNN's include image classification, object detection, and instance segmentation. Our proposed study evaluated the framework in the object detection domain. We describe more details about object detection in further sections.

2.1.3 Image Classification

This section describes the importance of image classification and how a classification algorithm is built using CNN's in Keras [8]. In general, image classification is classifying the categories of images using CNN. There are popular deep learning frameworks such as Keras [8], TensorFlow [5], and MXNET [19] to build image classification algorithms. For instance, to create a model that can classify dog or cat using Keras framework. Firstly, we need to download a dataset and prepare the structure of the dataset for training. A typical deep learning model divides the dataset into two folders, namely train and validation. We can use the training dataset to pass the images to the model for training and validation is used to build an accurate model by evaluating the trained model against validation images.

There are high-level APIs in Keras, PyTorch, and Gluon to build an image classification algorithm. To import libraries for data processing and to create a neural network, we have ImageDataGenerator and convolutional layers such as Sequential, Conv2D, MaxPooling2D, Activation, and Dropout in Keras framework. Once we have built a model, we have a fit method to train the model and save method to save the trained model.

The frameworks have made building deep learning models easy. There are similar methods in other frameworks that enable us to build deep learning models. However, this entire process of training a model takes several hours depends on the size of the dataset.

2.1.4 RESNET

We describe the ResNet classification CNN algorithm in this section, as we utilized this as a part of the object detection model to evaluate the proposed framework. ResNet [4] is a convolution neural network trained on the ImageNet database. It is one of the most popular neural network classification algorithms. Deep Learning is thought of as a hierarchical set of representations such that it learns low, mid, and high-level features. In images, this is analogous to learning like edge, shapes, and then objects. Theoretically, more layers should enrich the levels of the features, and previous models to Resnet have depths of 16 and 30 layers typically. So, the idea is should not be a building of neural networks because as adding more layers to the network. The first contribution of Resnet paper shows that stacking the convolution layers on top of activations and batch normalization; the training eventually gets worse, not better.

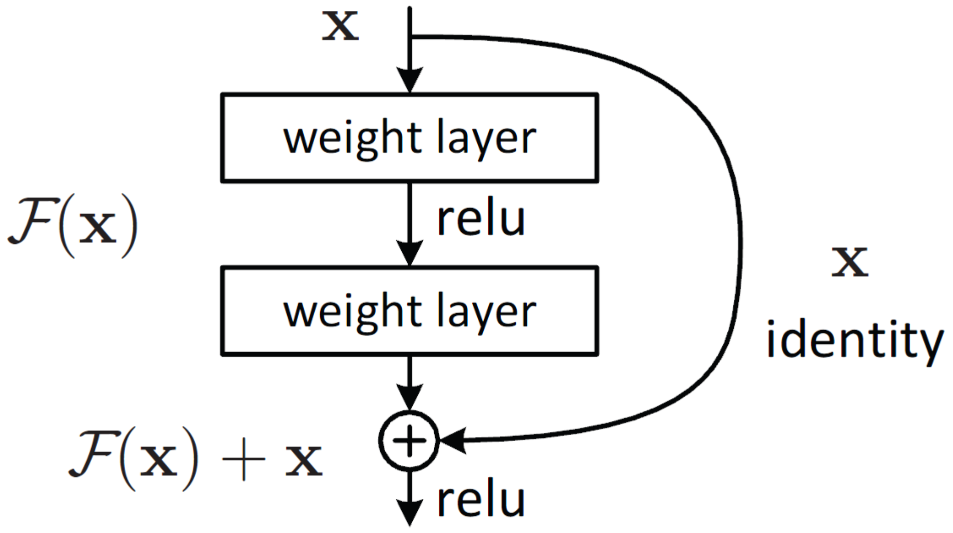


Figure 4 Residual Block in ResNet-50 [43]

To address this problem, Resnet introduced residual blocks in its architecture. Residual blocks are also known as shortcut connections. The intuition behind residual blocks is to retain the maximum accuracy at each layer. It does by considering the outputs at each stage along with the outputs of the shallow layer. There are different versions of ResNet such as ResNet-18, ResNet-34, ResNet-50, and ResNet-101. The value next to hyphen states the number of layers that a deep learning ResNet model contains. We used ResNet-50 to evaluate the proposed framework as it is a more accurate and lightweight model.

2.1.5 Object Detection

The proposed research work is conducted in the object detection domain. This section explains different object detection algorithms. It is necessary to understand about image classification and object localization to understand the object detection. We already studied image classification in the earlier sections.

Object localization uses the bounding boxes by localizing the position of the image in an image. To perform object detection, one must combine both classification + localization to detect the objects in images. There are several object detection algorithms out there, such as single-shot object detection, YOLO (you only look once), R-CNN, Fast R-CNN, and Faster R-CNN. The current study evaluates in object detection domain using Faster R-CNN.

2.1.6 Regional-Convolutional Neural Network

R-CNN stands for Regional-CNN. In R-CNN, they try to find out the regions and send them to a convolutional neural network to predict whether a region belongs to a specific class or not. The way it works is, it generates areas for each image based on the selective search algorithm, normalizes the image to a fixed size, and then will be sent to a convnet for extracting the features. Once the features are extracted, the output will be sent to support vector machines for classification. The bounding box regressors are used along with support vector machines (SVM) to refine the localization boxes.

In the selective search method, they merge the different pixels with neighboring pixels, and as they combine more and more, they get well-defined regions. In the training process of RCNN, we have to remove the last layer of the fully trained network such as Alexnet, resnet50, which has 1000 classes and replace it with the 20 types where 20 would represent the classes in Pascal VOC dataset. We then take the 2k regions of an image generated by a selective search algorithm and send it to a convolutional neural network and extract the features for that image.

Once we have those features, we do SVM classification on top of it. Besides, SVM classification, we can also add a corrective measure such as bounding box regressors to correct and learn from the imperfect bounding box detections.

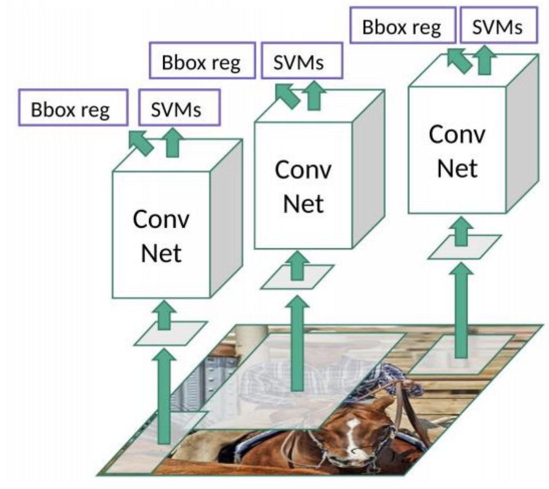


Figure 5 Regional-Convolutional Neural Network [44]

The Regional-CNN takes several hours to train as it generates and sends 2k regions for every image to convnet for feature extraction. To process feature extraction for every 2k images per image is time-consuming and wastage of computational power. To solve this problem, efficient algorithms such as Fast RCNN and Faster RCNN came into existence.

2.1.7 Fast Regional-Convolutional Neural Network

Fast RCNN is like the RCNN algorithm. However, in Fast RCNN, instead of running all the proposed image regions through the network independently, we need to run the entire image at one go. The way it works is we take the whole input image, send it to the convnet, which will output a feature map from the last layer. These feature maps are used to generate regions of interest.

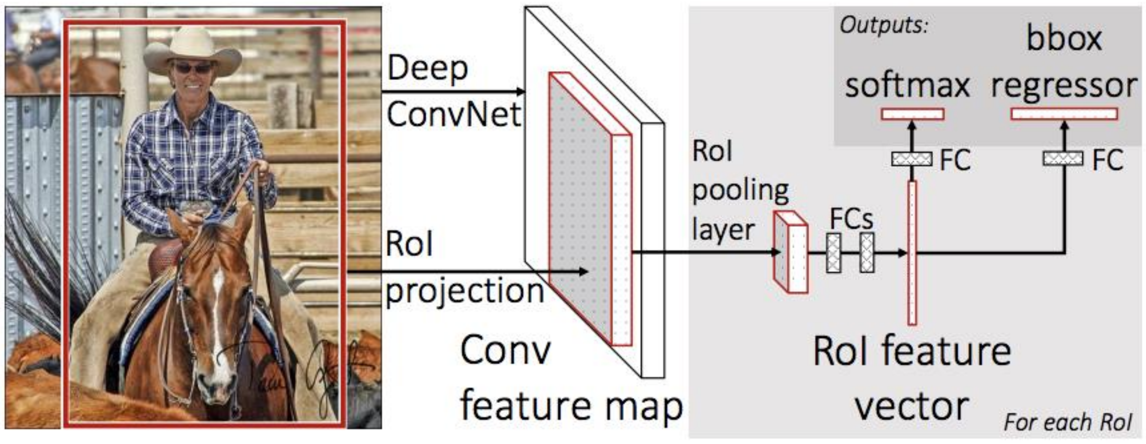


Figure 6 Fast Regional-Convolutional Neural Network [44]

In regions of interest, they scaled the size of the bounding box to the size of the feature map using region of interest pooling layer. Now, in the proposed areas that have been given by the selective search algorithm, we create a smaller/fixed part for the further layers for fully connected layers as they only accept fixed-size input. Once they fed to the fully connected layers, they calculate a SoftMax calculation way to find out the classes that it will belong to for the object detected. They also have a linear loss for the bounding box regressions.

To generate region proposals using a selective search algorithm is time-consuming and affects the performance of the network. Faster RCNN solves this problem by introducing an efficient algorithm replacing the selective search algorithm.

2.1.8 Faster Regional-Convolutional Neural Network

Faster R-CNN is a better approach compared to RCNN and Fast RCNN. A region proposal network replaces a selective search algorithm in Faster R-CNN. RPN is a neural network that learns to propose regions. After the convolutional layers generate the feature map, we need to slide a window upon it and create k anchor boxes on it. The anchor boxes are the proposed regions that are generated by the region proposal network.

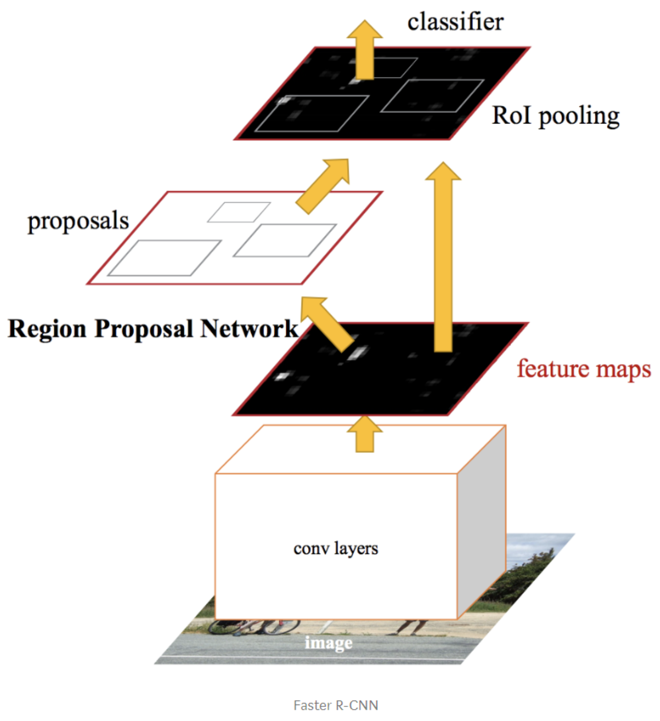


Figure 7 Faster Regional-Convolutional Neural Network [44]

In the Faster R-CNN paper, they have used three different scales with three different aspect ratios to generate the anchor boxes. In general, region proposals run a sliding window of size 3*3 on top of the feature map and create k anchor boxes. Those anchor boxes have to be classified whether they contain the object or not and also try to improve the bounding box coordinates for them. There are four losses in Faster R-CNN. Firstly, the classification loss for region proposal network, which tells there is an object or not. The bounding box proposal loss and object loss from the classification determines whether which class this object belongs to, and then there is another bounding box loss on top where we try to refine the bounding boxes that have been created by RPN. By comparison, Faster R-CNN is 250X times faster than RCNN and 25X times faster than RCNN. We have used the Faster R-CNN object detection model to evaluate the proposed framework.

2.1.9 Transfer Learning

We have optimized the proposed framework-based models using an optimized transfer learning approach. So, this section of the paper describes the traditional transfer learning algorithm. Without worrying much about data we have, transfer learning [20] is the fastest and easiest way to build a deep learning model. Training a deep learning model needs massive data and computational resources. In many cases, transfer learning enables us to take a model off the shelf and adapt it to our problem. For instance, we can make a model that has trained for one task, such as classifying objects and then fine-tune it to accomplish another task such as classifying scenes. Transfer learning is particularly prevalent in computer vision relevant tasks. Researches have shown that features learned from massive image sets, such as the ImageNet, are extremely interchangeable to a variety of image recognition tasks.

There are numerous ways we can transmit knowledge from one model to another. The easiest way is to remove the top layer of the trained model and initialize it with a randomly initialized one. Train only the top layer for the new task, while all other parameters remain fixed. This transferred model can be used as a feature extractor since the fixed parts can act as a feature vector, and the last layer acts like a traditional, fully connected layer. This methodology works best if our data and task are the same as the data.

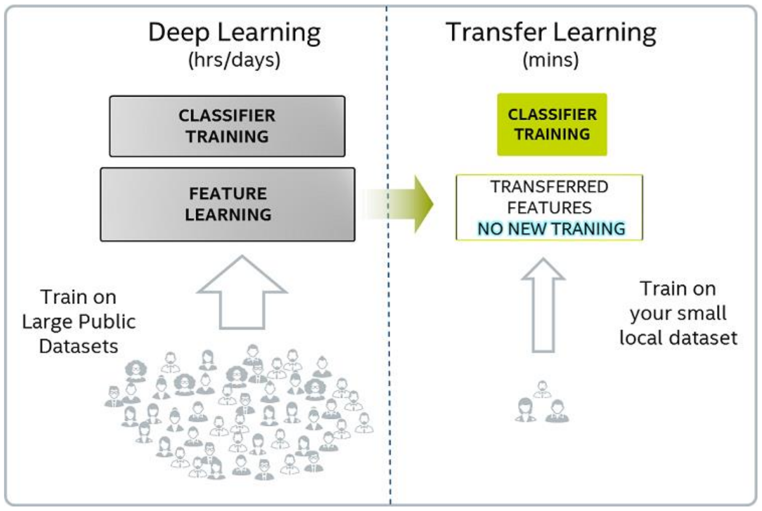


Figure 8 Transfer Learning Approach [45]

In situations where there is not much data to train a model for the target task, transfer learning is the only option to train a model without overfitting. Having fewer parameters also reduces the risk of overfitting. Train the entire network by unfreezing these transferred parameters. In this approach, we need to transfer the initial values of the parameters. Instead of initializing the random weights to our model, initialize the weights using a pre-trained model for a good start and speed up the convergence. To retain the initialization from pre-training, it is a common practice to lower the learning rate by order of magnitude. It is also common to start with frozen parameters, to prevent changing the transferred parameters too early, train only the randomly initialized layers until they converge, the unfreeze all the parameters and fine-tune the entire network.

Transfer learning is especially useful when we have a limited amount of data for one task but have a vast amount of data for a similar job or have a model that has already trained on the same data. Initializing the parameters with a pre-trained model can still be better than random initialization even if we have sufficient data to afford training a model from scratch, and tasks are not so close.

We might be wondering how transfer learning performs on this task. An exciting property of deep neural networks is that when they train on a broad set of images, the first layer parameters resemble each other regardless of the specific task they have trained. For instance, convolutional neural networks tend to learn textures, patterns, and edges in the earlier layers. These layers capture the features that are useful for natural features that detect corners, edges, shapes, textures, and different types of illuminants that considers as generic feature extractors we can use for different kinds of use cases. The closer we move to the output, the more precise features the layers lean to learn object parts and objects.

For instance, the last layer in a network trains to do classification would be particular to that classification task. If the model trains to classify breeds of dogs, one unit would respond only to pictures of specific breed. Transferring all layers except the last layer is probably the most common transfer learning, although not the only one. In general, it is possible to transfer first n layers from a pre-trained model to a target model and randomly initialize the rest. Technically, the assigned part does not even have to be the starting layers. It is possible to move the last layers as well if the tasks are similar, but the type of input is different. For example, we have a face recognition model trained on RGB images, and our goal is to construct a face recognition model that inputs pictures and has a depth channel in adding up to RGB data. Transfer learning does not work well if the data is vastly different; two model architectures are different.

2.2 Related Work

A. Object Detection

Object detection with deep learning: A review [11] analyzed multiple current state-of-the-art object detection algorithms. The paper explained about two types of object detection algorithms. Firstly, region proposal-based algorithm and secondly, classification/regression-based algorithm. Region proposal algorithm generates region proposals first and then classifies them in a later stage. It is a two-step process. In the classification/regression-based algorithm, the region proposals and classification did in a single step using unified, or grid approaches.

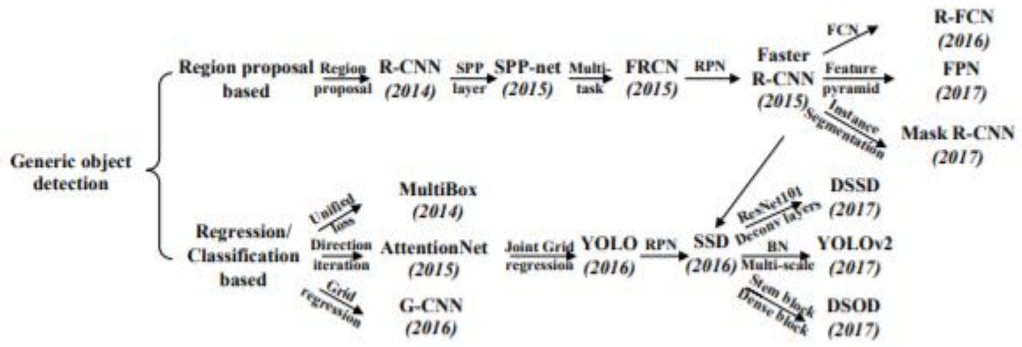


Figure 9 Object Detection with Deep Learning: A Review [11]

All the introduced object detection algorithms [11] focused on the internal architecture of the object detection algorithms. The paper [11] failed to address the cost-effective distributed environment for object detection that enables to customize the users with selected classes in the dataset to build a custom object, detection models.

B. Distributed framework and False Positive Suppression for Object Detection

Simpledet [21] performs object detection by dividing the dataset into sub-datasets and run the datasets parallelly on multiple GPU’s to perform training. We need high-end GPUs to build object distribution models using Simpledet. The Decoupled classification refinement: Hard false positive suppression for object detection [22] paper emphasizes on reducing the hard-false positives. This paper has analyzed false positives place an essential role in affecting the accuracy of the model. The proposed solution positioned a classification network in parallel with the localization network in the faster-RCNN architecture to build and refine false positives from the output detections.

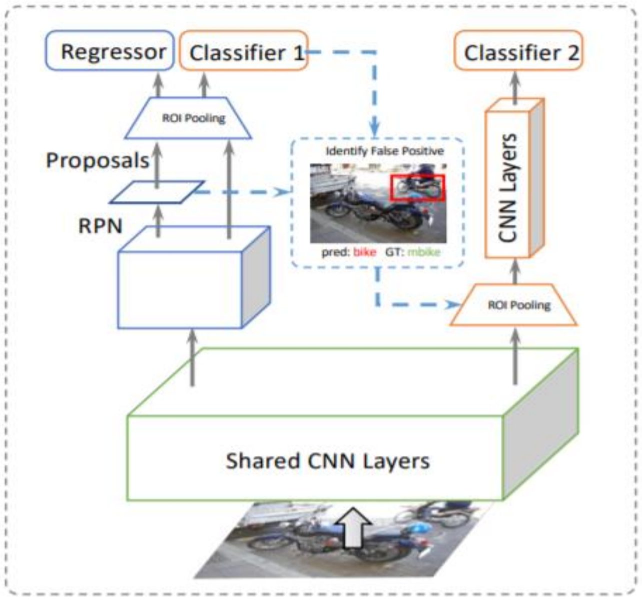
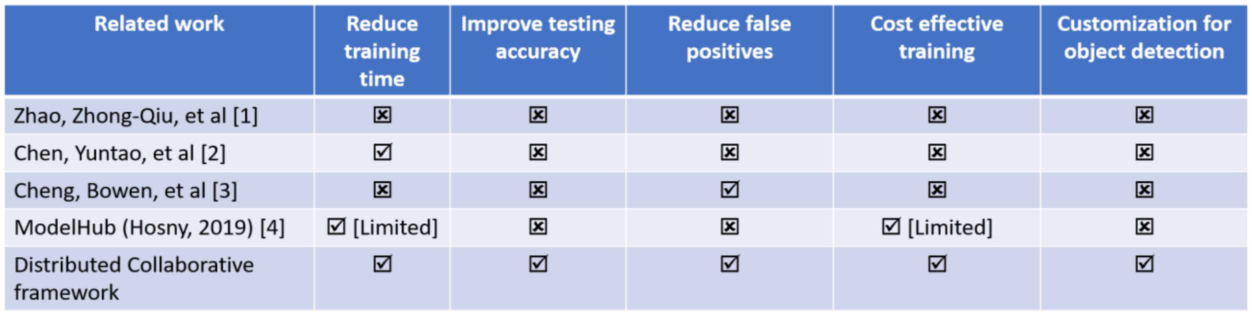


Figure 10 Decoupled Classification Refinement [22]

Simpledet [22] talks about a distributed environment for object detection, which is cost-ineffective and non-customizable. It does not allow the users to customize the models, and we also need practical machine learning algorithms to perform cost-effective learning. The paper [22] fails to address the advantages of creating a distributed environment and cost-effective learning for object detection. The following table compares the related work with the proposed framework, i.e., distributed collaborative framework.



Related work	Reduce training time	Improve testing accuracy	Reduce false positives	Cost effective training	Customization for object detection
Zhao, Zhong-Qiu, et al [1]	☒	☒	☒	☒	☒
Chen, Yuntao, et al [2]	☑	☒	☒	☒	☒
Cheng, Bowen, et al [3]	☒	☒	☑	☒	☒
ModelHub (Hosny, 2019) [4]	☑ [Limited]	☒	☒	☑ [Limited]	☒
Distributed Collaborative framework	☑	☑	☑	☑	☑

Figure 11 Comparison of related work

C. Transfer Learning

ModelHub [23] is an end-to-end container-based management tool that provides models to users to perform transfer learning. It contains several object detection deep learning models along with datasets for the developer to use. The transfer learning works well if we use a base pre-trained model that is close to the user's use case. ModelHub fails to address the recommendation of models for a user to perform cost-effective, efficient transfer learning.

Figure 11 states that all the current research work fails to address the cost-effective distributed environment to build object detection models. The papers [11] [21] [22] [23] also does not talk about the customization of the models with selected classes in the dataset. These problems motivated us to introduce a distributed, collaborative framework for object detection in deep learning.

CHAPTER. 3 METHODOLOGY

In this chapter, we introduced all the approaches presented in the proposed framework. The proposed framework focuses on building and testing cost-effective and customized object detection models in a distributed manner. This approach improved both the training and testing performance of the models. More details are explained in the following sections.

3.1 Effective Machine Learning Algorithm

An efficient machine learning algorithm aims at calculating the correlation score between any two deep learning models. This approach helps to build cost-effective object detection models. It is also applied during inference to improve the accuracy of object detection models.

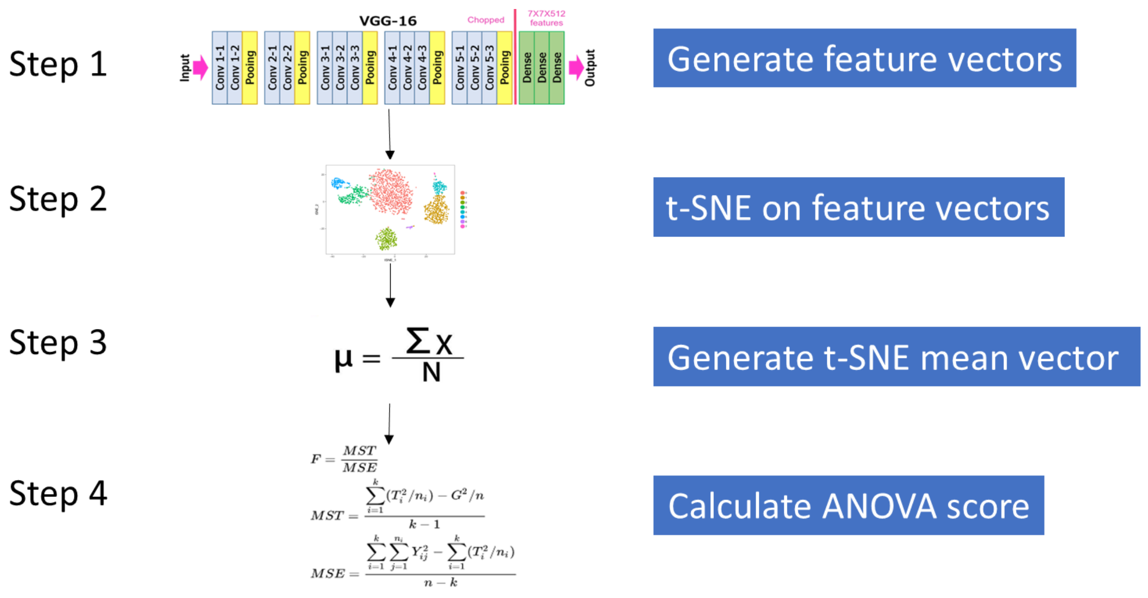


Figure 12 Effective Transfer Learning Approach

To apply, efficient machine learning algorithms in building and inferencing of deep learning models, we need to perform four steps that are described in Figure 12. This approach focuses on improving the training and testing accuracy of the models. It takes the source and target datasets as input and generates an Intra Class Correlation Coefficient Score (ICCS) [24]. Firstly, it creates feature vectors from both the source and target datasets. The generated feature vectors are passed to t-SNE for dimensionality reduction. We calculated a mean vector and then moved to Analysis of Variance (ANOVA) algorithm to make an ICCS score. This score tells us how much two deep learning models correlate to each other and identifies the most similar model from the deep learning model repository that can be used as a base model to perform efficient transfer learning.

3.1.1 Generate Feature Vectors

The first step in an effective machine learning algorithm is generating feature vectors for both source and target datasets. We have used VGG-16 [16] classification algorithm to create the feature vectors. The classification algorithm has two parts: feature extractor and classifier. We utilized feature vector layers to extract the features from the datasets.

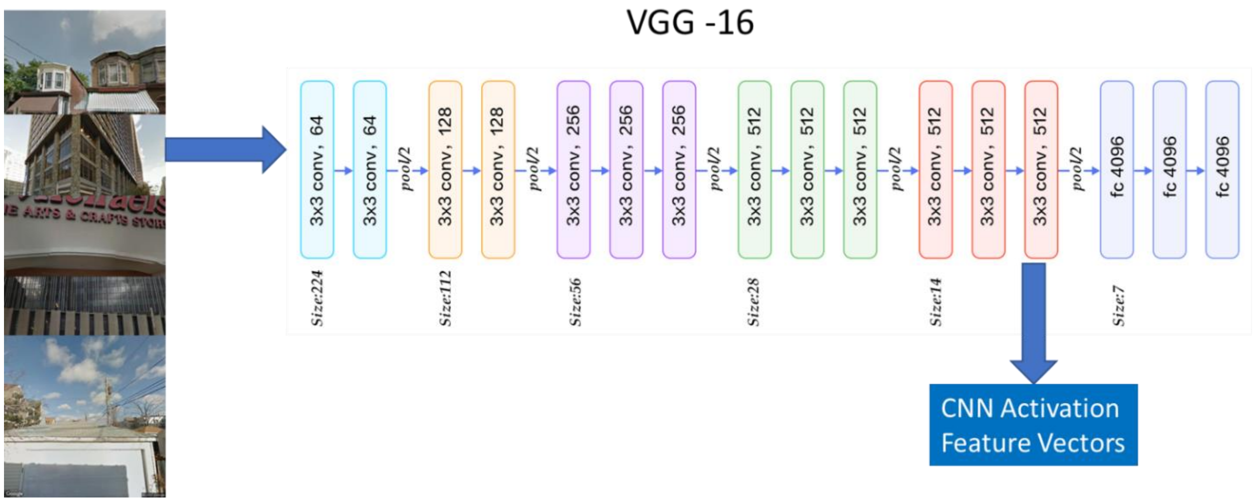


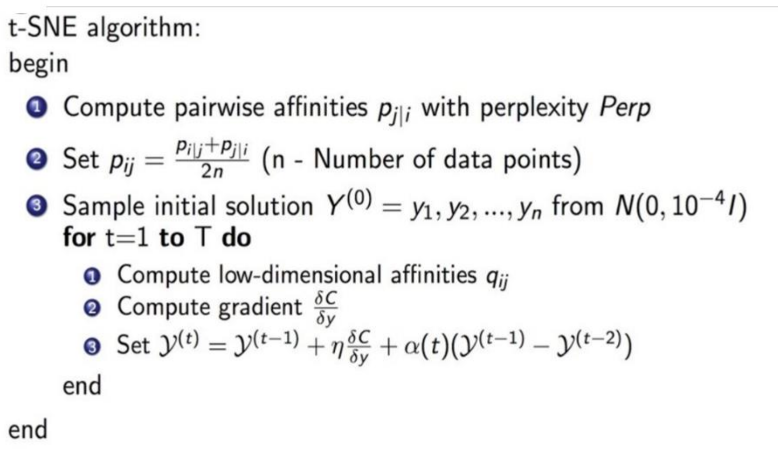
Figure 13 Feature Extraction from the Datasets [49]

The VGG-16 in Figure 13 is a convolutional neural network pre-trained model trained on the ImageNet dataset, which has 14 million images and 1,000 classes. The 16 in VGG-16 represents the number of layers in the architecture. The layers include convolution layers, max-pooling layers, activation layers, and fully connected layers. The FC layers in Figure 13 is the dense layers. The layers before FC layers in the above Figure are responsible for extracting the features from the dataset.

We have used VGG-16 as it is one of the accurate and lightweight classification models that are available in the deep learning classification models community. The way we generate feature vectors is, we pass the two datasets that we are interested in to calculate the correlation score. For instance, we pass the sofa and chair datasets to the VGG-16 model. It generates a 512-dimension feature vector for each of the images in both the sofa and chair datasets.

3.1.2 t- distributed Stochastic Neighbor Embedding on Feature Vectors (t-SNE)

t-SNE [25] stands for t-distributed stochastic neighborhood embedding is used for dimensionality reduction. It is one of the state-of-the-art dimensionality reduction algorithms. We applied t-SNE to the feature vectors generated by VGG-16 to remove the redundant features and reduce the number of computations in calculating an Intra Class Correlation Coefficient Score (ICCS).



```
t-SNE algorithm:
begin
  ❶ Compute pairwise affinities $p_{j|i}$ with perplexity $Perp$
  ❷ Set $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2n}$ (n - Number of data points)
  ❸ Sample initial solution $Y^{(0)} = y_1, y_2, ..., y_n$ from $N(0, 10^{-4}I)$
     for t=1 to T do
        ❶ Compute low-dimensional affinities $q_{ij}$
        ❷ Compute gradient $\frac{\delta C}{\delta y}$
        ❸ Set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta\frac{\delta C}{\delta y} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$
     end
end
```

Figure 14 t-Distributed Stochastic Neighborhood Embedding Algorithm [50]

The way t-SNE works is, it plots all the feature vectors in high dimensional space. It calculates the probabilistic distribution to find the nearest neighbors in high dimensional space. For every vector in high dimensional space, it generates a low dimensional vector ensuring the neighborhood embedding between the vectors.

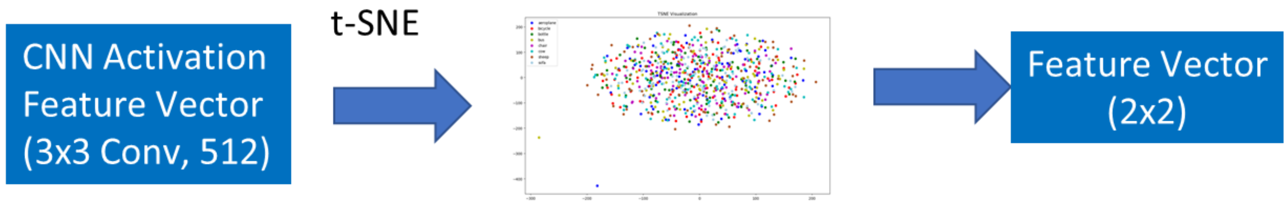


Figure 15 Feature Vectors to t-Distributed Stochastic Neighborhood Embedding [52]

The feature vectors of images in sofa and chair datasets that are extracted from VGG-16 in 3.1.2 are passed to t-SNE to generate a low dimensional vector for images in datasets. The t-SNE reduces the 512-dimension vector to a 2-dimension vector for all the features in both the datasets. We then calculated a mean vector from the low dimensional vectors generated by the t-SNE algorithm. These mean vectors are passed to an ANOVA [26] algorithm, which we will discuss in the following section.

3.1.3 Analysis of Variance Algorithm

ANOVA [26] is an Analysis of Variance. It is an extension of the t-test [27]. ANOVA [28] checks if a significant difference exists between 2 or more group means. For instance, we can compare mean test scores of groups, private and public schools. ANOVA assumes data follows a normal distribution, and variation within the group of interests are equal.

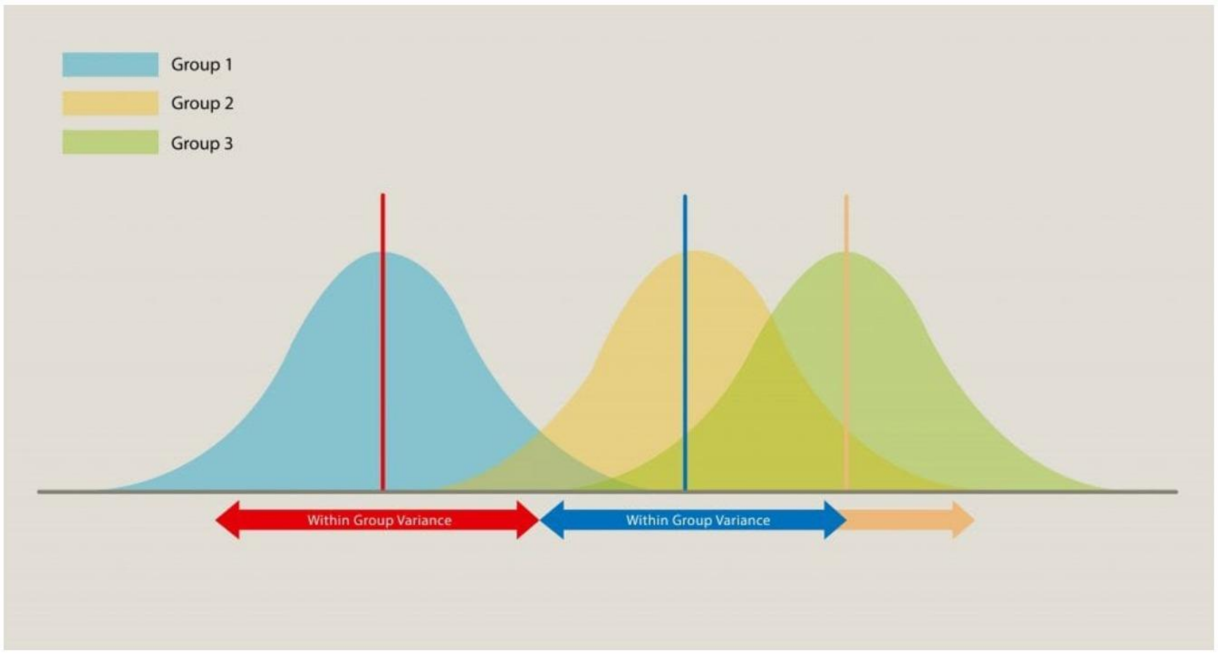


Figure 16 Analysis of Variance Comparison [51]

For an ANOVA, there is a NULL and an alternate hypothesis. The Null hypothesis asserts there is no difference between the group means, whereas an alternative theory states that there is a difference between the groups. The value resulting from ANOVA is a p-value that tells us how likely the difference between groups could have happened. If the p-value is significant, it fails to reject the NULL hypothesis. If it is small, it rejects the NULL hypothesis.

$$\bar{X}_{GM} = \frac{\sum x}{N}$$

Figure 17 Grand Mean in ANOVA

Figure 17 represents the Grand Mean (GM) that is present in the ANOVA formula. GM is the total data values divided by the total number of samples in the sample set. This GM is used to calculate the correlation score between the groups.

$$SS(B) = \sum n(\bar{x} - \bar{X}_{GM})^2$$

Figure 18 ANOVA Formula

Figure 18 represents the formula to calculate the correlation score between the two groups. SS(B) defines the Sum of Squares Between groups, whereas n is the sample size. X(GM) is calculated, as shown in Figure 17. The x bar is the total sum of the data values in a sample set.

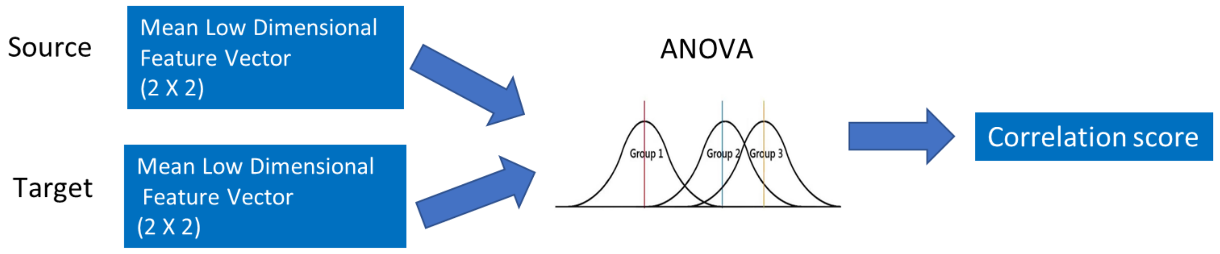

Source
Mean Low Dimensional Feature Vector (2 X 2)
Target
Mean Low Dimensional Feature Vector (2 X 2)
ANOVA
Group 1
Group 2
Group 3
Correlation score

Figure 19 Mean feature vectors to generate ICCS

In Figure 19, we can observe that the mean vector is calculated from the output of t-SNE dimensionality reduction vectors in Figure 15. The mean vectors are then passed to the ANOVA algorithm to generate a *p-value*. This *p-value* is interpreted as an Intra Class Correlation

Coefficient Score (ICCS) to compare the correlation between sofa and chair object detection models.

3.2 Building Single Class Single Model

We have introduced the Single Class Single Model (SCSM) approach to build high-performance customized, cost-effective object detection models. SCSM focuses on building class-wise independent object detection models in a distributed manner. We divided the dataset into sub-datasets based on each class present in the dataset.

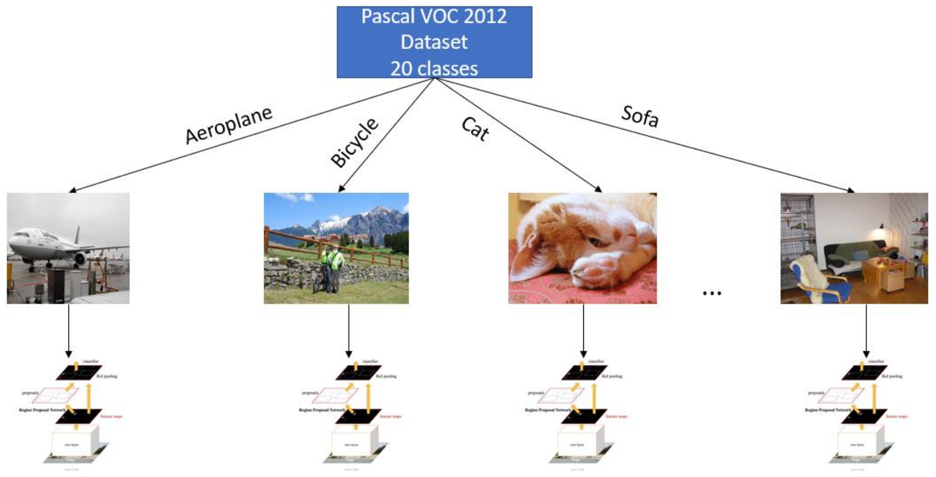


Figure 20 Single Class Single Model Architecture

The distributed way of building models helps to customize the models with selected classes in the future. It also improves the performance of the model, as there will be no confusion with other classes during the training phase. We can also optimize SCSM-based models with an efficient machine learning approach in 3.1 by loading a similar model as a base-model to make the model learn fast in less time. This optimization helps to converge the model more quickly. The steps to build SCSM models is shown below:

Steps to build:

1. Divide the dataset into class-wise sub-datasets based on the number of classes in the dataset.
2. The number of datasets/models is equal to the number of objects in the dataset.
3. Build faster R-CNN models for each independent class.
4. Calculate mAP from all the single class separate models.

We have used the Gluon framework to build SCSM-based models. Gluon framework is a high-level framework created on top of MXNET to develop and train deep learning models. Following is the pseudo code to build SCSM-based models with Gluon framework:

Pseudo-code:

```
1. Load pre-trained model with get_model(net_name,
pretrained_base=True, **kwargs)function.
2. Reset classes based on your usecase using
net.reset_class(CLASSES)method.
3. Prepare custom datasets with VOCLike(root='/voc dataset
path, splits=[(2007/2012,     'train/validation')])
4. Load training datasets in mini batches utilizing
mx.gluon.data.DataLoader(train_dataset.transform(train_tran
sform(net.short,net.max_size,net,ashape=net.ashape,multi_st
age=multi_stage))
5. Load validation datasets in mini batches with
mx.gluon.data.DataLoader(val_dataset.transform(val_transfor
m(short,net.max_size)),batch_size,False,batchify_fn=val_bfn
, last_batch='keep', num_workers=num_workers).
```

6. Use `gluon.Trainer()` method to initiate the training process.

7. Once the training is done, save your model using `save_params()` method.

3.3 Single Class Single Model Inferencing

The proposed Single Class Single Model (SCSM) inferencing enables us to perform highly accurate object detections. Single Class Single Model inferencing performs inference in a distributed collaborative environment. Each model is responsible for performing inference for a single object. The outcome detections of each model are filtered based on a confidence score higher than 95% and combined on a single image, which results as a final output.

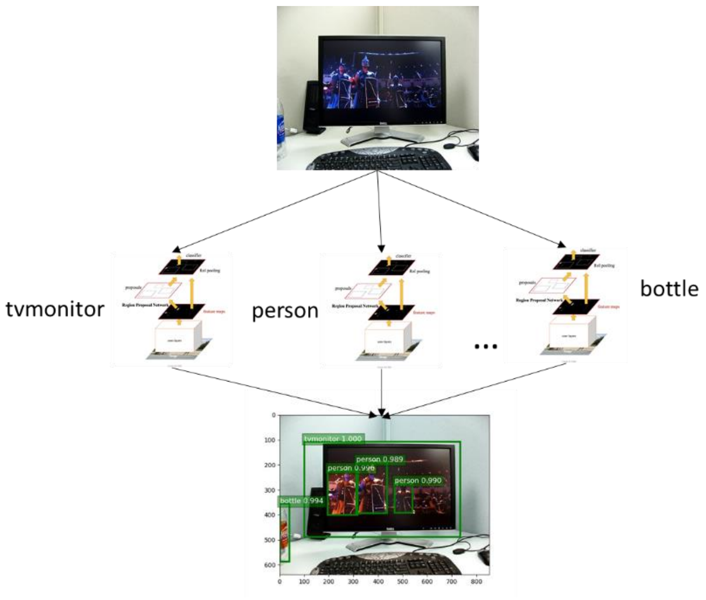


Figure 21 Single Class Single Model Inference Architecture

The SCSM inferencing helps to improve the testing accuracy of the models and allows us to customize the models with selected classes to perform inference. SCSM approach becomes difficult to handle more models for large datasets like MS COCO, which has 80 categories. We also observed SCSM has many false positives. To address these two problems, we introduced group-based inferencing to improve the performance of the models focusing on reducing false positives. The steps to perform SCSM inferencing is explained below:

Steps to perform SCSM inferencing:

1. Pass the test input image to all the individual class models.
2. Load the single class model to perform inference.
3. Filter detections of each class that has a confidence score higher than 95%.
4. Collaborative inference from all the individual class-based models.

Following is the pseudo code to perform SCSM model inferencing with Gluon framework:

Pseudo-code:

```
1. Load the input test image using glob.glob() in python.
2.Transform the input image using gcv.data.transforms.presets.rcnn.load_test() method.
3. for i in range(0, len(all the scscm models))
     -Load the pre-trained model with
     gcv.model_zoo.get_model(pretrained=False) method.
     -Load the saved params file using net.load_parameters().
     -Perform  inference with net(transformed input image)
4. Filter detections that has confidence score greater than 95% with bbox.plot_bbox(threshold=0.95)
```

5. Merge the filtered output detections with `patches.Rectangle()` present in matplotlib.

3.4 Group Based Inferencing

Group based inferencing is introduced to improve the testing accuracy of the object detection models. This approach aims at reducing the false positives that are created during the inference. The group-based method uses an effective machine learning algorithm in 3.1 along with K-Means clustering [29] to perform inference.

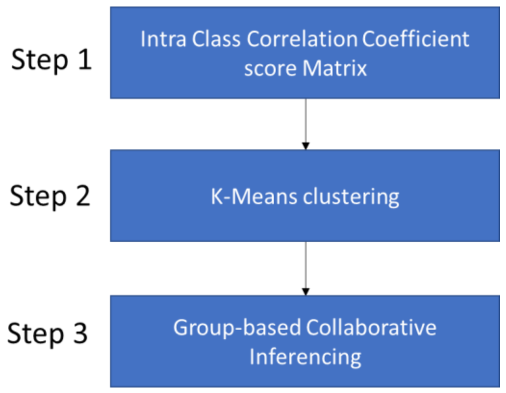

Step 1
Intra Class Correlation Coefficient score Matrix
Step 2
K-Means clustering
Step 3
Group-based Collaborative Inferencing

Figure 22 Group Based Inferencing Approach

With an effective machine learning algorithm, we generated the Intra Class Correlation Coefficient Score (ICCS) symmetric matrix for all the classes in the dataset. This matrix is then passed to the K-Means clustering algorithm. K-means is a statistical algorithm that uses Euclidean distance to perform clustering. It randomly selected centroids based on the k-value. From the centroid, it identifies the closest neighbors from the centroid based on the Euclidean distance. It generates the new centroid and identifies the neighbors using Euclidean distance repetitively based on the number of iterations that the user provides. The number of groups it creates depends on the k-value we provide.

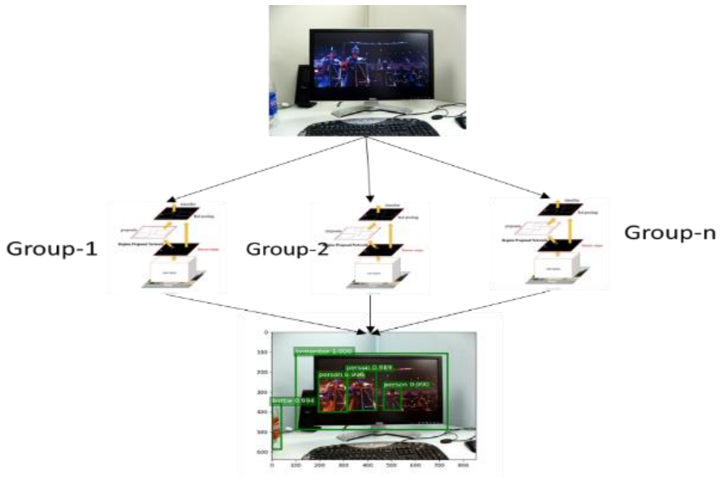


Figure 23 Group Based Inference Architecture

Using k-means clustering, we generated the groups by passing an ICCS symmetric matrix as an input. It then creates three groups as we provided k value as three. In group-based inferencing, the number of models is equal to the number of groups. Grouping similar correlated objects force the model to distribute the probability between two confusing classes, thus reduces the false positives during inference. The steps to perform group inferencing is shown below:

Steps to perform group inferencing:

1. Build models based on groups generated using an efficient machine learning algorithm along with K-Means clustering.
2. Pass an input test image to all group-based models.
3. Load all group models.
4. Filter detections of each group that has a confidence score higher than 95%.
5. Collaborative inference from all the group-based models.

Following is the pseudo code to perform group-based inferencing in Gluon framework:

Psuedo-code:

```
1.Load the input test image using glob.glob() in Python.
2.Transform the input image using gcv.data.transforms.presets.rcnn.load_test() method.
3. for i in range(0, len(all the group models))
          -Load the pre-trained model with gcv.model_zoo.get_model(pretrained=False) method.
          -Load the saved params file using net.load_parameters().
          -Perform  inference with net(transformed input image)
4. Filter detections that has confidence score greater than 95% with bbox.plot_bbox(threshold=0.95)
5. Merge the filtered output detections with patches.Rectangle() present in matplotlib.
```

CHAPTER. 4 EXPERIMENTATION & RESULTS

4.1 Datasets

To conduct our experiments in this research, we have used the Pascal VOC datasets and Neighborhood dataset. Pascal VOC dataset contains 20 classes, and the neighborhood dataset comprises five types. The neighborhood dataset is the one we created at the UMKC Distributed Intelligent Computing group (UDIC) lab.

To evaluate SCSM training models along with efficient machine learning algorithms, we have used 7994 training images and 1999 validation images from Pascal VOC 2012 and 2000 training images, 500 validation images from the neighborhood dataset.

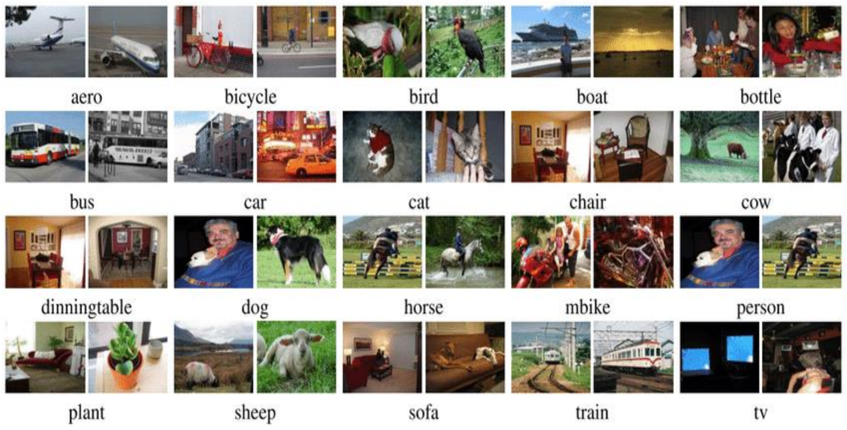


Figure 24 Pascal VOC Dataset [48]

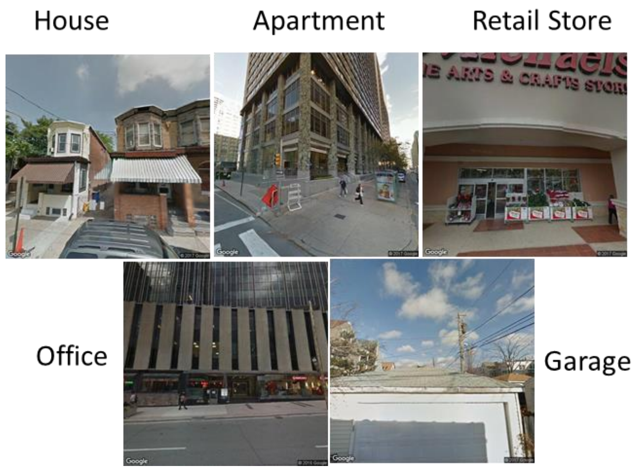


Figure 25 Neighborhood Dataset

We created a neighborhood dataset from Google Streetview images. Google provides a static street view API that helps to capture the snapshot of a real-world image provided with latitude and longitude positions.

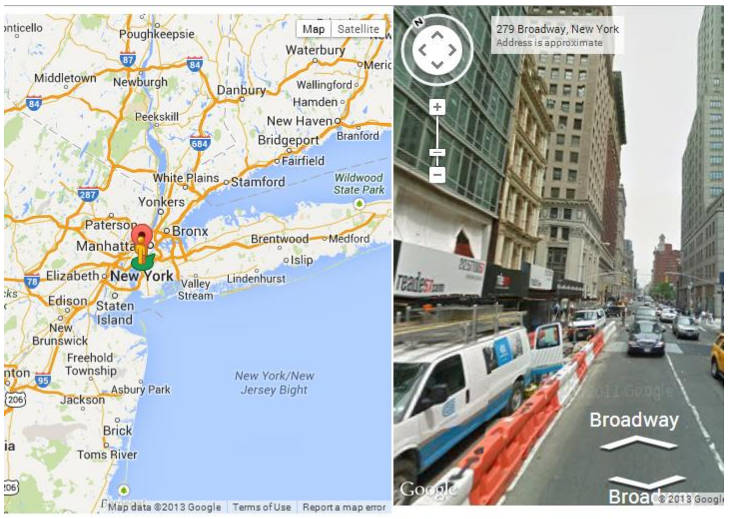


Figure 26 Google Streetview Image [47]

We annotated images using a supervisely [30] tool. Supervisely [30] is a tool that allows the users to generate a ground truth bounding boxes and produces a JSON metadata file for each image. We have used a python script to convert JSON formatted training data into a pascal VOC dataset format.

4.2 Evaluation Metrics

To evaluate the deep learning object detection models, we considered mean average precision (mAP), precision, recall, and F1-score. The mAP tells us how accurate the predicted detections are, whereas precision focuses on the percentage of true positives in the output detections. The recall score provides the rate of relevant true positives, and the F1 score balances both precision and recall scores.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Figure 27 Precision Formula [53]

Precision tells us how accurate our models are from predicted outcomes. It is an excellent measure to identify if the cost of false positives is high. For instance, to detect non-spam email as spam is a false positive in the spam detection model. A low precision score, in this case, leads the email user to lose useful information.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figure 28 Recall Formula [53]

Recall metric helps to identify if there is a high cost involved in false negatives. For instance, if a sick patient is detected as a non-sick patient, and if the disease is contagious, the consequences are going to be worse. So high recall plays an essential role in evaluating the performance of a model.

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Figure 29 F1 Score Formula [53]

F1 score balances both precision and recall. As accuracy contributes a ton of true negative, we also need to balance both false positives and false negatives by using an F1 score. It also helps in many business circumstances.

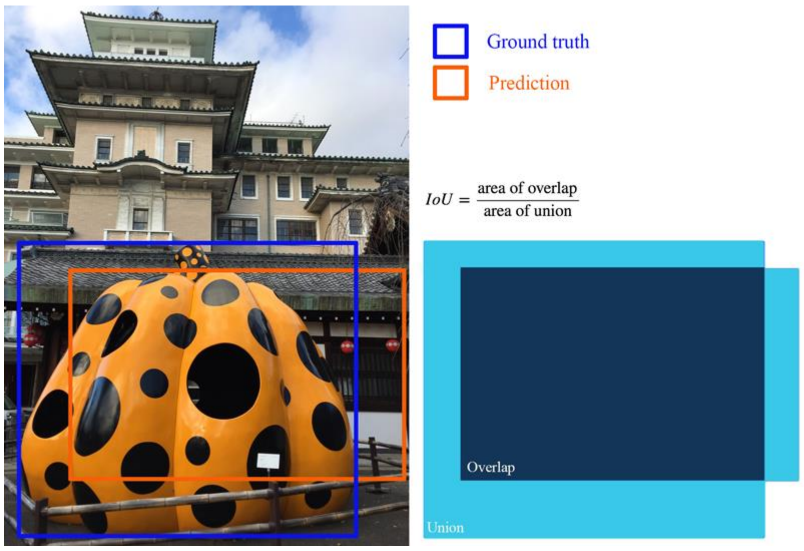


Figure 30 Intersection Over Union [46]

mAP [31] is calculated based on the Intersection Over Union (IOU) score. We calculated the IOU score based on ground truth and predicted bounding boxes. If IOU > 0.5, we take the predicted bounding box as true positive else, we consider it as false positive. We calculated precision and recall for all the test images based on true positives, false positives, and false negatives. Once we have the precision and recall score, the area under the precision-recall curve is called average precision (AP). The average of average precision considered as mean Average Precision (mAP).

4.3 Effective Machine Learning Algorithm

We evaluated the efficient transfer learning on the neighborhood domain dataset. The assumption is we have two models, namely apartment and garage, along with datasets in the neighborhood domain.

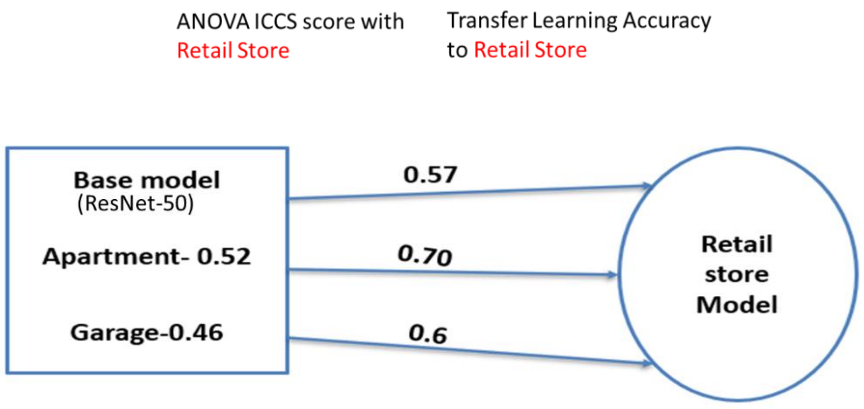
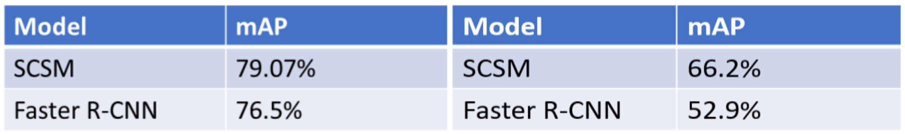


Figure 31 Effective Transfer Learning Results

The value over the arrow describes the accuracy gained by the retail-store model using ResNet-50, apartment, and garage as base models. We can observe that the retail-store model has achieved the highest accuracy using the apartment as a base model. By loading the similar base model that is close to the target, models improve the efficiency as it helps for faster convergence of the model.

4.4 Building Single Class Single Model

To evaluate the SCSM approach, we used Pascal VOC 2012 [32] and neighborhood [4] datasets. The following are the results using SCSM based approach Vs. Faster R-CNN



Model	mAP	Model	mAP
SCSM	79.07%	SCSM	66.2%
Faster R-CNN	76.5%	Faster R-CNN	52.9%

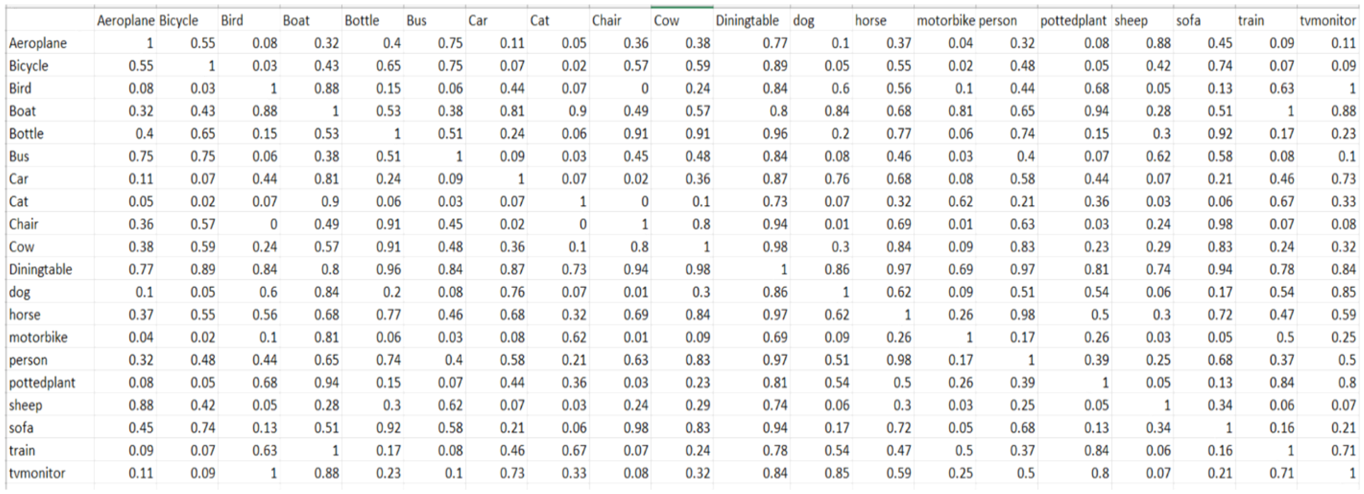
Figure 32 Faster R-CNN Vs. SCSM for VOC 2012 and Neighborhood Dataset

The SCSM approach beats the accuracy of the regular faster-RCNN model in both scenarios. The reason is that SCSM based models are independent models that focus only on one class. During the training, the model is not confused with any of the other categories in the classification stage. Therefore, it improves the training accuracy and reduces the training time as we are performing model building in a distributed environment.

4.5 Single Class Single Model and Group based Inferencing

Single Class Single Model inferencing performs class wise inference collaboratively, whereas group-based models perform group-wise inference and filter the detections based on confidence score higher than 95%. This approach beats the testing accuracy of faster-RCNN models.

ICCS matrix of all the classes in pascal VOC 2007 dataset is shown below:



	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor
Aeroplane	1	0.55	0.08	0.32	0.4	0.75	0.11	0.05	0.36	0.38	0.77	0.1	0.37	0.04	0.32	0.08	0.88	0.45	0.09	0.11
Bicycle	0.55	1	0.03	0.43	0.65	0.75	0.07	0.02	0.57	0.59	0.89	0.05	0.55	0.02	0.48	0.05	0.42	0.74	0.07	0.09
Bird	0.08	0.03	1	0.88	0.15	0.06	0.44	0.07	0	0.24	0.84	0.6	0.56	0.1	0.44	0.68	0.05	0.13	0.63	1
Boat	0.32	0.43	0.88	1	0.53	0.38	0.81	0.9	0.49	0.57	0.8	0.84	0.68	0.81	0.65	0.94	0.28	0.51	1	0.88
Bottle	0.4	0.65	0.15	0.53	1	0.51	0.24	0.06	0.91	0.91	0.96	0.2	0.77	0.06	0.74	0.15	0.3	0.92	0.17	0.23
Bus	0.75	0.75	0.06	0.38	0.51	1	0.09	0.03	0.45	0.48	0.84	0.08	0.46	0.03	0.4	0.07	0.62	0.58	0.08	0.1
Car	0.11	0.07	0.44	0.81	0.24	0.09	1	0.07	0.02	0.36	0.87	0.76	0.68	0.08	0.58	0.44	0.07	0.21	0.46	0.73
Cat	0.05	0.02	0.07	0.9	0.06	0.03	0.07	1	0	0.1	0.73	0.07	0.32	0.62	0.21	0.36	0.03	0.06	0.67	0.33
Chair	0.36	0.57	0	0.49	0.91	0.45	0.02	0	1	0.8	0.94	0.01	0.69	0.01	0.63	0.03	0.24	0.98	0.07	0.08
Cow	0.38	0.59	0.24	0.57	0.91	0.48	0.36	0.1	0.8	1	0.98	0.3	0.84	0.09	0.83	0.23	0.29	0.83	0.24	0.32
Diningtable	0.77	0.89	0.84	0.8	0.96	0.84	0.87	0.73	0.94	0.98	1	0.86	0.97	0.69	0.97	0.81	0.74	0.94	0.78	0.84
dog	0.1	0.05	0.6	0.84	0.2	0.08	0.76	0.07	0.01	0.3	0.86	1	0.62	0.09	0.51	0.54	0.06	0.17	0.54	0.85
horse	0.37	0.55	0.56	0.68	0.77	0.46	0.68	0.32	0.69	0.84	0.97	0.62	1	0.26	0.98	0.5	0.3	0.72	0.47	0.59
motorbike	0.04	0.02	0.1	0.81	0.06	0.03	0.08	0.62	0.01	0.09	0.69	0.09	0.26	1	0.17	0.26	0.03	0.05	0.5	0.25
person	0.32	0.48	0.44	0.65	0.74	0.4	0.58	0.21	0.63	0.83	0.97	0.51	0.98	0.17	1	0.39	0.25	0.68	0.37	0.5
pottedplant	0.08	0.05	0.68	0.94	0.15	0.07	0.44	0.36	0.03	0.23	0.81	0.54	0.5	0.26	0.39	1	0.05	0.13	0.84	0.8
sheep	0.88	0.42	0.05	0.28	0.3	0.62	0.07	0.03	0.24	0.29	0.74	0.06	0.3	0.03	0.25	0.05	1	0.34	0.06	0.07
sofa	0.45	0.74	0.13	0.51	0.92	0.58	0.21	0.06	0.98	0.83	0.94	0.17	0.72	0.05	0.68	0.13	0.34	1	0.16	0.21
train	0.09	0.07	0.63	1	0.17	0.08	0.46	0.67	0.07	0.24	0.78	0.54	0.47	0.5	0.37	0.84	0.06	0.16	1	0.71
tvmonitor	0.11	0.09	1	0.88	0.23	0.1	0.73	0.33	0.08	0.32	0.84	0.85	0.59	0.25	0.5	0.8	0.07	0.21	0.71	1

Figure 33 Intra Class Correlation Coefficient Score Matrix for Pascal VOC 2007

Figure 33 is a symmetric matrix for all the classes present in Pascal VOC 2007. The matrix is generated by taking 500 images from each class into consideration. The correlation score is shown for every class to the remaining other classes in the dataset. The highest score states that classes are highly correlated to each, and low scores state that classes are less likely correlated to each other.

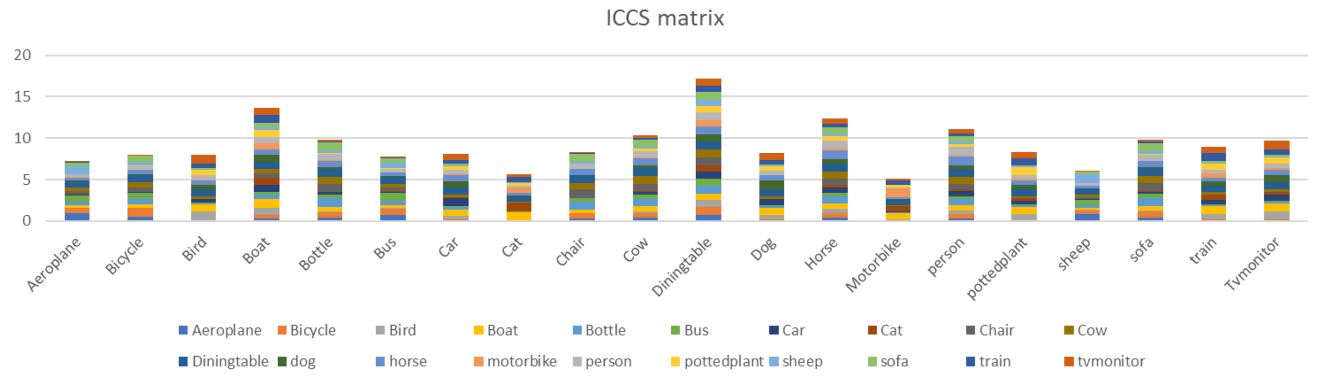


Figure 34 Intra Class Correlation Coefficient Score Matrix Visualization for Pascal VOC 2007

The highest score in the graph explains that a specific class is correlated to a higher number of classes in the dataset. The colors represent different objects present in the dataset.

The x-axis in Figure 34 represents the class names, and the y-axis represents the correlation score. Charts for each of the groups generated by K-Means clustering based on the Intra Class Correlation Coefficient score matrix:

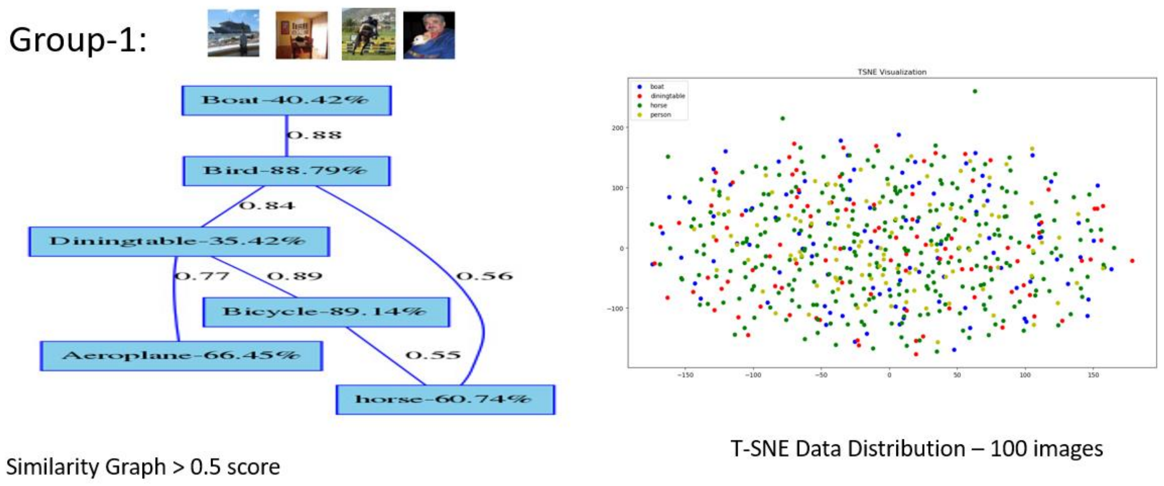


Figure 35 Intra Class Correlation Coefficient Score and t-SNE– Group-1

Figure 35 shows the visualizations for group 1 generated by K-Means clustering. The group-1 objects comprise of boat, dining-table, horse, and person. The left-side diagram shows the correlation score of these four objects from group-1 to all the objects in the dataset. The right-side picture shows the scatter plot of low dimensional feature vectors generated by the t-

SNEalgorithm.

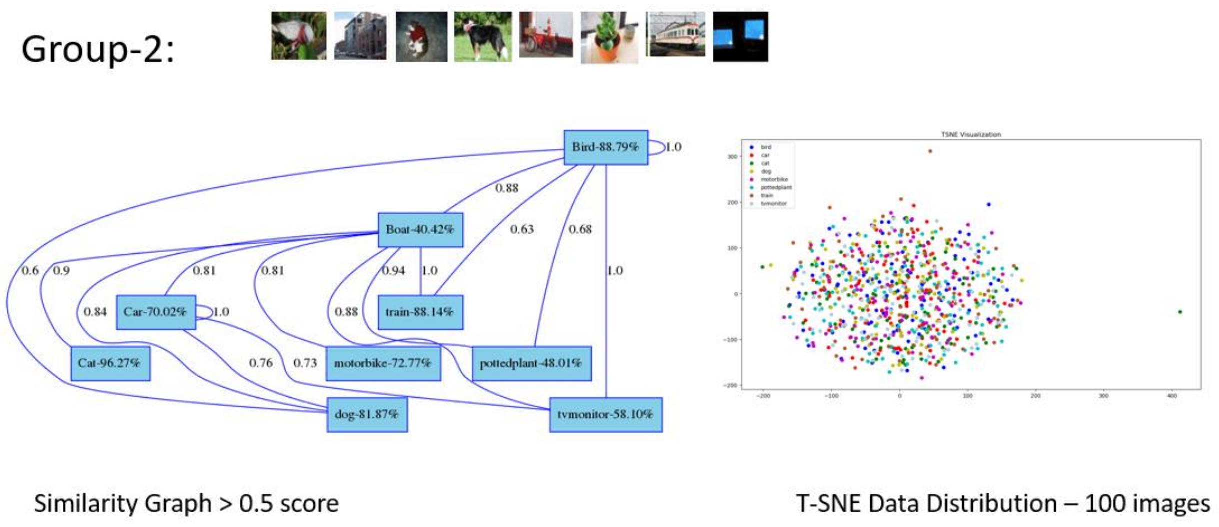
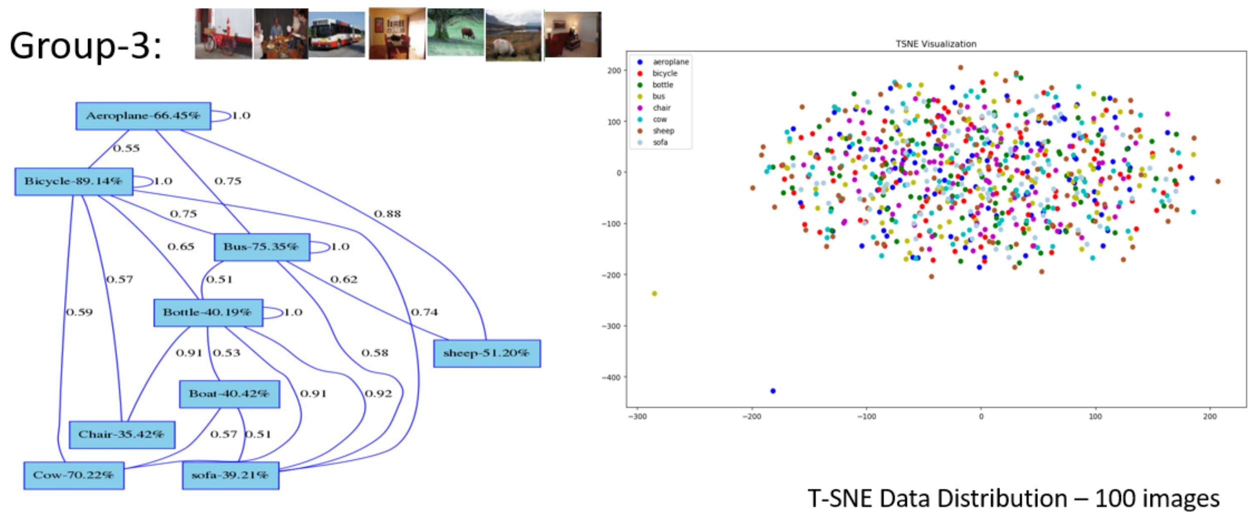


Figure 36 Intra Class Correlation Coefficient Score and t-SNE– Group-2

Figure 36 shows the visualizations for group 2 generated by K-Means clustering. The objects in group-2 include Bird, Car, Cat, Dog, Motorbike, Potted plant, Train, and Tv monitor. The value over the arrow in the left-side diagram represents the correlation score generated by the effective machine learning algorithm described in 3.1.



Similarity Graph > 0.5 score

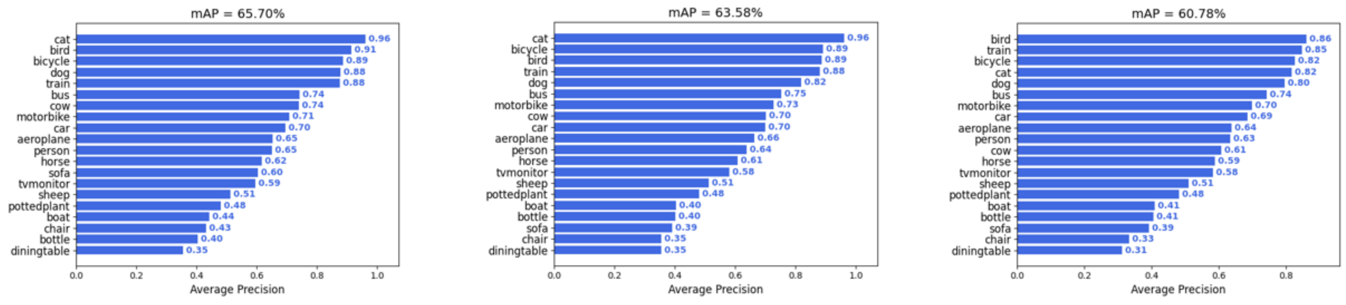
T-SNE Data Distribution – 100 images

Figure 37 Intra Class Correlation Coefficient Score and t-SNE– Group-3

Figure 37 shows the visualizations for group 3 generated by K-Means clustering. The group-3 consists of Aeroplane, Bicycle, Bottle, Bus, Chair, Cow, Sheep, and Sofa. Like Figures 35 and 36, the left-side diagram represents the correlation score graph, whereas the right-side chart shows the feature vector distribution. The overlapping of the feature vectors in the right-side picture tells us these objects in the dataset are correlated to each other.

We generated all the above three groups using K-means clustering. The ICCS matrix is passed as an input to K-means to create these groups. The left-side graph shows all the connections between classes in each group that has ICCS more than 0.5. The right-side Figure represents the feature distribution of the images in the dataset that belongs to the respective group. Each color in the chart signifies a class, as shown in the graph. The mAP score comparison for the SCSM, group-based inferencing, and Faster R-CNN is shown below:

mAP Score:
SCSM > Group > Faster R-CNN
65.70% > 63.58% > 60.78%



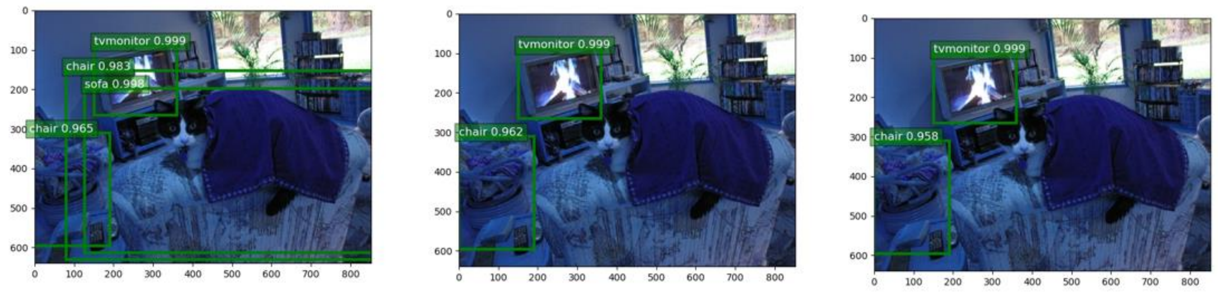
SCSM Group Inferencing Faster R-CNN

Figure 38 Inference for Single Class Single Model Vs. Group Vs. Faster R-CNN

Some of the output detections for SCSM, group-based inferencing and Faster R-CNN are shown below:

Ground-truth: [cat, tv monitor, chair]

SCSM > Group > Faster R-CNN



SCSM Group Inferencing Faster R-CNN

Figure 39 Output Detections Single Class Single Model Vs. Group Vs. Faster R-CNN-1

In Figure 37, we can observe both SCSM and group-based inferencing beats the Faster R-CNN model detections. The ground truth objects that are present in the original image are a cat, tv monitor, and a chair. The SCSM and group-based inference have the highest confidence score for each of the true positive detections in an image comparing with the Faster R-CNN model. This is because the SCSM and group-based inferencing have less confusion with other classes during the inference.

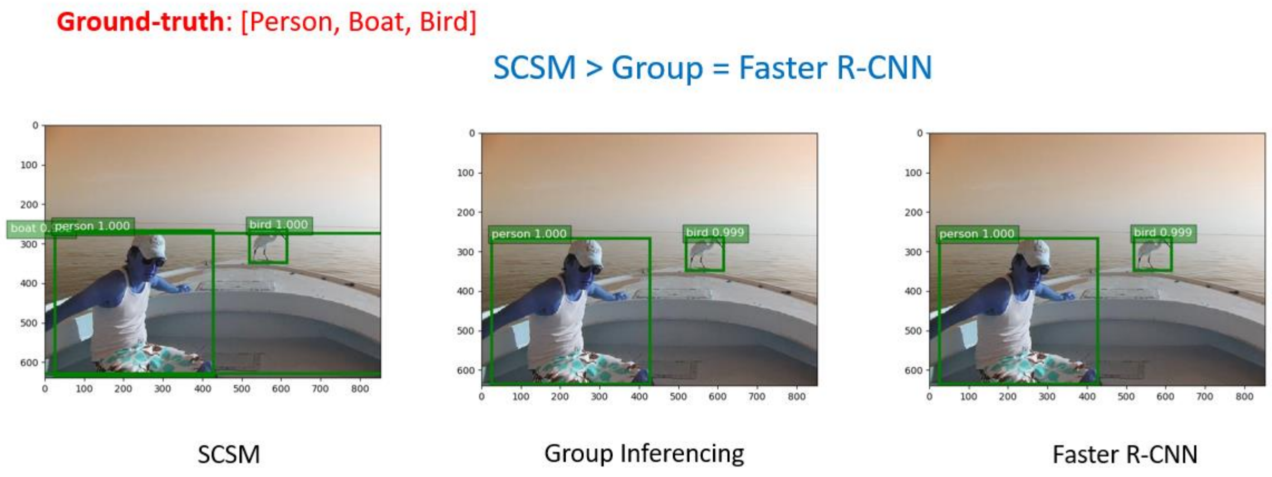


Figure 40 Output Detections Single Class Single Model Vs. Group Vs. Faster R-CNN-2

The SCSM model beats the group-based and faster R-CNN inferencing in Figure 40. The ground truth objects for the above test image are person, boat, and bird. The SCSM can identify the boat, whereas group-based and Faster R-CNN fails to detect. It Is because the SCSM is not confused with any classes during the inference, whereas group-based and Faster R-CNN are forced to distribute the probability between multiple classes and thereby missed a true positive.

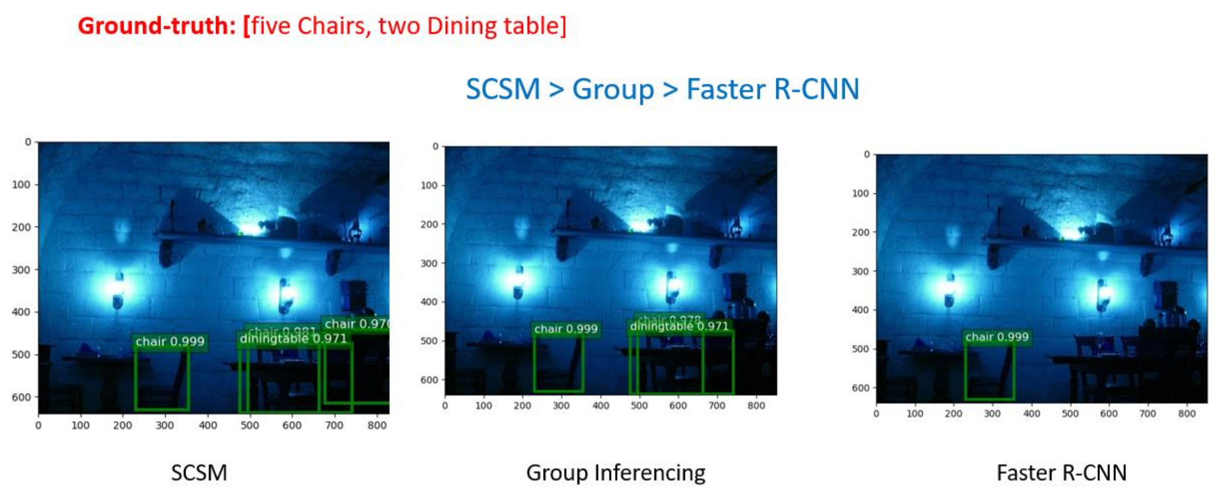


Figure 41 Output Detections Single Class Single Model Vs. Group Vs. Faster R-CNN-3

SCSM and group-based inferencing perform exceptionally well compared to the Faster R-CNN in Figure 41. The ground truth for this test image is five chairs, and two dining tables as the SCSM and group-based focuses on single and few classes, respectively, during the inference. Therefore, SCSM and group-based detected more true positives.

From the above output mAP score and detections, we can conclude SCSM, and group-based inferencing beats the accuracy of the Faster R-CNN model. It is because both SCSM and group-based inferencing deals with a fewer number of classes compared with faster R-CNN during the inference process. Therefore, it increases the number of true positives, thereby expanding the testing accuracy of the deep learning models.

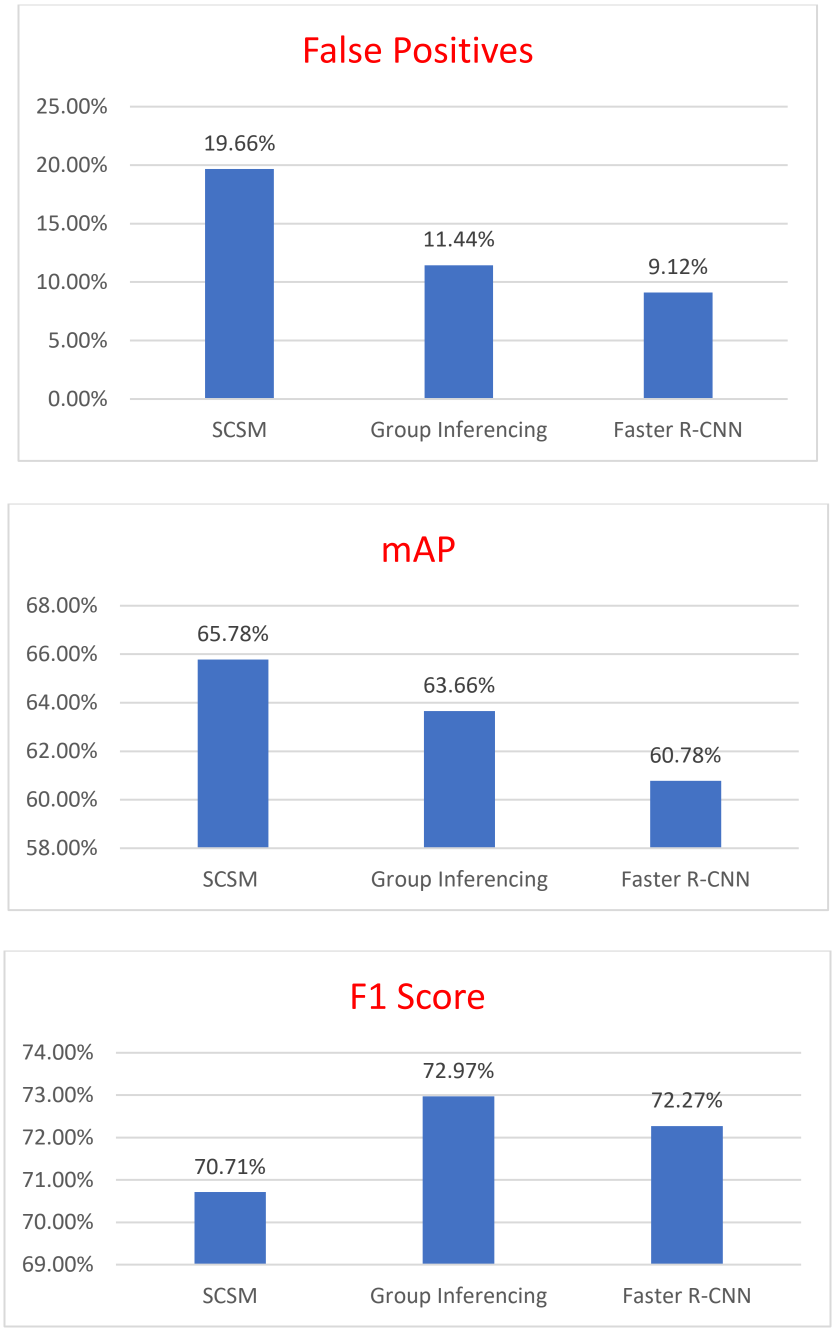


Figure 42 False Positives Vs. Mean Average Precision Vs. F1 Score

Figure 42 shows some insights for SCSM, group-based inferencing, and Faster R-CNN based models. As we can see, the SCSM and group inferencing beat the accuracy of the models compared to the Faster R-CNN model. Group inferencing approach models help to reduce the false positives compared to SCSM approach-based models. Each group in the group inferencing approach knows about a higher number of correlated classes; this forces the model to choose a single category among all confused types. Therefore, this approach reduces false positives.

Reducing false positives improves the precision score. AS f1 score is the harmonic mean of precision and recall; therefore, increasing the precision score increases the f1 score as well. The Faster R-CNN model has a less false-positive rating compared with SCSM and group-based inferencing models. Nevertheless, to build such models, it takes much time and requires high-end GPUs.

CHAPTER. 5 CONCLUSION & FUTURE WORK

5.1 Conclusion

To conclude, the proposed framework aims to build distributed object detection models using Single Class Single Model approach. These models are highly efficient and customized with selective classes based on your use case. The efficient machine learning algorithms applied to SCSM models for cost-effective learning.

We can also perform collaborative inferencing using SCSM based models in a distributed environment. We applied efficient machine learning algorithms along with clustering mechanisms to reduce false positives during inference. The collaborative inferencing models are evaluated based on several metrics such as mAP, F1 score, and false positives to perform efficiently in detecting objects in images. The proposed framework based constructed models helps to quickly build cost-effective state-of-the-art models and real-world applications such as NeighborNet [4].

5.2 Future Work

In this section, we would like to elaborate on the future scope of our research work to build robust deep learning object detection models—multiple clustering mechanisms such as hierarchical and spectral clustering to group the models to perform group-based inferencing. We are also interested in generating groups based on different parameters like similarity, context, dissimilarity, and semantic behaviors.

Furthermore, we want to evaluate the proposed framework models based on false - negatives and confusion scores. We also look forward to testing the deep learning models on several other benchmark datasets like MS COCO and video datasets.

BIBLIOGRAPHY

[1] Y. LeCun, Y.B. Yann, Y. Bengio, and G. Hinton. "Deep learning." *Nature* 521.7553 436-444, 2015.

[2] Deep Learning “Rise of deep learning from 2010-2019”, [Online], Available: https://thenextweb.com/artificial-intelligence/2020/01/02/2010-2019-the-rise-of-deep-learning/

[3] D. A. Forsyth, and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[4] D. H. Ho, R. Marri, S. Rella and Y. Lee, "DeepLite: Real-time deep learning framework for neighborhood analysis," *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, 2019, pp. 5673-5678, doi: 10.1109/BigData47090.2019.9005651

[5] M. Abadi, et al. "Tensorflow: A system for large-scale machine learning." *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[6] N. Ketkar. "Introduction to pytorch." *Deep learning with python*. Apress, Berkeley, CA, 2017. 195-208.

[7] J. Guo, et al. "Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing." *Journal of Machine Learning Research* 21.23 1-7, 2020.

[8] N. Ketkar. "Introduction to keras." *Deep learning With Python*. Apress, Berkeley, CA, 2017. 97-111

[9] J.D. Owens, et al. "GPU computing." *Proceedings of the IEEE* 96.5,2008, 879-899.

[10] Datasets, “Source for open-source datasets”, [Online] Available :https://analyticsindiamag.com/15-open-datasets-for-deep-learning-enthusiast

[11] Z. Zhao, et al. "Object detection with deep learning: A review." *IEEE Transactions on Neural Networks and Learning Systems* 30.11,2019, pp. 3212-3232.

[12] K.P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[13] P. Jain, “Complete guide of activation functions,” [Online] Available: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

[14] D. Maclaurin, D. Duvenaud, and R.P. Adams. "Autograd: Effortless gradients in numpy." *ICML 2015 AutoML Workshop*. vol. 238, 2015,

https://indico.lal.in2p3.fr/event/2914/contributions/6483/subcontributions/180/attachments/6060/7185/automl-short.pdf

[15] Y. LeCun, “Backpropagation applied to handwritten zip code recognition," *Neural Computation*, 1, pp. 541–551, 1989.

[16] S. Liu, and W. Deng, “Very deep convolutional neural network based image classification using small training sample size", *3rd IAPR Asian Conference on Pattern Recognition (ACPR),* November 2015, pp. 730-734.

[17] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747* (2017)

[18] A. Dertat, “ Applied deep learning -Part 4: convolutional neural networks “, [Online] Available: https://towardsdatascience.com/Applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[19] T. Chen, et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." *arXiv preprint arXiv:1512.01274* (2015).

[20] L. Torrey, and J. Shavlik. "Transfer learning." *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010,242-264.

[21] Y. Chen, et al. "Simpledet: A simple and versatile distributed framework for object detection and instance recognition." *Journal of Machine Learning Research* 20.156, 1-8, 2019.

[22] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang, "Decoupled classification refinement: Hard false positive suppression for object detection." *arXiv preprint arXiv:1810.04002* (2018)

[23] A. Hosny, M. Schwier, C. Berger, EP. Ornek, M. Turan, PV. Tran, L. Weninger, F. Isensee, KH. Maier-Hein, R. McKinley, Lu MT. "ModelHub. AI: Dissemination platform for deep learning models." *arXiv preprint arXiv:1911.13218* (2019)

[24] JJ. Bartko. "The intraclass correlation coefficient as a measure of reliability." *Psychological Reports*, vol. 19, no. 1, 3-11,1966.

[25] L. vander Maaten, and G. Hinton. "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, 2579-2605, 2008. [online] http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[26] E.R. Girden, *Sage University papers. Quantitative applications in the social sciences, Vol. 84.ANOVA: Repeated measures.* Sage Publications, Inc., 1992.

[27] T.K. Kim. "T test as a parametric statistic." *Korean Journal of Anesthesiology* 68.6, 540, 2015.

[28] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, "How transferable are features in deep neural networks?", *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3320-3328.

[29] K. Alsabti, S. Ranka, and V. Singh. "An efficient k-means clustering algorithm." http://www.cise.ufl.edu/ranka/ (1997).

[30] Supervisely, “Leading platform for entire computer vision lifecycle ”, [Online] Available: https://supervise.ly/, 2017

[31] mAP, “Map calculation for object detection”, [Online], Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, 2018

[32] S. Shetty. "Application of convolutional neural network for image classification on Pascal VOC challenge 2012 dataset." *arXiv preprint arXiv:1607.03785* (2016).

[33] T. Mikolov, et al. "Recurrent neural network based language model." *Eleventh Annual Conference of the International Speech Communication Association*. 2010, https://www.isca-speech.org/archive/archive_papers/interspeech_2010/i10_1045.pdf

[34] C. Francois. “Keras”, [Online] Available:https://github.com/fchollet/Keras, 2015.

[35] Google Streetview, “Google streetview static API ”, [Online], Available: https://developers.google.com/maps/documentation/streetview/intro

[36] S. Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02, 107-116, 1998

[37] Deep Learning, “Deep neural network”, [Online], Available: https://www.pyimagesearch.com/2016/09/26/A-Simple-Neural-Network-with-Python-and-Keras/

[38] RNN, “Recurrent neural network”, [Online], Available: https://www.stratio.com/blog/Deep-Learning-3-Recurrent-Neural-Networks-Lstm/

[39] Y. Wang, W. Liao, and Y. Chang. "Gated recurrent unit network-based short-term photovoltaic forecasting." *Energies* 11.8, 2163, 2018.

[40] S. Hochreiter, and J. Schmidhuber. "Long short-term memory." *Neural Computation* 9.8, 1735-1780, 1997.

[41] Introduction to Deep Learning “What is deep learning?”, [Online], Available: https://www.youtube.com/watch?v=aircAruvnKk&t=436s

[42] Deep Learning “Hidden layers in deep learning”, [Online], Available: https://www.i2tutorials.com/hidden-layers-in-neural-networks/

[43] ResNet-50 “Short cut connections in ResNet-50”, [Online], Available: https://neurohive.io/en/popular-networks/Resnet/

[44] S. Ren, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in Neural Information Processing Systems*. 2015, pp. 91-99.

[45] Transfer Learning “Introduction to transfer learning”, [Online], Available: https://www.freecodecamp.org/news/Asl-Recognition-using-Transfer-Learning-918ba054c004/

[46] Object Detection mAP “Intersection over union”, [Online], Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[47] Google Streetview Image “ Google streetview static API”, [Online], Available: http://agichevski.com/2013/12/07/Google-Map-and-Google-Street-View-Api/

[48] M. San Biagio, et al. "Encoding structural similarity by cross-covariance tensors for image classification." *International Journal of Pattern Recognition and Artificial Intelligence* 28.07 (2014): 1460008

[49] VGG-16, “VGG-16 diagram”, [Online], Available: https://www.quora.com/What-is-the-VGG-Neural-Network

[50] t-SNE “t-SNE algorithm”, [Online], Available: https://www.immunology.pitt.edu/sites/default/files/3.pdf

[51] Machine Learning Algorithms “Analysis of variance”, [Online], Available: https://analyticsindiamag.com/10-machine-learning-algorithms-every-data-scientist-know/

[52] X. Chen, et al. “Flight state identification of a self-sensing wing via an improved feature selection method and machine learning approaches.” *Sensors (Basel, Switzerland)* vol. 18,5 1379. 29 Apr. 2018, doi:10.3390/s18051379

[53] Precision and Recall “Formulas for precision and recall”, [Online], Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

VITA

Sirisha Rella completed her bachelor’s degree in Computer Science from the Vellore Institute of Technology, Vellore, India. After her bachelor’s degree, she worked as a software engineer at Cerner Corporation for two years in Bangalore, India. She started her master’s program in Computer Science Electrical Engineering at the University of Missouri-Kansas City in Fall 2018. During her master’s program, she worked as a graduate research assistant for an NSF project and as a graduate teaching assistant for several classes. She is currently working as an AI marketing engineer intern at NVIDIA Corporation. After completing her master’s degree, she will work as an AI technical product marketing manager at NVIDIA Corporation.