Robotics Research Lab
Department of Computer Science
University of Kaiserslautern

Master Thesis

Simultaneous Semantic Segmentation and Object Detection using a Common Pipeline

Iyer Venkatesh R

March 24, 2020

Master Thesis

Simultaneous Semantic Segmentation and Object Detection using a Common Pipeline

Robotics Research Lab
Department of Computer Science
University of Kaiserslautern

Iyer Venkatesh R

Day of issue : June 26, 2019
Day of release : January 27, 2020

First Reviewer : Prof. Dr. Karsten Berns
Supervisor : M.Sc. Axel Vierling

Hereby I declare that I have self-dependently composed the Master Thesis at hand. The sources and additives used have been marked in the text and are exhaustively given in the bibliography.

March 24, 2020 – Kaiserslautern

(Iyer Venkatesh R)

Acknowledgement

I want to thank Prof. Dr Karsten Berns for providing this opportunity to write my thesis at RR Lab. Also, I would like to especially thank my supervisor Axel Vierling for providing all the support needed. I want to thank my peers at RR Lab, who were kind enough to help with software and hardware issues.

I want to thank my friends Saurabh, Ashutosh, Sadique, Sridhar and Kriti for all the technical help and suggestions. Finally, and most importantly, my family for continuous support and encouragement. This wouldn't have been possible without them.

Abstract

This work explores computer vision problems such as semantic segmentation and object detection and how both these different architectures can be combined in order to solve these problems in parallel. This work is helpful in the area of autonomous vehicles where the segmented map can show the path to move and object detection can detect cars and pedestrians. This approach is based on a common encoder that combines both the different architectures. While training the network, a dataset containing segmentation ground-truth and bounding box coordinates for each object in the image are passed such that segmented map and bounding boxes are predicted in parallel. This approach is also compared with stand-alone architectures of object detection and semantic segmentation in order to understand the flaws or benefits of using combined architecture. Also, a number of experiments are conducted for trying to increase the accuracy of both the predictions.

Contents

Acronyms

SOTA State-of-Art

ML Machine Learning

AI Artificial Intelligence

DL Deep Learning

CNN Convolution Neural Network

FCN Fully Convolutional Networks

SSD Single shot multibox detector

KITTI KITTI Vision Benchmark

VGG Visual Geometry Group

1 Introduction

Progress in the field of Artificial Intelligence (AI) is one of the hot topics of the ongoing boom in teaching sytems and robots to see the world. Systems and robots use this vision to perform complex tasks in both the physical and virtual worlds. It is found that the investment and work in AI are accelerating at an unprecedented rate in areas like search and optimization, computer vision, machine learning, probabilistic reasoning, neural networks.

The report [Perrault 19] shows that research about cognition related performance - a game playing AI outsmart a human opponent is leading the research category in the number of papers published. While not far behind is, the research in computer vision, which is pushing this progress in autonomous vehicles, including power, augmented reality and object detection.

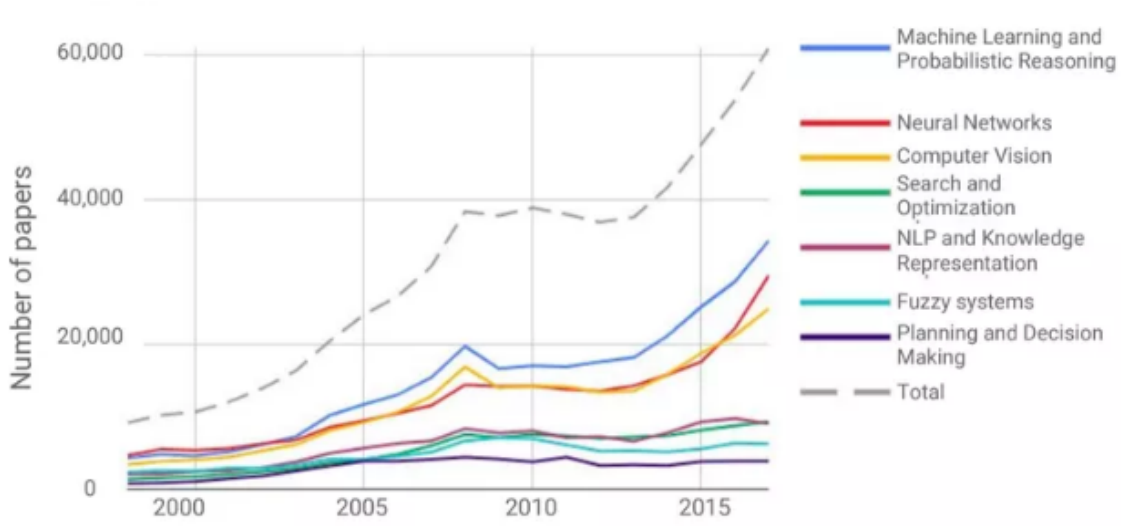


Figure 1: Graph showing the statistics of papers published in recent years

Also, in terms of performance, AI continues to escalate to new heights in the field of computer vision. By measuring benchmark performance for the widely-used image training database [Fei-Fei 13], the [Perrault 19] shows that the time taken by a model to classify the images to achieve good accuracy has come down to few minutes from an hour.

This revolution that is happening in terms of research and hardware backing such research is pushing the current State-of-Art (SOTA) technology towards a point where such innovations will become part of everyday human lives. Thus, the increasing investment of monetary funds and workforce in the research in this field only seeks to accelerate human progress in the direction of bringing AI into our everyday lives, exclusively in the field of computer vision which is seen to be revolutionizing rapidly.

1.1 Motivation

Computer vision in the field of AI is a booming industry applied to many of our day-to-day products. The potential gains are high when a computer in certain areas replaces humans. This is because a computer can see many things at once, in great detail and can analyze it

parallelly. The accuracy of analysis done by computer can bring incredible time savings and quality improvements and also free up resources that require human communication.

Some applications of computer vision in practice today are:

- Autonomous vehicles
- Translation app
- Facial recognition
- Healthcare
- Real-time sports tracking
- Agriculture
- Manufacturing

Out of all the above-mentioned applications, we shift our focus towards autonomous vehicles. According to [World], more than 1.25 million people die each year because of traffic accidents. Nearly 50% of road casualties are pedestrians, cyclists and motorcyclists. The larger part majority of these accidents are due to human error. In such cases, computer vision techniques within Deep Learning (DL) like object detection and semantic segmentation can be used to prevent such accidents. Using such deep networks, the vehicles learn to analyze situations like detecting pedestrians, give way for cyclists, mapping the road and act accordingly.

Figure 2 shows how an autonomous vehicle detects vehicles and maps the path.

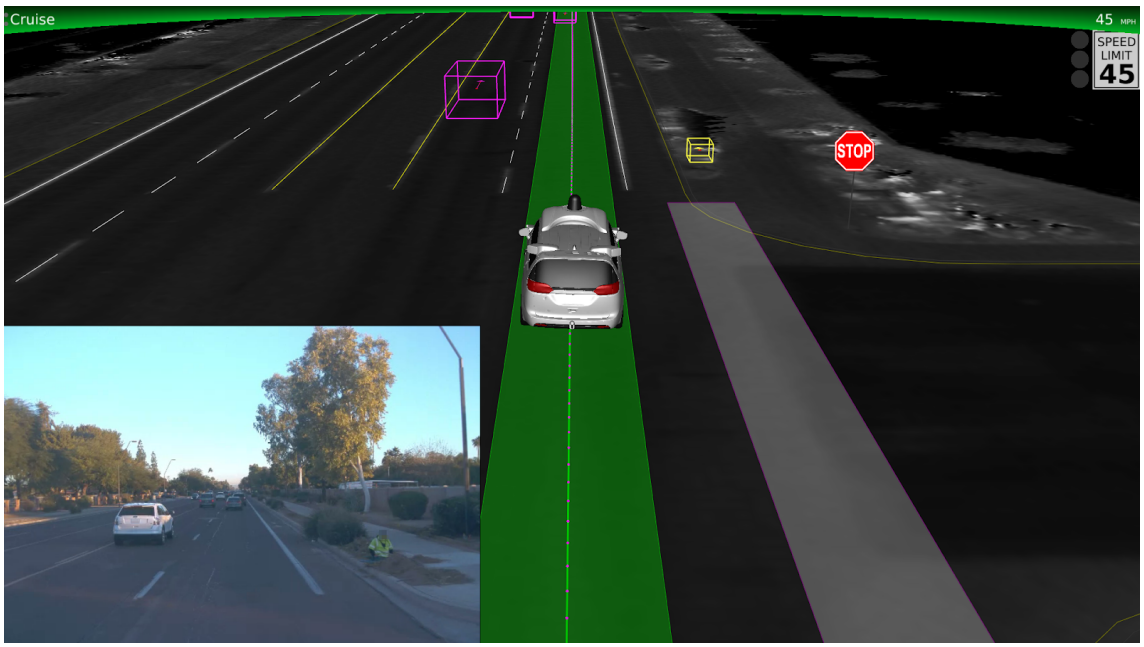


Figure 2: Autonomous vehicle [Waymo 09]

Also, semantic segmentation is important in the field of biomedicine, where relevant regions in the body can be classified, making tests easier and simpler. Whereas, object detection is important in the fields of urban planning, ship tracking, and monitoring the crops.

1.2 Tasks

The above section shows the importance of computer vision techniques such as semantic segmentation and object detection in autonomous vehicles. The objective of this work is to solve both semantic segmentation and object detection using a single common pipeline.

Basic overview of tasks are:

- Combine a dataset such that it consists segmentation ground truth (explained in section 2.6) as well as object bounding box coordinates (explained in section 2.7).
- Combine object detection and semantic segmentation architectures such that it gives bounding box and segmentation map predictions.
- Evaluate the model.

2 Background

This chapter covers topics which will help, in understanding the approach and methodology that are used, in this work. Section 2.1 gives an explanation of machine learning in the area of AI and how different learning techniques in machine learning are applied. Subsequently, section 2.3 gives an explanation about how the transition from a biological neuron to the state-of-art Convolution Neural Network (CNN), occurs. Sections 2.6 and 2.7 are about computer vision problems, namely, semantic segmentation and object detection, upon which this work is mainly based.

2.1 Machine learning

Machine learning is the field of AI that provides systems, the ability to learn automatically without being programmed to do so. It allows machines to learn through observations of data or, instructions to look for some specific pattern in data. Also, learning involves the machines rectifying itself using past learning experiences, such that it can make better decisions in the future. Hence, the practice of machine learning has become widespread in many fields where profitable opportunities and dangerous risks can be recognized without any human intervention or assistance.

The *'learning'* can be categorized into 4 types:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

2.2 Types of learning

2.2.1 Supervised learning

Supervised learning algorithms are designed to learn by examples. The name *'supervised'* implies an algorithm, that learns under the supervision of a teacher. To make a supervised algorithm learn, data (training data) is provided in the form of input and correct output pairs. During the training, the algorithm learns a mapping function that searches for patterns and structures in the inputs that, can be co-related with the desired outputs. The ultimate goal for an trained model is to predict the correct label or class for unseen data (test data) based on the training data.

Supervised learning algorithm is given as:

$$Y = f(X) \tag{1}$$

Provided an input X, the mapping function assigns a label to it and gives the output (prediction) Y

Supervised learning algorithms can be further categorized into two approaches:

- **Classification**: Classification is the problem of identifying the label for new unseen input sample based on the training data.

  Classification problem can further be categorized into two problems:

  1. Binary classification: It is the task of classifying an input sample into two given class/labels.
     For example - Will it rain today or not? Is this cat or not?
  2. Multi-class classification: The input samples are classified into three or more classes/ labels.
     For example - Is this cat, dog or a lion? Is this mail spam, important or promotional?

- **Regression**: A regression model attempts to predict a continuous output variable. In this case, the output Y would be a real value that ranges from $-\infty$ to $+\infty$.
  For example - What is the value of a stock? Price of a house in Kaiserslautern?

Some common algorithms used in supervised learning are:

- Linear regression for regression problems
- Random forest for regression problems
- Support vector machine for classification problems
- K-Nearest neighbour for classification problems

For the sake of completeness, sections 2.2.2, 2.2.3 and 2.2.4 cover the remaining learning techniques in machine learning. These learning techniques are not in the scope of this work; hence, only a shallow understanding is required.

2.2.2 Unsupervised learning

In this type of learning, training data consists of only inputs X and no correct output Y. It is for the algorithm to learn the function f to describe the hidden structure from the given data.

Some common algorithms used in this type of learning are:

- Clustering
- Principal Component Analysis(PCA)
- Singular Value Decomposition(SVD)

2.2.3 Semi-supervised learning

Semi-supervised learning includes a large amount of inputs X, wherein only a part of the data, has output Y. This type of learning is described as the hybridization of supervised and unsupervised learning types.

Some common algorithms used in this type of learning are:

- Generative models
- Transductive algorithms
- Graph-based algorithms

2.2.4 Reinforcement learning

Reinforcement learning is about action and the reward associated with that action. This type of learning is employed in various machines to, find the best possible reaction to a specific situation. The machines or the agents, learn from their own experience, unlike supervised learning, where the models are trained with the correct data.

Some common algorithms used in this type of learning are:

- Q-Learning
- State-Action-Reward-State-Action(SARSA)
- Deep Q Network(DQN)

From the above section, it can be stated that Machine Learning (ML) is a set of algorithms that analyses the data and learns from the analyzed data to discover patterns. In the same way, a neural network is also a set of algorithms used in machine learning to learn from data and discover patterns in it using graphs of neurons. This work is mainly based on neural networks, so, we shift our focus towards it.

2.3 Neural network

2.3.1 From Biology to Artificial Intelligence

The human brain is made up of 86 billion cells called neurons. Each neuron is made up of a cell body connected with many dendrites and a single axon. Dendrites receive information from other neurons and pass it to the cell body, while an axon, sends the information from the cell body to other neurons. Axons are connected to synapses, which are connected to dendrites. A neuron receives electrical inputs at the dendrite, and if the sum of these inputs is sufficient enough to activate the neuron, it transmits an electrical signal along the axon and passes this signal to other neurons. A neuron in the brain looks like figure 3.

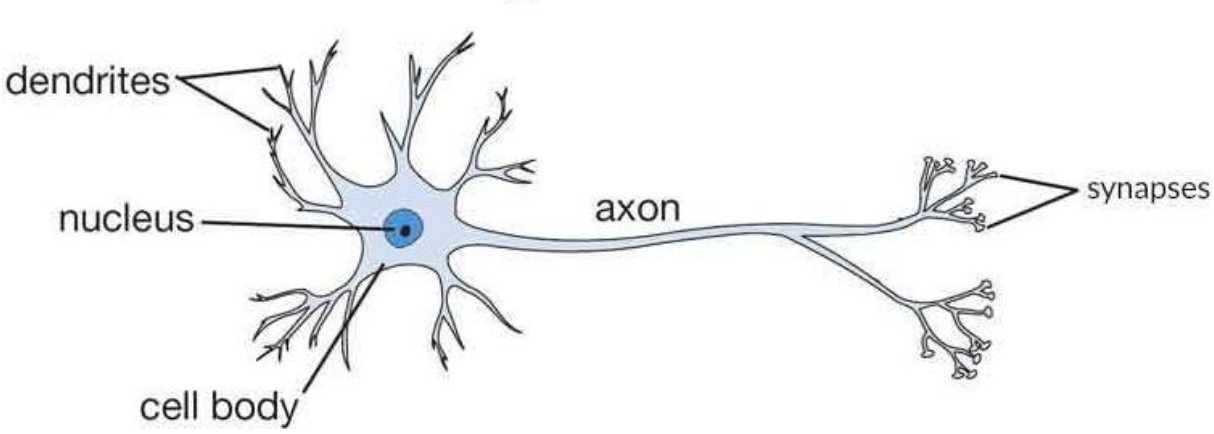


Figure 3: Biological neuron [Fei-Fei 17]

While there has been lots of progress in the area of Artificial Intelligence and machine learning in recent years, the groundwork for everything had been laid out more than 60 years ago. An artificial neuron was developed by Neurophysiologist Warren McCulloch and mathematician Walter Pitts using electrical circuits in the year 1943. Taking inspiration from the human brain and [McCulloch 43], in the year 1958, [Rosenblatt 60] introduced artificial neuron named perceptron. A biological neuron that is mathematically modelled is known as a perceptron.

In a biological neuron, axons from a neuron transmit electrical signals to dendrites of other neurons. In the same way, in perceptron, these electrical signals are represented in the form of numerical values. Between the dendrites and axons, the synapses which modulate these signals by various amounts, exist. Similarly, in perceptron, each input signal is multiplied by a value called weight. If this value exceeds a certain threshold, only then, an actual neuron fires an electrical signal. Similarly, in a perceptron, the weights are calculated with the input signal and then it applies an activation function on the weighted sum ($w_1 \times x_1 + w_2 \times x_2 + ...w_n \times x_n$) to calculate the output. This output signal is then, fed to other perceptrons. A single perceptron looks like figure 4.

The idea behind the neural network is to simulate lots of interconnected brain cells inside a system to make it learn things, recognize structures and make judgement, the way humans do it. Hence, with the same biological motivation behind neurons in the brain, perceptrons that are stacked in several layers form an artificial neural network. In an artificial neural network, the neurons are organized into three layers, namely input, hidden and output. The input layer brings the data into a model and is passed on to the subsequent layers. The hidden layer is between the input and output layer where the neurons take in a set of weighted inputs, apply the activation function (shown in figure 4) to the sum and pass on the output to the output layer.

2.3.2 Activation functions

The activation function operates as a mathematical gate in between the input for the current neuron and its output going to the next neuron. It can be as a function that turns the neuron's output on and off, depending on the threshold. Or it can be a mapping function that maps the input signals into output signals such that the neural network can perfrom action.

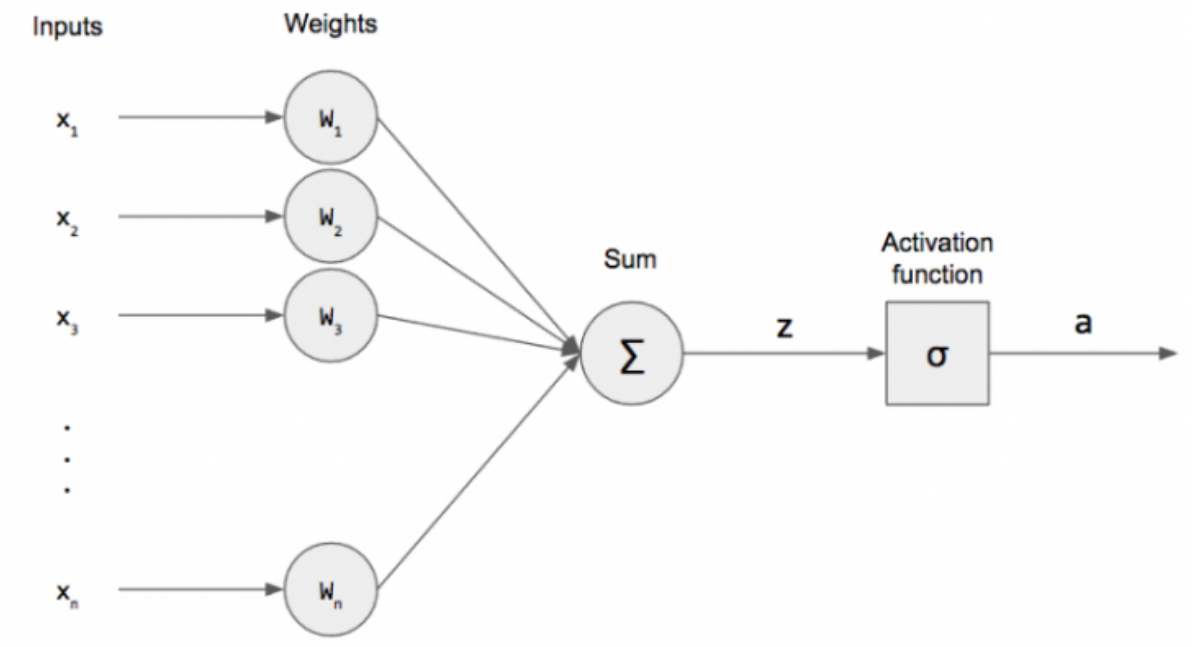


Figure 4: Artificial neuron: Perceptrons [Deshpande 17]

As the activation function is attached to each neuron in every layer, it is desired that the function be computationally efficient. Additionally, the function needs to be differential for training the neural network (covered in section 2.3.5)

There are 3 types of activation functions:

- Binary activation function
- Linear activation function
- Non-linear activation function

Binary activation function

This is a threshold-based activation function. A neuron is only activated if the input value reaches a certain threshold. Once it reaches this threshold, it sends the same signal to the next layer of neurons.

Problem:

1. This activation function does not support the classification of multiple inputs. Hence, it is a binary decision-maker, i.e. 'yes' or 'no.'

Linear activation function

In this activation function, each input is multiplied by the weights for each neuron, and it creates an output signal that is equivalent to the given input.

Problems:

1. The derivative of this activation function always remains a constant and has no relation to the input. Hence, it's not possible to go back and adjust the weights across all the layers through which we can attain better prediction.

2. With linear activation functions, it does not matter how many layers we have in the neural network because the last layer would also be a linear function of the first layer(linear aggregate of linear functions is still a linear function). Hence, all the layers just turn into a single-layered neural network that has limited power and the ability to handle complex input data.

Non-linear activation function

Most of the modern neural networks use non-linear activation functions. Non-linear activation functions are responsible for the model to create complex mappings between the input and the output. Such kind of functions are essential for modelling complex data such as images, videos, and audios which are high dimensional.

Advantages over the above mentioned activation functions:

1. Differential: Methods like gradient descent depend on continuously differentiable functions for finding an optimal minimum. If the functions are not differential, it brings uncertainty in the direction and magnitude of updates to be made in the weights.

2. Monotonic: A monotonic function, guarantees convexity in the loss or cost function that the model tries to minimize.

3. Multiple layered neural networks: Expands the possibility of making a neural network multi-layered or deep such that it can learn complex data and perform well.

4. Approximately behaves as an identity function: Using non-linear activation functions, the neural network gets the ability to learn efficiently when its weights are randomly initialized otherwise we need to initialize the weights for the neural network in a different manner.

5. Non-linearity: Non-linearity ensures that the output cannot be reconstructed from a linear combination of inputs. In the absence of non-linearity, it is possible to replicate the entire network with just one linear combination of inputs. Also, non-linearity is required such that the model can learn a complex function that can map an input to output

Some of the most used non-linear activation functions are:

- Sigmoid:
  - Maps the output in the range of [0,1]
  - The Sigmoid activation function gives rise to the vanishing gradient problem as the larger output contributes to lesser gradients due to which the learning of the model slows down.
  - Sigmoid activation function is given as : $\sigma(x) = \dfrac{1}{1+e^{-x}}$

- Tanh:
  - This activation function squishes the output in the range [-1,1]
  - Tanh activation function is given as : $\sigma(x) = tanh(x)$
- ReLu:
  - This activation function does not allow any negative output components to be propagated; only the positive outputs are allowed.
  - It stills maintains a non-linear shape but its not a completely continuous function.
  - ReLu activation function is given as : $\sigma(x) = max(0, x)$
- Softmax:
  - A softmax activation function is used as the output function of the last layer in neural networks. It turns the score of the last layer into values that can be understood by humans.
  - This function is the same as the argmax function, which returns the position of the largest value from the score vector. This value is interpreted as a probability of the class.
  - Softmax activation function is given as: $s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$; $x_1 ... x_n$ are numbers

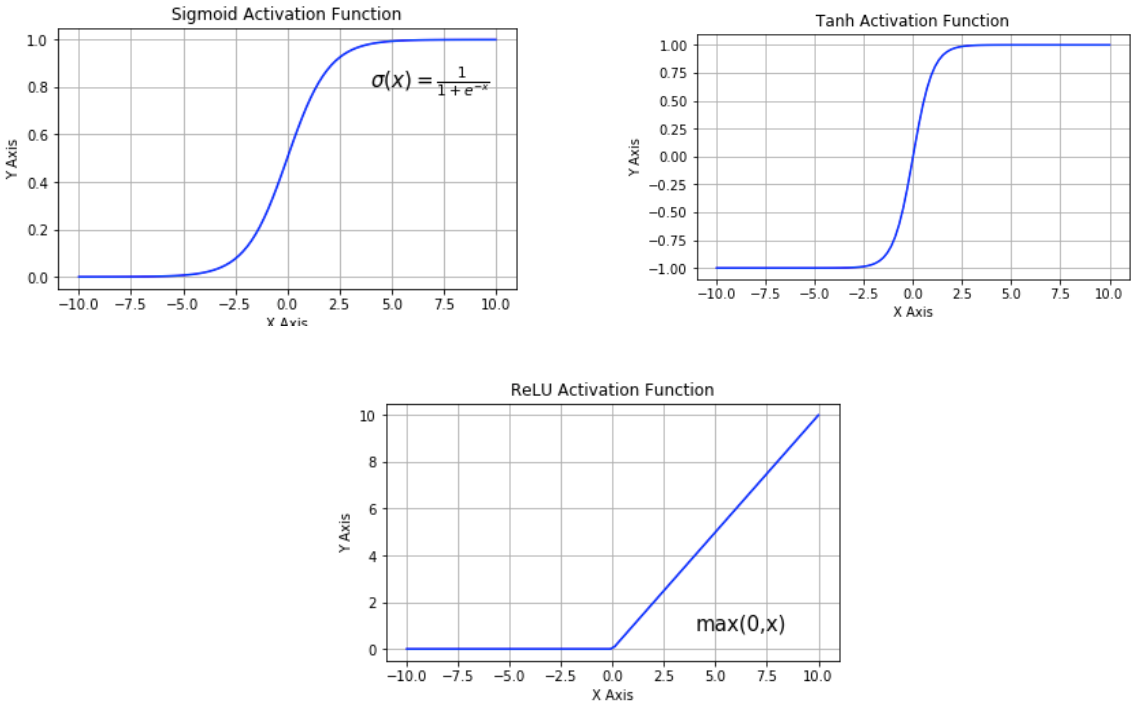


Figure 5: Graphical representation of Sigmoid, Tanh and ReLu functions. [Sharma 17]

2.3.3 Error and loss function

Error is the variation between the actual output and predicted output from the network (output layer). The function that calculates the error is known as the loss function. In simple terms, loss function evaluates the performance of a learning algorithm. Different problems use different loss functions and each of these loss functions give different errors for prediction. Hence, loss functions have a considerable effect on the performance of the model.

Different loss functions are used for different types of problems. Some of the commonly used loss functions are:

- Mean Squared Error (MSE): It is one of the most basic loss functions. This error function calculates the loss by measuring the difference between prediction and actual output, squares it and averages it across the whole data set.
  Equation can be given as:

$$\frac{1}{n}\sum_{i=1}^{n}(prediction - actualoutput)^2 \tag{2}$$

  n is the total size of data set

- Likelihood loss: This loss is mostly used in classification problems. If a model outputs probabilities of [0.1,0.2,0.3,0.4] for the actual output of [1,0,0,1], the likelihood loss would be calculated as $(0.1) \times (0.1) \times (0.4) \times (0.4) = 0.0016$. The loss function only considers the output probabilities where ground truth label is 1 (correctly classified) and for 0 (incorrect classification) $(1 - p)$ is taken as probability.

2.3.4 Optimization

Once the loss is calculated using the loss function, the next step is to reduce the loss such that, the difference between actual output and the prediction decreases. To reduce the loss, we need to find a set of optimal weights for the model. Gradient descent optimization algorithm is used to minimize the loss function and reach the minimum of the function by moving in the direction of steepest descent. The partial derivative of the cost function is calculated with respect to each parameter such that the new gradient gives the slope of the loss function and the direction in which, we need to move to reduce the loss and improve the output of the model. While doing this, we are also updating the weights of our model.

The learning rate determines the size of the steps needed to reach the minimum of the function. If the learning rate is too high, it may require a lesser number of steps to reach the minimum; however, there is also a chance of passing over the lowest point of the function. A low learning rate is highly time-consuming, but, it is more precise, and hence, the required minimum is achieved.

2.3.5 Training neural network

Gradient descent optimization algorithm is iterative; hence it runs over the entire dataset many times. This iteration is known as an epoch. One epoch means, the entire dataset is iterated over the whole model. Each epoch consists of a forward pass and backpropagation. The loss is calculated with the forward pass, and weights are updated during backpropagation.

- During forward pass, the network initializes some random weights for all neurons, for example, some random values.
- Output from the forward pass (predicted output) is compared to the actual output (ground truth), and the error is calculated.
- The error is backpropagated through all the previous layers. The gradients of weights and related activation function are calculated.
- The gradient is calculated using the chain rule. The derivative of the error is calculated with respect to the parameters (weights and activation function)
- With calculated gradient, parameters are updated as follows:

$$\theta = \theta - \alpha \frac{d}{d\theta} J(\theta) \tag{3}$$

  α is learning rate, θ is parameters and J is the loss function.

2.4 Convolution Neural Network

As discussed in the above sections, the neural network or multi-layered network of neurons can learn the relation between the input and output using non-linear mapping function. But in the case of images, a neural network would, for a given task, take into account the entire image. This, in turn, makes the neural network highly inefficient in terms of pragmatic quality as well as neural network quality, as, the number of parameters to learn increases. Hence, to solve this problem, another type of neural network called Convolution Neural Network (CNN) is used,which is specifically designed for computer vision-related tasks such as image classification, semantic segmentation, object detection, etc.

The inception of CNN happened in the year 1960 by D.H Hubel and T.N Wiesel in their paper [Hubel 62] where they described two types of cells in the human brain (specifically in the visual cortex), simple cells, and complex cells. Simple cells are activated when they identify basic shapes in a fixed area and a definite angle. The complex cells have bigger receptive fields, and their output is not sensitive to specific positions in the field. Taking inspiration from [Hubel 62], in the year 1998, CNN was re-introduced in the paper [LeCun 98] called LeNet-5 which was able to classify digits from hand-written numbers. Since it's inception to the present era, the CNN has been used in various state of the art applications, especially in the field of computer vision.

2.4.1 Components of CNN

There are four main operations in CNN. These operations are the essential building blocks in every CNN.

- Convolution operation
- Non-linearity
- Pooling step
- Fully connected layer

2.4.2 Convolution operation

The name convolution is a mathematical operation computed on two different functions x and y which produces a third function expressing how the shape of one is modified by the other. The primary goal of convolution is to extract different features from the input.

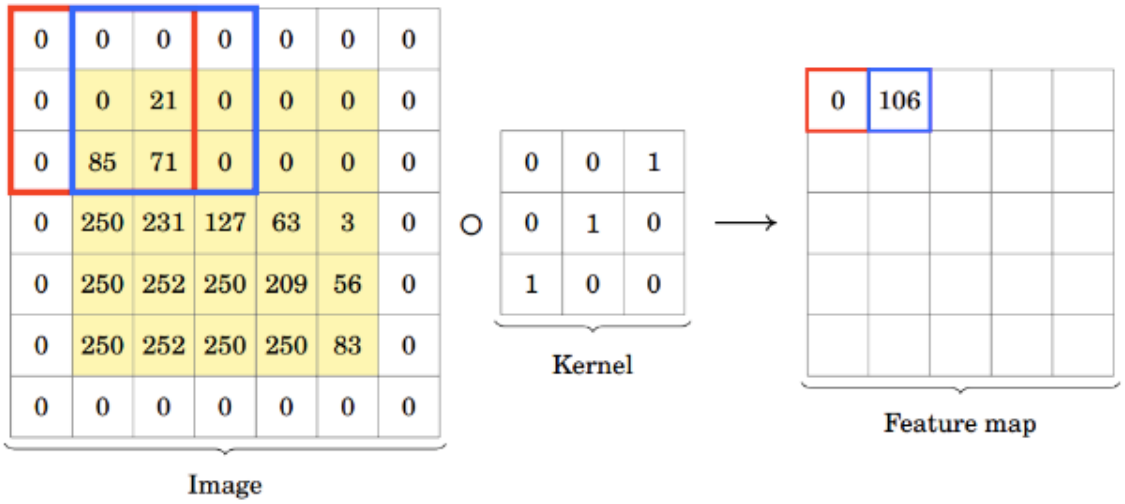


Figure 6: A conolution operation [Pavlovsky 17]

Consider the above example, where there is an image (x) of 7×7 and a kernel or filter (y) of 3×3. Slide the filter over the image pixel-by-pixel and for every position and compute the element-wise multiplication between the two matrices and then, add the outputs to get a final integer which forms a single element in the output matrix. The output matrix is called an activation map or feature map. Hence, a CNN investigates only a piece of the image rather than the entire image at once; making it different from other neural networks. Figure 6 shows how the convolution works.

A convolution neural network learns the values (weights) of the filters on its own during the training process (gradient descent). We can have multiple numbers of filters such that more features from the image get extracted, and better the network becomes in observing and recognizing patterns.

The size of the feature map can be controlled by three parameters, they are:

- Depth - Depth - Depth is the number of filters we use for the convolution operation. For example, if we use three distinct filters on the input image, we will have three feature maps that are stacked together.
- Stride - It determines the number of pixels by which the filter slides over the input matrix. For example, if the stride value is 1, we move the filter over the input by one pixel at a time. If the stride value is 3, we move the filter over the input by jumping 3 pixels at a time.
- Zero-padding - We pad the input matrix with zeros around the border such that filter can be applied to the bordering elements of the input matrix. Applying

zero-padding is called as wide-convolution and not using zero-padding is called as narrow-convolution.

2.4.3 Non-linearity

As explained in the section 2.3.2, non-linear activation functions are responsible for the model to create complex mappings between the input and the output such that modelling complex data such as images, videos, audios, which are, high dimensional in nature is possible.

2.4.4 Pooling step

Pooling step is done to reduce the dimensionality of the feature map and preserve the most important features from the map. This process is also called as downsampling of upsampling.

There are three types of pooling:

- Max pooling - Define a spatial neighbourhood 2×2 window and take the largest element from each window of the feature map to form a new output matrix. Figure 7 shows how max-pooling works.
- Average pooling - Define a spatial neighbourhood 2×2 window and take the average of all the elements in each window from the feature map, to form a new output matrix with the average values.
- Sum pooling - Define a spatial neighbourhood 2×2 window and take the sum of all the elements in the window from the feature map, to form the new output matrix with the sum values.

Pooling has further advantages like:

- It makes the input representations smaller and hence, more manageable.
- It reduces the number of parameters and therefore, controls overfitting.
- It makes the model unvaried to small distortions, translations in the input image.
- It makes an invariant representation of the image due to which, the network can detect objects at any located position in the image.

2.4.5 Fully connected layer

The term 'fully connected' implies that every neuron in the layer is connected with the neurons of the previous layer. The convolutional layers and pooling layers are used to extract features from the input image while, the fully connected layer uses these extracted features to, classify the input image into separate classes based on the training dataset. A softmax function is used as the activation function in order to get the probabilities of each class in the dataset.

Figure 8 shows how all the components are linked to each other in CNN.

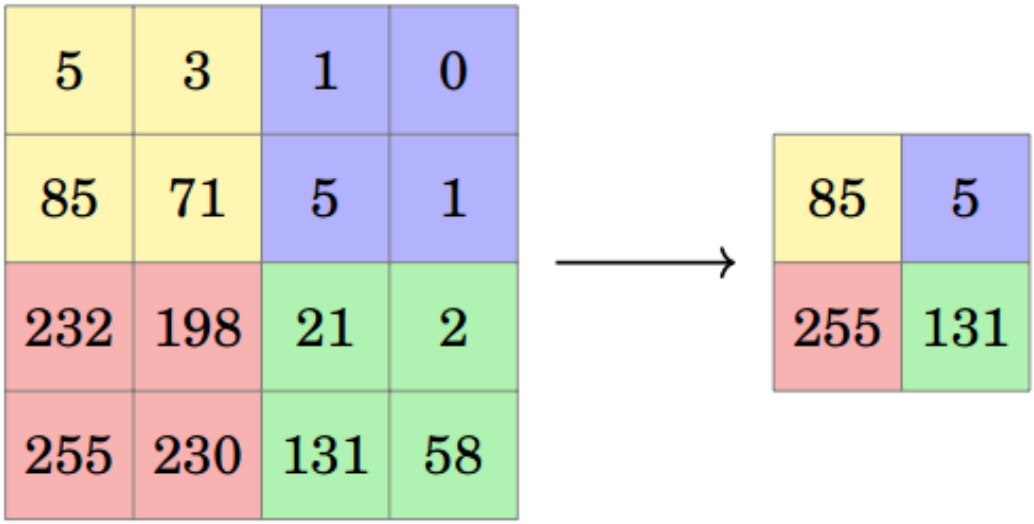


Figure 7: [Pavlovsky 17]

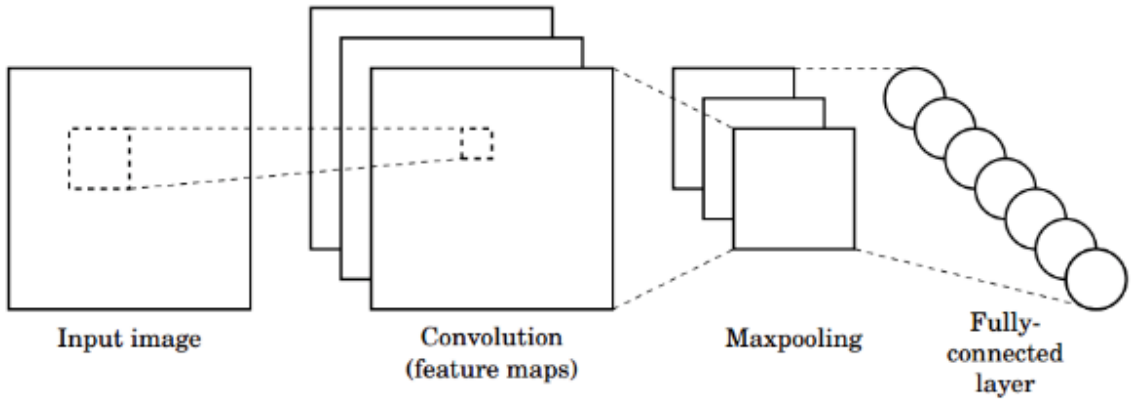


Figure 8: [Pavlovsky 17]

2.5 Feature extraction using CNN

The previous section gives us an understanding of the components in CNN. A CNN can be thought of, as a combination of two components where convolution and pooling layers extract features from an image, while, fully connected layers do the classification using the softmax function. A feature is a measurable piece of data in the image specific to some object. It can be a distinct colour or a specific shape like a line, an edge or an image segment. These convolution layers are capable of learning such complex features in an image. The initial layers detect features such as lines and edges. The next layers combine these basic features to detect shapes and the following layers combine these information

to identify the object.

Below figure 9 shows how features are extracted from a given image.

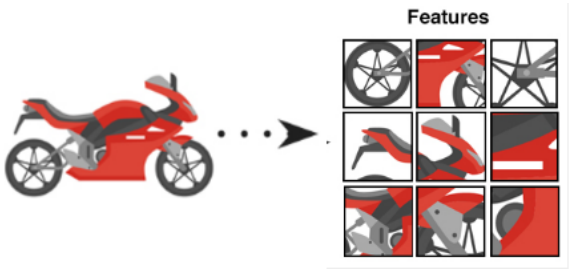


Figure 9: [Elgendy 20]

Such feature extractors are trained and fine-tuned on a very large dataset (for e.g. ImageNet dataset which contains 1.2 million images with 1000 classes). Then, these learned features are directly incorporated into new networks to perform new tasks. This process is known as transfer learning using a pre-trained network. Transfer learning process generally tends to work if the features are suitable to both the base and the new tasks.

Some well known feature extractors are:

- VGG16
- ResNet
- DenseNet

Some advantages of using transfer learning are listed below:

- It saves training time in comparison to train an entire CNN from scratch.
- It works when the labelled data is limited.

2.6 Semantic Segmentation

Semantic segmentation is about understanding an image at the pixel level. To be precise, it is the task of classifying each and every pixel in an image from a set of predefined classes. This process is also referred to as pixel-level classification.

Applications that use semantic segmentation are:

- Autonomous vehicles
- Medical diagnostics
- Facial segmentation
- Geo-Sensing

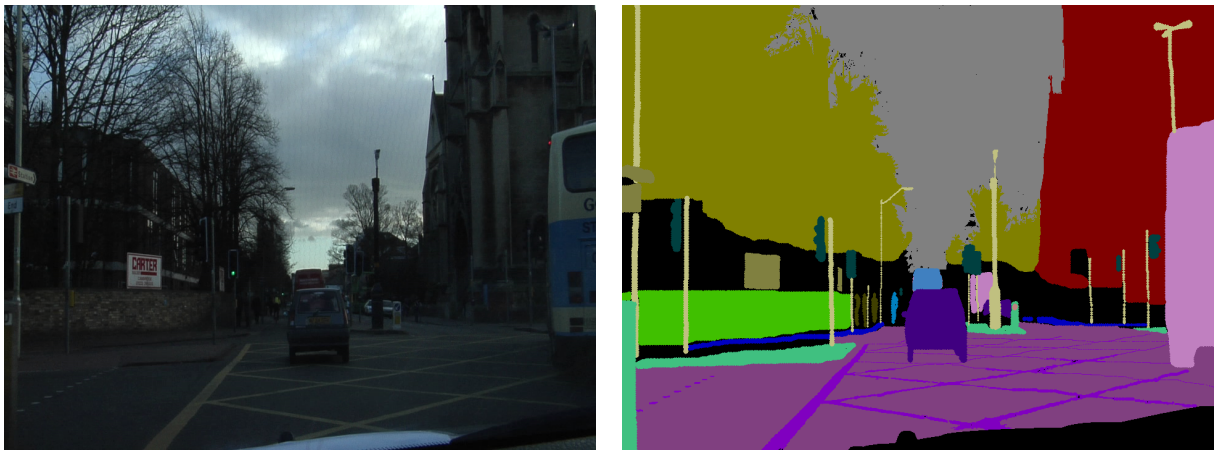


Figure 10: Example of semantic segmentation

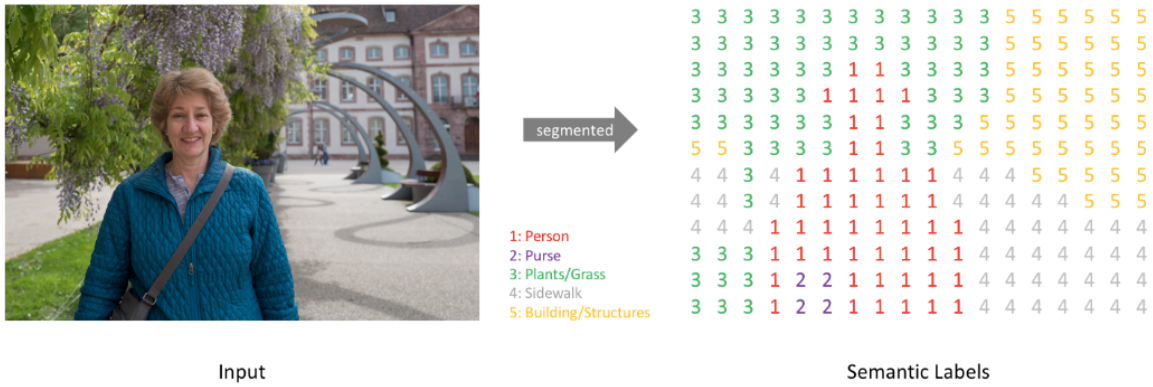


Figure 11: Each class represented as integers [Jordan 18b]

2.6.1 Representation of the task

The goal in semantic segmentation is to take an image and output a colour map where each pixel contains a class label represented as an integer (as shown in figure 11.)

Every class label is one-hot encoded, and an output channel is created for each of these classes. One hot encoding means a way of representing the data in a binary string where only a single bit can be 1 and rest all the bits are 0. By doing this, we stack all output channels in such a way that 1's show the existence of a particular class.

Figure 12 shows one hot encoding of figure 11.

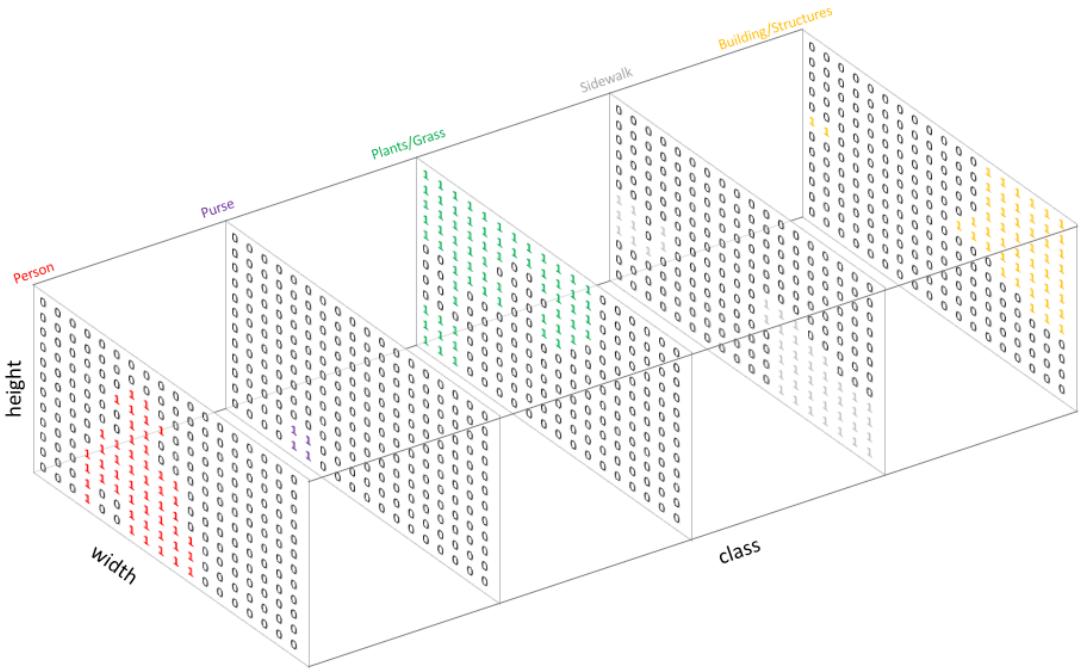


Figure 12: One hot encoding [Jordan 18b]

The prediction is obtained by collapsing all the class maps into a segmentation map by taking the argmax of each depth-wise pixel vector.

2.6.2 Architecture

The most naive approach for semantic segmentation architecture is to stack a specific number of convolution layers with the same padding in order to preserve the dimensions of the image and output a final segmentation map. Figure 13 shows the most basic architecture used for semantic segmentation.

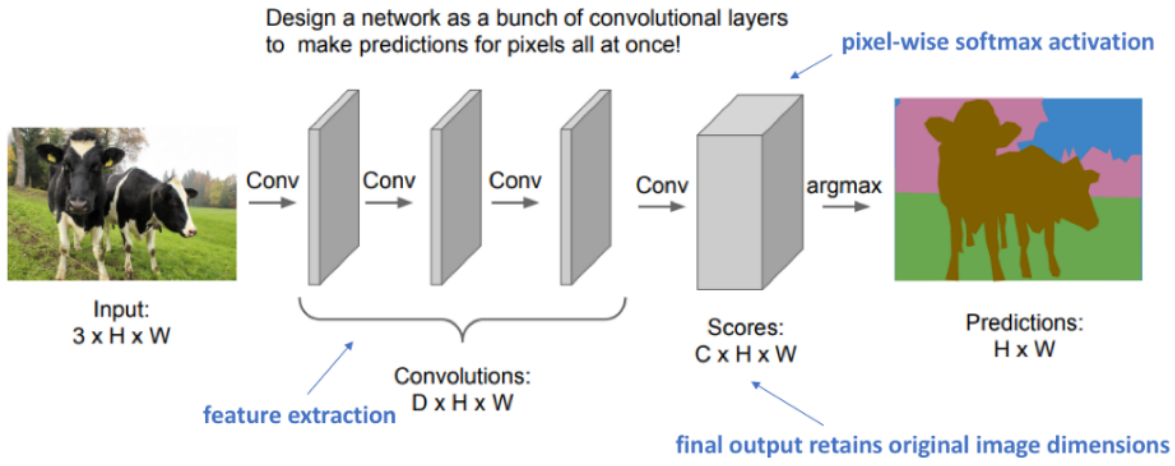


Figure 13: Preserving spatial resolution [Fei-Fei 17]

As the above method is computationally very expensive because of preserving the dimensions throughout, a newer approach of segmentation was developed using the encoder-decoder architecture. Encoder downsamples the spatial resolution of the input image, develops lower resolution feature maps which are highly efficient at discriminating classes. The decoder does the upsampling of the feature maps into a full resolution segmentation map. Figure 14 shows the encoder-decoder architecture.

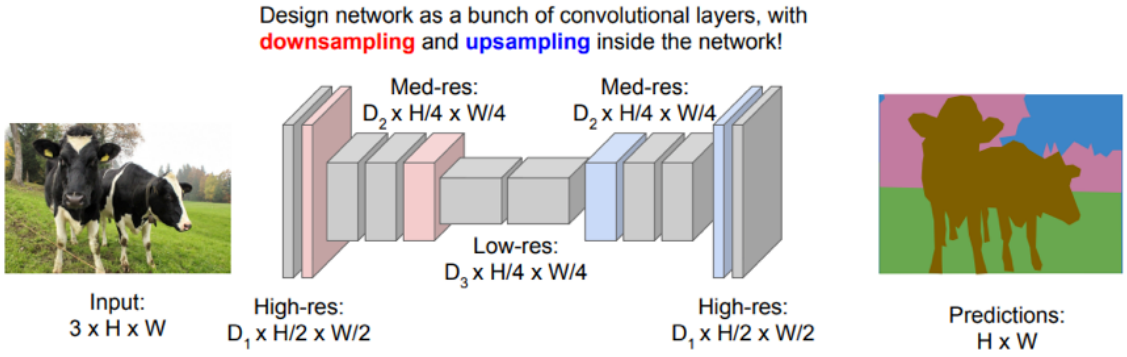


Figure 14: Encoder-Decoder architecture [Fei-Fei 17]

Recent architectures that perform semantic segmentation use pre-trained feature extractor as an encoder such that a rich feature representation of the original input image is achieved.

Methods of upsampling

The above sections give an understanding as to how an encoder and decoder architecture works in semantic segmentation. The pooling operation downsamples the spatial resolution by summarizing a local area using a single value (e.g. max pooling), while unpooling operations upsample the feature map resolution by distributing a single value to a higher resolution. Some unpooling approaches are listed below:

- Nearest neighbour: This is one of the most basic approaches to upsample the feature map. It copies the value from the neighbouring pixel and does upsampling.
- Max unpooling: This technique works by storing the locations of a maximum element within each pooling window and placing it at the same location while unpooling. Rest of the values contained in the pooling window are zeroed out.
- Transposed convolution: This is one of the most popular approaches amongst the upsampling techniques. This technique allows learned upsampling. A typical convolution operation does the element-wise dot product of the image and filter's view and produces a single value in the output matrix. While a transpose convolution does the complete opposite, we take a single value from the low-resolution feature map and multiply it with all values in the filter and project the filter values onto the output feature map. Figure 15 below gives a 1-D example of how transposed convolution works.

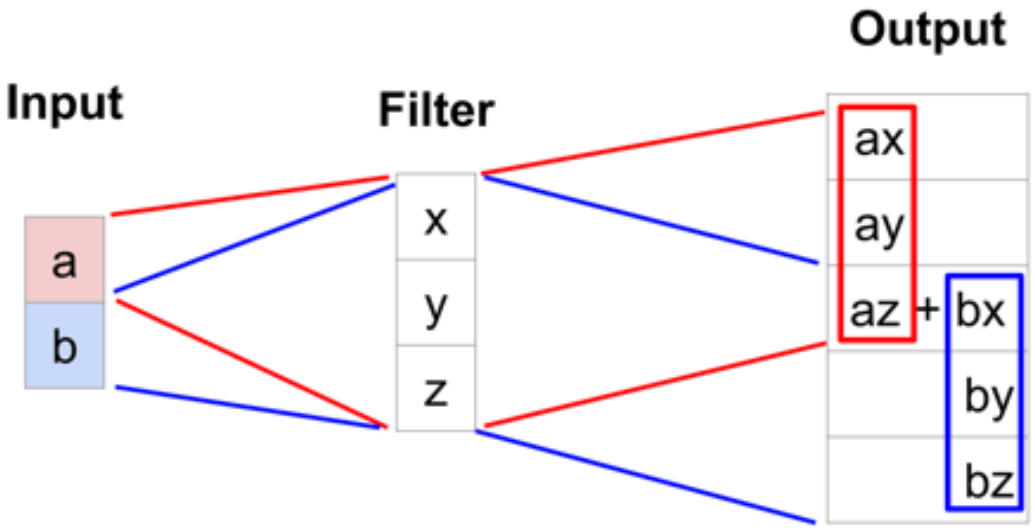


Figure 15: 1-D transpose convolution [Jordan 18b]

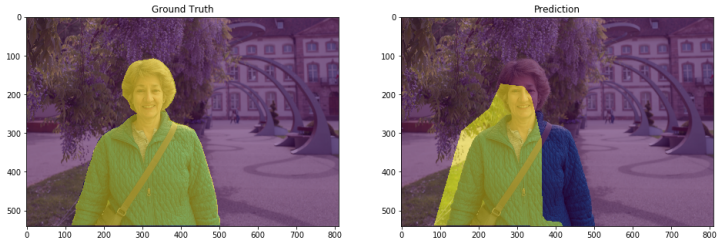
2.6.3 Evaluation metrics

- Pixel accuracy: It is the percent of pixels that are correctly classified in the image. If a pixel is correctly predicted to belong to a certain class, it is counted as a true positive sample, whereas true negative represents a pixel that is correctly identified as not belonging to the correct class. Even though this metric is really easy to understand, this metric does not give a proper measure of performance. Class imbalance, means, the dominance of certain class in an image affects this metric the most.

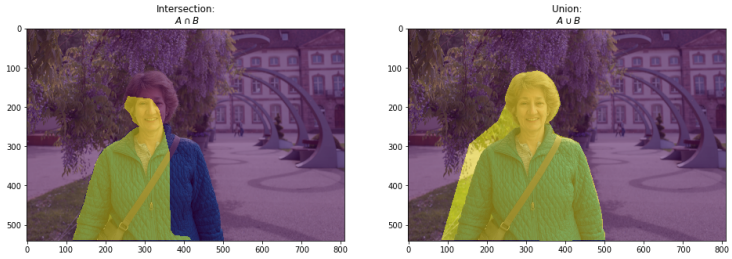
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

- Intersection-over-Union(IoU): It is one of the most commonly used metrics in semantic segmentation. IoU is the area of overlap divided by the area of union between the predicted segmentation map and the actual output. This metric ranges from 0-1 (0-100%) where 0 signifies no overlap and 1 signify perfect segmentation map according to the actual output.

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{5}$$



(a) Actual output and prediction sample



(b) Intersection and union between actual output and prediction

Figure 16: [Jordan 18b]

2.7 Object detection

Object detection is a technique used for finding objects of interest in an image. It is about finding multiple objects, classify them and locate them in the image using a bounding box.

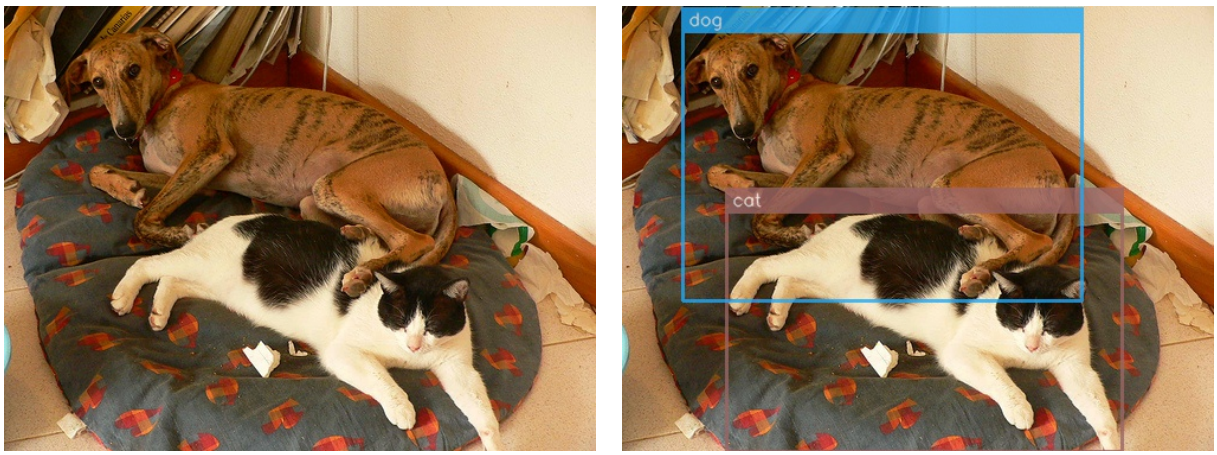


Figure 17: Example of object detection

Applications that use object detection are:

- Face detection
- Visual search engine
- Aerial image analysis

- Counting objects

Object detection can be categorized in two different types of approaches:

- Two-stage object detection: In this approach, object detection happens in two stages. First, the model proposes a set of a region of interests using a region proposal network. The second part of the model does the classification and bounding box regression over region proposals.

  Some well known two-stage detection architectures are:

  - R-CNN
  - Fast R-CNN
  - Faster R-CNN

- One stage object detection: This approach requires only a single pass through the CNN to detect all the objects in an image in one go. Due to this factor, these models are simpler and faster at the performance.

2.7.1 Direct object prediction using grid

One stage object detectors use pre-trained feature extractor to build a rich feature representation of the original input image. We remove last few layers from the feature extractor such that the output is a collection of stacked feature maps which describe the original input image in low spatial resolution. Consider an example of $7 \times 7 \times 512$ representation of the original input image where each of the stacked feature maps describes different characteristics of the image. This 7×7 obtained from the feature extractor roughly locates the object in the original input image. This feature map can be represented as a grid responsible for detecting the object as it contains the centre point of coordinates of the bounding box.

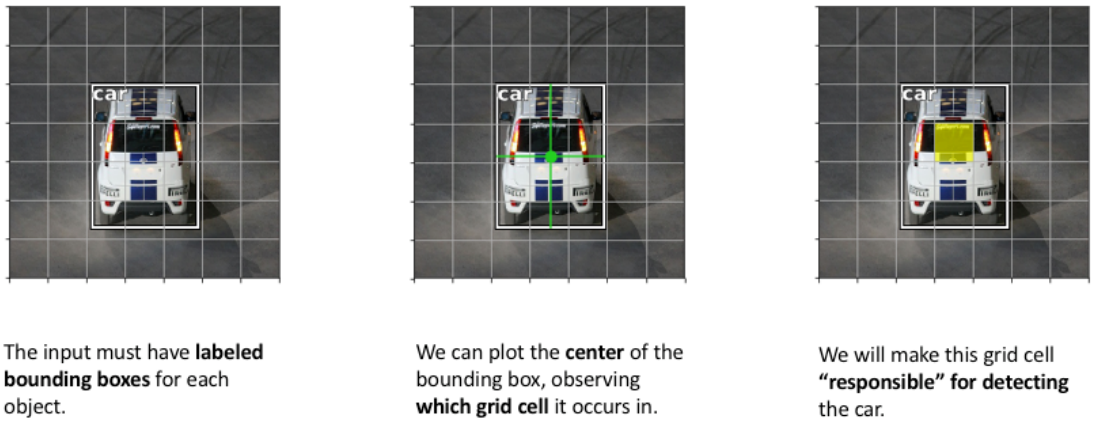


Figure 18: [Jordan 18a]

We combine all the feature maps in order to produce an activation corresponding to the particular grid cell containing the object. Considering we might have more that one object per image, we should have multiple activations on specific grid cells.

However, to fully describe the detected object, we need

- The probability of an object contained in the grid cell P_{obj}
- Class of the object $(c_1, c_2, ...c_c)$
- And the co-ordinates of the bounding box for the object (x, y, w, h)

Hence, as per the requirements, we need to learn a convolution filter for each of the above-mentioned attributes such that it outputs $5 + C$ output channels in order to describe a single bounding box at each grid cell location. The figure 19 shows us the output on applying $5 + C$ convolution filters.

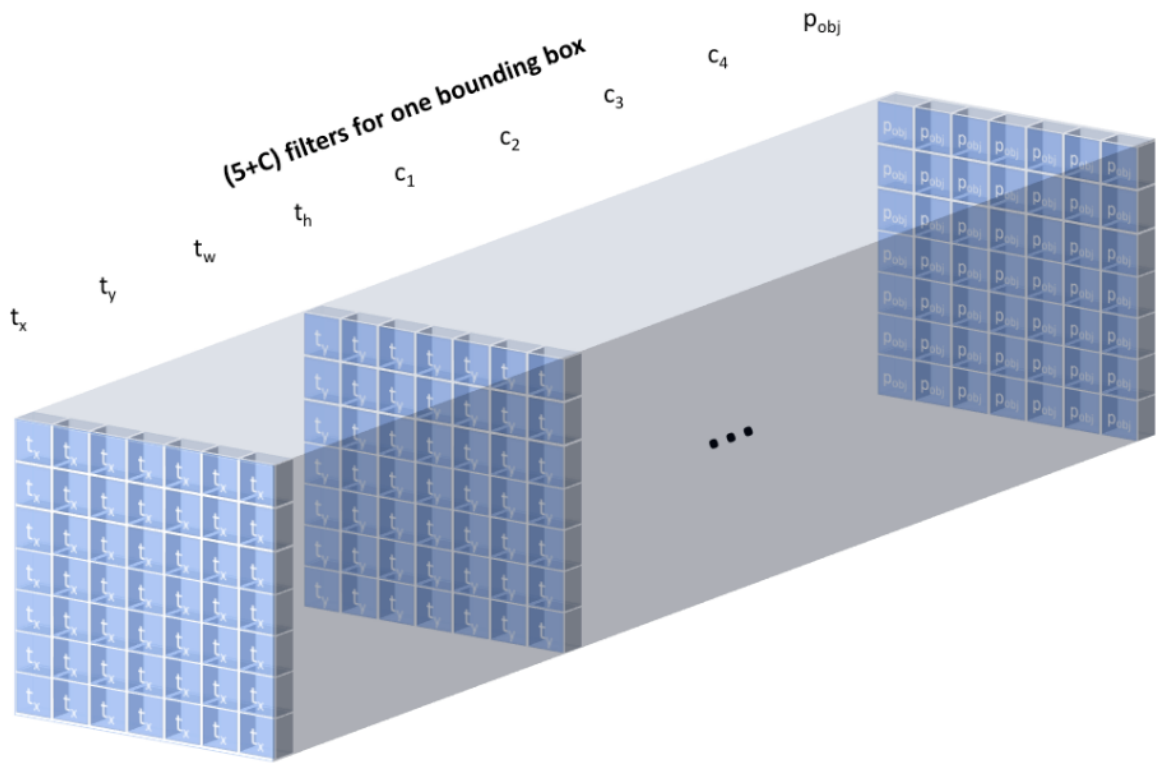


Figure 19: [Jordan 18a]

Considering we might have multiple objects belonging to the same grid cell, we need to alter our layer such that it produces B$(5+C)$ filters for B bounding boxes for each grid cell. Hence, our model will produce a fixed number of $N \times N \times B$ predictions for a given image. By setting a threshold for P_{obj} we can limit our predictions. However, we end up introducing a problem by creating a large imbalance between the predicted bounding boxes containing an object and bounding boxes that do not contain any object.

2.7.2 Non Maximum Suppression

In order to resolve the problem of imbalance between predicted bounding boxes containing object and predicted bounding boxes that don't contain any object, we use a filtering technique called Non-Maximum Suppression. We filter out the predictions that are noisy and may not contain any object. Plus, we want just a single bounding box prediction for each object detected.

Hence, we set the P_{obj} threshold and filter out most of the bounding box predictions that are below the set threshold. However, we might still be left with multiple bounding box predictions describing the same object. Hence, we perform the following steps in order to remove the redundant bounding boxes:

- We need to select the bounding box prediction with the highest confidence score
- Calculate the IoU score between the selected (highest confidence score) and all the remaining predictions
- Remove the prediction boxes which have an IoU score above some defined threshold
- Repeat the above-mentioned steps until no more prediction boxes are remaining to be suppressed.

Fig. 20 shows how an image looks before and after applying Non Maximum Suppression.

Figure 20: [Sambasivarao 19]

We perform Non-Maximum Suppression on each of the class separately.

2.7.3 Evaluation metrics

Average precision

The accuracy of object detection models is measured in terms of classification and localization. The metric used to suit this purpose is the Mean Average Precision (mAP). Average precision is calculated for each class separately under the area of the precision-recall curve.

- Precision: It is the percentage measure of correct predictions by the model.

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP)} \tag{6}$$

- Recall: It is the percentage measure of possible ground-truth bounding boxes being detected.

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{7}$$

For both precision and recall, True Positive is the measure of predictions that has IoU greater than 0.5 with ground-truth bounding boxes, while, False Positive (FP) is the measure of predictions with IoU less than 0.5 with the ground-truth bounding boxes and False Negative (FN) is the number of ground-truth bounding boxes that are not detected by the model.

The predictions by the model are sorted by the confidence score from highest to lowest. Then, 11 different confidence thresholds (known as ranks) are chosen such that recall at each of those rank values have 11 values ranged from 0 to 1 with 0.1 intervals. These thresholds should be in a way that recalls at those confidence values is $0, 0.1...1.0$. Average precision is now calculated as the average of maximum precision values at those 11 selected recall values.

Average precision for class c is defined as:

$$AP_c = \frac{1}{11} \sum_{r \in (0,0.1...1)}^{R} \max(P(r)) \tag{8}$$

Here $P(r)$ is the precision value for one of the 11 recall values (r)

Mean average precision is the average of AP's over all the classes. The mAP is given as:

$$mAP = \frac{1}{C} \sum_{c}^{C} AP_c \tag{9}$$

Here, C is the total number of classes and AP_c is the AP for particular class c.

3 Related work

This chapter gives a brief understanding of the evolution of semantic segmentation and object detection solved using ML based techniques to DL based techniques. This section also gives an insight into the approaches used for semantic segmentation and object detection. Taking inspiration from these approaches, we decide upon the architecture that can be used for this work.

3.1 Semantic Segmentation

Section 2.6 describes the process of linking each pixel of an image with a class. This task is one of the grand challenges in the field of computer vision. It all started with researchers using traditional machine learning algorithm [Dollár 09] with the help of techniques such as edge detection [Huang 10], clustering [Zheng 18], region growing [Gómez 07] and SIFT [Suga 08]. But all of these techniques required the features to be extracted manually and hence, this becomes a tedious job and requires domain expertise.

Such ML-based techniques slowed down around the era of DL as it started to take over the world of computer vision because it needs only data. Also, amongst different learning algorithms in the field of DL, CNN got a tremendous amount of success in the area of semantic segmentation.

In DL, R-CNN [Girshick 14] used selective search algorithm [Uijlings 13] to extract region proposals from the image and then applied CNN upon each region proposal and achieved record result for PASCAL VOC dataset. This technique was a leading hand in the area of semantic segmentation at that time. Around the same time [Gupta 14] used CNN along with geocentric embedding on RGB-D images for semantic segmentation. After this phase, Fully Convolutional Networks (FCN) [Long 15] gained the highest attention as it achieved SOTA result. FCN used base model Visual Geometry Group (VGG)16 as the feature extractor and bilinear interpolation technique for the upsampling of feature maps. It also used skip connections for combining low and high layer features in the final feature map to achieve fine-grained segmentation map. FCN used only local information which makes semantic segmentation quite ambiguous. Hence, to reduce the ambiguous information from the image, [Mostajabi 14] used contextual features and achieved SOTA result. Lately, [Ronneberger 15] used a U shaped network known as U-Net, which consists of a contracting and expansive pathway approach to semantic segmentation. The contracting path is responsible for extracting image features and reduce spatial information; expansive pathway upsamples the contracted feature map. In each upsampling step, the network concatenates the reduced up-convolved feature map with corresponding cropped feature map from contracting pathway. By using both high level and low-level spatial information, U-Net achieves precise segmentation map. Segnet [Badrinarayanan 15] also follow the same footsteps of U-Net by using an encoder-decoder network. An encoder is used to extract image features, and the decoder uses un-pooling operation to get a segmentation map of a size similar to the input image in order to get a precise localization of the segmented object.

This work mainly focuses on encoder-decoder architecture. As mentioned above, the encoder is used to extract features, and give a spatially reduced feature map from images, and the decoder does upsampling of the feature maps to achieve segmented map.

3.2 Object Detection

Section 2.7 describes the process of detecting instances of visual objects in an image. This task is one of the most major and challenging problems in the area of computer vision. Nearly all of the early object detection algorithms were based on extracting features manually due to the lack of effective image representations. [Viola 01] achieved real-time detection of human faces that is faster than other algorithms and with better detection accuracy. This detector follows the most straight-forward way of sliding windows through all possible scales and locations in an image to check if any of the windows contain a human face. Histogram of Oriented Gradients (HOG) by [Dalal 05] was the next revelation in this area, which was considered as an important improvement over other detectors. In order to detect multiple objects of different sizes, the HOG detector re-scales the input image numerous times while keeping the size of the sliding window same. HOG detector has proven to be an important foundation for many object detectors. Such traditional methods became saturated slowly and steadily.

In 2012, the re-birth of DL happened, which lead to the application of CNN in the field of object detection. RCNN [Gupta 14] was the first object detector in the DL era. The idea behind RCNN was the extraction of a set of object proposals using selective search [Van de Sande 11] and each of these proposals is fed into a CNN model to extract features. Lastly, linear SVM classifiers are used to predict the presence of an object within each proposal and identify the class of the object. Fast RCNN [Girshick 15a] detector was an improvement over RCNN that enables to simultaneously train a detector and bounding box regressor under similar network configurations. Faster RCNN by [Ren 15] was an upgrade of Fast RCNN. It introduced Region Proposal Network (RPN) that enables complimentary region proposals. Components like feature extraction, bounding box regression, proposal detection were integrated into a unified end-to-end learning framework. After a variety of improvements over the above-mentioned networks, [Lin 17] proposed a top-down architecture with lateral connections in order to build high-level semantics at all scales. This type of architecture shows great advances for detecting objects at various scales. Above mentioned DL methods to detect object are categorized as two-stage detectors where one stage extracts object proposals, and the next stage classifies objects contained in the proposals.

DL methods also consist of one-stage object detectors which skip the object proposal step and runs detection directly over an image. YOLO proposed by [Redmon 15] is a one-stage detector that divides the input image into grids and predicts bounding boxes and probabilities for each of the grid simultaneously. Due to this, YOLO is speedy at detecting objects. However, YOLO suffers from a drop in localization accuracy for small objects. SSD proposed by [Liu 16] has paid attention to this problem. SSD uses multi-reference and multi-resolution detection techniques which makes it better at detecting small objects.

This work mainly focuses on one-stage object detection networks. As mentioned above, one-stage detectors skip the region proposal step and run detection directly.

3.3 Combining semantic segmentation and object detection

The above-mentioned steps give a high-level understanding of the architecture to be used for performing both the tasks. This section involves a brief background as to how both the architectures (semantic segmentation and object detection) can be connected using a single common pipeline.

[Salscheider 19] uses an encoder-decoder architecture for both the tasks parallelly. An encoder is a feature extractor whose last feature map is shared in between the semantic segmentation head and object detection head. Also, [Teichmann 16] also uses an encoder-decoder architecture where the encoder is shared among 3 tasks, namely, object detection, semantic segmentation and image classification.

4 Approach and Implementation

This chapter covers the approach, methodology and design of architecture used in this work. Section 4.1 gives an explanation about datasets used for training and evaluating the model. Sections 4.4 and 4.5 explain different architectures using feature extractor as encoders in semantic segmentation and object detection. Section 4.6 shows how we can combine different architectures through a common feature extractor/ encoder that can solve both the problems mentioned above parallelly.

4.1 Dataset

Datasets play a vital role in the field of DL. A useful dataset gives a significant boost to an algorithm's performance. Preparing a sizeable amount of good and standard datasets require time and knowledge to gather relevant information. However, there are several publicly available standard datasets in the field of image processing.

Out of all the publicly available datasets, we opt for PASCAL-VOC 2012 [Everingham] and KITTI Vision Benchmark [Geiger 12] datasets to train and evaluate our model. The following sections consist of a brief description of both the datasets mentioned above.

4.1.1 PASCAL-VOC 2012

PASCAL VOC is a benchmark challenge in computer vision tasks, which provides the computer vision and machine learning association with a standard dataset of images, annotations, and evaluation procedures [Everingham]. Since it's inception in the year 2005, the dataset consisted of only 1578 images (4 classes: bicycle, car, motorbike, people) for the task of classification and detection; while in the year 2012, the number of images went up more than 10,000 images consisting of 20 classes. The 2012 challenge includes dataset consisting of images that are used for detection, classification and segmentation, which is categorized as:

- Detection and classification - 17, 125 images
- Segmentation - 2,913 images

4.1.2 KITTI Vision Benchmark

KITTI dataset is a novel challenging real-world computer vision benchmark. This dataset was captured by driving autonomous cars developed by [AnnieWAY 06]. These cars were driven in rural areas and on highways around the mid-size city of Karlsruhe, Germany. Two high-resolution colour and grayscale video cameras attached to these cars are used to capture images for the dataset. Each image in the dataset consists of up to 15 cars and 30 pedestrians. KITTI consists of images that can be used for detection and segmentation, categorized as:

- Detection - 7,481 images
- Segmentation - 200 images

Some sample images of both the datasets are shown in the table 1 and 2.

Table 1: Sample images from the PASCAL-VOC 2012 dataset

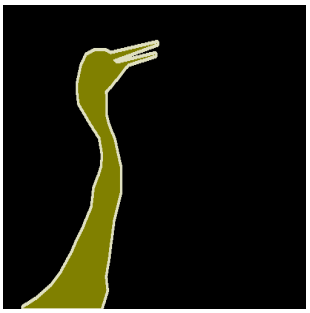
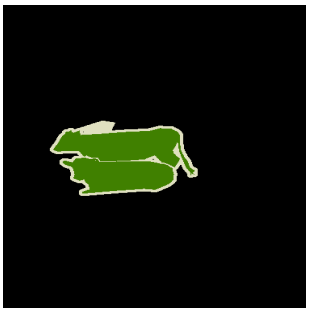
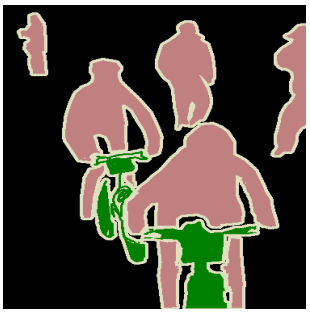
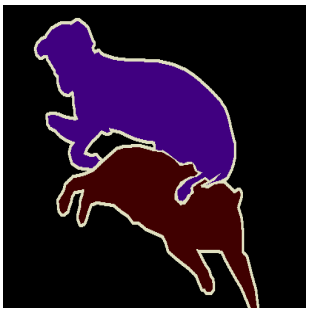
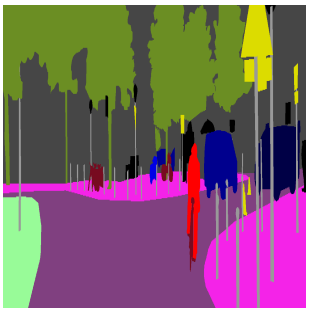
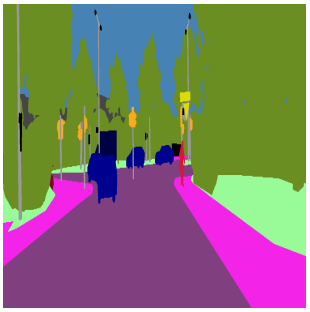
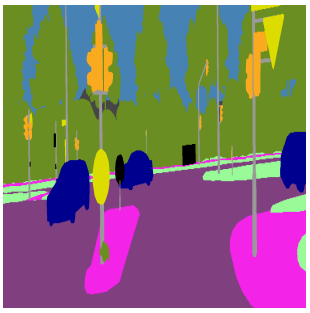
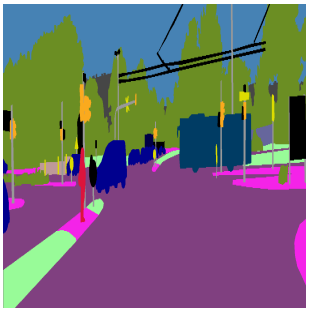
	
	
	
	

Table 2: Sample images from the KITTI dataset

This work is based on creating a common pipeline to perform semantic segmentation and object detection. This poses a challenge to create a dataset that contains an equal number of ground truth for semantic segmentation and object bounding box coordinates for object detection. In case of PASCAL-VOC 2012 dataset, the bounding box coordinates for each object in the image are already available in XML format, but it's not the same in the case of KITTI dataset. Hence, a possible solution to this is using [Tzutalin 15] to create bounding box coordinates for each object manually and save it in XML format.

Figure 21 shows how bounding box coordinates can be created using LabelImg tool.

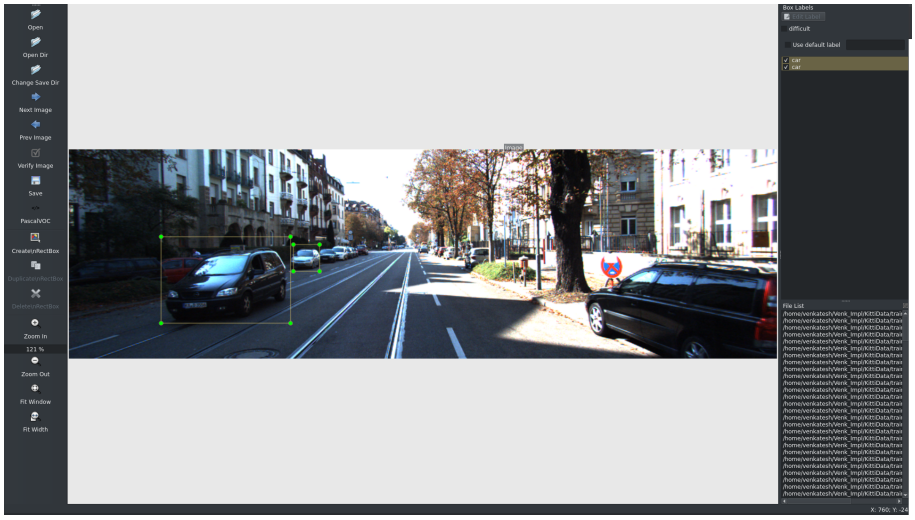


Figure 21: LabelImg tool

Training-validation split for both the datasets are:

- PASCAL-VOC 2012 - Training: 2310 Validation: 583
- KITTI - Training: 160 Validation: 40

4.2 Data augmentation

Data augmentation is a strategy that enables the diversity in data available for training DL models. This strategy proves to be important because it helps reducing overfitting and improves the generalization of the model. In order to achieve this, a series of augmentations on the data can be applied. Augmentation of data can either be performed offline or on the fly such that it can cover most of the variances and scenarios expected to appear in the real world.

Some of the augmentations applied randomly to the data in our work are:

- Brightness transform: -30 to 30
- Contrast transform : 0.5 to 1.5
- Hue transform: -18 to 18
- Saturation transform: 0.5 to 1.5
- Horizontal flip

4.3 Feature extractor/ Encoder

Section 2.5 mentions how CNN is used to extract features from images. As this approach is based on using a common encoder to do semantic segmentation and object detection, choosing an effective image feature extractor is of utmost importance. [Simonyan 14] neural network from VGG group, University of Oxford is used for extracting image features in this work. This architecture is an improvement over [Krizhevsky 12] where large-sized filters are replaced with multiple small-sized filters stacked one after the other and increasing the depth of the network. The arrangement of the layers are consistent and built using 3×3 convolutional layers with stride 1 to increase the depth, and the volume is reduced by using max-pooling of 2×2 filter with stride 2. In the end, 2 fully connected layers, each with 4,096 nodes, followed by a softmax function for the output.

Some pros and cons using VGG16 are stated below:

Pros:

- Multiple, stacked, smaller sized filters increases the depth of network hence, enabling it to learn more complex features at a lower cost.
- Very simple and consistent architecture.

Cons:

- Consists of too many weight parameters making it very memory consuming.
- Too many parameters makes the training process slow.

Figure 22 shows the architecture of VGG16.

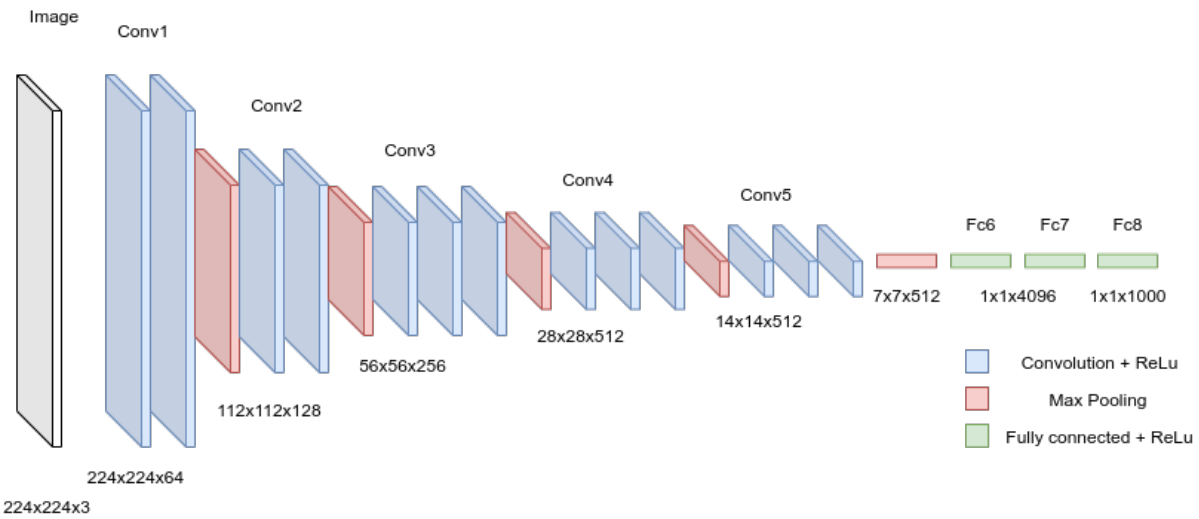


Figure 22: VGG16

4.4 Semantic segmentation

Section 2.6 gives an understanding as to how semantic segmentation is performed using an encoder-decoder architecture. FCN [Long 15] is used for semantic segmentation in this work. FCN has revolutionized the area of semantic segmentation as it can process the whole image in a single forward pass and hence, exploits contextual information in images efficiently and better way. FCN is an encoder-decoder architecture, where, VGG16 is used as a pre-trained encoder (feature extractor). This architecture consists of only convolutional and pooling layers, giving them the ability to produce arbitrary sized predictions. Due to this reason, the fully connected layers (Fc6 and Fc7) of VGG16 are converted to 1×1 convolution. The extracted features are upsampled using transposed convolutions using bilinear interpolation features. Skip connection is introduced after each convolution block to merge features from various resolution levels that helps in combining context information with spatial information. Figure 23 shows the architecture diagram of FCN.

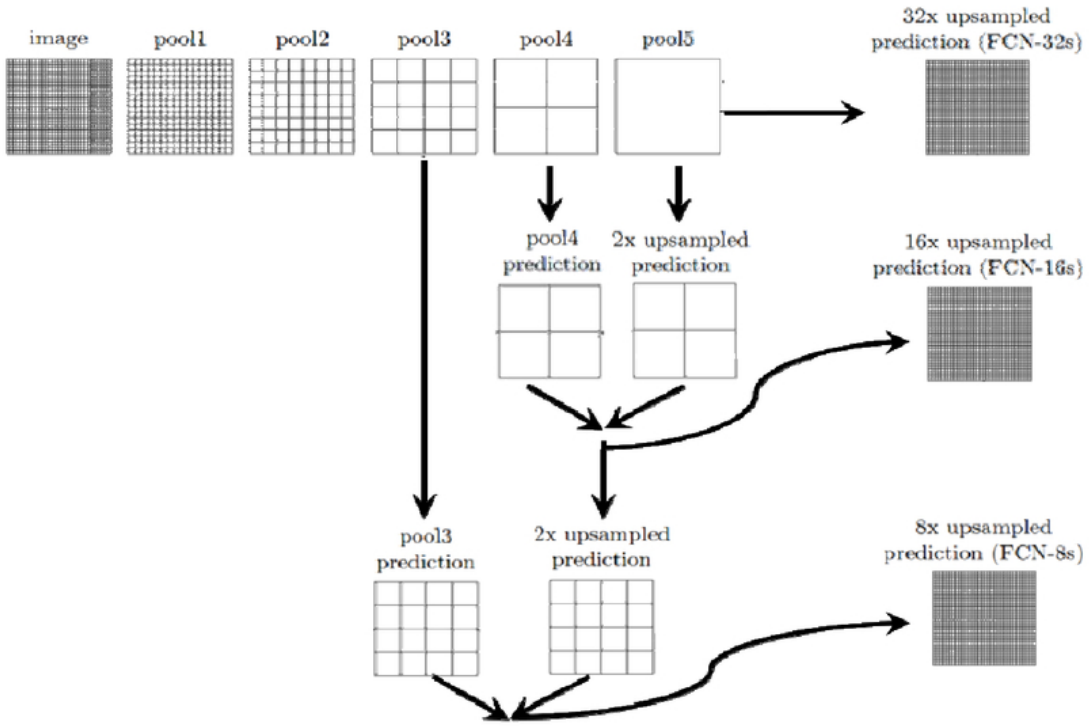


Figure 23: Architecture of FCN-32, FCN-16 and FCN-8

There are 3 variants in FCN. These variants have common downsampling path but different respective upsampling paths; they are:

- FCN-32: Segmentation map is directly created from conv7 layer by using a transposed convolution layer with stride 32.
- FCN-16: The prediction from conv7 is upsampled using the transposed convolution of stride 2 (2x upsampling) and added to the pool4 feature map. This summed feature map is upsampled again using the transposed convolution of stride 16 to produce segmentation map.

- FCN-8: The summed feature map of conv7 (2x) and pool4 is upsampled using the transposed convolution of stride 2 and added to pool3 feature map. This summed feature map is upsampled using transposed convolution with stride 8 to produce segmentation map.

4.4.1 Calculating loss

Pixel-wise cross entropy loss

This is one of the most generally used loss function for the task of semantic segmentation. It compares each pixel of the class prediction (depth-wise pixel vector) to the one-hot encoded class label. Figure 24 gives an example of how loss is calculated.

Pixel-wise cross-entropy loss can be calculated as:

$$-\Sigma_{classes}\, y_{true}\, log(y_{pred}) \tag{10}$$

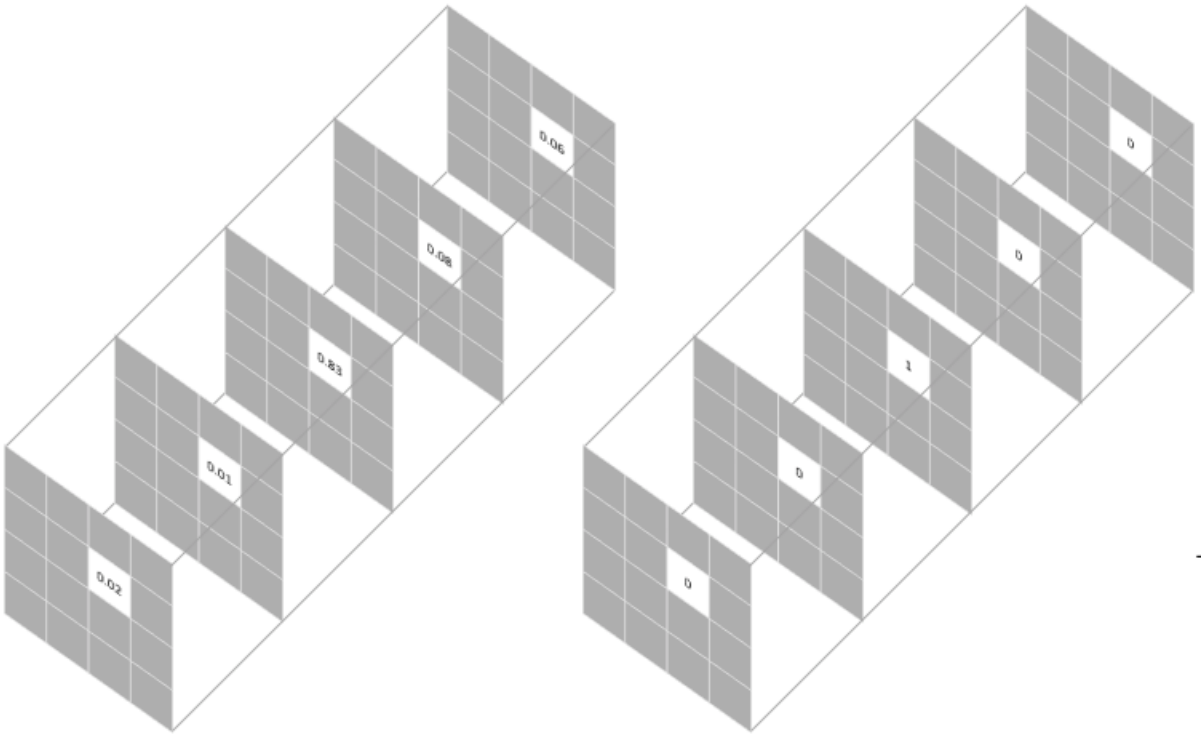


Figure 24: (Left) Prediction for a selected pixel and (Right) one hot vector of the corresponding pixel

Pixel-wise loss is calculated as the log loss and is summed over all classes contained in the dataset. This calculation is averaged after being calculated for all pixels.

4.5 Object Detection

Section 2.7 gives an understanding of the approaches of object detection and how one stage object detector works. Single shot multibox detector (SSD) [Liu 16] is used for object detection in this work. SSD is a one stage detector that learns to map classification and regression problem directly from the raw image to bounding box coordinates in a single global step, hence the name 'single shot'. Figure 25 shows the SSD architecture

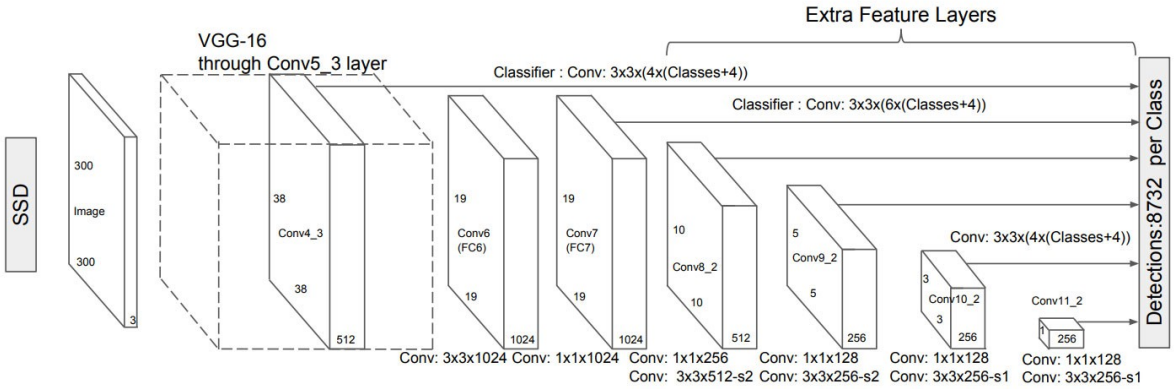


Figure 25: Architecture of SSD

SSD network is composed of six major parts that play an important role in detection objects; they are:

- Feature extractor
- Convolutional box predictor
- Anchors generation
- Matching priors to actual bounding box coordinates
- Hard negative mining
- Non-maximum suppression as a post-processing step.

Feature extractor

SSD is based on VGG16 which works as a feature extractor.

Convolutional box predictor

For object detection, last fully connected layers of VGG16 are discarded, and a set of auxiliary convolution layers are added which are called as convolutional box predictor. These layers are responsible in producing multiple feature maps of shape 19×19, 10×10, 5×5, 3×3 and 1×1 in successive order stacked after the feature map coming from last convolutional layer from VGG16 of size 38×38. These six auxiliary layers extract features from the image at multiple scales and progressively decrease the spatial resolution. Due to performing convolution operation at multiple scales, objects of various sizes can be detected. For example, feature maps of 10×10 perform better in detecting smaller objects while feature maps of size 3×3 perform better in detecting large objects. Each of these layers is connected to two heads, one that performs regression to find the bounding boxes and the other one to a classifier to classify each of the detected boxes.

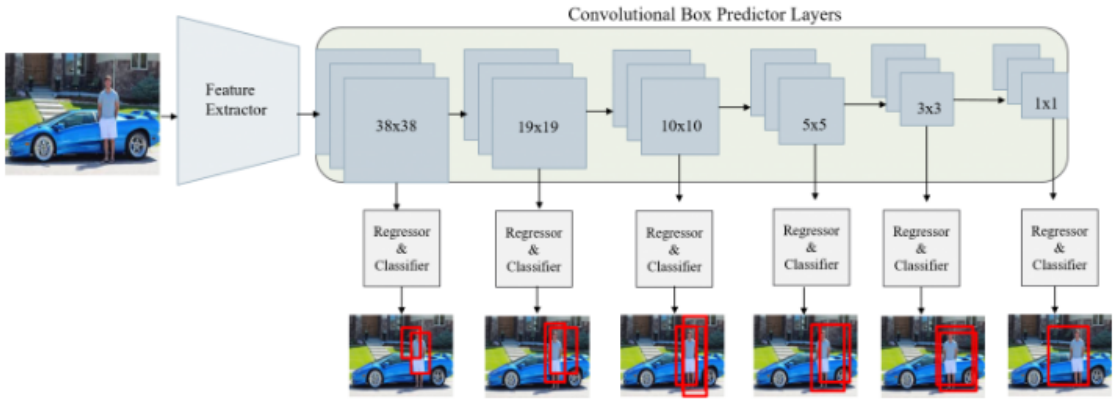


Figure 26: Auxiliary box predictor layers [Reza 19]

Anchors generation

Anchors generation is one of the most critical steps that can significantly affect the overall performance of the model. Anchors are a set of pre-defined boxes overlayed on the input as well as the generated feature maps at different spatial locations, scales and aspect ratios that act as reference points for the model to predict ground truth boxes. Anchors are significant because they provide a strong starting point for the regressor to predict ground truth boxes rather than starting to predict with random co-ordinates. Each feature map is divided into several grids, and each grid is associated with a set of default anchor bounding boxes of different dimensions and aspect ratios. Each of the priors is responsible for predicting exactly one bounding box.

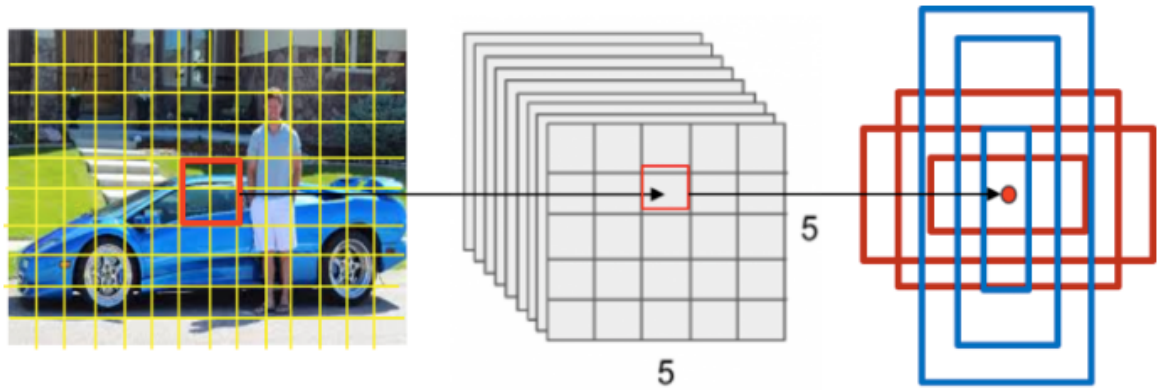


Figure 27: Anchors overlayed on 5×5 map [Reza 19]

Figure 27 shows each grid of a 5×5 feature map having 6 anchors which mostly matches the objects that need to be detected. For example, the red coloured anchors are most likely to detect the car, whereas blur coloured anchors are likely to detect the person and car of both smaller and larger sizes.

Matching priors to actual bounding box co-ordinates

Each ground truth bounding box coordinates is matched to anchors having the largest Intersection over Union (IoU) or Jaccard index. Also, unassigned anchors having IoU

greater than a threshold (usually 0.5) are also matched to the ground truth boxes. All matched anchors are called positive samples, and the rest are considered as negative samples.

Hard negative mining

Among a large number of anchors, only a few will be considered as positive samples, and the rest of the anchors will be considered as negative samples. Hence, from a training point of view, this imbalance between the positive and negative samples will make the training biased towards negatives. To avoid such a situation, some of the negative samples are randomly sampled from the negative pool such that the training set has a negative to positive ratio as 3:1. This process is known as hard negative mining. This reason to keep the negative samples is to let the network know about the features that lead to incorrect detection.

Non-maximum suppression

As explained in section 2.7, non-maximum suppression is used to attain top predictions for each class. This makes certain that only the most probable predictions are kept by the model, while the noisy ones are discarded.

As SSD is a single-shot object detector, all of these components are trained simultaneously.

4.5.1 Calculating loss

The overall loss function is a weighted sum of localization loss (loc) and the confidence loss (conf) where localization loss is the mismatch between ground truth box and predicted bounding box. Confidence loss is the loss in assigning class labels to the predicted bounding boxes. The equation of the loss function is given as:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{11}$$

Here x = 1 if the anchor matches the determined ground truth box, and 0 otherwise. N is the number of matched anchors, l is predicted bounding box, g is ground truth box, c is class, L_{conf} and L_{loc} are confidence and localization loss. α is the weight for localization loss. Smooth-L1 loss [Girshick 15b] is used for localization on l and g, softmax loss is used for optimization of confidence loss over multiple class confidence c

4.5.2 L2 regularization

This is a method used to reduce model complexity and overfitting on training data. During the phase of training, a handful of weights can get too large and thus influence the prediction heavily by suppressing the rest of the weights. This makes the model lose the power to generalize and in turn, overfit on training data. Hence, this method penalizes such large weights and makes sure the model learns smaller and consistent weights, through-out all the layers.

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda \sum_i \sum_j W_{i,j}^2 \tag{12}$$

Here, N is the number of classes in the dataset, L_i is a loss for particular class *i*. $W^2_{i,j}$ is the regularization function that calculates the sum of squares of each weight from the weight matrix W and λ is the regularization parameter which specifies the strength of regularization.

4.5.3 Batch Normalization

Batch normalization [Ioffe 15] is also a regularization technique that accelerates the model training by allowing it to use higher learning-rates. During the training phase, a large dataset is passed as batches to the model. The distribution of the batches usually differ from each other, and this affects the behaviour of the model. This change is distribution is known as covariate shift. Due to covariate shift, layer activations change quickly to adapt to changing inputs. As activations of the previous layer are passed as output to the next layer, all layers of the model get affected inconsistently for each batch. This results in a long time for the model to converge. To avert this, activations of each layer are normalized to have zero mean and unit variance. This helps each layer to learn the more stable distribution of inputs in all batches, which speeds up the model training. However, forcing activations to be fixed can limit the representation of input data. Instead, batch normalization allows the model to learn parameters β and γ during the training phase to convert the mean and variance according to the input data.

$$y^(k) = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{13}$$

$\hat{x}^{(k)}$ is a single activation and $\gamma^{(k)}$, $\beta^{(k)}$ are parameters that scale and shift the normalized value during training.

4.6 Combining architectures

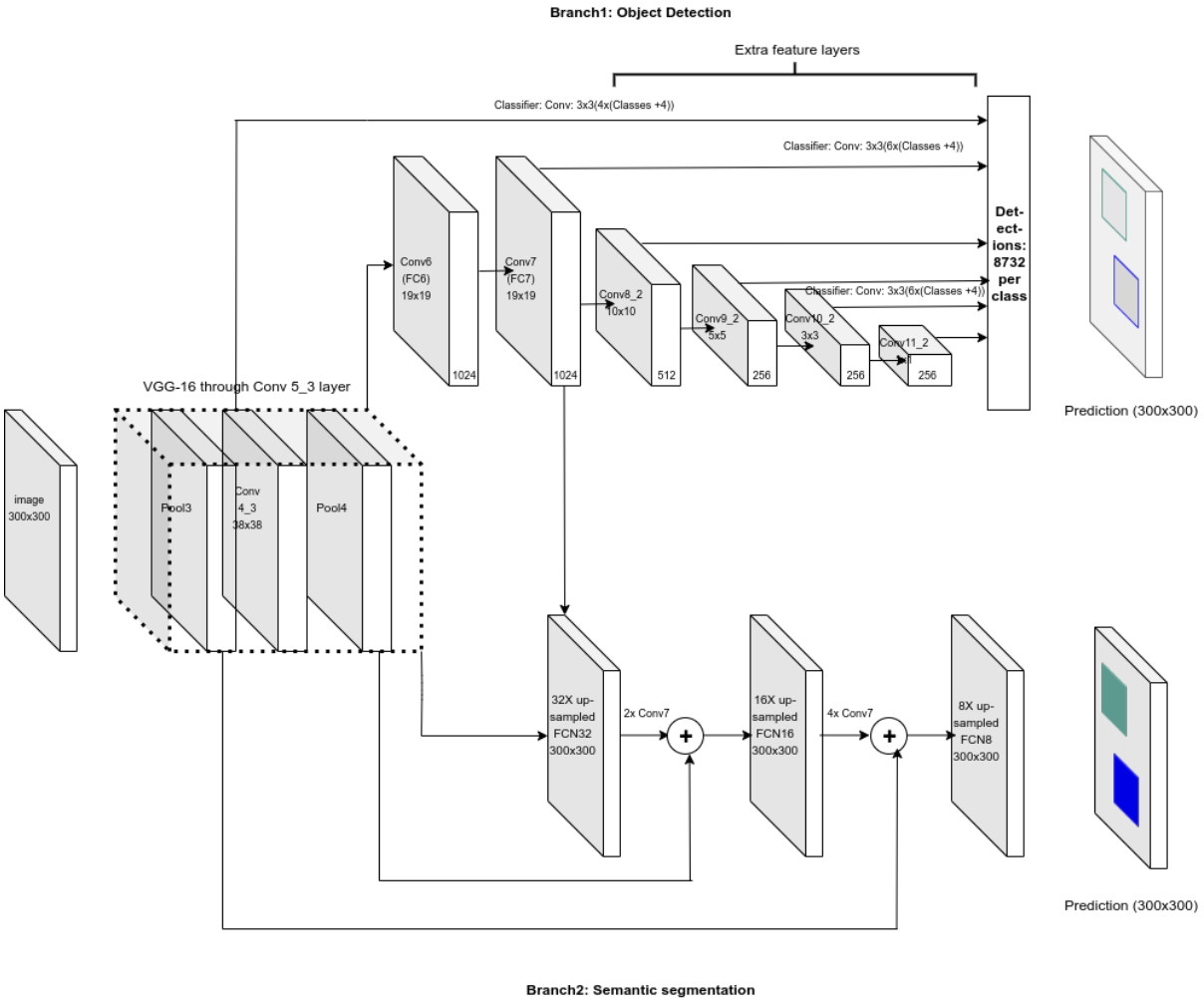


Figure 28: Combined architecture

4.6.1 Combined architecture

Sections 4.4 and 4.5 give an understanding on how semantic segmentation and object detection work separately. Figure 4.6 shows how both the architectures can be combined using a common feature extractor/ encoder. The combined architecture uses VGG16 as the common encoder and transfers the image features to both the branches of the architecture. 'Branch1' is the SSD network that performs object detection and 'Branch2' is the FCN network that performs semantic segmentation.

'Branch1' consists of six auxiliary convolutional layers responsible for producing feature maps of shape 19×19, 10×10, 5×5, 3×3 and 1×1 in succession stacked after the last feature map produced by VGG16 encoder. These layers further extract image features from the image at multiple scales such that objects of different sizes can be detected. For example, feature maps of 3×3 are useful to detect big sized objects. Each of these layers is connected to a regressor and a classifier. Regressor predicts the bounding boxes for objects present in the image. Anchors play an important role for the regressors to work. Anchors are a set of pre-defined boxes that are overlayed on the input and the feature maps at different locations, scales and aspect ratios. They provide a starting point for the regressor to predict ground-truth boxes. Once the bounding boxes are predicted, the classifier predicts the class of the object contained in the predicted bounding box. Lastly, non-maximum suppression is applied to keep top predictions and remove the noisy ones.

'Branch2' consists of transposed convolutional layers responsible for producing segmentation maps from the last feature map of VGG16 encoder. These transposed convolutional layers upsample the feature maps to the original size as the input. Upsampling can be done in 3 ways: There are 3 ways to upsample the feature map. One way is to upsample the feature map directly by applying the transposed convolution to produce the segmentation map. The other two techniques involve skipping connections to merge features from different resolution levels and applying transposed convolution to produce a segmentation map.

Combined loss

Once both the architectures are combined, the respective losses also need to be combined such that the network can learn both detection and segmentation parallelly. 'Branch1' consists of two losses, localization loss and confidence loss. Localization loss measures the mismatch between ground truth bounding box coordinates and predicted bounding box coordinates. Confidence loss measures the mismatch between ground truth class labels and predicted class labels inside the predicted bounding box. Similarly, 'Branch2' consists of pixel-wise cross-entropy loss. This loss calculates the difference between the predicted vector and one hot encoded class label. The combined loss for our architecture is given as:

$$L = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) + (-\Sigma_{classes}\ y_{true}\ log(y_{pred})) \tag{14}$$

The left part of the loss calculates the localization and confidence loss used for object detection while the right side calculates the pixel-wise loss, which is used for semantic segmentation.

N is the number of correctly matched anchors with the ground truth bounding box. Value of x=1 if the anchor matches with the ground truth bounding box and 0 if it doesn't

match. l is the predicted bounding box while g is the ground truth bounding box. y_{true} is the actual ground truth of semantic segmentation and y_{pred} is the prediction from the model.

5 Experiments and Results

In this chapter, experiments and results of standalone architectures and combined architectures are discussed. Section 5.1 is about experiments using PASCAL-VOC 2012 dataset followed by section 5.2 about experiments using KITTI dataset.

5.1 Evaluation on PASCAL-VOC 2012

Section 4.6.1 shows how combining the architectures (*Branch1* and *Branch2*) is possible using a common encoder. Before evaluating the combined architecture, it is necessary to check the performance evaluation of each branch. This technique is useful to know the flaw or benefit of using the combined architecture in comparison to using stand-alone architectures.

Hyperparameters

Hyperparameters are kept constant throughout all experiments to get an ideal comparison.

Hyperparameter	Value
Learning rate	0.0001; 0.001; 0.0001
Intervals	18,300; 36,600
Batch size	8
Weight decay	0.000001
Epochs	200

Branch1 - Standalone - Object detection

Branch1 - standalone only involves training of *Branch1* while the *Branch2* is freezed. During this phase, only the localization and confidence loss is calculated. Images are resized to 300×300, and random augmentation is applied on the fly.

This setup gives an mAP score of **46.9%**

Figure 29: Loss vs epochs - Branch1

Figure 29 shows the loss vs epochs curve for standalone object detection. It is seen that after the 120^{th} epoch, the loss increases neither decrease, which means the network has saturated.

Some sample predictions are shown in table 3

Table 3: Predicted images from Branch1 - standalone - object detection

Branch2 - Standalone - semantic segmentation

Branch2 - standalone only involves the training of *Branch2* keeping the *Branch1* freezed. During this phase only the pixel-wise cross entropy is calculated. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Images are resized to 300×300 and random augmentation is applied on the fly.

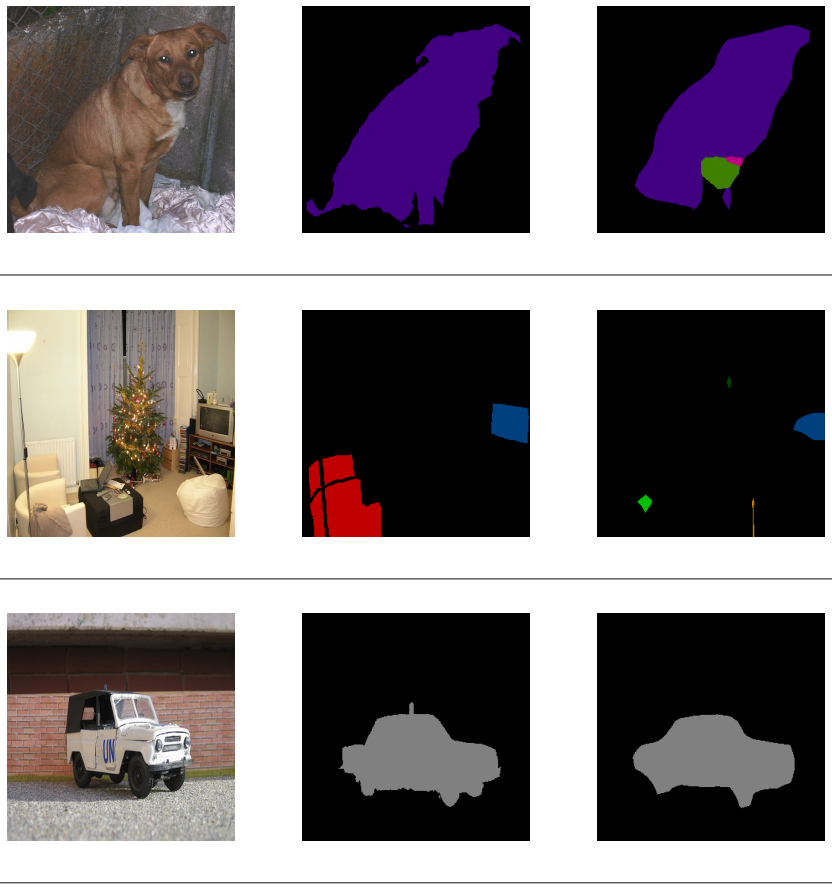
This setup gives a mean IoU score of **62.01%** and mean pixel accuracy score of **85.4%**.

Figure 30: Loss vs epochs - Branch2

Figure 30 shows the loss vs epochs curve for semantic segmentation. The loss value at the end of 200^{th} epoch is **1.04**. Also, it is seen that after 65^{th} epoch we reach the minima of function where the value of loss is **0.54**.

Some sample predictions are shown in table 4

Table 4: Predicted images from Branch2- standalone - semantic segmentation (Extreme left - RGB image, Bottom - ground truth, Extreme right - prediction)



Combining Branch1 and Branch2 (FCN32)

This phase involves the training of both the branches such that one branch gives the object detection result and the other branch give segmentation map. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Localization, confidence and pixel-wise cross-entropy are calculated in this phase. Images are resized to 300×300, and random augmentation is applied on the fly.

Training both the branches parallelly gives mean AP score of **46.6%** for object detection and mean IoU score of **56.85%**, mean pixel accuracy score of **83.3%** for semantic segmentation.

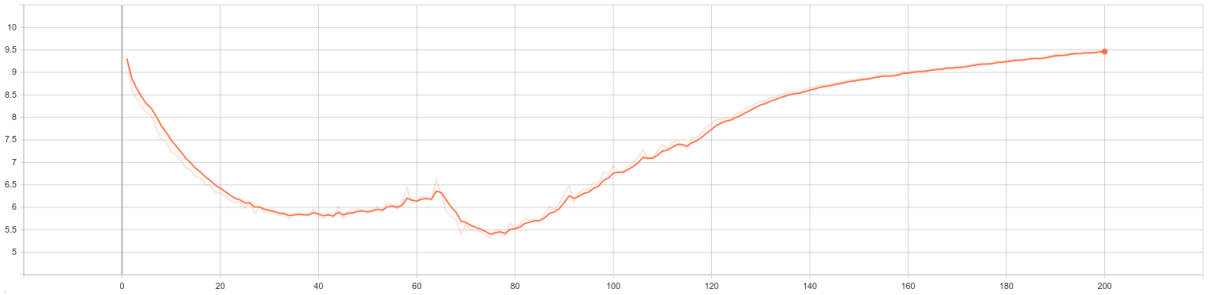


Figure 31: Loss vs epochs

Figure 31 shows the loss vs epochs curve for the combined training. As per the curve, the loss is increasing gradually at a high rate after 80^{th} epoch.

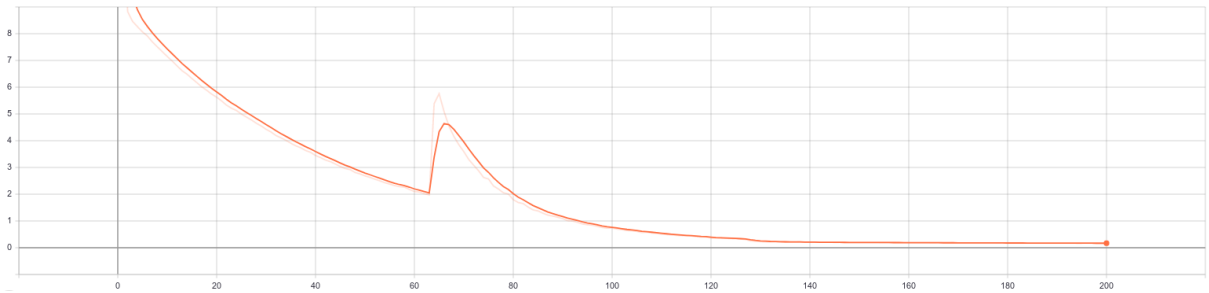


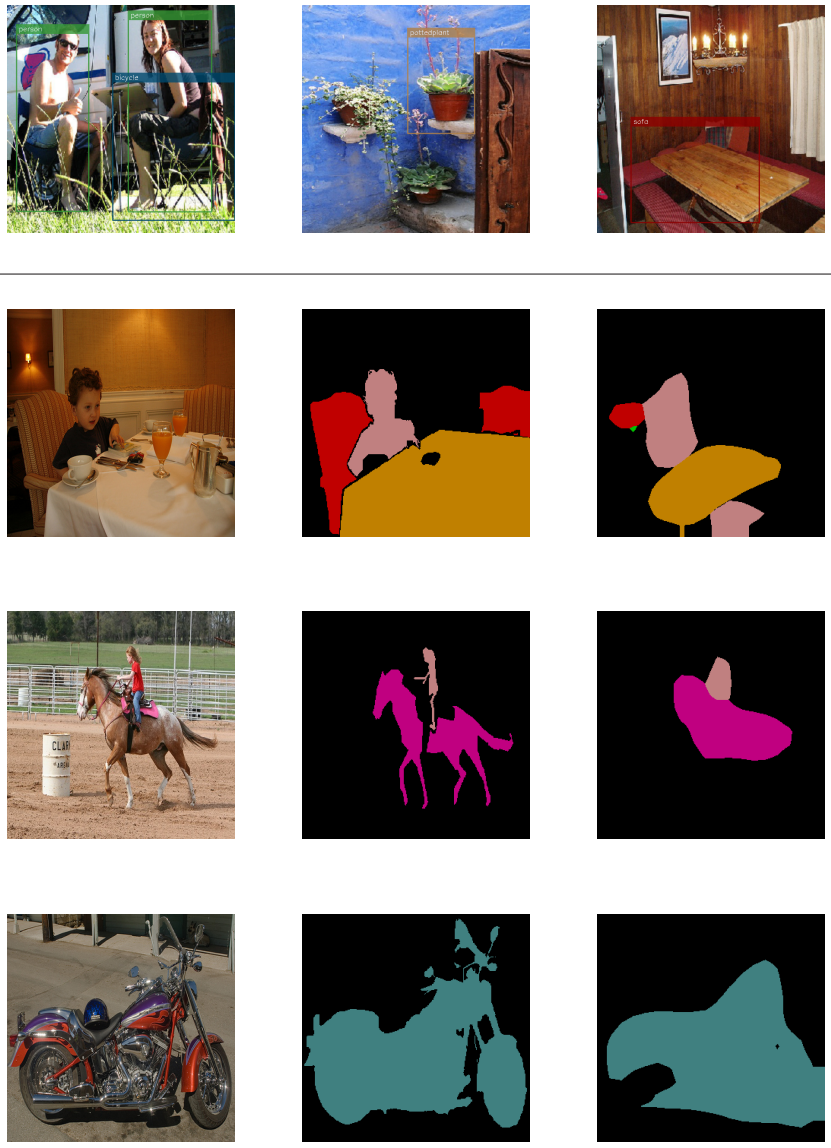
Figure 32: Loss vs epochs (Training)

Figure 32 shows the training loss vs epochs curve. By comparing both the above graphs, we can deduce that the network has overfitted on training data.

Hence, scores at the 80^{th} are reported as network starts to overfit after that. The mean AP score for object detection is **35.65%** while the mean IoU score and mean pixel-accuracy score are **46.92%** and **80.1%** for semantic segmentation.

Some sample predictions are shown in table 5 5

Table 5: Predicted images from combined architecture - object detection + semantic segmentation



Combining Branch1 and Branch2 using Lambda in loss function - (FCN32)

Section 4.6.1 shows how the combined loss is calculated for semantic segmentation and object detection. This experiment shows how using λ in the combined loss function will work with the model.

$$Combinedloss = \lambda(lossofBranch1) + \lambda(lossofBranch2) \tag{15}$$

The combined architecture can be seen as a multi-task setting where object detection and semantic segmentation are being solved with the help of common parameters and very few task-specific parameters. This idea of using a combined loss function then can be seen as a generalization strategy, where one of the above-mentioned tasks put pressure on the

common parameters such that the other task can be generalized better. Of course, in an ideal scenario where the two tasks are highly related, adding the loss functions with $\lambda = 1$ would suffice. However, by adding a small value of λ for one of the tasks will add as a regularizer (generalizes the parameters by putting some constraints) for the other task.

The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Images are resized to 300×300, and random augmentation is applied on the fly.

- Case1: $\lambda_1 = 1$ and $\lambda_2 = 0.01$

$$L = \lambda_1(-\Sigma_{classes}\ y_{true}\ log(y_{pred})) + \lambda_2(\frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))) \tag{16}$$

As per the loss function above, the object detection part will act as a regularizer for the semantic segmentation part.

This setup of λ value gives a mean IoU score of **62.52%** and mean pixel-accuracy score of **86.3%** for semantic segmentation. The mAP score for object detection drops down to only **3%**.

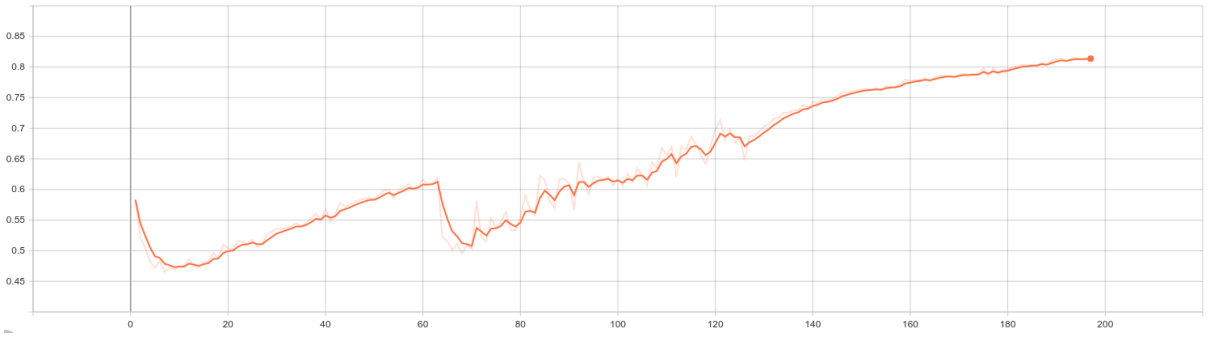


Figure 33: Semantic segmentation loss vs epochs

Figure 33 shows the loss vs epochs curve. It is seen that loss is increasing after the 70^{th} epoch.

Figure 34 shows the training loss vs epochs curve. By comparing both the above graphs, we can deduce that the network has overfitted on training data.

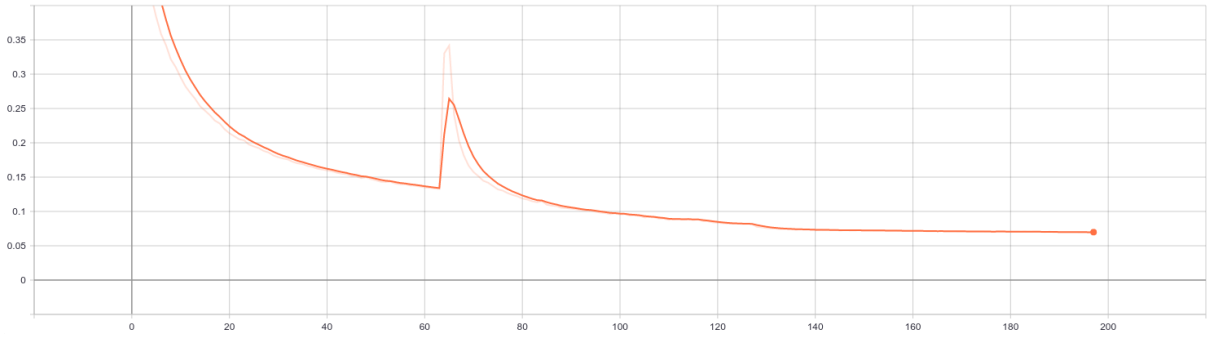


Figure 34: Semantic segmentation loss vs epochs (Training)

Hence, scores at 70^{th} epoch are reported as network starts to overfit after that. The mean AP score for object detection is **0.37%** while the mean IoU score and mean pixel-accuracy for semantic segmentation are **48.65%** and **79.82%**.

- Case2: $\lambda_1 = 0.01$ and $\lambda_2 = 1$

$$L = \lambda_1(-\Sigma_{classes}\ y_{true}\ log(y_{pred})) + \lambda_2(\frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))) \tag{17}$$

As per the loss function above, the semantic segmentation part will act as a regularizer for the object detection part. This setup of λ value gives a mean AP score of **45.02%** while mean IoU and mean pixel-accuracy scores for semantic segmentation are **32.48%** and **73.7%**.

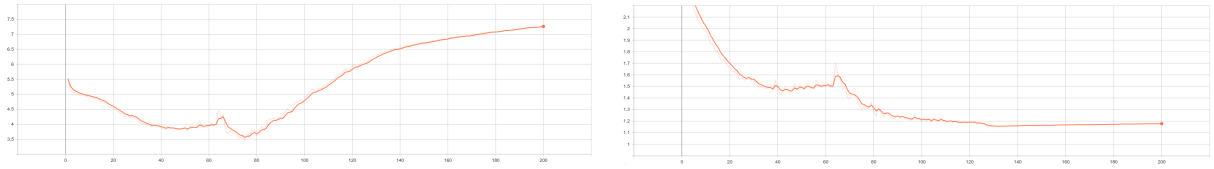


Figure 35: Confidence and localization loss vs epochs

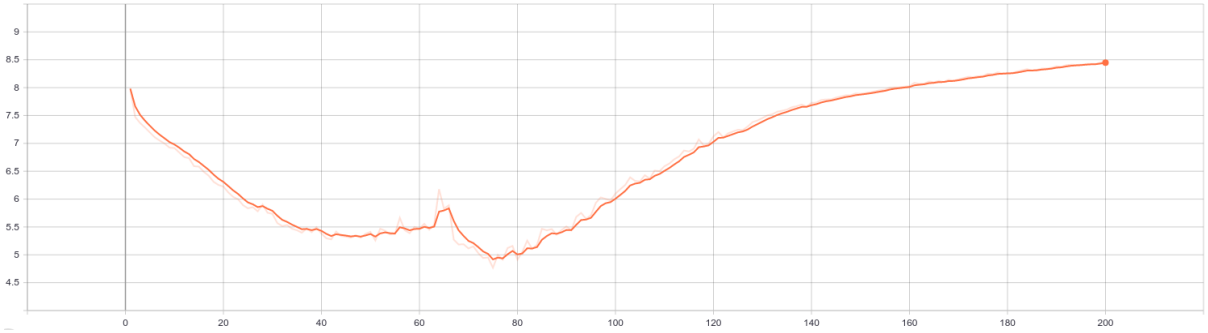


Figure 36: Total loss vs epochs

Figures 35 and 36 are confidence, localization and total loss curves. It shows that the total loss gradually increases at a high rate after the 76^{th} epoch. The reasons for increase in total loss can be because of the confidence loss increasing and also might be because the network is overfitting the training data.

Hence, scores from the 77^{th} epoch are reported. The mean AP score for object detection is **36.46%** while the mean IoU and mean pixel-accuracy scores are **32.48%** and **73.7%**.

Combining Branch1 and Branch2 - (FCN32)

This experiment is carried out by resizing the images to 512×512 and passing it to the network. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Augmentation of images are done on the fly.

Training both the branches gives mean AP score of **47.66%** for object detection while mean IoU and mean pixel-accuracy scores of **61.5%** and **47.6%** for semantic segmentation.

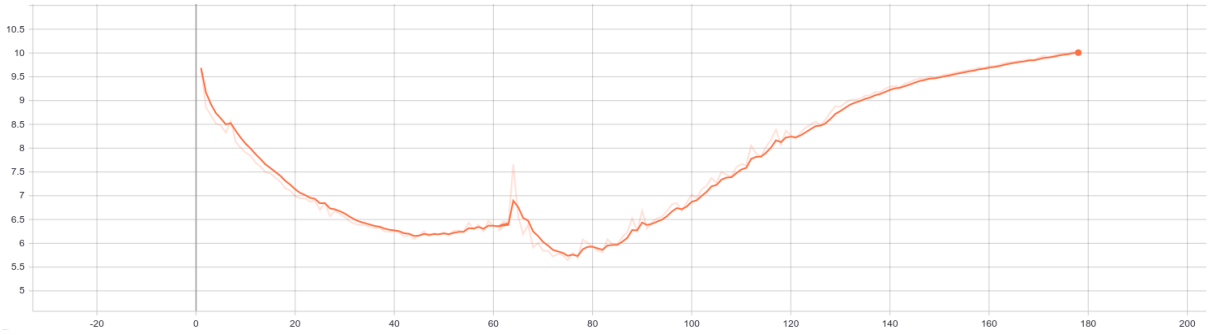


Figure 37: Total loss vs epochs

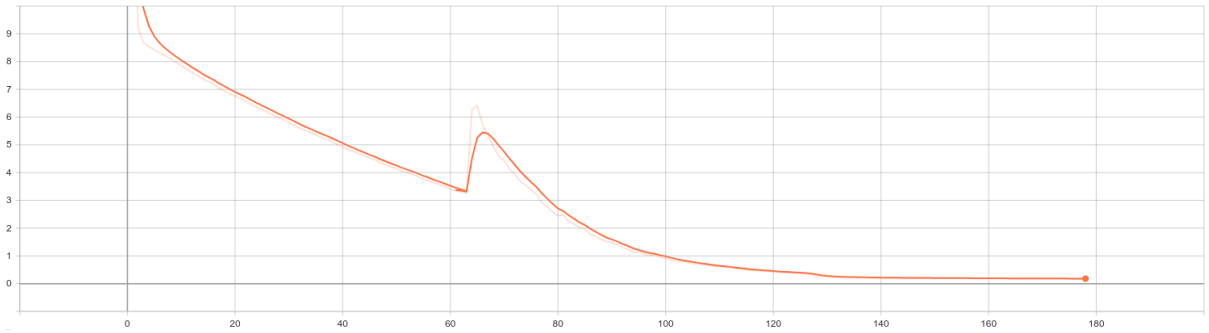


Figure 38: Total loss vs epochs (Training)

Figure 37 shows the loss vs epochs curve. It is seen that loss is increasing after the 77^{th} epoch.

Figure 38 shows the training loss vs epoch. By comparing both the above graphs, we can deduce that the network has overfitted on training data.

Hence, scores at 77^{th} epoch are reported. The mean AP is **38.39%** for object detection while mean IoU score and mean pixel accuracy scores are **49.96%** and **80.3%**.

Combining Branch1 and Branch2 - (FCN8)

This experiment is carried out by resizing the images to 300×300 and passing it to the network. Augmentation of images are done on the fly. The predicted segmented map is achieved by upsampling and using skip connections as mentioned in section 4.4

Training both the branches parallelly gives mean AP score of **0.6%**for object detection while mean IoU and mean pixel-accuracy of **32.48%** and **73.7%**for semantic segmentation.

Summary of results for PASCAL-VOC

Standalone			
	mAP	**mIoU**	**mPA**
Branch1 (object detection) (300 x 300)	46.9%	-	-
Branch2 (semantic segmentation) (300 x 300)	-	62.01%	85.4%
Combined			
	mAP	**mIoU**	**mPA**
Combined (FCN32) (300 x 300)	35.65%	46.92%	80.1%
Combined [((λ=1)segmentation loss)+ ((λ=0.01)detection loss)] - (FCN32)) (300 x 300)	0.37%	48.65%	79.82%
Combined [((λ=0.01)segmentation loss)+ ((λ=1)detection loss (FCN32))] (300 x 300)	36.46%	32.48%	73.7%
Combined (FCN8) (300 x 300)	0.6%	32.48%	73.7%
Combined (FCN32) (512 x 512)	**38.39%**	**49.96%**	**80.3%**

5.2 Evaluation on KITTI

Hyperparameters

Hyperparameter	Value
Learning rate	0.0001
Batch size	4
Weight decay	0.000001
Epochs	50

Branch1 - Standalone - Object detection

Branch1 - standalone only involves training of *Branch1* while the *Branch2* is freezed. Only the localization and confidence loss is calculated in this phase. Images are resized to 300×300 and random augmentation is applied on the fly.

This setup gives an mAP score of **35.95%**

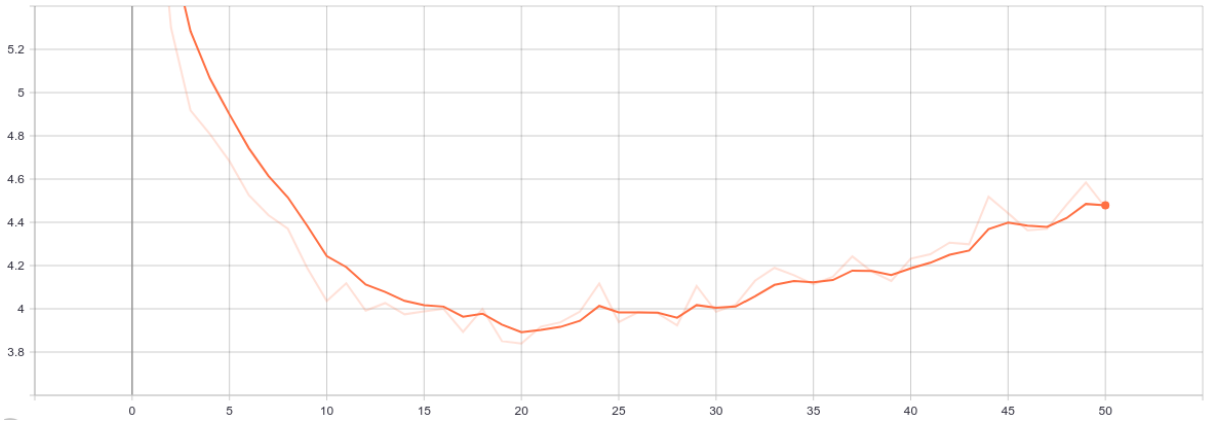


Figure 39: Loss vs epoch

Figure 39 shows the loss vs epochs curve.

Some sample predictions are shown in table 6

Table 6: Predicted images from Branch1 - standalone - object detection

Branch2 - Standalone - Semantic segmentation

Branch2 - standalone only involves the training of *Branch2* keeping the *Branch1* freezed. During this phase only the pixel-wise cross entropy is calculated. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Images are resized to 300×300 and random augmentation is applied on the fly.

This setup gives a mean IoU score of **60.83%** and mean pixel accuracy score of **95.6%**.

Figure 40 shows the loss vs epochs curve.

Some sample predictions are shown in table 7.

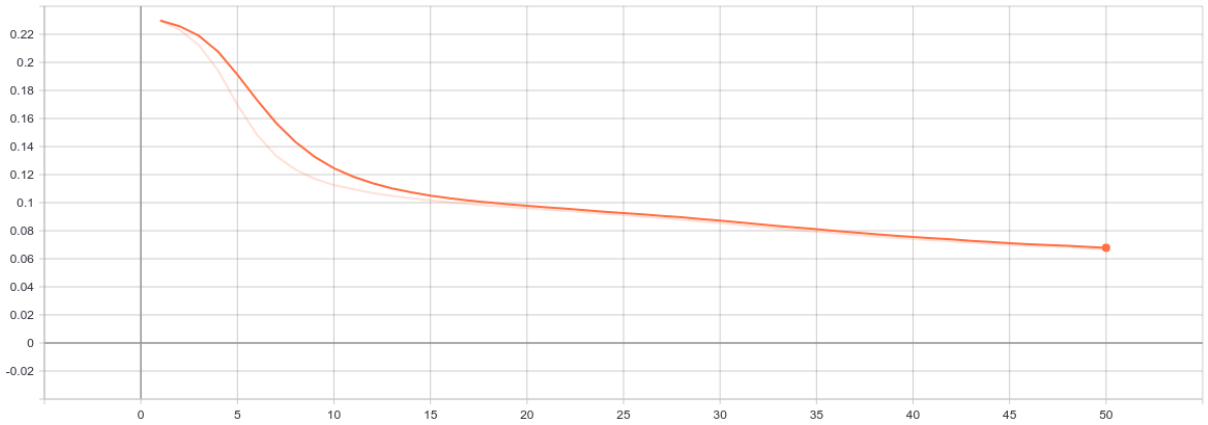
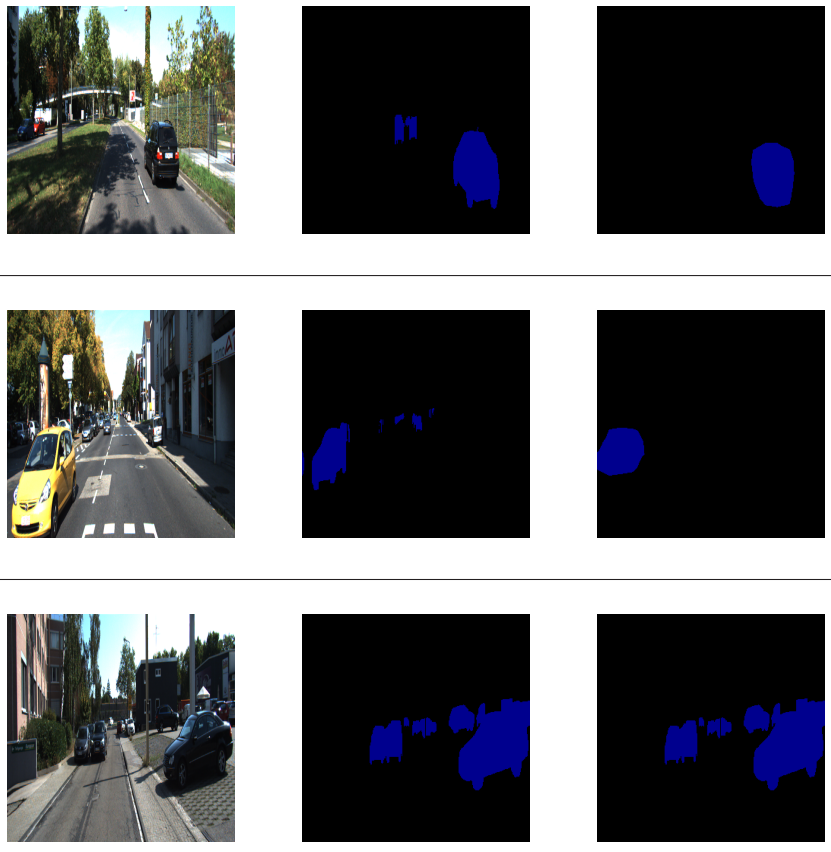


Figure 40: Loss vs epoch

Table 7: Predicted images from Branch2- standalone - semantic segmentation (Extreme left - RGB image, Bottom - ground truth, Extreme right - prediction)



Combining Branch1 and Branch2

This phase involves the training of both the branches such that one branch gives the object detection result and the other branch give segmentation map. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Localization, confidence and pixel-wise cross-entropy are calculated in this phase. Images are resized to 300×300, and random augmentation is applied on the fly.

Training both the branches parallelly gives mean AP score of **35.8%**for object detection and mean IoU score of **60.9%**, mean pixel accuracy score of **95.8%** for semantic segmentation.

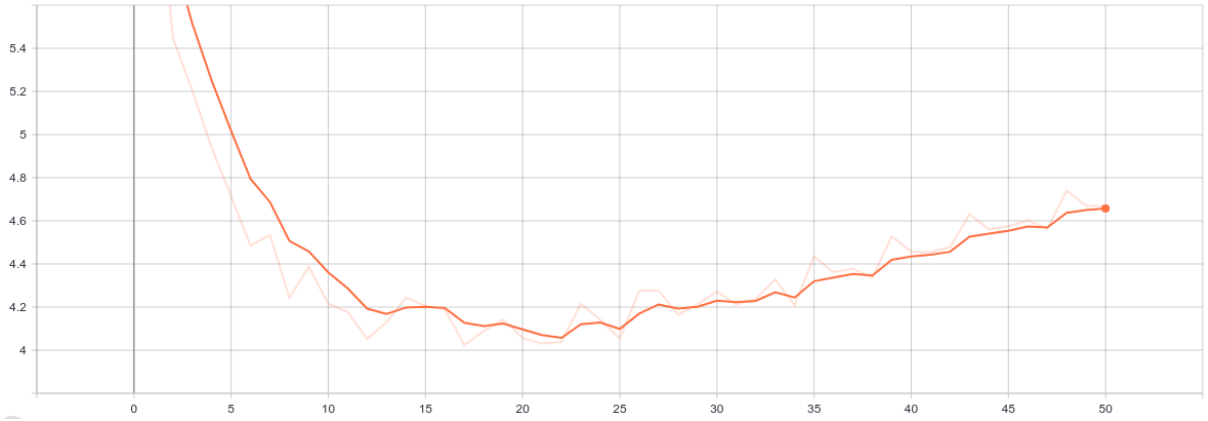


Figure 41: Loss vs epoch

Figure 41 shows the loss vs epochs curve.

Some sample predictions are shown in table 8

Combining Branch1 and Branch2 using Lambda in loss function - (FCN32)

The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Images are resized to 300×300, and random augmentation is applied on the fly.

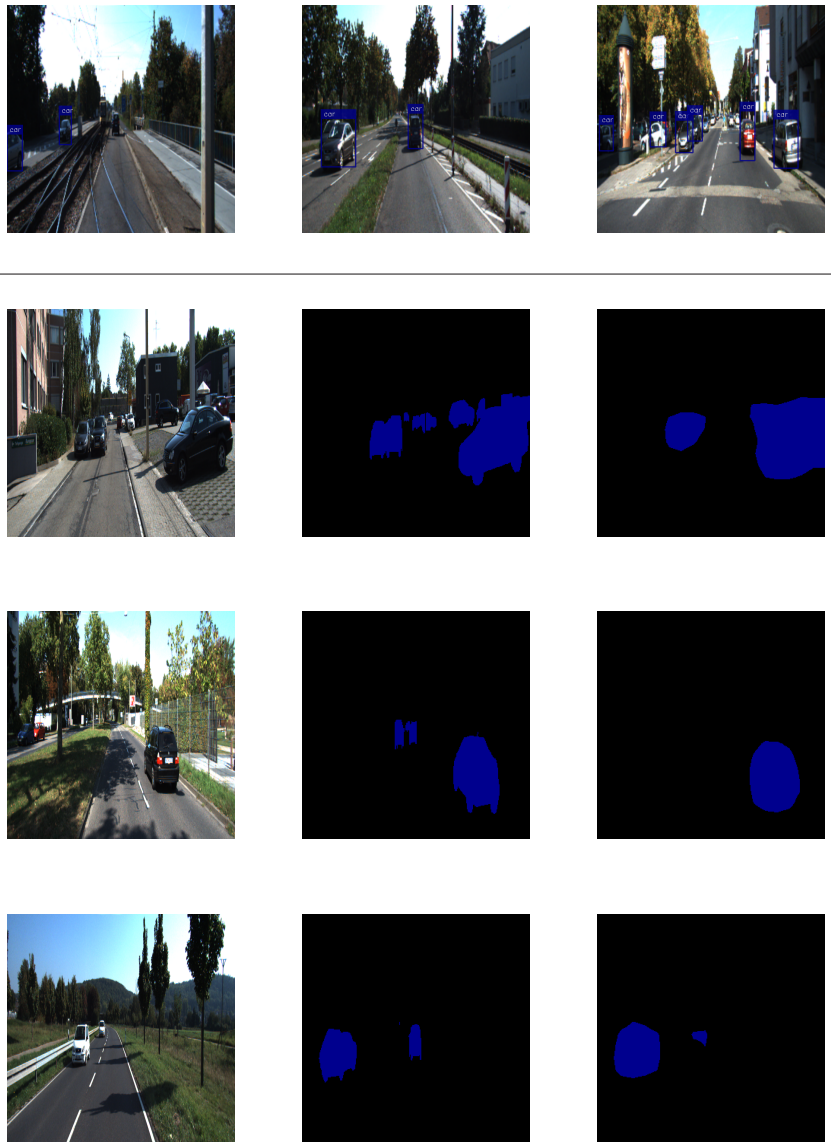
- Case1: $\lambda_1 = 1$ and $\lambda_2 = 0.01$

$$L = \lambda_1(-\Sigma_{classes}\ y_{true}\ log(y_{pred})) + \lambda_2(\frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))) \tag{18}$$

  Here, the object detection part will act as a regularizer for the semantic segmentation part. This setup of λ value gives a mean IoU score of **61.56%** and mean pixel-accuracy score of **95.7%** for semantic segmentation. The mAP score for object detection drops down to only **0.6%**.

  Figure 42 shows the loss vs epoch curve

Table 8: Predicted images from combined architecture - object detection + semantic segmentation



- Case2: $\lambda_1 = 0.01$ and $\lambda_2 = 1$

$$L = \lambda_1(-\Sigma_{classes}\ y_{true}\ log(y_{pred})) + \lambda_2(\frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))) \tag{19}$$

As per the loss function above, the semantic segmentation part will act as a regularizer for the object detection part.

Figures 43 and 44 are confidence, localization and total loss curves.

The mean AP score for object detection is **35.21%** while the mean IoU and mean pixel-accuracy scores are **46.91%** and **93.8%**.

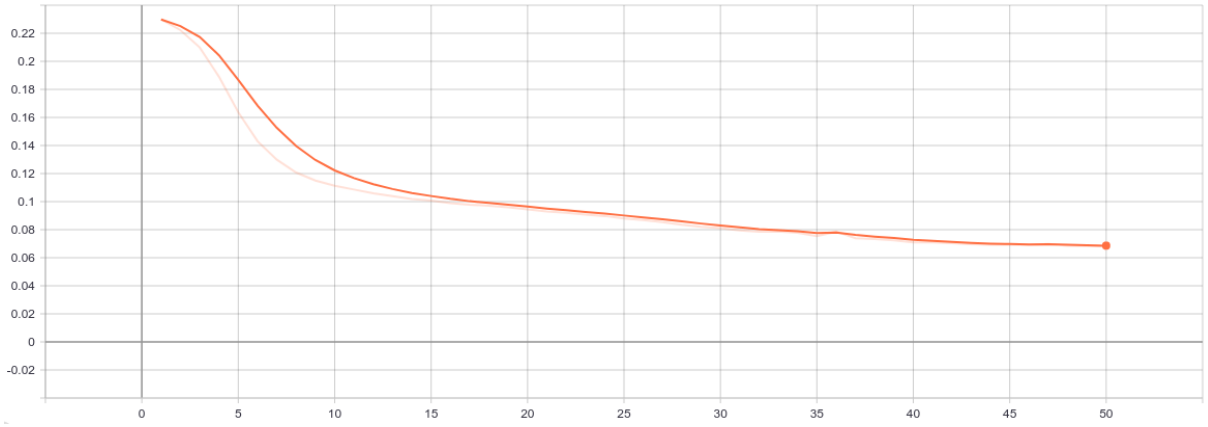


Figure 42: Segmentation loss vs epochs

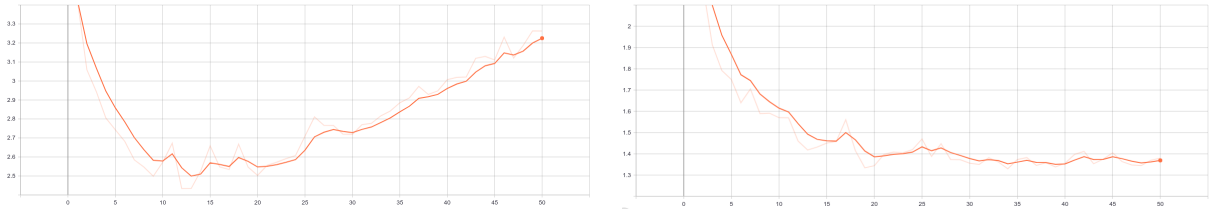


Figure 43: Confidence and localization loss vs epochs

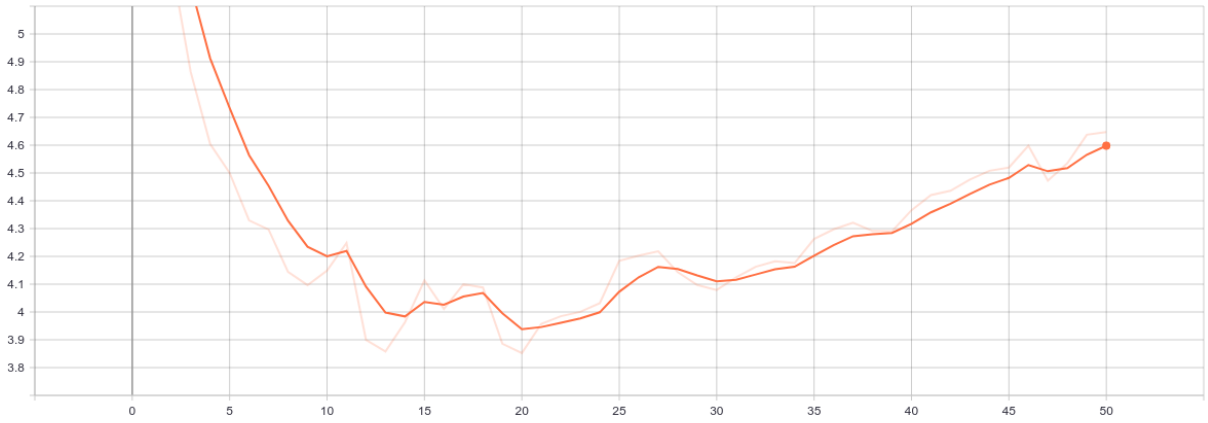


Figure 44: Total loss vs epochs

Combining Branch1 and Branch2 - (FCN32)

This experiment is carried out by resizing the images to 512×512 and passing it to the network. Augmentation of images are done on the fly. The predicted segmentation map is achieved by upsampling the last feature map of the VGG16 encoder by 32 times (FCN32). Augmentation of images are done on the fly.

Training both the branches gives mean AP score of **42.4%** for object detection while mean IoU and mean pixel-accuracy scores are **62.60%** and **95.7%** for semantic segmentation.

Figure 45 shows the loss vs epoch curve.

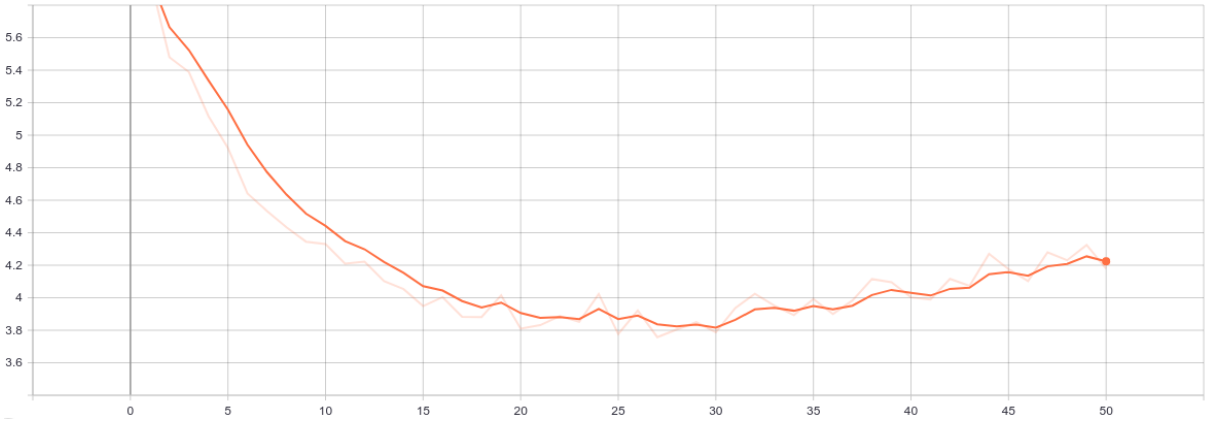


Figure 45: Total loss vs epochs

Combining Branch1 and Branch2 - (FCN8)

This experiment is carried out by resizing the images to 300×300 and passing it to the network. Augmentation of images are done on the fly. The predicted segmented map is achieved by upsampling and using skip connections as mentioned in section 4.4

Training both the branches parallelly gives mean AP score of **0.6%**for object detection while mean IoU and mean pixel-accuracy of **50.6%** and **93.2%**for semantic segmentation.

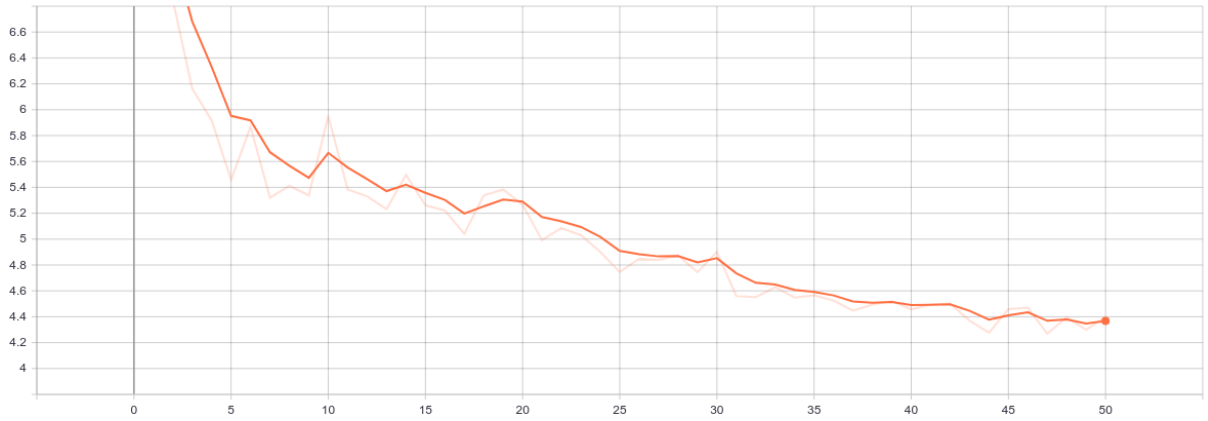


Figure 46: Total loss vs epochs

Figure 46 shows the loss vs epoch curve.

Summary of results for KITTI

Standalone			
	mAP	**mIoU**	**mPA**
Branch1 (object detection) (300 x 300)	35.95%	-	-
Branch2 (semantic segmentation) (300 x 300)	-	60.83%	95.6%
Combined			
	mAP	**mIoU**	**mPA**
Combined (FCN32) (300 x 300)	35.8%	60.9%	95.8%
Combined ((λ=1)segmentation loss)+ (λ=0.01)detection loss (FCN32) (300 x 300)	0.6%	61.56%	95.7%
Combined ((λ=0.01)segmentation loss)+ (λ=1)detection loss (FCN32) (300 x 300)	35.21%	46.91%	93.8%
Combined (FCN8) (300 x 300)	0.6%	50.6%	93.2%
Combined(FCN32) (512x512)	**42.4%**	**62.60%**	**95.7%**

6 Discussion and summary

We propose an approach to join object detection and semantic segmentation architecture using a common encoder in this work. [Liu 16] is used for object detection in our work. It is a one-stage detector that skips the region proposal step and runs detection directly. It consists of six auxiliary layers of different scales that help it detecting objects of different size. All of these layers are connected to two different heads. One of the head does the classification part, and the other head does the regression part. Also, anchors are overlayed at every grid of the feature map or the image. These anchors provide a starting point for the regressor to predict bounding boxes.

For semantic segmentation we use [Long 15]. This architecture is based on fully convolutional networks that consist of only convolutional layer and max-pooling layers. It is based on an encoder-decoder architecture. The encoder does the feature extraction part of the images by reducing the spatial resolution of the image, and the decoder does the upsampling of the feature map to regain the same size as the image.

Both the above-mentioned architectures use an encoder to extract image features, and these features are used for object detection as well as semantic segmentation. Using this encoder, we combined both the architectures such that the last feature map is shared between both the architectures. A pre-trained VGG16 encoder is used to extract features for the combined architecture.

In order to train the combined architecture, the losses of object detection and semantic segmentation have to be combined. Hence, localization and confidence loss of object detection is combined with the pixel-wise cross-entropy loss of semantic segmentation.

Several experiments are done to understand the flaws or benefits of using a combined architecture as compared to stand-alone architecture.

References

[AnnieWAY 06] AnnieWAY, "Team AnnieWAY, KIT, Germany", `https://www.mrt.kit.edu/annieway/team.html`, 2006.

[Badrinarayanan 15] V. Badrinarayanan, A. Kendall, R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", *CoRR*, vol. abs/1511.00561, 2015.

[Dalal 05] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection", in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, IEEE. 2005, pp. 886–893.

[Deshpande 17] M. Deshpande, "Perceptrons: The First Neural Networks", 2017.

[Dollár 09] P. Dollár, Z. Tu, P. Perona, S. Belongie, "Integral channel features", 2009.

[Elgendy 20] M. Elgendy, *Deep Learning for Vision Systems*, Manning Publications, 2020.

[Everingham] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results", http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[Fei-Fei 13] L. Fei-Fei, O. Russakovsky, "Analysis of Large-Scale Visual Recognition", 2013.

[Fei-Fei 17] L. Fei-Fei, "Detection and segmentation", `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf`, 2017.

[Geiger 12] A. Geiger, P. Lenz, R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[Girshick 14] R. Girshick, J. Donahue, T. Darrell, J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Girshick 15a] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[Girshick 15b] R. Girshick, "Fast R-CNN", in *The IEEE International Conference on Computer Vision (ICCV)*. December 2015.

[Gómez 07] O. Gómez, J. A. González, E. F. Morales, "Image Segmentation Using Automatic Seeded Region Growing and Instance-Based Learning", in *Progress in Pattern Recognition, Image Analysis and Applications*, L. Rueda, D. Mery, J. Kittler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 192–201.

[Gupta 14] S. Gupta, R. B. Girshick, P. Arbelaez, J. Malik, "Learning Rich Features from RGB-D Images for Object Detection and Segmentation", *CoRR*, vol. abs/1407.5736, 2014.

[Huang 10] Y.-R. Huang, C.-M. Kuo, "Image segmentation using edge detection and region distribution", in *2010 3rd International Congress on Image and Signal Processing*, vol. 3, IEEE. 2010, pp. 1410–1414.

[Hubel 62] D. H. Hubel, T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex", *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[Ioffe 15] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167*, 2015.

[Jordan 18a] J. Jordan, "An overview of object detection: one-stage methods.", `https://www.jeremyjordan.me/object-detection-one-stage/`, 2018.

[Jordan 18b] J. Jordan, "An overview of semantic image segmentation.", `https://www.jeremyjordan.me/semantic-segmentation/`, 2018.

[Krizhevsky 12] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[LeCun 98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[Lin 17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, "Feature pyramid networks for object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[Liu 16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, "Ssd: Single shot multibox detector", in *European conference on computer vision*, Springer. 2016, pp. 21–37.

[Long 15] J. Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[McCulloch 43] W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[Mostajabi 14] M. Mostajabi, P. Yadollahpour, G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features", *CoRR*, vol. abs/1412.0774, 2014.

[Pavlovsky 17] V. Pavlovsky, "Introduction To Convolutional Neural Networks", `https://www.vaetas.cz/posts/intro-convolutional-neural-networks/`, 2017.

[Perrault 19] R. Perrault, Y. Shoham, E. Brynjolfsson, J. Clark, J. Etchemendy, B. Grosz, T. Lyons, J. Manyika, S. Mishra, J. C. Niebles, "The AI Index 2019 Annual Report", Stanford University, Tech. Rep., 2019.

[Redmon 15] J. Redmon, S. K. Divvala, R. B. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", *CoRR*, vol. abs/1506.02640, 2015.

[Ren 15] S. Ren, K. He, R. B. Girshick, J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *CoRR*, vol. abs/1506.01497, 2015.

[Reza 19] Z. N. Reza, "Object Detection with Single Shot Multibox Detector", `https://ai-diary-by-znreza.com/object-detection-with-single-shot-multibox-detector`, 2019.

[Ronneberger 15] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", *CoRR*, vol. abs/1505.04597, 2015.

[Rosenblatt 60] F. Rosenblatt, "Perceptron simulation experiments", *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960.

[Salscheider 19] N. O. Salscheider, "Simultaneous object detection and semantic segmentation", *arXiv preprint arXiv:1905.02285*, 2019.

[Sambasivarao 19] K. Sambasivarao, "Non-maximum Suppression (NMS)", `https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c`, 2019.

[Sharma 17] A. Sharma, "Understanding Activation Functions in Deep Learning", `https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/`, 2017.

[Simonyan 14] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[Suga 08] A. Suga, K. Fukuda, T. Takiguchi, Y. Ariki, "Object recognition and segmentation using SIFT and Graph Cuts", in *2008 19th International Conference on Pattern Recognition*, IEEE. 2008, pp. 1–4.

[Teichmann 16] M. Teichmann, M. Weber, J. M. Zöllner, R. Cipolla, R. Urtasun, "MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving", *CoRR*, vol. abs/1612.07695, 2016.

[Tzutalin 15] Tzutalin, "LabelImg", `https://github.com/tzutalin/labelImg`, 2015.

[Uijlings 13] J. R. Uijlings, K. E. Van De Sande, T. Gevers, A. W. Smeulders, "Selective search for object recognition", *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[Van de Sande 11] K. E. Van de Sande, J. R. Uijlings, T. Gevers, A. W. Smeulders, "Segmentation as selective search for object recognition", in *2011 International Conference on Computer Vision*, IEEE. 2011, pp. 1879–1886.

[Viola 01] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, IEEE. 2001, pp. I–I.

[Waymo 09] Waymo, "Waymo - Google self-driving car", 2009.

[World] World, Health, Organization.

[Zheng 18] X. Zheng, Q. Lei, R. Yao, Y. Gong, Q. Yin, “Image segmentation based on adaptive K-means algorithm”, *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, p.6̃8, 2018.