ArNet A Deep Learning Architecture for Pixel-wise Semantic Segmentation of Images

A Thesis Submitted
By
Arik Md. Isthiaque

ID: 1103096

Under the Supervision of

Arifa Islam Champa
Lecturer
Department of Computer Science and Engineering

Bangladesh Army International University of Science and Technology

Bachelor of Science in Computer Science and Engineering

Bangladesh Army International University of Science and Technology

Fall, 2019 Semester

ARNET A DEEP LEARNING ARCHITECTURE FOR PIXEL-WISE SEMANTIC SEGMENTATION OF IMAGES

A Thesis submitted to the Department of Computer Science and Engineering Faculty, Bangladesh Army International University of Science and Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

Arik Md. Isthiaque **ID: 1103096**

Department of
Computer Science and Engineering

Bangladesh Army International University of Science and Technology

Fall Semester 2019

July, 2020

Declaration

This is to certify that this thesis is my original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Arik Md. Isthiaque

Approval

The Thesis titled “**ArNet A Deep Learning Architecture for Pixel-wise Semantic Segmentation of Images**” has been submitted to the following respected members of the Board of Examiners in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering on June 2020 by the following student and has been accepted as satisfactory.

Arik Md. Isthiaque	**ID: 1103096**

Supervisor	External Supervisor
Arifa Islam Champa	Dr. Fakir Sharif Hossain
Lecturer	Head of the Department
Department of CSE	Department of EEE
Bangladesh Army International University of Science and Technology	Bangladesh Army International University of Science and Technology

Mohammad Asaduzzaman Khan

Head, Department of CSE

Bangladesh Army International University of Science and Technology

Acknowledgment

First and foremost, I am very grateful to Almighty ALLAH for without his graces and blessings this thesis would not have been possible. Immeasurable appreciation and deepest gratitude for the help and support are extended to following persons who are in one way and another have contributed to making this thesis possible.

Arifa Islam Champa, Lecturer, Department of Computer Science and Engineering, Bangladesh Army International University of Science & Technology, my honorable thesis supervisor. I am very grateful to her for her support and words of encouragement and also for giving a long period of time to make this thesis possible.

Mohammad Asaduzzaman Khan, Associate Professor, Head of the Department of Computer Science and Engineering, Bangladesh Army International University of Science & Technology, a special thanks to him for his kind advice.

Golam Muktader Nayeem, Associate Professor, Department of Computer Science and Engineering, Bangladesh Army International University of Science & Technology, my mentor and the person closest to my heart who guided me in my whole undergraduate life. I would like to thank him for his guidance, inspiration, and constructive suggestions which are very helpful in the preparation of this thesis.

I would also like to express my gratitude to all the teachers of the Department of Computer Science and Engineering, Bangladesh Army International University of Science & Technology for their valuable guidance in every step of mine during the four years of the learning stage.

Md Omar Hasan, Software Engineer, Ferntech Solutions, an expert in machine learning and computer vision and a friend of mine, who provided me the guidelines in the journey of machine learning. I would like to make an appreciation for him to make this thesis possible.

My acknowledgment would be incomplete without thanking the biggest source of my strength, my family. The blessings of my parents **Mrs. Shamsun Nahar Akonje** & **Mr. Abdul Awal Talukder** and the love and care of my younger brother **Afrid Md Isfar**. I thank them for putting up with me in difficult moments where I felt stumped and for goading me on to follow my dream. This thesis

would not have been possible without their unwavering and unselfish love and support is given to me at all times.

Finally, I would like to thank all my friends, senior, junior, and the staff of the Department of Computer Science and Engineering, Bangladesh Army International University of Science & Technology for their valuable suggestions and assistance that has helped in the successful completion of the thesis.

Arik Md. Isthiaque

TABLE OF CONTENTS

DECLARATION..I
APPROVAL II
ACKNOWLEDGMENT...III
ABSTRACT X
CHAPTER 1...1
INTRODUCTION..1
1.1. Introduction...1
1.2. Historical Background..3
1.2.1. Earlier Research..4
1.2.2. Recent Research...6
1.2.3. State of the art technology..8
1.3. Future Scope of This Study...9
1.3.1. Future Scopes 9
1.3.2. Recommendations..10
1.4. Limitation of the Study..11
1.5. Advantage over Traditional Method 11
1.6. Objective of this Work 12
1.6.1. Primary objectives 12
1.6.2. Secondary Objectives...12
1.7. Introduction to this Thesis 12
CHAPTER 2...14
SYSTEM OVERVIEW..14
2.1. Introduction...14
2.2. ArNet Architecture..14
2.3. Deep Learning...17
2.3.1. Convolutional Neural Network 17
2.3.2. Advantage of Convolutional Neural Network..19
2.3.3. Applications of Convolutional Neural Network...19
2.3.3.1. Semantic Segmentation...20
2.4. 2D Convolution..21
2.5. 2D Deconvolution 22
2.6. Batch Normalization..23
2.7. 2D MaxPooling 24
2.8. 2D UnPooling..25

2.9. Activation Function 25
2.9.1. ReLU (Rectified Linear Unit) Activation Function 25
2.9.2. SoftMax Activation 26
2.10. Summary 26
CHAPTER 3 27
IMPLEMENTATION 27
3.1. Introduction 27
3.2. Gathering the Dataset 27
3.2.1. 4.2 Preprocessing the Dataset 28
3.3. Creating the Architecture 31
3.3.1. The Encoder Layer 31
3.3.2. The Decoder Layer 33
3.4. Implementation Tools 35
3.4.1. Environments 35
3.4.2. Packages 36
3.4.3. Frameworks 36
3.5. Novelty of Our Work 37
3.6. Summary 38
CHAPTER 4 39
EXPERIMENTAL RESULTS 39
4.1. Introduction 39
4.2. Evaluation of the Architecture 39
4.3. Comparisons 42
4.4. Summary 42
CHAPTER 5 43
DISCUSSIONS AND CONCLUSIONS 43
5.1. Discussions 43
5.2. Suggestion for Future Work 43
5.3. Conclusions 44
REFERENCES 45

LIST OF FIGURES

FIGURE 1.1: A REAL-TIME SEGMENTED ROAD SCENE FOR AUTONOMOUS DRIVING [5]………..2

FIGURE 1.2: A CHEST X-RAY WITH THE HEART (RED), LUNGS (GREEN), AND CLAVICLES (BLUE) ARE SEGMENTED [6]. 3

FIGURE 1.3: VISUALIZATION OF HOW SEMANTIC SEGMENTATION WORKS [7]. 4

FIGURE 1.4: FULLY CONVOLUTIONAL NETWORK ARCHITECTURE [13].....................6

FIGURE 1.5: SEGNET ARCHITECTURE [18]. 8

FIGURE 1.6: A DENSENET ARCHITECTURE WITH 5-LAYER CONNECTION [19]...........9

FIGURE 2.1: OVERVIEW OF ARNET ARCHITECTURE. 14

FIGURE 2.2: THE ENCODER LAYER. 15

FIGURE 2.3: THE DECODER LAYER. 16

FIGURE 2.4: A CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE [21]............17

FIGURE 2.5: AN OVERVIEW OF SEMANTIC IMAGE SEGMENTATION [7]. 20

FIGURE 2.6: A 2D CONVOLUTION OPERATION [23]. 21

FIGURE 2.7: 2D MAXPOOLING [27]..24

FIGURE 3.1: OVERVIEW OF CAMVID DATASET IMAGES [18]. 27

FIGURE 3.4: A ZERO-PADDED 4 X 4 MATRIX BECOMES A 6 X 6 MATRIX [24]............31

FIGURE 4.1: PLOTTED CATEGORICAL ACCURACY PER EPOCH OF THE TRAINING AND VALIDATION DATA...39

FIGURE 4.2: PLOTTED LOSS PER EPOCH OF THE TRAINING AND VALIDATION DATA…… 40

FIGURE 4.3: SOME PREDICTED RESULTS OF ARNET ARCHITECTURE. 41

LIST OF TABLES

TABLE 4.1: THE CATEGORICAL ACCURACY AND LOSS ON VALIDATION AND TRAINING DATA OF THE ARNET ARCHITECTURE…………………………………….36

TABLE 4.2: VALIDATION ACCURACY COMPARISON BETWEEN SOME THE STATE-OF-THE-ART ARCHITECTURE AND ARNET………………………………………..38

ABSTRACT

Detecting objects from images and making decisions from those collected data is a major part of an intelligent machine. Nowadays many real-time machines like robots, autonomous vehicle works on the basis of detecting objects from visual scenarios. So, the technological advancement of computer vision is of utmost importance. The semantic segmentation of images is a part of computer vision to understand a visual scenario. Here we present ArNet a deep learning architecture for semantic image segmentation that is motivated by SegNet and Fully Convolutional DenseNet for semantic image segmentation for achieving state-of-the-art accuracy. Our main focus of ArNet is to reduce the network complexity of the convolutional neural network by reducing the number of layers without reducing the accuracy of the architecture. We have a global accuracy of 91.00% in the current “CamVid” data set we are working on.

Chapter 1
Introduction

1.1.Introduction

Computer vision is an interdisciplinary scientific field that deals with how computers can gain a high-level understanding of digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do. Computer vision tasks include methods for acquiring, processing, analyzing, and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information in the forms of decisions. Understanding in this context means the transformation of visual images into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory [1].

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multi-dimensional data from a 3D scanner, or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual surveying, 3D scene modeling, and image restoration [2].

In computer vision, image segmentation is the process of classifying a digital image into multiple classes or sets of pixels, also known as objects. The purpose of segmentation is to simplify the image representation to an image that is more understandable and easier to analyze and more recognizable for the machines. Image segmentation is mostly used to identify objects and their edges in images. In detail, image segmentation is the method of assigning a class label to every single pixel that is present in an image so that pixels that fall under the same class label have certain characteristics in common [3].

The output of an image segmentation process is a set of classes that cover the entire image with a set of contours that are extracted from the image. Each of the pixels in the same class label is similar to the other pixels of that class and they also share some common characteristics, such as color, intensity, or texture. The pixels that are in the adjacent regions are significantly different from those characteristics [4].

Segmentation processes are useful for a variety of tasks, including:

- **Autonomous vehicles**
  It needs to equip cars with the necessary perception to understanding their environment so that self-driving cars can safely integrate into our existing roads. In Figure 1.1, we can see a segmented image of DeepLab architecture which is used for autonomous driving [5].

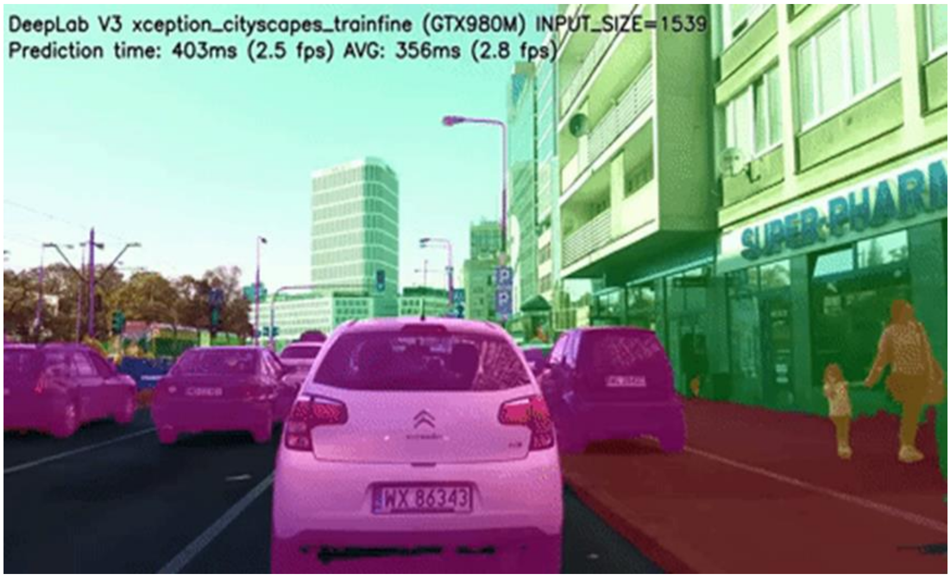

DeepLab V3 xception_cityscapes_trainfine (GTX980M) INPUT_SIZE=1539
Prediction time: 403ms (2.5 fps) AVG: 356ms (2.8 fps)
SUPER-PHARM
WX 86343

Figure 1.1: A real-time segmented road scene for autonomous driving [5].

- **Medical image diagnostics**
  Machines can augment analysis performed by radiologists, greatly reducing the time required to run diagnostics tests. In figure 1.2, an image of a chest x-ray is shown. We can see that in the image the heart and lungs can be easily identified [6].

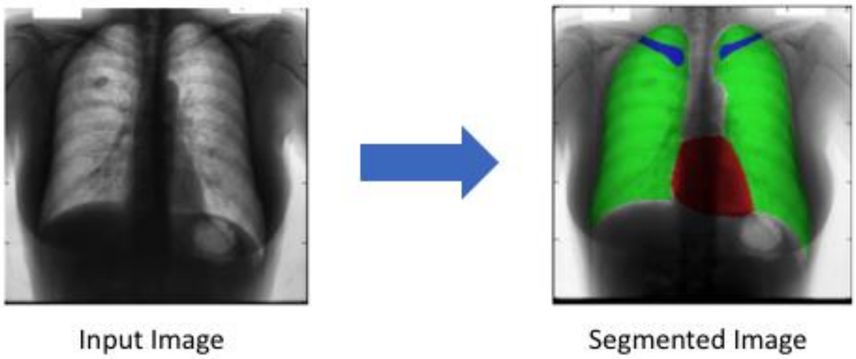
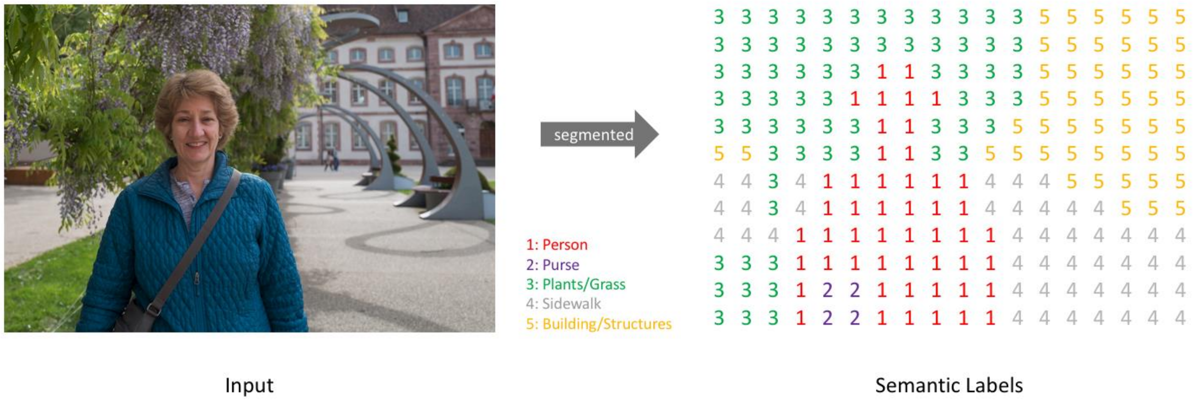


Figure 1.2: A chest x-ray with the heart (red), lungs (green), and clavicles (blue) are segmented [6].

1.2. Historical Background

Semantic segmentation means the segmentation of all objects of the image and classifies them into different distinguish classes based on their weights. Semantic segmentation is one of the most fundamental processes in the area of machine learning and machine vision. This area is important because semantic segmentation can be modeled as a fundamental preprocessing for other processes, that includes object modeling, object detection, scene understanding, and scene parsing. In figure 1.3, we can see the semantic segmented image is by comparing the real image and how the machine did the segmentation by simply marking the portion with different numbers [7].



3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Figure 1.3: Visualization of how semantic segmentation works [7].

1.2.1. Earlier Research

Alexander G. Schwing and Raquel Urtasun unified the two-stage process into a single joint training algorithm. They demonstrated their method on the semantic image segmentation task and had shown encouraging results on the challenging PASCAL VOC 2012 dataset [8].

Bharath Hariharan, Pablo Arbel´aez, Ross Girshick, and Jitendra Malik detected all instances of a category in an image and for each instance, they marked all the pixels that belong to it. They called this task Simultaneous Detection and Segmentation (SDS). They had built it on recent work that uses convolutional neural network R-CNN, introducing a novel architecture named SDS. They achieved state-of-the-art performance on semantic segmentation and state-of-the-art performance in object detection. In figure 1.4 the detailed architecture of SDS is shown [9].

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari presented a generic objectness measure. These included an innovative cue to measure the closed boundary characteristic. In experiments on the challenging PASCAL VOC 07 dataset, they showed this new cue to outperform a state-of-the-art saliency measure and the combined objectness measure to perform better than any cue alone [10].

Camille Couprie, Clement Farabet, Laurent Najman, and Yann Lecun had addressed the multiclass segmentation of indoor scenes with RGB-D inputs. While this area of research has gained much attention recently, most works still rely on hand-crafted features. In contrast, they had applied a multiscale convolutional network to learn features directly from the images and the depth information. They had obtained state-of-the-art on the NYU-v2 depth dataset with an accuracy of 64.5% [11].

Christian Szegedy, Alexander Toshev, and Dumitru Erhan showed outstanding performance on image classification tasks. They had presented a simple and yet powerful formulation of object detection as a regression problem to object bounding box masks. They achieved state-of-the-art performance of the approach had shown on Pascal VOC [12].

Jonathan Long, Evan Shelhamer, and Trevor Darrell had shown that convolutional networks by themselves, trained end-to-end, pixels to pixels, exceed the state-of-the-art in semantic segmentation. Their key insight was to build "fully convolutional" networks that take input of the arbitrary size and produce correspondingly-sized output with efficient inference and learning. They had defined and detailed the space of fully convolutional networks, explained their application to spatially dense prediction tasks, and had drawn connections to prior models. They had adapted contemporary classification networks like AlexNet, the VGG net, and GoogLeNet into fully convolutional networks and transfer their learned representations by fine-tuning to the segmentation task. Their fully convolutional network had achieved state-of-the-art segmentation of PASCAL VOC, NYUDv2, and SIFT Flow, while inference takes less than one-fifth of a second for a typical image. In figure 1.5, the detailed architecture of FCN is shown and we can see that the architecture can also learn backward [13].

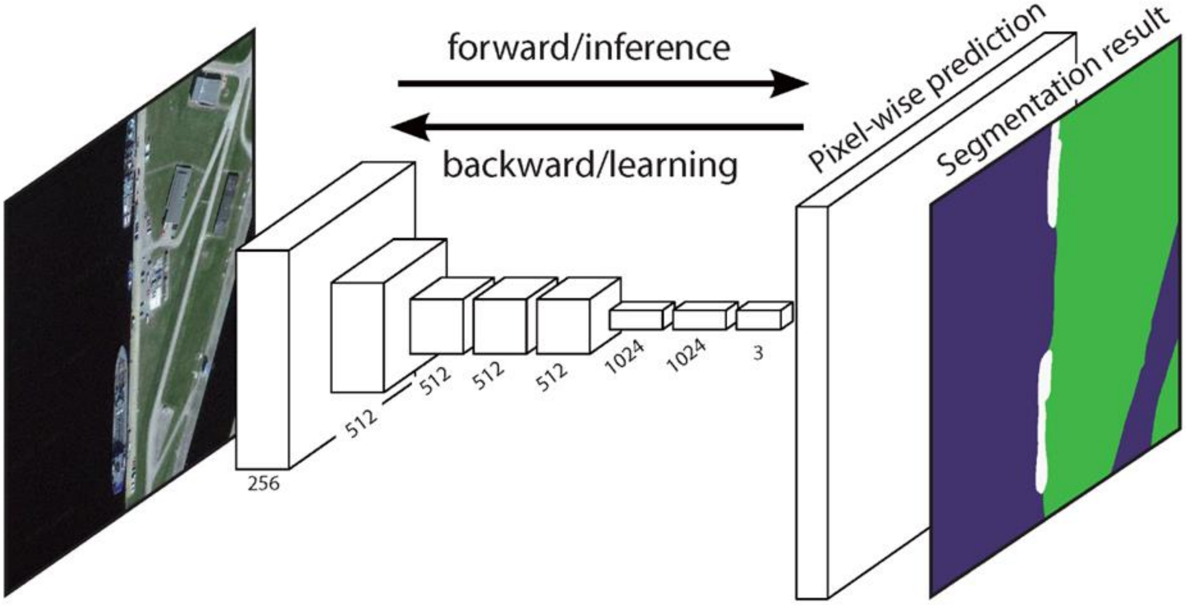


Figure 1.4: Fully Convolutional Network architecture [13].

Clement Farabet, Camille Couprie, Laurent Najman, Yann Lecun had proposed a method that had used a multiscale convolutional network trained from raw pixels to extract feature vectors on each pixel. The system had generated record accuracy on the Sift Flow Dataset (33 classes) and the Barcelona Dataset (170 classes) and near-record accuracy on Stanford Background Dataset (8 classes), while being an order of magnitude faster than competing approaches, producing a 320 × 240 image labeling in less than a second, including feature extraction [14].

1.2.2. Recent Research

Convolutional neural networks with many layers have recently been shown to achieve excellent results on many high-level tasks such as image classification, object detection, and more recently also semantic segmentation. Particularly for semantic segmentation, a two-stage procedure is often employed. Hereby, convolutional networks are trained to provide good local pixel-wise features for semantic segmentation of images. Some of the recent works on the semantic image segmentation are given below.

Jifeng Dai, Kaiming He, and Jian Sun had proposed a method to exploit shape information via masking convolutional neural network features. The proposal is treated as masks on the convolutional feature maps. The CNN features were directly used to train classifiers for recognition. They had further proposed a joint method to handle objects in the same framework. State-of-the-art results are demonstrated on benchmarks of PASCAL VOC and new PASCALCONTEXT, with a compelling computational speed [15].

Joseph J. Lim C. Lawrence and Zitnick Piotr Dollar had proposed a novel approach to both learning and detecting local contour-based representations for mid-level features. Their features, called sketch tokens, were learned using supervised mid-level information in the form of hand-drawn contours in images. They had achieved large improvements in detection accuracy for tasks of pedestrian and object detection as measured on INRIA and PASCAL, respectively. These gains were due to the complementary information provided by sketch tokens [16].

Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich had introduced a purely feed-forward architecture for semantic segmentation. They mapped small image elements (superpixels) to rich feature representations. This approach exploited statistical structure in the image. Their architecture had achieved state-of-the-art performance in semantic segmentation, obtaining 64.4% average accuracy on the PASCAL VOC 2012 test set [17].

Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla presented a novel and practical deep fully convolutional neural network architecture for semantic pixel-wise segmentation termed SegNet. This core trainable segmentation engine consisted of an encoder network, a corresponding decoder network followed by a pixel-wise classification layer. The architecture of the encoder network was topologically identical to the 13 convolutional layers in the VGG16 network. The role of the decoder network is to map the low-resolution encoder feature maps to full input resolution feature maps for pixel-wise classification. In figure 1.6, the architecture of the SegNet shown and here we can see that the architecture is divided into encoder and decoder layers. [18].

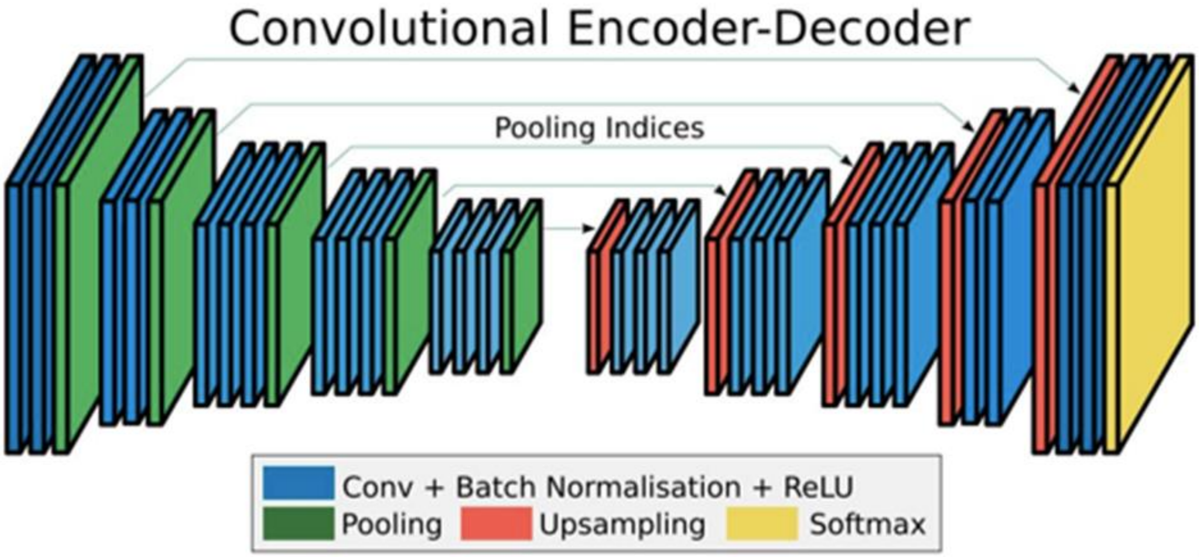

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax

Figure 1.5: SegNet architecture [18].

1.2.3. State of the art technology

Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger introduced the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages, they alleviate the vanishing gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. In figure 1.7, a five-layer DenseNet is shown, but we can easily expand these layers up to 100 layers [29].

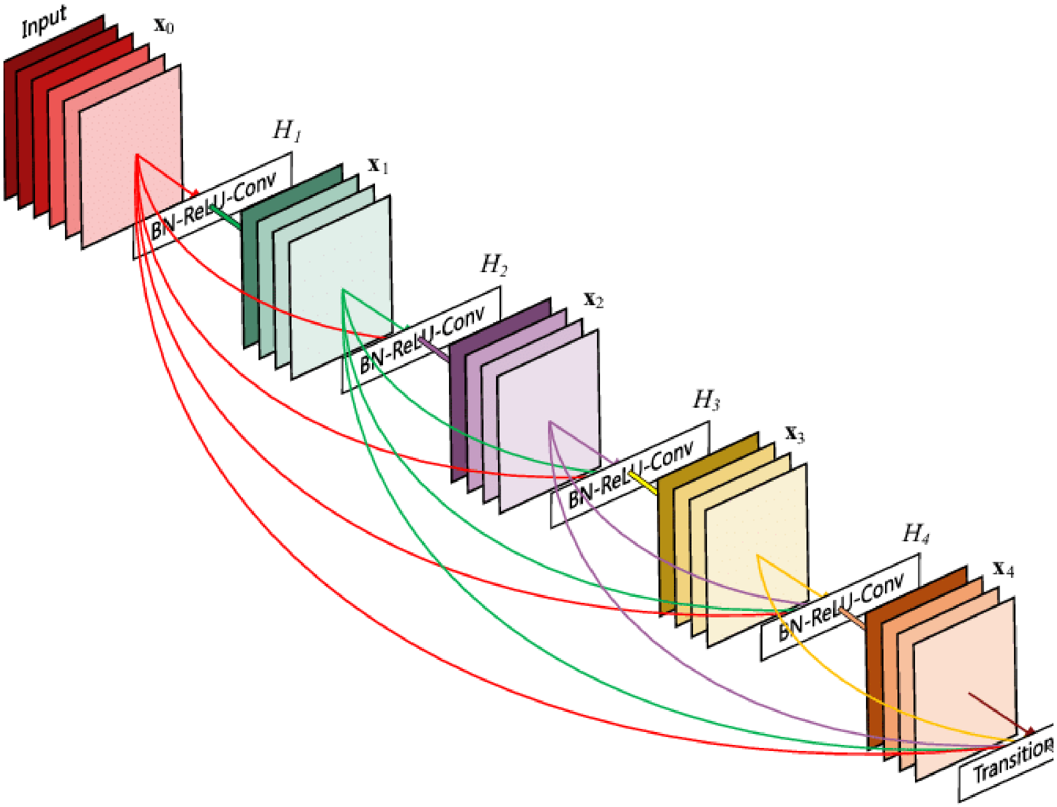


Figure 1.6: A DenseNet architecture with 5-layer connection [19].

1.3. Future Scope of This Study

The ability to predict is an important attribute of intelligence. It is also of utmost importance in real-time systems, e.g. in robotics or autonomous driving, which depend on visual scene understanding for decision making. Some of the future scopes of this work are given below-

1.3.1. Future Scopes

Image segmentation is being applied in many fields in today's world,

- **Automotive sector**: In developing advanced drivers assist for semi-autonomous cars and also heavily used in autonomous/driver-less cars.

- **Image enhancing:** The camera apps in smartphones and digital cameras using image segmentation to enhance image quality, video stabilization and noise removal, etc.
- **Robotics:** Mobile robot's navigation in an unknown environment (SLAM), control of the robot by processing the video feed from the camera on the robot to extract the live scene around it.
- **Gaming:** Advanced gaming consoles like Xbox Kinect uses image segmentation from motion analysis of the human player.
- **Problem specific solutions:** Image segmentation is used as a solution to a variety of problems, starting from facial recognition access to defects identification in manufacturing industries.
- **Manufacturing:** To identify defects in the processes and also to control the robots in performing certain tasks. for example, defects in the manufacturing of a Printed Circuit Board (PCB) can be observed using high-resolution image segmentation.
- **Human-machine interface:** Machines are made smart by adding a gestural interface, or human action response interfaces, which decodes the actions of the human user to perform certain tasks.

1.3.2. Recommendations

Image semantic segmentation is a huge area that is a combination of machine learning and digital image processing. Day by day our machines are getting smart and more autonomous. We know an autonomous machine can take its own decisions itself. For taking any kind of decision a machine needs to be perfectly aware of its environment. So, computer vision and image processing have a future ahead. Our work in the semantic segmentation of images can help to build more and more autonomous machines like robotics and self-driving vehicles. This ArNet architecture is particularly built for the road scenes to recognize the things that are present in a general roadway environment. So, it is recommended that it can be used in robotics and self-driving vehicles in the future by optimizing the architecture. From FCN8 to Fully Convolutional DenseNet many things are changed for achieving state-of-the-art accuracies and make the semantic image segmentation more efficient. In our architecture, we try to make this more efficient by decreasing some of these layers. We are still working on the layers and trying to make it more efficient if possible. In the future, we have a plan of working with the “Canny Edge Detector” algorithm. If we can interface the algorithm the efficiency of object detection and labeling will be easier as we know this algorithm can easily define the edges of any object from an image.

1.4. Limitation of the Study

In the way of building any architecture, there are several limitations that have to be faced. Maximum deep learning architecture is machine-oriented. In our work, we have a big issue with machine power. As we are working on the RGB image it takes a huge time to process the image in the NumPy array. Also, in any traditional convolutional neural network, it takes a huge time to complete the process. To make a lesser burden we need to use a low number of dataset images which is 367 images. We know that by using a big dataset can give us more accurate results. But using a large dataset will be a worse decision to complete the work. We utilize the GPU power over CPU for our work to make faster result but this also takes time more than expected. In the future, we will expand the dataset if we have more machine power.

1.5. Advantage over Traditional Method

In a convolutional neural network adding more layers will help us to extract more features. But we can do that up to a certain extent. There is a limit. After that, instead of extracting features, it tends to ‘overfit’ the data. Overfitting can lead to errors in some of the other forms like false positives. In the DenseNet there 100’s of layers to generate a more accurate result. In our ArNet the number of layers is lower than the other traditional architecture like SegNet or DenseNet but it can still manage state-of-the-art accuracy with this low number of layers. It also takes a lower amount of computation time, which also a very big advantage over any other traditional architecture.

1.6. Objective of this Work

To accomplish thesis work we make a list of some objectives that we must achieve during the work. Our primary and secondary objectives are stated below-

1.6.1. Primary objectives

The primary and main objective of this research is to build a deep convolutional neural network architecture for semantic segmentation of RGB images.

1.6.2. Secondary Objectives

To obtain this primary objective the following sub-objectives must also be achieved:

- Build a convolutional neural network architecture
- Prepare the dataset compatible so that it can be fed into the architecture
- Evaluate the architecture
- Compare the result with other mainframe architectures

1.7. Introduction to this Thesis

Image segmentation is a process where the objects of an image are classified by their pixel information. In an image, certain pixels share some common characteristics. By identifying them we can label the pixels, so they can be detected from the image. We use image segmentation to find objects in images like text (OCR), cars (automated vehicle), cancer, tumor (medical image). In recent days, deep learning algorithms got success in handwritten OCR, NLP [1] [2] [3]. It has also a huge active interest in pixel-wise semantic segmentation. Semantic segmentation is a machine learning process where a machine can learn from a visual scenario by adding class to every pixel that it contains. It is not like other instance segmentations. Instance segmentation can differentiate between objects of the same classes. In semantic segmentation, it can only detect and it doesn't care about the instances. Our ArNet is based on the SegNet architecture which is a deep convolutional encoder-decoder architecture for robust semantic image segmentation [4]. Our model is also based on encoder-decoder network architecture. One of the major drawbacks of SegNet is that it had too many layers. As we know by increasing the layers of an architecture, we can get higher accuracies but the cost will be more increased [5]. So, we build an architecture called ArNet with lower network layers with more accuracy.

Chapter 2
System Overview

2.1. Introduction

In this thesis, a model has been designed that can predict the semantic segmentation of any RGB images. The model has been designed after tinkering with different configurations of the convolutional neural network. This chapter mainly focuses on the architecture of the proposed model and the reason why this model has been chosen.

2.2. ArNet Architecture

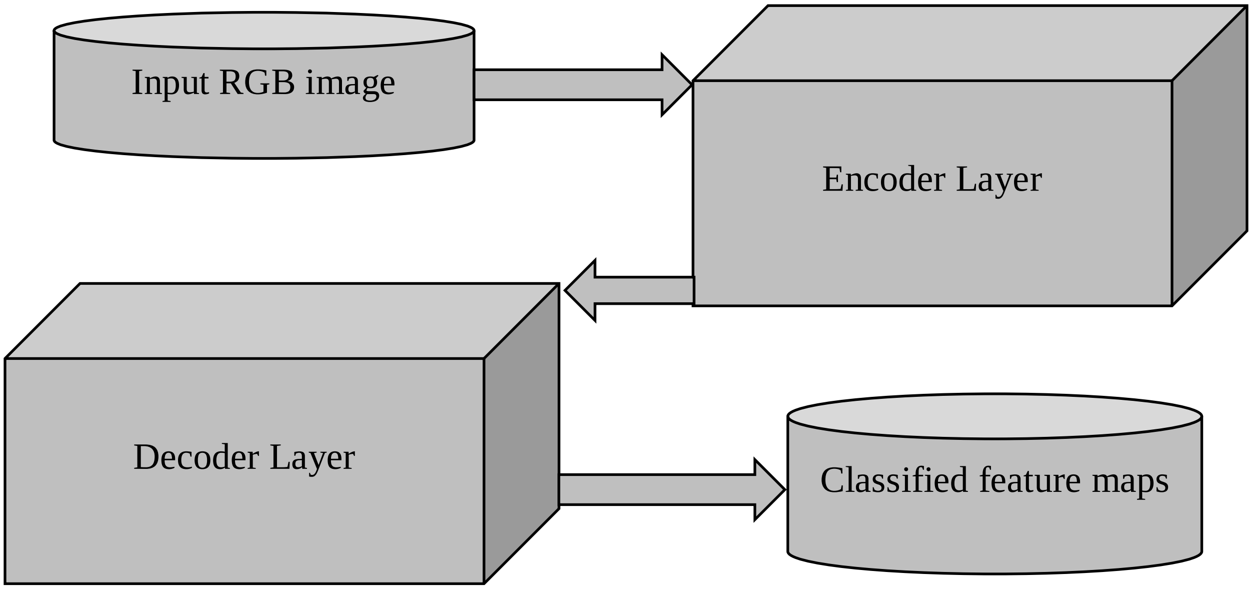


Figure 2.1: Overview of ArNet architecture.

Figure 2.1 describes the basic model of our ArNet architecture. In our architecture, there will be two main layers and they are the encoder layer and the decoder layer. The layers are corresponding to each other. In figure 2.2 the encoder layer is shown in detail. Here we can see that there are 4 2D convolutional blocks and corresponding to them there are also a batch normalization block, a

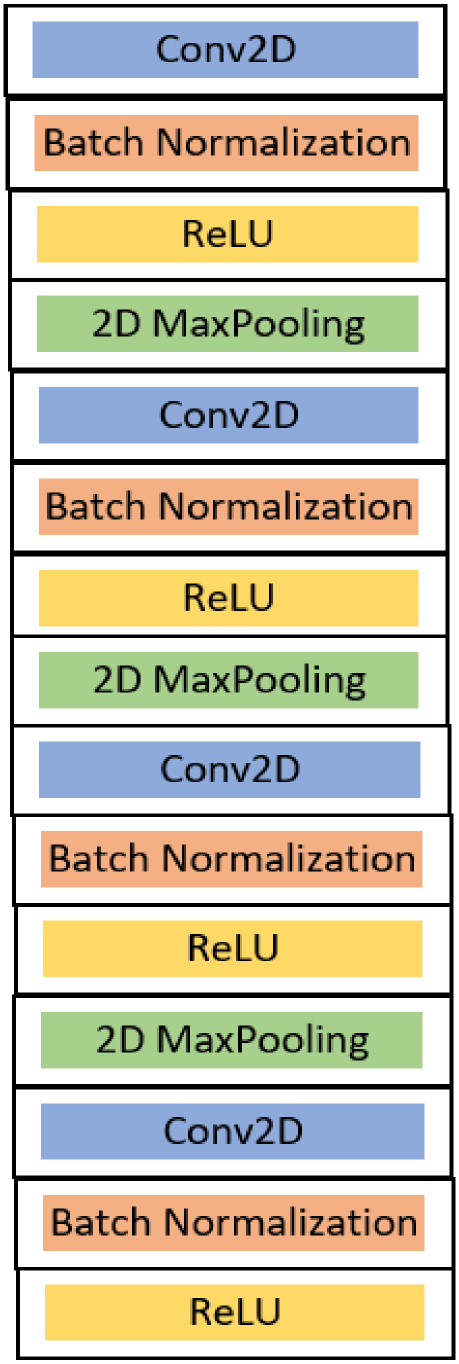


Figure 2.2: The encoder layer.

a ReLU activation block, and a 2D MaxPooling block for each 2D convolution block. In figure 2.3 the decoder layer is shown in detail. from figure 2.3 we can see that there are 4 2D deconvolution blocks are present in the decoder layer. Each block has a corresponding batch normalization, 2D UnPooling blocks. At the end of the decoder, there is a SoftMax activation block is present.

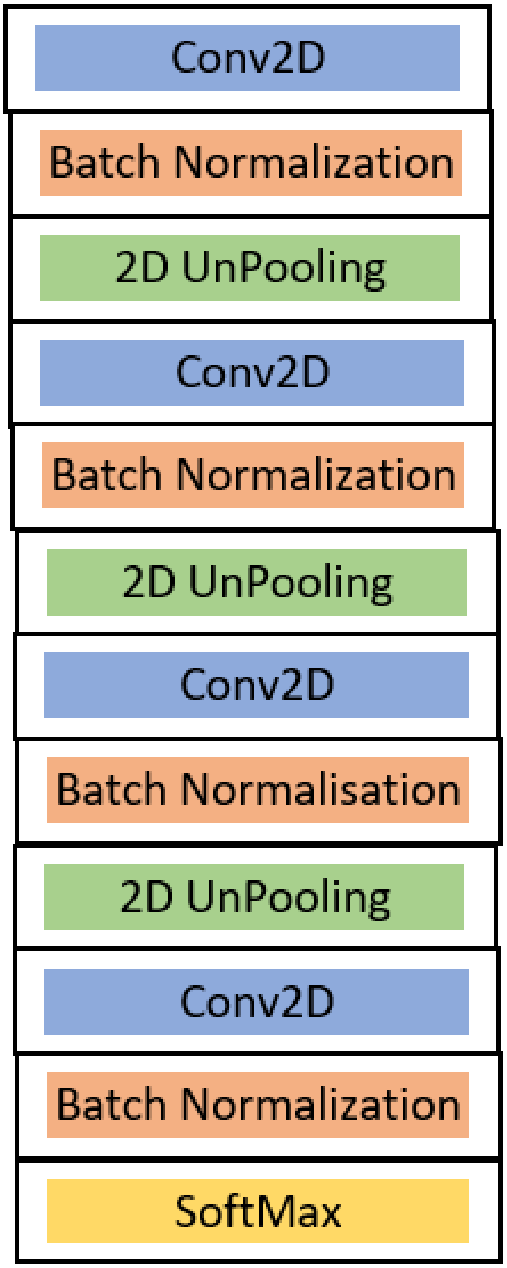


Figure 2.3: The decoder layer.

2.3.Deep Learning

Deep learning also is known as deep structured learning or hierarchal learning is a part of a broader family of machine learning methods based on learning data representations, as opposed to task-specified algorithms. Learning can be supervised, semi-supervised, or unsupervised. Deep learning methods are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains especially human brains, which make them incompatible with neuroscience evidence [20].

2.3.1. Convolutional Neural Network

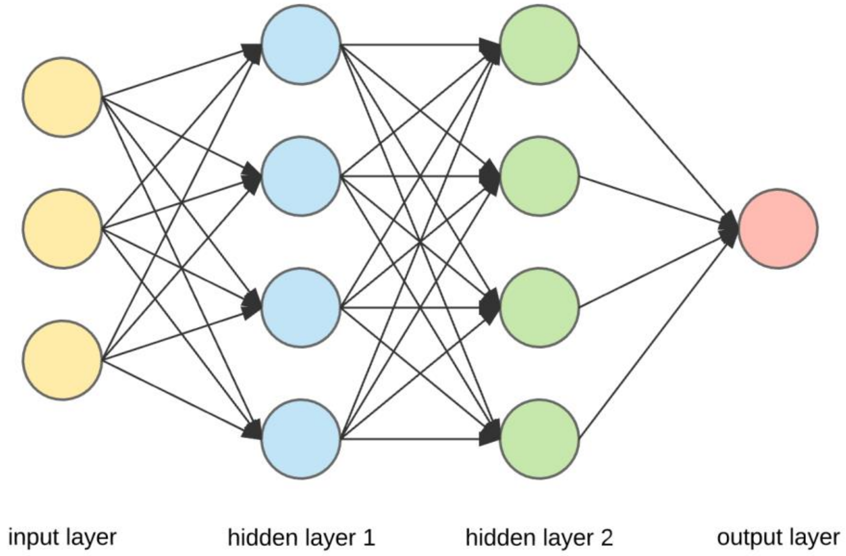


Figure 2.4: A convolutional neural network architecture [21].

A convolutional neural network is a class of deep learning. It is used mostly for image recognition, image or video processing, and for natural language processing. The convolutional neural network is used for both descriptive and generative purposes. A traditional neural network is either a combination of hardware or software or sometimes both. It was mainly built to mimic the structure of the human neural network but a traditional neural network is not completely for image processing and also for image recognition, whereas the model has to feed images in a reduced resolution process. In figure 2.4 a basic architecture of the convolutional neural network is shown, here we can see that a convolutional neural network may have several hidden layers. The accuracy of the convolutional neural network depends on the number of hidden layers.

A convolutional neural network is a regularized version of a multilayer perceptron. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. A convolutional neural network takes a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, a convolutional neural network is on the lower extremity [21].

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. A convolutional neural network uses relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a convolutional neural network typically consist of a series of convolutional layers that convolve with multiplication or other dot product. The activation function is commonly a RELU layer and is subsequently followed by additional convolutions such as pooling layers, fully connected layers, and normalization layers referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point [22].

2.3.2. Advantage of Convolutional Neural Network

Deep learning has efficiently gained an impressive performance on visual and speech recognition. The convolutional neural network is trained in a supervised way so while training the network it is required in which supervised way is required to train. A convolutional neural network requires few preprocessing compared to other models. One of the best advantages of the convolutional neural network is it does not require to use the handcrafting feature vector. As the convolutional neural network is trained in a supervised manner, therefore it does not require handcrafting to compute the feature vector. It is because of the design architecture of the convolutional neural network. It learns feature vector from the training dataset which is provided in the form of an image. The convolutional neural network is used to train a large class problem. As the convolutional neural network is a part of machine learning so according to the traditional machine learning it is predicted that training and testing dataset has the same distribution [22].

2.3.3. Applications of Convolutional Neural Network

There are several applications of convolutional neural networks based on the image. Some of them are multimedia retrieval, visual input and access, instance segmentation, industrial automation, semantic segmentation and etc [22].

2.3.3.1. Semantic Segmentation

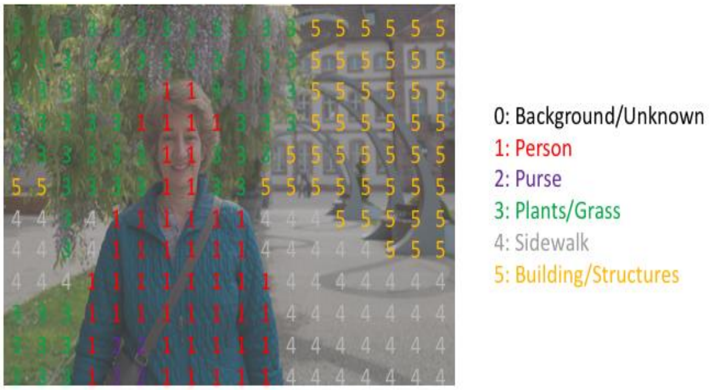


Figure 2.5: An overview of semantic image segmentation [7].

Semantic image segmentation describes the task of partitioning an image into regions that delineate meaningful objects and labeling those regions with an object category label. An example of semantic segmentation is given in Figure 2.5. It can be seen as a generalization of figure segmentation where one segments a particular object, say a building, from the other objects of the image [7].

The task is usually approached as supervised or semi-supervised machine learning, using a set of training images that are manually segmented and labeled. The learning algorithm then discovers relevant image features that help discriminate regions belonging to different categories in unseen test images. Semantic image segmentation has been of interest since at least 1989, but only fairly recently have processor speeds started to allow the rich models for high accuracy across many object categories. the semantic segmentation task can be treated as a pixel-labeling problem. The different methods proposed for solving this problem can be categorized based on the relationships they encode between different pixels (Figure 2.5). Some methods for semantic segmentation solve the pixel-labeling problem by classifying each pixel independently. Another class of methods works by grouping pixels into segments and assigning a single label to each group. Superpixels are computed from the image in a bottom-up fashion and can aid computational efficiency but may lead to final incorrect labeling. Accurately segmenting and recognizing image objects open up many potential applications. Not only does it tell us what things are in an image but also where they are and how they look. This information can be used in, for example, image search, image editing, augmented reality, robot navigation, and medical image analysis [8].

2.4. 2D Convolution

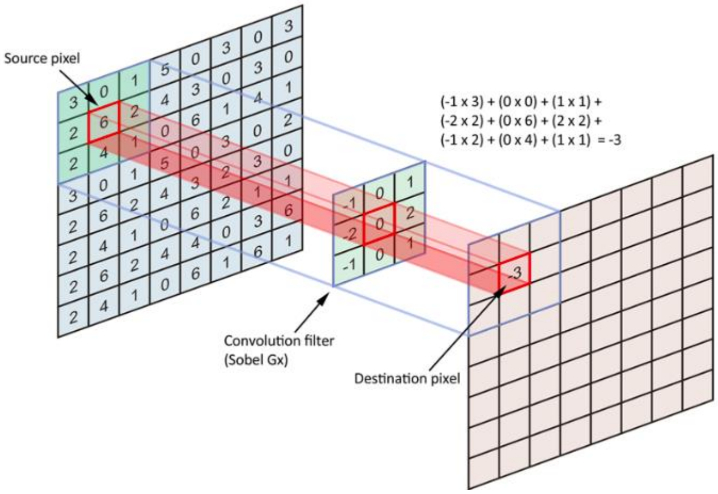

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 2.6: A 2D convolution operation [23].

Convolution is a mathematical operation on two functions that produces a third function expressing how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions after one is reversed and shifted. Convolution has applications that include probability, statistics, computer vision, natural language processing, image and signal processing, engineering, and differential equations. Computing the inverse of the convolution operation is known as deconvolution. In figure 2.6 a calculation of the convolution operation is given.

In normal dense network or multi-layer perceptron, all the neurons in one layer are connected to all neurons in the next layer, which means the weight params of the network is the multiplication product of the number of neurons in the connected layers, so if we were to process the image of higher resolutions the network params will be very high and require higher computational power and won’t scale for larger images. No one wants to train millions of params for the small images. This problem can be resolve by using a convolution neural network. Here the output is related to the closer input neurons and not all of them in the previous layer. convolution and pooling operations help us achieve this.

In simple terms convolution is just a matrix multiplication operation, Here the image will be represented as a numeric array which indicates the pixel values of the image. we will be multiplying the part of the image (depends on the size of the filter) with a filter over and over until we cover each and every pixel of the image. This process will shrink the image [23].

2.5. 2D Deconvolution

In optics and imaging, the term "deconvolution" is specifically used to refer to the process of reversing the optical distortion that takes place in an optical microscope, electron microscope, telescope, or other imaging instruments, thus creating clearer images. It is usually done in the digital domain by a software algorithm, as part of a suite of microscope image processing techniques. Deconvolution is also practical to sharpen images that suffer from fast motion or jiggles during capturing. Early Hubble Space Telescope images were distorted by a flawed mirror and were sharpened by deconvolution [24]. In mathematics, deconvolution is an algorithm-based process used to enhance signals from recorded data. Where the recorded data can be modeled as a pure signal that is distorted by a filter (a process known as convolution), deconvolution can be used to restore the original signal. The concept of deconvolution is widely used in the techniques of signal processing and image processing.

The 2D Deconvolution represents a layer that performs an opposite operation to 2D Convolution. This is also referred to as transposed convolution, which better reflects the actual mathematical operation. Convolution is not itself an invertible operation, which means we cannot simply go from the output back to the input. Deconvolution has instead to learn weights in the same way as Convolution layers. Deconvolutional operates similarly to convolution blocks, but stride has the opposite effect making the output bigger, not smaller. The deconvolution operation is used when we transform the output of a convolution back into a tensor that has the same shape as the input of that convolution. This is useful if we want a model that transforms images into other images, instead of just giving categorical or scalar predictions. Autoencoders and image segmentation models are examples of such models [25].

2.6. Batch Normalization

Batch normalization also known as a batch norm is a technique for improving the speed, performance, and stability of artificial neural networks. Batch normalization was introduced in a 2015 paper. It is used to normalize the input layer by re-centering and re-scaling. While the effect of batch normalization is evident, the reasons behind its effectiveness remain under discussion. It was believed that it can mitigate the problem of the internal covariate shift, where parameter initialization and changes in the distribution of the inputs of each layer affect the learning rate of the network. Recently, some scholars have argued that batch normalization does not reduce internal covariate shift, but rather smooths the objective function, which in turn improves the performance. Others sustain that batch normalization achieves length-direction decoupling, and thereby accelerates neural networks.

Each layer of a neural network has inputs with a corresponding distribution, which is affected during the training process by the randomness in the parameter initialization and the randomness in the input data. The effect of these sources of randomness on the distribution of the inputs to internal layers during training is described as an internal covariate shift. Although a clear-cut precise definition seems to be missing, the phenomenon observed in experiments is the change in means and variances of the inputs to internal layers during training. Batch normalization was initially proposed to mitigate the internal covariate shift. During the training stage of networks, as the parameters of the preceding layers change, the distribution of inputs to the current layer changes accordingly, such that the current layer needs to constantly readjust to new distributions. This problem is especially severe for deep networks because small changes in shallower hidden layers will be amplified as they propagate within the network, resulting in a significant shift in deeper hidden layers. Therefore, the method of batch normalization is proposed to reduce these unwanted shifts to speed up training and to produce more reliable models.

Besides reducing the internal covariate shift, batch normalization is believed to introduce many other benefits. With this additional operation, the network can use a higher learning rate without vanishing or exploding gradients. Furthermore, batch normalization seems to have a regularizing effect such that the network improves its generalization properties, and it is thus unnecessary to use dropout to mitigate overfitting. It has been observed also that with batch norm the network becomes more robust to different initialization schemes and learning rates [26].

2.7. 2D MaxPooling

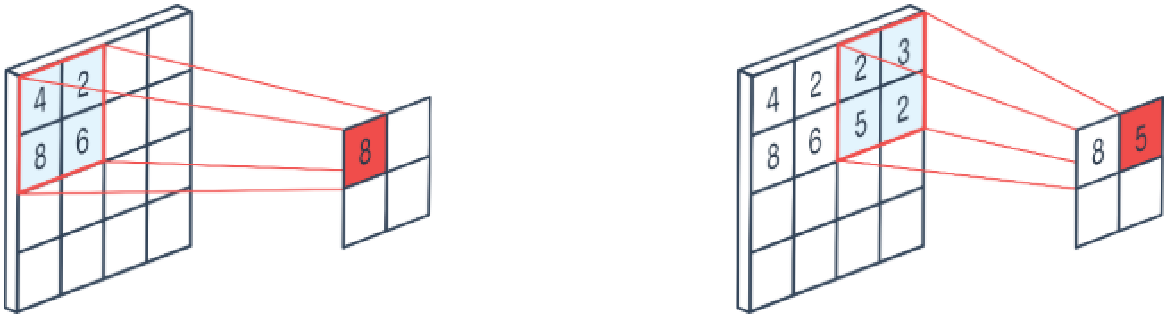


Figure 2.7: 2D MaxPooling [27].

MaxPooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. MaxPooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation.

In figure 2.7 we have a 4x4 matrix representing our initial input. Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a stride of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions. For each of the regions represented by the filter, we will take the max of that region and create a new, output matrix where each element is the max of a region in the original input [27].

2.8. 2D UnPooling

During the pooling operation, create a matrix that records the location of the maximum value, unpool operation will insert the pooled value in the original place, with the remaining elements being set to zero. UnPooling captures example-specific structures by tracing the original locations with strong activations back to image space. As a result, it effectively reconstructs the detailed structure [27].

2.9. Activation Function

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on the input. This is similar to the behavior of the linear perceptron in neural networks. However, only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes [28].

2.9.1. ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now. Since it is used in almost all the convolutional neural networks or deep learning. The rectified linear activation function is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. The function and it is derivative both are monotonic. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turn affects the resulting graph by not mapping the negative values appropriately [28].

2.9.2. SoftMax Activation

SoftMax is an activation function like other activation functions include RELU and Sigmoid. It is frequently used in classifications. SoftMax output is large if the score is large. Its output is small if the score is small. The proportion is not uniform. SoftMax is exponential and enlarges differences - push one result closer to 1 while another closer to 0. It turns scores aka logits into probabilities. Cross entropy like cost function is often computed for output of SoftMax and true labels encoded in one hot encoding. SoftMax function outputs a vector that represents the probability distributions of a list of potential outcomes [28].

2.10. Summary

A detailed overview of our architecture has been given in this chapter. The overview also includes the explanation of the different layers and blocks that are used in the deep convolutional neural network.

Chapter 3
Implementation

3.1. Introduction

The implementation of the architecture had done in 3 stages:

1. Gathering the dataset
2. Preprocessing the dataset
3. Creating the architecture

3.2. Gathering the Dataset

For training and evaluating our model, we used the CamVid Dataset which is a dataset for semantic image segmentation. The Cambridge-driving Labeled Video Database (CamVid) is the first collection of videos with object class semantic labels, complete with metadata. The database provides ground truth labels that associate each pixel with one of 32 semantic classes.

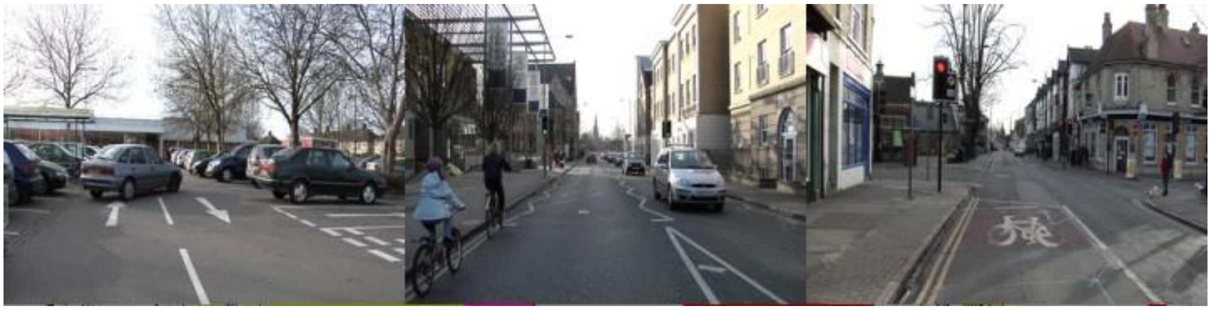


Figure 3.1: Overview of CamVid Dataset images [18].

The database addresses the need for experimental data to quantitatively evaluate emerging algorithms. While most videos are filmed with fixed-position CCTV-style cameras, our data was captured from the perspective of a driving automobile. The driving scenario increases the number and heterogeneity of the observed object classes. In figure 3.1 some of the images are shown form the CamVid dataset.

The CamVid Database offers four contributions that are relevant to object analysis researchers. First, the per-pixel semantic segmentation of over 700 images was specified manually and was then inspected and confirmed by a second person for accuracy. Second, the high-quality and large resolution color video images in the database represent valuable extended duration digitized footage to those interested in driving scenarios or ego-motion. Third, we filmed calibration sequences for the camera color response and intrinsic and computed a 3D camera pose for each frame in the sequences. Finally, in support of expanding this or other databases, we offer custom-made labeling software for assisting users who wish to paint precise class-labels for other images and videos. We evaluated the relevance of the database by measuring the performance of an algorithm from each of three distinct domains: multi-class object recognition, pedestrian detection, and label propagation [18].

3.2.1. 4.2 Preprocessing the Dataset

We have an RGB image dataset in our hands. Now, we need to prepare them as a NumPy array so they can feed into the architecture. We use the NumPy package of python to simply do that operation. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python [30]. In simple words, the NumPy will convert our image data into some array of a matrix. So that a computer can easily identify them. As we know the computer cannot understand any image data.

In our data set, there is a set of 367 images. We use a small amount of data because it will take too much time to process our data as the architecture is machine-oriented and it needs the best GPU support. At first, the data will be divided into train, test, and validation data using the NumPy package. After that, we will normalize our NumPy arrays with the histogram equalization operation. Histogram equalization is an image processing procedure that reassigns image pixel intensities. The basic idea is to use interpolation to map the original CDF of pixel intensities to a CDF that is almost a straight line. In essence, the pixel intensities are spread out and this has the practical effect of making a sharper, contrast-enhanced image. This is particularly useful in astronomy and medical imaging to help us see more features. It also used to improve the lighting of any low contrast image.

For our use, the dataset already provides the labels pre-calculated. The following 12 labels will be categorized by our architecture.

1. Sky
2. Building
3. Pole
4. Road
5. Pavement
6. Tree
7. Road sign signal
8. Fence
9. Car
10. Pedestrian
11. Bicycle
12. Unlabeled

After declaring all the labels now, but now the data is the form of Categorical Data. Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Categorical variables are often called nominal. Some algorithms can work with categorical data directly. Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some applications. There are several ways to convert categorical data into numerical data. We use one-hot encoding for converting categorical data to numerical data.

For categorical variables where no such ordinal relationship exists, this encoding will allow the model to assume a natural ordering between categories may result in poor performance or unexpected results. In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

At last, we need to calculate the weights of each label or class by median frequency balancing. So that the architecture can easily categorize the labels by their weights.

Before the one-hot encoding the variable look like,

```
label_values = [Sky, Building, Pole, Road, Pavement, Tree, SignSymbol,
Fence, Car, Pedestrian, Bicyclist, Unlabelled];
```

After one hot encoding the are turn into binary recognizable format,

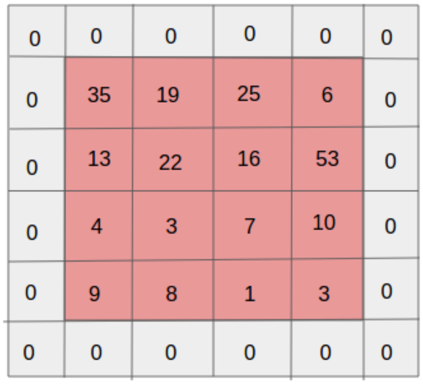
```
label_values[0]  = [1,0,0,0,0,0,0,0,0,0,0,0];
label_values[1]  = [0,1,0,0,0,0,0,0,0,0,0,0];
label_values[2]  = [0,0,1,0,0,0,0,0,0,0,0,0];
label_values[3]  = [0,0,0,1,0,0,0,0,0,0,0,0];
label_values[4]  = [0,0,0,0,1,0,0,0,0,0,0,0];
label_values[5]  = [0,0,0,0,0,1,0,0,0,0,0,0];
label_values[6]  = [0,0,0,0,0,0,1,0,0,0,0,0];
label_values[7]  = [0,0,0,0,0,0,0,1,0,0,0,0];
label_values[8]  = [0,0,0,0,0,0,0,0,1,0,0,0];
label_values[9]  = [0,0,0,0,0,0,0,0,0,1,0,0];
label_values[10] = [0,0,0,0,0,0,0,0,0,0,1,0];
label_values[11] = [0,0,0,0,0,0,0,0,0,0,0,1];
```

3.3. Creating the Architecture

The model has been created using Keras library. Keras is an open-source neural network library that has been written in Python and capable of running on top of TensorFlow and Theano [31]. The model is sequential. It has a linear stack of layers. A sequential model implements each layer in a sequential manner per epoch. The model can be trained using a NumPy array of input data. There are two layers in this model, first is the encoder layer and a corresponding decoder layer. The output of the encoder layer is the input of the decoder layer.

3.3.1. The Encoder Layer

The encoder layer starts with a 2D convolution block. The 2D convolution is a fairly simple operation start with a kernel size of 3x3, which is simply a small matrix of weights. This kernel slides over the 2D input data, performing an element-wise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel. The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essential, the weighted sums with the weights being the values of the kernel itself of the input features located roughly in the same location of the output pixel on the input layer. Here, we need to do a zero-padding operation because pixels on the edge are never at the center of the kernel because there is nothing for the kernel to extend to beyond the edge.



0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figure 3.4: A zero-padded 4 x 4 matrix becomes a 6 x 6 matrix [24].

Zero-padding refers to the process of symmetrically adding zeroes to the input matrix. It’s a commonly used modification that allows the size of the input to be adjusted to our requirement. It is mostly used in designing the CNN layers when the dimensions of the input volume need to be preserved in the output volume [24]. In figure 3.4 a zero matrix 4x4 matrix is shown which as becoming 6x6 is size.

After the 2D convolution block, a batch normalization block appears in the next block. The output of the 2D convolutional block is feed as input in the batch normalization block. Batch normalization also known as the batch norm is a technique for improving the speed, performance, and stability of artificial neural networks. It is used to normalize the input layer by re-centering and re-scaling. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). Batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers. We can use higher learning rates because batch normalization makes sure that there’s no activation that’s gone really high or really low. It reduces overfitting because it has a slight regularization effect. Similar to dropout, it adds some noise to each hidden layer’s activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information.

VGG16 doesn’t have a batch norm layer in it because batch normalization didn’t exist before VGG16. If we train it with it from the start, the pre-trained weight will benefit from the normalization of the activations. So, adding a batch norm layer actually improves our architecture. To summarize everything, we can think about batch normalization as doing preprocessing at every layer of the network [26].

The output of the batch normalization block is feed into a ReLU block. ReLU means a rectified linear unit. In the context of artificial neural networks, the rectifier is an activation function. In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. The rectified linear activation function is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

ReLU is linear (identity) for all positive values, and zero for all negative values. This means it's cheap to compute as there is no complicated math. The model can, therefore, take less time to train or run. It converges faster. It's sparsely activated. Data sparsity (missing information) is different and usually bad. Since ReLU is zero for all negative inputs, it's likely for any given unit to not activate at all. This is often desirable [28].

After the completion of the ReLU block, a 2D MaxPooling block comes into action. MaxPooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. MaxPooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation [27].

3.3.2. The Decoder Layer

The output of the encoder layer is directly fed into the decoder layer. The decoder layer does the total opposite work of the encoder layer. The first block of the decoder layer is the 2D deconvolutional layer. The 2D Deconvolution represents a block that performs an opposite operation to 2D Convolution. This is also referred to as transposed convolution, which better reflects the actual mathematical operation. Convolution is not itself an invertible operation, which means we cannot simply go from the output back to the input. Deconvolution has instead to learn weights in the same way as Convolution layers. Deconvolutional operates similarly to convolution blocks, but stride has the opposite effect making the output bigger, not smaller [24].

After the 2D deconvolution block, a batch normalization block appears in the next block. The output of the 2D deconvolutional block is feed as input in the batch normalization block. Batch normalization also known as the batch norm is a technique for improving the speed, performance, and stability of artificial neural networks. It is used to normalize the input layer by re-centering and re-scaling. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). Batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers. We can use higher learning rates because batch normalization makes sure that there's no activation that's gone really high or really low. It reduces overfitting because it has a slight regularization effect. Similar to dropout, it adds some noise to each hidden layer's activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information [26].

After the completion of the ReLU block, a 2D MaxPooling block comes into action. During the pooling operation, create a matrix that records the location of the maximum value, unpool operation will insert the pooled value in the original place, with the remaining elements being set to zero. UnPooling captures example-specific structures by tracing the original locations with strong activations back to image space. As a result, it effectively reconstructs the detailed structure [27].

After completing all the blocks, the output goes to a SoftMax activation function. SoftMax is an activation function like other activation functions include RELU and Sigmoid. It is frequently used in classifications. SoftMax output is large if the score is large. Its output is small if the score is small. The proportion is not uniform. SoftMax is exponential and enlarges differences - push one result closer to 1 while another closer to 0. It turns scores aka logits into probabilities. Cross entropy like cost function is often computed for output of SoftMax and true labels encoded in one hot encoding. SoftMax function outputs a vector that represents the probability distributions of a list of potential outcomes [28].

3.4. Implementation Tools

To implement and evaluate the deep convolutional neural network several development environments, packages, and frameworks are used. Here is a list of tools used in the total research process.

3.4.1. Environments

- **Anaconda Environment**: This the main python environment that has been used to develop our deep convolutional neural network architecture. Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and MacOS. Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface. The big difference between conda and the pip package manager is how to package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists [32].

- **Jupyter Notebook**: This notebook is used to develop and evaluate the deep convolutional neural network architecture as it can save the output result so that we don’t need to start compiling the architecture every time.

  Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text, and multimedia resources in a single document. Computational notebooks have been around for decades, but Jupyter, in particular, has exploded in popularity over the past couple of years. This rapid uptake has been aided by an enthusiastic community of user-developers and a redesigned architecture that allows the notebook to speak dozens of programming languages -- a fact reflected in its name, which was inspired, according to co-founder Fernando Pérez, by the programming languages Julia (Ju), Python (Py) and R [33].

3.4.2. Packages

- OpenCV: Our RGB images need to be processed for the deep convolutional neural network architecture so we use this OpenCV to do that. OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez which was later acquired by Intel. The library is cross-platform and free for use under the open-source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch after converting to an ONNX model and Caffe according to a defined list of supported layers [34].

- Numpy: For manipulating the images as arrays we Numpy package of python. It is a popular Python library used for performing array-based numerical computations. The canonical implementation of NumPy used by most programmers runs on a single CPU core and is parallelized to use multiple cores for some operations. This restriction to a single-node CPU-only execution limits both the size of data that can be handled and the potential speed of NumPy code [35].

3.4.3. Frameworks

- TensorFlow: It is the main framework that we used to implement the deep convolutional neural network architecture. TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer whereas in previous “parameter server” designs the management of the shared state is built into the system, TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks. Several Google services use TensorFlow in production, we have released it as an open-source project, and it has become widely used for machine learning research [31].

- Keras: It is the main library of neural networks that have been used to develop the deep convolutional neural network. Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer are François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model [36].

3.5. Novelty of Our Work

In a convolutional neural network adding more layers will help us to extract more features. But we can do that up to a certain extent. There is a limit. After that, instead of extracting features, it tends to ‘overfit’ the data. Overfitting can lead to errors in some of the other forms like false positives. In the DenseNet there 100’s of layers to generate a more accurate result. In our ArNet the number of layers is lower than the other traditional architecture like SegNet or DenseNet but it can still manage state-of-the-art accuracy with this low number of layers. It also takes a lower amount of computation time, which also a very big advantage over any other traditional architecture.

3.6. Summary

This chapter explains the implementation details of the entire system. It also describes the dataset we used for the model. The chapter explains how we make our data compatible with the model and how we processed it. Finally, it describes all the blocks that will come to action during the training process of the model.

Chapter 4
Experimental Results

4.1. Introduction

This chapter discloses the results we obtain from our architecture. It also includes a comparison with some other traditional architecture in the form of a table.

4.2. Evaluation of the Architecture

The architecture had been evaluated as it had been training. This evaluation used the training data and validation data. The validation had generated validation accuracy and validation loss.

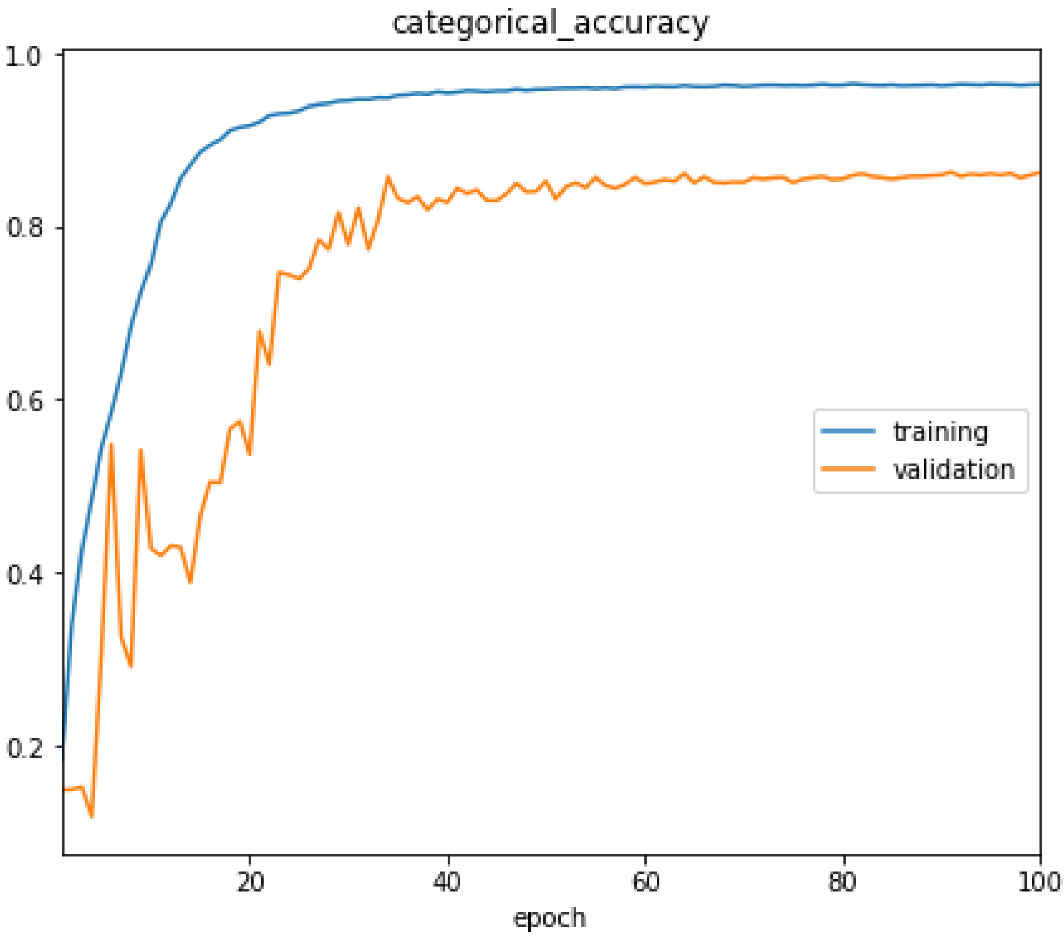


Figure 4.1: Plotted categorical accuracy per epoch of the training and validation data.

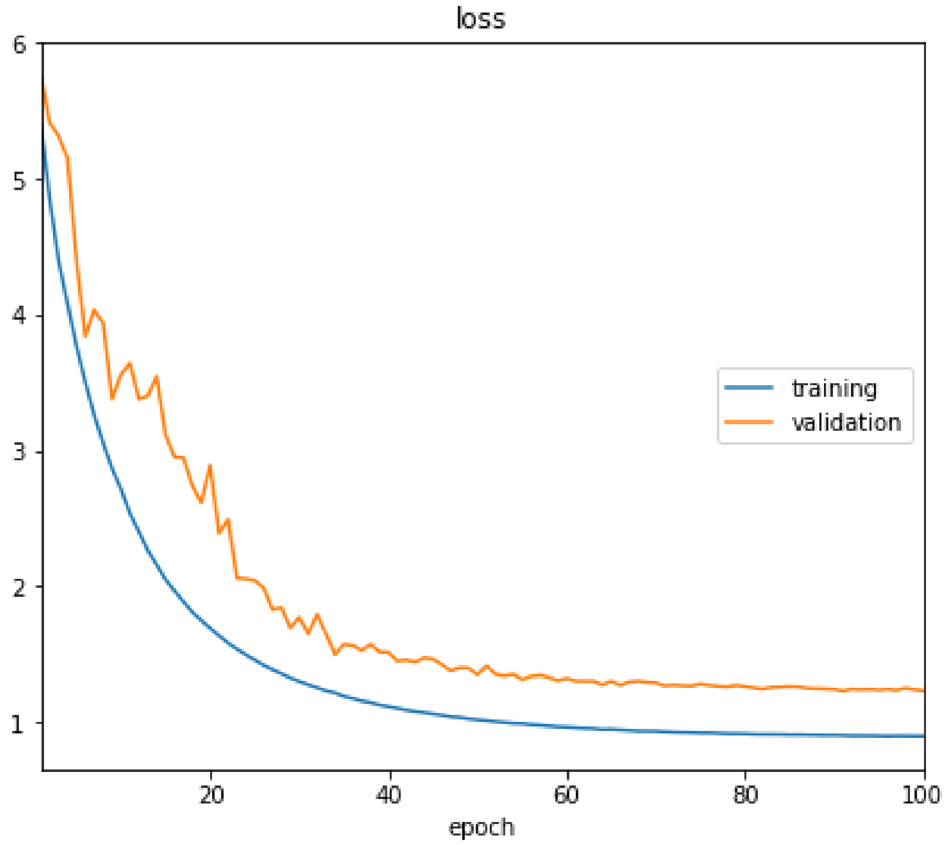


Figure 4.2: Plotted loss per epoch of the training and validation data.

Here, we plotted the data of training which had achieved a good state-of-the art result. Figure 4.1 provides a graphical representation of the model getting more and more accurate with every epoch at semantic segmentation. On the other hand, figure 4.2 provides a graphical representation of the reduction of the loss value with every epoch.

Table 4.1: The categorical accuracy and loss of validation and training data of the ArNet architecture.

ArNet	Training	Validation
Accuracy	0.963	0.887

In table 4.1 we have shown the result of our experiment's best results. Here, the training accuracy is 96.3% and training validation is 88.7% which is higher than most of the traditional architecture.

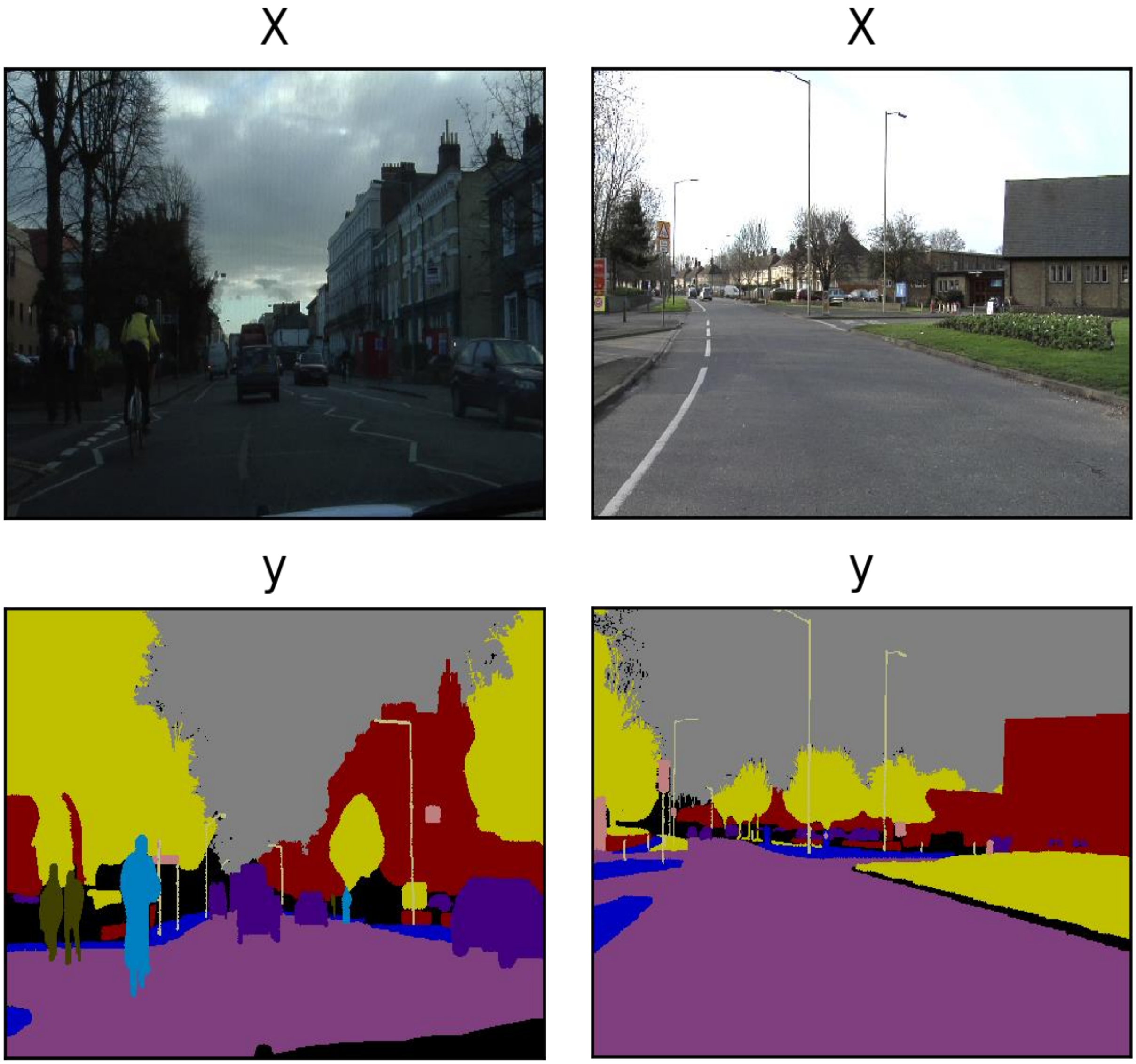


Figure 4.3: Some predicted results of ArNet architecture.

In figure 4.3 the result of the prediction of our ArNet architecture is shown. Here, x is the input image and y are the predicted image. If we compare the images, we can observe that every element or object in the images has been segmented into distinguishable segments or colors.

4.3.Comparisons

In the introduction chapter, we learned about some state-of-the-art architecture of semantic image segmentation. Here, is a comparison between the ArNet and them. In the table 4.2 the comparison is shown. Here, we can see that the validation accuracy of FCN8 is 62.5% accuracy which takes around 12 minutes per epoch on average. The SegNet have 88% accuracy with 1.6 hours per epoch on average. We have a shortage of machine power. So, the FC-DenseNet100 was unable to complete the training as a single epoch takes around13 hours. At last our ArNet architecture got 88.7% accuracy with around 32 minutes per epoch on average.

Table 4.2: Validation accuracy comparison between some the state-of-the-art architecture and ArNet.

Architecture	Validation Accuracy	Time Per Epoch (avg)
FCN8 [13]	62.5	12 minutes
SegNet [18]	88.0	1.6 hours
FC-DenseNet100 [19]	91.2	13 hours (only one epoch was done completely)
ArNet	88.7	32 minutes

4.4. Summary

This chapter shows the output we get from the experiment of our ArNet architecture. Here, we also make a comparison between some of the major architecture for semantic segmentation of the image.

Chapter 5
Discussions and Conclusions

5.1. Discussions

In the way of building architecture, there are several limitations that we have to face. One of the major problems is that the architecture needs more and more epochs to be accurate or to generate greater accuracy. We are only able to do 100 epochs so far and we did it only once because we have very big limitations of powerful machines. We train the architecture on a machine having 2.30GHz CPU, 16GB of RAM, 8GB of Graphics Card. At first, we tried to train the architecture with the CPU power but it takes more than 42 minutes to complete an epoch. So, we moved to the CUDA platform means we started to use the power of the graphics card to train the architecture. It took around 32 minutes for one epoch. The time for 100 epochs is around 54 hours for our architecture. It’s a huge amount of time for the laptop to operate. So, we trained our model once. In the future, if we have a more powerful machine in our hands, we will try to overcome this problem as we know in machine learning can have more accuracy by training the architecture over and over. For the limitations of the machine, we afford to add more datasets. If we can more datasets, we will be able to generate more accurate semantic segmentation. But more datasets will need more machine power. So, a low dataset is also a limitation of our work.

5.2. Suggestion for Future Work

Image semantic segmentation is a huge area that is a combination of machine learning and digital image processing. Day by day our machines are getting smart and more autonomous. We know an autonomous machine can take its own decisions itself. For taking any kind of decision a machine needs to be perfectly aware of its environment. So, computer vision and image processing have a future ahead. Our work in the semantic segmentation of images can help to build more and more autonomous machines like robotics and self-driving vehicles. This ArNet architecture is particularly built for the road scenes to recognize the things that are present in a general roadway environment. So, it can be used in robotics and self-driving vehicles in the future by optimizing the architecture. From FCN8 to Fully Convolutional DenseNet many things are changed for achieving state-of-the-art accuracies and make the semantic image segmentation more efficient. In our architecture, we try to make this more efficient by decreasing some of these layers. We are still working on the layers and trying to make it more efficient if possible. In the future, we have a plan of working with the “Canny Edge Detector” algorithm. If we can interface the algorithm the efficiency of object detection and labeling will be easier as we know this algorithm can easily define the edges of any object from an image.

5.3. Conclusions

Semantic segmentation of the image is one of the key processes of machine automation like robotics and autonomous vehicle. So, it is important to develop these basic processes more efficient to make our machines smarter. This need motivated us to work with this architecture. Our ArNet is a deep architecture that has a lower number of blocks from other traditional architecture but it can give the same accuracy as them. In the future, we will be looking for more optimization for our architecture and then it could be able to make a benchmark in the world of computer vision.

REFERENCES

[1] D. H. Ballard, C. M. Brown (1982). "*Computer Vision*". Prentice-Hall. ISBN 978-0-13-165316-0.

[2] B. Jahne; H. Haubecker (2000). "*Computer Vision and Applications, A Guide for Students and Practitioners*". Academic Press. ISBN 978-0-13-085198-7.

[3] L. G. Shapiro and G. C. Stockman (2001): "*Computer Vision*", pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3.

[4] B. Lauren, and L. W. Lee. "*Perceptual information processing system*." Paravue Inc. U.S. Patent Application 10/618,543, filed July 11, 2003.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, "*DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*", Computer Vision and Pattern Recognition, (Submitted on 2 Jun 2016 (v1), last revised 12 May 2017 (this version, v2)).

[6] Z. Stefan, M. Zilske, and H.-C. Hege. "*3D reconstruction of individual anatomy from medical image data: Segmentation and geometry processing*." (2007).

[7] A. A. Novikov, D. Lenis, D. Major, J. Hladůvka, M. Wimmer, K. Bühler, "*Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs*", Computer Vision and Pattern Recognition, (Submitted on 30 Jan 2017 (v1), last revised 13 Feb 2018 (this version, v4)).

[8] A. G. Schwing, R. Urtasun, "*Fully Connected Deep Structured Networks*", Computer Vision and Pattern Recognition, (Submitted on 9 Mar 2015), arXiv:1503.02351.

[9] Hariharan B., Arbeláez P., Girshick R., Malik J. (2014) "*Simultaneous Detection and Segmentation*". In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8695. Springer, Cham.

[10] B. Alexe, T. Deselaers, V. Ferrari, "*Measuring the Objectness of Image Windows*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34/11 (2012), pp. 2189-2202.

[11] C. Couprie, C. Farabet, L. Najman, Y. LeCun, "*Indoor Semantic Segmentation using depth information*", Computer Vision and Pattern Recognition, (Submitted on 16 Jan 2013 (v1), last revised 14 Mar 2013 (this version, v2)).

[12] D. Erhan, C. Szegedy, A. Toshev, D. Anguelov, “*Scalable Object Detection using Deep Neural Networks*”, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 2147-2154.

[13] J. Long, E. Shelhamer, and T. Darrell, "*Fully convolutional networks for semantic segmentation*," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 3431-3440.

[14] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "*Learning Hierarchical Features for Scene Labeling*," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1915-1929, Aug. 2013.

[15] J. Dai, K. He, J. Sun, “*Convolutional Feature Masking for Joint Object and Stuff Segmentation*” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3992-4000

[16] J. J. Lim, C. L. Zitnick, P. Dollar, “*Sketch Tokens: A Learned Mid-Level Representation for Contour and Object Detection*” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3158-3165.

[17] M. Mostajabi, P. Yadollahpour, G. Shakhnarovich, “*Feedforward semantic segmentation with zoom-out features* “, Computer Vision and Pattern Recognition, (Submitted on 2 Dec 2014). arXiv:1412.0774.

[18] V. Badrinarayanan, A. Handa and R. Cipolla "*SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling*.", Computer Vision and Pattern Recognition (Submitted on 2 Nov 2015 (v1), last revised 10 Oct 2016 (this version, v3)), arXiv preprint arXiv:1505.07293, 2015.

[19] G. Huang, Z. Liu, L. v. d. Maarten and K. Q. Weinberger, "*Densely Connected Convolutional Networks,*" 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269.

[20] Y. Bengio (2009), "*Learning Deep Architectures for AI*", Foundations and Trends in Machine Learning: Vol. 2: No. 1, pp 1-127. http://dx.doi.org/10.1561/2200000006.

[21] M.Masakazu, K. Mori, Y. Mitari, Y. Kaneda. "*Subject independent facial expression recognition with robust face detection using a convolutional neural network*". Neural

Networks. 16 (5): 555–559. DOI:10.1016/S0893-6080(03)00115-1. PMID 12850007. Retrieved 17 November 2013.

[22] A. G. Schwing, R. Urtasun, “*Fully Connected Deep Structured Networks*”, Computer Vision and Pattern Recognition, (Submitted on 9 Mar 2015), arXiv:1503.02351.

[23] D. Torres, Alejandro (Nov 2, 2010). "*Origin and history of convolution*". 41 pgs. http://www.slideshare.net/Alexdfar/origin-adn-history-of-convolution. Cranfield, Bedford MK43 OAL, UK. Retrieved March 13, 2013.

[24] T. O'Haver, "*Intro to Signal Processing - Deconvolution*". The University of Maryland at College Park. Retrieved 2007-08-15.

[25] N. Wiener, (1964). “*Extrapolation, Interpolation, and Smoothing of Stationary Time Series*.” Cambridge, Mass: MIT Press. ISBN 0-262-73005-7.

[26] A.V. Knyazev, K. Neymeyr, (2003). "*A geometric theory for preconditioned inverse iteration III: A short and sharp convergence estimate for generalized eigenvalue problems*". Linear Algebra and Its Applications. 358 (1–3): 95–114. DOI:10.1016/S0024-3795(01)00461-X.

[27] Q. Zhao, S. Lyu, B. Zhang, and W. Feng, “*Multiactivation Pooling Method in Convolutional Neural Networks for Image Recognition*”, Big IoT Data Analytics in Fog Computing, Volume-2018, Article ID-8196906, https://doi.org/10.1155/2018/8196906.

[28] K. Eckle, J. Schmidt-Hieber, “*A comparison of deep networks with ReLU activation function and linear spline-type methods*”, Neural Networks, Volume 110, 2019, Pages 232-242, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2018.11.005.

[29] M. Maurício, and D. Stringhini. “*High-Level Computer Vision Using OpenCV*.” 2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials (2011): 11-24.

[30] Y. Bengio (2009), "*Learning Deep Architectures for AI*", Foundations and Trends in Machine Learning: Vol. 2: No. 1, pp 1-127. http://dx.doi.org/10.1561/2200000006.

[31] M. Bauer, M. Garland. 2019. “*Legate NumPy: accelerated and distributed array computing*”. In Proceedings of the International Conference for High-Performance Computing, Networking, Storage and Analysis (SC ’19). Association for Computing Machinery, New York, NY, USA, Article 23, 1–23. DOI: https://doi.org/10.1145/3295500.3356175.

[32] J. Wang and A. L. Yuille, "*Semantic part segmentation using a compositional model combining shape and appearance*," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1788-1797.

[33] D. Christine, "*Conda works with Linux, OSX, and Windows, and is language-agnostic, which allows us to use it with any programming language or even multi-language projects*.", (21 May 2015). "Conda for Data Science". Archived from the original on 16 June 2015. Retrieved 16 Jun 2015.

[34] J. B. Hamrick, "*Creating and Grading IPython/Jupyter Notebook Assignments with NbGrader",* SIGCSE '16: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, February 2016, Page 242. https://doi.org/10.1145/2839509.2850507.

[35] Marangoni, Maurício, and Denise Stringhini. "*High-Level Computer Vision Using OpenCV*." 2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials (2011): 11-24.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K.Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, "*TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*", (Submitted on 14 Mar 2016 (v1), last revised 16 Mar 2016 (this version, v2)).