Environmental Adaptation for the Urban Scene Understanding of Autonomous Vehicles

Author:

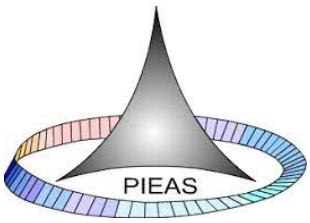
Aroosha Pervaiz

Supervisor:

Dr. Irfan Ul Haq

A thesis submitted in fulfillment of the requirements

for the degree of Bachelors in Computer and Information Sciences



Department of Computer and Information Sciences

Pakistan Institute of Engineering and Applied Sciences

June 26, 2022

Declaration of Authorship

I, Aroosha Pervaiz, declare that this thesis titled, "Environmental Adaptation for the Urban Scene Understanding of Autonomous Vehicles" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

"Once you trust a self-driving car with your life, you pretty much will trust artificial intelligence with anything."

Dave Waters

PAKISTAN INSTITUTE OF ENGINEERING AND APPLIED SCIENCES

Abstract

Department of Computer and Information Sciences

Environmental Adaptation for the Urban Scene Understanding of Autonomous Vehicles

by Aroosha Pervaiz

When robots are tested in real environments, they are prone to encountering obstacles that are specular and unknown. To tackle this, they utilize a map of their environment using the numerous sensors situated in different parts of the vehicle. Urban Scene Understanding is when autonomous vehicles use this information by classifying the objects around them into different categories, so it helps them avoid obstacles and understand the traffic more. There has been a lot of research in this domain, but the proposed models and techniques fail to adapt and generalize to the real world.

To combat this, we performed the reconstruction of available datasets into a composite, unified one to tackle the issue of domain generalization. This dataset contains 194 classes. Using this dataset, we trained a condition generative adversarial network, which can not only generalize to real-data but also to simulation data. In order to further enhance the efficacy of our model, we propose a modified Pix2Pix that tells our model "where to look at." This is done by adding an attention mechanism in our generator. Not only have we been able to perform satisfactory results, we have, so far, achieved the state of the art results as compared to our baseline methodology. The satisfactory results of our model on the simulation data extend its usefulness in the gaming world as well as domains like 'Metaverse.'

Acknowledgements

I acknowledge the help provided by our supervisor Dr. Irfan ul Haq. He was always available whenever I needed any assistance. Without his farsightedness and vision, we would not have achieved even a fraction of what we did.

Special thanks to Dr. Imran Nadeem for his continued support and for always being there. Without his technical support, the completion of this project would have been next to impossible.

Special thanks to Dr. Naeem Akhtar for his continuous evaluation, guidance, and feedback that enabled me to attain my project goals.

Special thanks to Azeem Sarwar and Amina Batool for their friendship, love and for enduring the endless rants about the workload. Without their support and encouragement, the completion of this project would have been difficult.

Contents

List of Figures

List of Tables

Dedicated to my parents for their endless love, support, and encouragement.

Chapter 1

Introduction

1.1 Motivation

With the advent of new technologies and improved resources, there has been extensive development in the realization of autonomous vehicles in urban traffic environments. Autonomous driving is being researched with the expectation that it will alleviate the gruesome externalities associated with traffic, including traffic congestion and traffic accidents (Martnez-Daz and Soriguera, 2018). It should also be noted that autonomous vehicles remain a challenging research topic because of the complexities posed by the intricate and complex nature of traffic and the expectations for these vehicles to mimic human behavior that is ethical, strategical, and traffic management. Let us now discuss the terminologies associated with the project in detail.

1.2 Vision

In our project, our main emphasis is on Environmental Adaptation for the Urban Scene Understanding of Autonomous Vehicles. So, we want our model to properly classify and adapt, and then properly generalize on an unknown environment.

1.2.1 Informal Statement

We want to propose such an algorithm that assists the autonomous vehicles for a hassle-free drive.

1.2.2 Formal Statement

1.2.2.1 Task (T)

After supervised training of the model on a particular dataset, the algorithm can environmentally adapt to classify another unknown domain.

1.2.2.2 Experience (E):

The algorithm trains using a generalized, composite dataset comprising of various datasets.

1.2.2.3 Performance

We will make use of measures such as Generator Loss, Discriminator Loss, and mean Intersection Over Union (mIoU).

1.2.3 Domains of Our Project

1.2.3.1 Generalized Approach

a. Reconstruction and regeneration of the MSeg dataset [4] that comprises of the following datasets: KITTI (Geiger et al., 2013), CityScapes (Cordts et al., 2016), COCO (Lin, 2014) + COCO STUFF (Caesar, Uijlings, and Ferrari, 2018), ADE20K (Zhou et al., 2017), MAPILLARY datasets (Neuhold et al., 2017), and PASCAL VOC (Everingham et al., 2010), PASCAL CONTEXT (Mottaghi et al., 2014), CAMVID (Richter et al., 2016), (Richter et al., 2016), (Geiger et al., 2013), SCANNET-20 (Dai et al., 2017). b. Relabeling of the composite data using mseg-mturk (Lambert et al., 2020) labeling scripts. c. Training of

our specialized Graph-Adversarial Network on this dataset. d. Testing on a state-of-the-art unknown dataset.

1.2.3.2 Major Aspects of our Methodology

Our domain generalization approach can be extended in to four different aspects, but the scope of our project is limited to the first two categories only.

1.2.3.2.1 Training on real data, testing on real data

In this approach, we train our model on data that is purely real and expect it to generalize well on data that is unseen and belongs to a terrain that's unknown but purely real.

1.2.3.2.2 Training on real data, testing on simulation data

In this approach, we train our model on data that is purely real and expect it to generalize well on data that is unseen and belongs to a terrain that's unknown but purely simulated.

1.2.3.2.3 Training on simulation data, testing on simulation data

In this approach, we train our model on data that is purely simulated and expect it to generalize well on data that is unseen and belongs to a terrain that's unknown but purely simulated.

1.2.3.2.4 Training on simulation data, testing on real data

In this approach, we train our model on data that is purely simulated and expect it to generalize well on data that is unseen and belongs to a terrain that's unknown but purely real.

1.3 Thesis Structure

1.3.1 Chapter 3 - Basic Concepts and State of The Art

In this chapter, we discuss the basic concepts that are imperative for understanding the basics behind our work. Furthermore, we present a comprehensive analysis regarding the previously related work.

1.3.2 Chapter 4 - Problem Context and Scope

In this chapter, we discuss the challenges posed by the problem we are trying to resolve. Then, we define the scope and context within which our thesis is confined.

1.3.3 Chapter 5 - Comprehensive Analysis of the Composite Dataset

To elaborate on our methodology, we start with a comprehensive survery and exploratory data analysis of the datasets that we utilize in our composite dataset. Thereon, we present the exact methodology that we utilized to place all these datasets into a composite, unified taxonomy for the creation of a single dataset. Onwards, we explain the reasoning for using these datasets.

1.3.4 Chapter 6 - Proposed Methodology

In this chapter, we define and explain the algorithms that we utilize for training our deep-learning model. Precisely, we describe pix2pix, and then go on to explain the modified pix2pix: an attention based pix2pix introduced by us.

1.3.5 Chapter 7 - Implementation

In this chapter, we demonstrate the results and compare them with pur baseline papers. We also present the results from our cross-validation results across different training and testing datasets. Furthermore, we provide some qualitative results to show the efficacy of our trained model.

Chapter 2

Basic Concepts and State of The Art

2.1 Problem Statement

In order to fully describe the problem that we are trying to solve, we can divide it into three basic components, which are as follows:

- Autonomous Driving
- Urban Scene Understanding
- Environmental Adaptation and Generalization

TABLE 2.1: Summary of our Problem Statement.

Autonomous Driving	**Urban Scene Understanding**	**Adaptation and Generalization**
Our main aim is to assist the autonomous vehicles for a hassle-free, safe driving	Semantic Segmentation Based Approach; Assist in the understanding and classification of the objects	Train on X, Test on Y

2.1.1 Autonomous Driving

Autonomous driving cars are driverless cars that don't need human intervention for their mobility– they can drive just like a human driver does, and they can go anywhere just like a normal vehicle does. The self-driving pipeline of these autonomous vehicles can be divided into four components: perception, localization, planning, and control. One of the most challenging technical aspects of autonomous vehicles is the detection of obstacles unequivocally and indubitably at high speeds and long distances. This is related to the "perception" component. To mitigate this problem, the companies have released extensive sets of datasets that are finely labeled to assist in accelerating the development of autonomous vehicles.

2.1.2 Urban Scene Understanding

It is incredibly important for the autonomous vehicle to be perceptive to the environment. It has been observed that the Urban Scene Understanding has moved from a neglected area to a much more focused area of research in recent years (Hoiem et al., 2015). There are three hierarchies of Urban Scene Understanding: Object Detection, Semantic Segmentation, and Instance Segmentation. Object Detection is the broadest category and refers to the identification of different objects in the image using bounding boxes. Whereas in semantic segmentation, we assign each pixel a class label. When it comes to instance-based segmentation, it is similar to semantic segmentation, but we also treat multiple objects belonging to the same class as separate entities. Object Detection offers less complexity but consumes less computational power, while Instance-based Segmentation provides us with high knowledge of surroundings but is also computationally huge. To make a decent trade-off between the complexity and computational resources, we will be focusing on Semantic Segmentation to ensure that we have adequate details of the

scenery and our computational costs don't exceed too much.

2.1.3 Environmental Adaptation and Generalization

With the advent and ever-increasing popularity in Advanced Driver's Assistance Systems (ADAS), there has been a surge in interest in autonomous vehicles, hence making it an important topic in the research domain. There has been tremendous progress in the Semantic Segmentation domain, but the models and algorithms proposed cannot generalize well to different datasets or a different location. This presents an issue known as covariate shift (Shimodaira, 2000) or selection bias (Heckman, 1979) in which the models are more biased towards the training dataset, so they don't generalize well when it comes to real-world data. In adaptation, we use the dataset from different test environments to train a robust algorithm that has seen various test environments. In generalization, we expect the data to perform satisfactorily to previously unseen data that was not available during the training process.

2.2 Literature Survey

There has been some progress when it comes to the aggregation of the datasets, but it is mostly related to a single area or domain. One of the earliest contributions is by Ros et al. who managed to aggregate six datasets related to driving (Ros et al., 2016). Another notable contribution is by Bevandic et al., 2019 who combined the datasets Mapillary Vistas, ImageNet-1KBB, Cityscapes, the WildDash validation set and performed joint segmentation, and then subsequently, used WildDash for outlier detection (Heckman, 1979). When it comes to domain adaptation techniques, Triantafillou et al. used few shot learning techniques on a meta dataset (Triantafillou et al., 2019). Vol et al. has used domain difference as a measure of noise to achieve a very robust optimization approach (Volpi et al., 2018). One of the most notable contributions

is from Lambert et al., which is also one of our baseline papers (Lambert et al., 2020). The primary purpose of this paper is to present a composite dataset for semantic segmentation from different domains. They do this by reconciliation of domains and object masks in more than 1,00,000 images. However, there methodology has the biggest flaw which arises from the need to manually relabel the data. We want to propose such a methodology that resolves the needs for this relabeling that's labor-intensive and tedious.

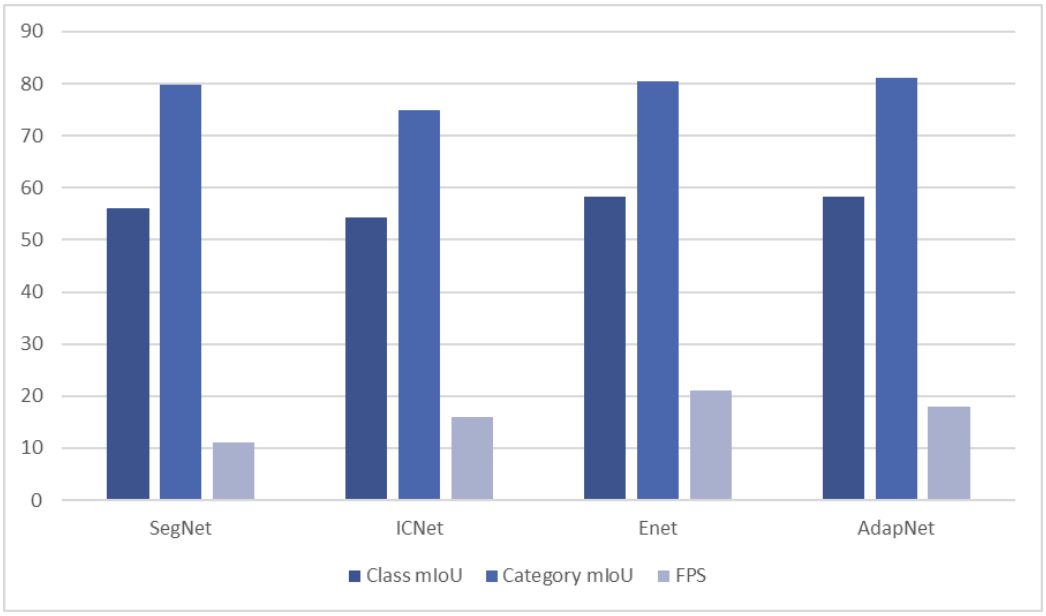


FIGURE 2.1: A comparison of Classical Deep Learning approaches for Semantic Segmentation. The x-axis shows the Class mIOU, Category mIoU, FPS for each dataset. The y-axis represents the corresponding percentages.

When we talk about Deep-Learning approaches to semantically segment the view for the purpose of better understanding the view, in particular, the traffic scenes, there have been many advancements, but they usually limit their scope to fewer datasets with labels containing no more than 34 classes. Semantic segmentation enables us to categorize each pixel into different classes, or in formal term, into different semantic labels. Semantic segmentation paves a way for us to achieve a complete understanding of environment, hence allowing us to apply it to domains like autonomous driving, virtual as well as augmented reality, and even indoor navigation. When we

talk about the networks proposed for Semantic Segmentation, ENet (Paszke et al., 2016), SegNet (Badrinarayanan, Kendall, and Cipolla, 2017), AdapNet (Valada et al., 2017), or Mask RCNN (He et al., 2017) follow the encoder decoder patterns, and many further algorithms use these as building blocks to propose newer versions. Apart from improving the accuracy, people have come up with solutions that provide a trade-off between the inference speed and accuracy. Such example is that of (Treml et al., 2016) and (Canziani, Paszke, and Culurciello, 2016). They propose a lighter network which greatly improves the inference speed. (Treml et al., 2016) propose a modified SqueezeNet for improving inference speed while maintaining reliable accuracy. The major computational parts are the fire modules consisting of three convolutions. They substituted Rectified Linear Units (ReLUs) with Exponential Linear Units (ELUs) (Clevert, Unterthiner, and Hochreiter, 2015). Their reason for doing that is to make efficient use of features, and it also reserves the negative part of the activation function. They achieved a consistently better inference speeds at the frame sizes of 480x320, 640x360, and 1280x720. Even their accuracy was better than both the Enet (Paszke et al., 2016) and SegNet (Badrinarayanan, Kendall, and Cipolla, 2017).

Chapter 3

Problem Context and Scope

3.1 Challenges

When it comes to the problems in Semantic Segmentation, they have been rife. These issues include the lack of data containing adequate labels, the changing taxonomies, the size of the views, incorrect labels, datasets with extremely unbalanced instances, and so on. Let us discuss each of this problem in details.Martnez-Daz and Soriguera, 2018

3.1.1 Lack of Data containing Adequate Labels

There has been enormous success in the successful segmentation of images related to autonomous vehicles, however, the classes are sometimes often inadequate. When we switch from one domain to another, there is a significant variation in the scenery that an autonomous vehicles has to adapt to. The type of roads change, the ambience changes, there can be an influx of other objects like animals that our model has never seen. In such cases, it won't be able to properly identify such objects, thus resulting in situations that can be hazardous and unexpected.

3.1.2 The changing taxonomies

When we move from one terrain to another, our view changes drastically. Even within cities, there can be an enormous shift in the taxonomies. In order to cater that, we have used datasets that belong to different terrains.

3.2 Scope

As robots evolve in unknown environments, they are prone to encountering specular obstacles. Urban Scene Understanding is when autonomous vehicles make use of this information by classifying the objects around them into different categories, so it helps them avoid obstacles and understand the traffic more. There has been a lot of research in this domain, but the proposed models and techniques fail to adapt and generalize to the real world. To combat this, we will perform the reconstruction of Mseg dataset that recently became available. We also want to utilize this newer dataset, which has, so far, no publications. This is the Audi Autonomous Driving Dataset(A2D2) (Geyer et al., 2020). We wish to achieve a model that can achieve domain adaptation and generalization without the need for manually relabeling the testing dataset.

Chapter 4

Comprehensive Analysis of the Composite Dataset

4.1 Exploratory Data Analysis

4.1.1 KITTI (Geiger et al., 2013)

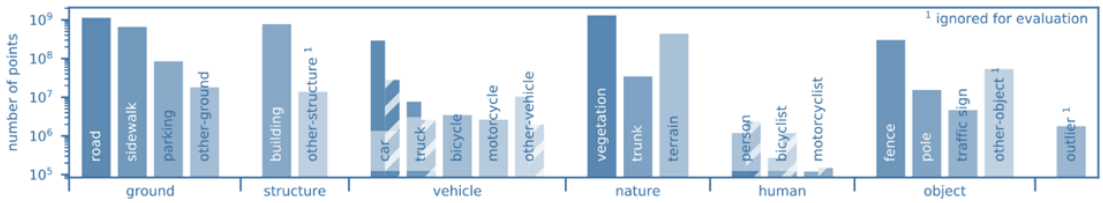


FIGURE 4.1: Classes distribution for KITTI dataset.

The KITTI dataset comprises of 28 classes to distinguish and locate moving as well as non-moving objects. These are divided into 6 major categories and their subsequent sub-categories, namely:

- Ground: 'road', 'sidewalk', 'parking', 'other-ground'
- Structure: 'building', 'other-structure'
- Vehicle: 'car', 'truck', 'bicycle', 'motorcycle', 'other-vehicle'
- Nature: 'vegetation', 'trunk', 'terrain'
- Human: 'person', 'bicyclist', 'motorcyclist'

- Object: 'fence', 'pole', 'traffic', 'other-object'
- Outliers: *ignored for evaluation

4.1.2 CityScapes (Cordts et al., 2016)

- Flat: 'road', 'sidewalk', 'parking', 'rail track'
- Human: 'person', 'rider'
- vehicle : 'car', 'truck', 'bus', 'on rails', 'motorcycle', 'bicycle', 'caravan', 'trailer'
- Construction: 'building', 'wall', 'fence', 'guard rail', 'bridge', 'tunnel'
- Object: 'pole', 'pole group', 'traffic sign', 'traffic light'
- Nature: 'vegetation', 'terrain'
- Sky: 'sky'
- Void: 'ground', 'dynamic', 'static'

4.1.3 COCO (Lin, 2014) + COCO STUFF (Caesar, Uijlings, and Ferrari, 2018)

This dataset consists of fifteen different classes, which are further divided into several other classes. Here's the list:

- Water: 'water-other', 'waterdrops', 'sea', 'river', 'fog'
- Ground: ground-other', 'playingfield', 'platform', 'railroad', 'pavement', 'road', 'gravel', 'mud', 'dirt', 'snow'
- Solid: solid-other', 'hill', 'mountain', 'stone', 'rock', 'wood'
- Sky: 'sky-other', 'clouds'

- Vegetation: 'vegetation-other', 'straw', 'moss', 'branch', 'flower', 'leaves', 'bush', 'tree'
- Structural: 'structural-other', 'railing', 'net', 'cage', 'fence'
- Building: 'building-other', 'roof', 'tent', 'bridge', 'skyscraper', 'house'
- Food-Stuff: 'food-other', 'vegetable', 'salad', 'fruit'
- Textile: textile-other', 'banner', 'pillow', 'blanket', 'curtain', 'clothes', 'napkin', 'towel', 'mat', 'rug'
- Furniture-Stuff: furniture-other', 'stairs', 'light', 'counter', 'mirror', 'cupboard', 'cabinet', 'shelf', 'table', 'desk', 'door'
- Window: window-other', 'window-blind'
- Floor: 'floor-other', 'floor-stone', 'floor-marble', 'floor-wood', 'floor-tile', 'carpet'
- Ceiling: 'ceiling-other', 'ceiling-tile'
- Wall: wall-other', 'wall-concrete', 'wall-stone', 'wall-brick', 'wall-wood', 'wall-panel', 'wall-tile'
- Raw-Material: 'metal', 'plastic', 'paper', 'cardboard

4.1.4 ADE20K (Zhou et al., 2017)

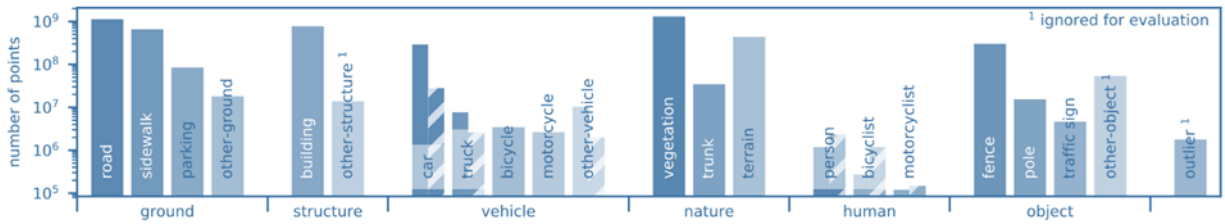


FIGURE 4.2: Classes distribution for ADE20K dataset

- Person: 'head', 'right arm', 'left arm', 'left leg', 'right leg', 'right hand', 'torso'
- Building: 'window', 'door', 'balcony', 'column', 'shop', 'window', 'roof', 'shutter'
- Car: 'wheel', 'window', 'door', 'headlight', 'license plate', 'taillight', 'windshield'
- Chair: 'leg', 'back seat', 'apron', 'arm', 'stretcher', 'seat', 'cushion'
- Light source: 'aperture', 'diffusor', 'shade', 'backplate', 'canopy', 'bulb'
- Window: 'pane sash', 'lower sash', 'upper sash', 'muntin', 'rail', 'casing'
- Table: 'leg', 'drawer', 'top', 'apron', 'door', 'base', 'side'
- Cabinet: 'door', 'drawer', 'top', 'front', 'side shelf', 'skirt'
- Lamp: 'shade', 'column', 'base', 'tube', 'canopy', 'cord', 'arm'
- Door: 'door', 'frame', 'handle', 'knob', 'pane', 'muntin', 'lock', 'hinge'
- Bed: 'headboard', 'leg', 'footboard', 'side', 'rail', 'bedpost', 'side', 'ladder'
- Shelf: 'shelf', 'door', 'side', 'top', 'leg'
- Armchair: 'armleg', 'back seat', 'cushion', 'back', 'pillow', 'seat'
- Sofa: 'base seat', 'seat'
- Desk: 'cushion', 'back pillow', 'armleg', 'seat'
- House: 'base', 'back skirt'
- Swivel Chair: 'drawer', 'leg', 'top', 'door', 'side', 'shelf'

- Stool: 'window', 'door', 'roof', 'chimney', 'railing', 'column', 'shutter'
- Coffee Table: 'back', 'base', 'seat', 'arm', 'piston', 'armrest'
- Van: 'leg', 'seat', 'stretcher', 'apron'
- Chandelier: 'footrest', 'leg', 'top', 'apron', 'shelf', 'drawer'
- Truck: 'wheel', 'window', 'door', 'license plate', 'taillight', 'windshield', 'headlight'
- Dresser: 'shade', 'bulb', 'arm', 'chain', 'canopy', 'wheel'
- Kitchen Stove: 'windshield', 'headlight', 'window', 'license', 'plate', 'mirror'
- Computer: 'drawer', 'side', 'skirt', 'front', 'top', 'leg', 'base'
- Telephone: 'receiver', 'base', 'screen', 'buttons', 'cord', 'keyboard'
- Fan: 'blade', 'motor', 'canopy', 'shade', 'tube'
- Bus: 'window', 'wheel', 'windshield', 'headlight', 'door', 'license plate'
- Wardrobe: 'mirror', 'door', 'drawer', 'shelf', 'side', 'top', 'leg', 'door'
- Microwave: 'button', 'panel', 'dial', 'screen', 'buttons', 'display'
- Airplane: 'button', 'landing', 'gear', 'stabilizer', 'wing', 'fuselage', 'turbine', 'engine'
- Pool Table: 'corner', 'pocket', 'leg', 'side', 'pocket', 'bed', 'rail', 'cabinet', 'base'
- Washing Machine: 'door', 'dial', 'buttons', 'detergent', 'dispenser', 'button', 'panel', 'screen'
- Kitchen Island: 'door', 'drawer', 'work', 'surface', 'side', 'top'

4.1.5 MAPILLARY Vistas dataset (Neuhold et al., 2017)

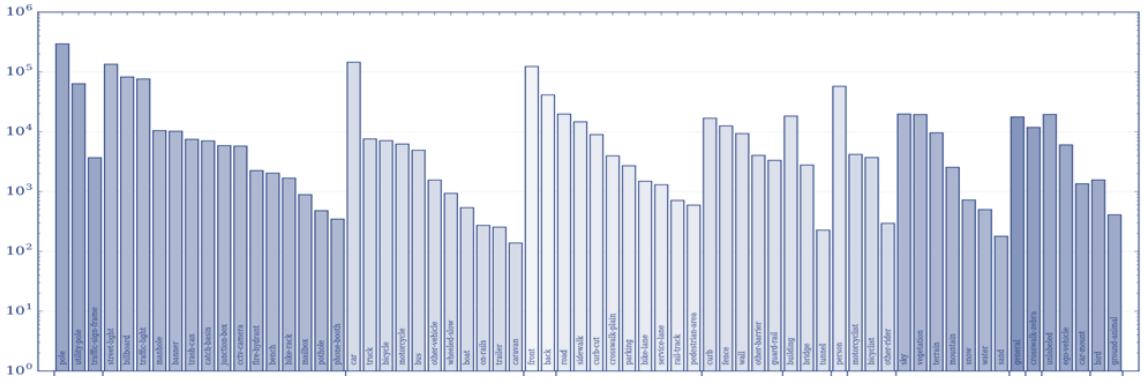


FIGURE 4.3: Classes distribution for Mapillary Vistas dataset

- Flat: 'road', 'sidewalk', 'curb-cut', 'crosswalk-plain', 'parking', 'bike-lane', 'service-lane', 'rail-track', 'pedestrian-area'
- Rider: 'bicyclist', 'motorcyclist', 'other-rider'
- Nature: 'sky', 'vegetation', 'terrain', 'mountain', 'snow', 'water', 'sand'
- Marking: 'general'
- Marking, discrete: 'crosswalk-zone'
- Void: 'ego-vehicle', 'car mount', 'general'
- Animal: 'bird', 'ground-animal'
- Barrier: 'curb', 'fence', 'wall', 'other-barrier', 'guard-rail', 'building'
- Structure: 'building', 'bridge', 'tunnel'
- Human: 'person'
- Support: 'pole', 'utility-pole', 'traffic-sign-frame'
- Object: 'street-light', 'billboard', 'traffic-light', 'mobile', 'banner', 'trash-can', 'catch-basic', 'junction-box', 'cctv-camera', 'fire-hydrant', 'bench', 'bike-rack', 'mailbox', 'pothole', 'phone-booth'

- Vehicle: 'car', 'truck', 'bicycle', 'motorcycle', 'bus', 'other-vehicle', 'boat', 'on-rails', 'trailer', 'caravan'
- Traffic-sign: 'front', 'back'

4.1.6 PASCAL VOC (Everingham et al., 2010)

In Pascal VOC, classes are divided into 20 categories, namely:

- Aeroplane
- Bicycle
- Bird
- Boat
- Bottle
- Bus
- Car
- Cat
- Chair
- Cow
- Dining table
- Dog
- Horse
- Motorbike
- Person
- Potted plant

- Sheep
- Sofa
- Train
- Tv/monitor

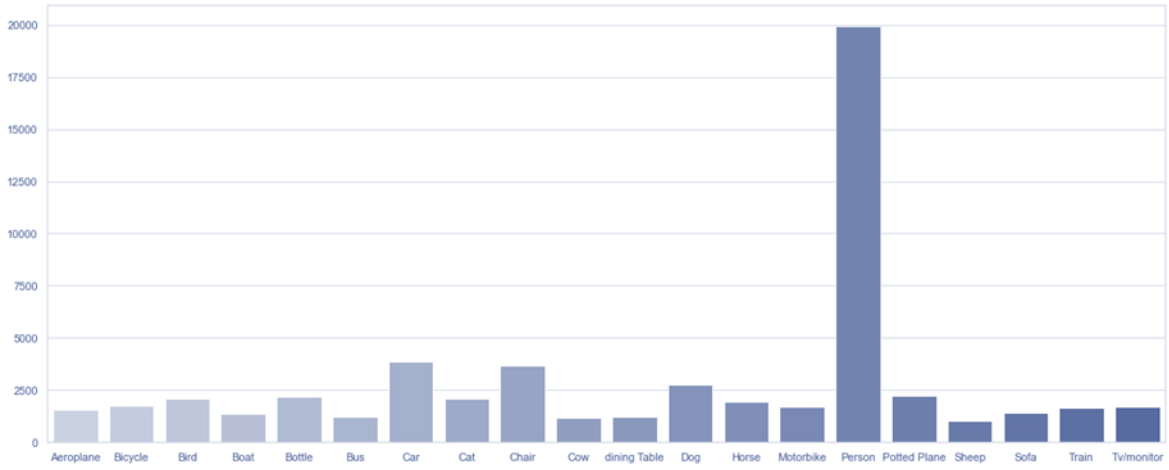


FIGURE 4.4: Classes distribution for PASCAL VOC dataset

4.1.7 PASCAL CONTEXT (Mottaghi et al., 2014)

Pascal Context has the same statistics as the original dataset, with the training set containing 10,103 images and the testing set comprising 9637 images. The list of the classes is present in the Appendix Section.

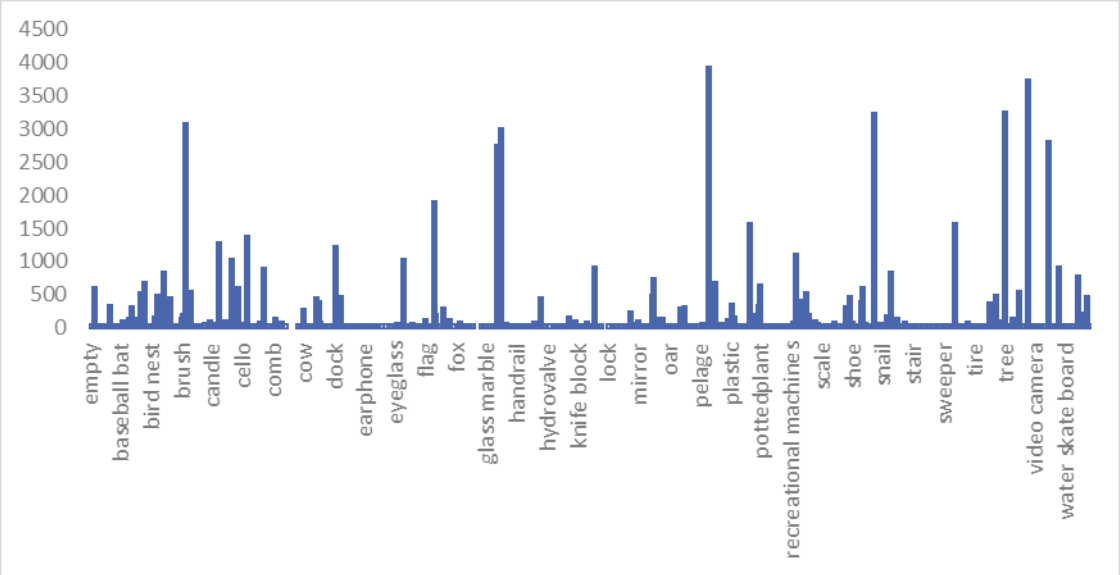


FIGURE 4.5: Classes distribution for PASCAL CONTEXT dataset

4.1.8 CAMVID (Richter et al., 2016)

CAMWID stands for Cambridge-driving Labeled Video Database containing dataset with labels containing 32 classes. There are 367 images in training set, 101 images in validation set, and 233 images in the test set.

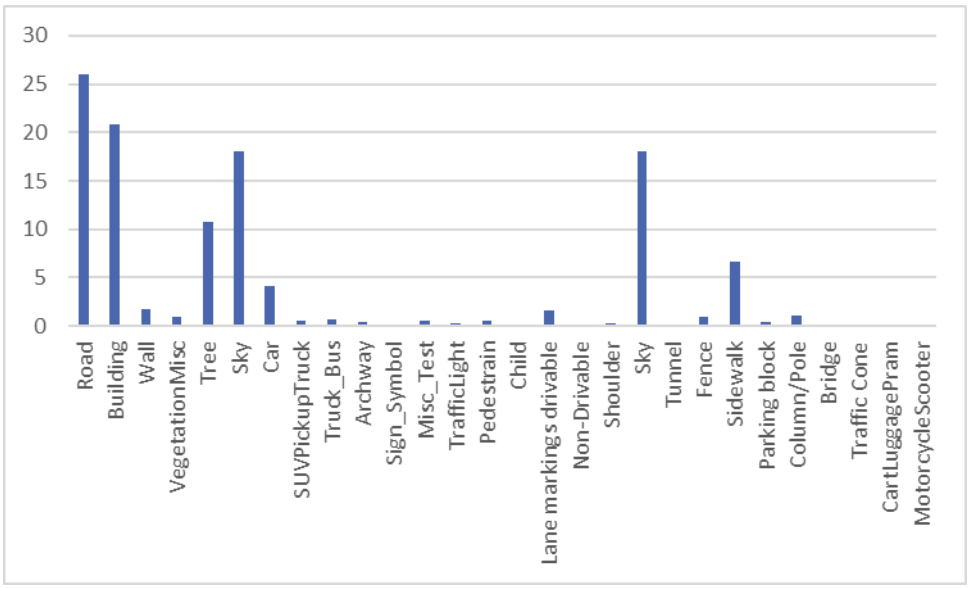


FIGURE 4.6: Classes distribution for CAMVID dataset

Number of classes on the x-axis, Percentage Area on y-axis The classes are as follows:

- Road
- Building
- Wall
- VegetationMisc
- Tree
- Sky
- Car
- SUVPickupTruck
- Truck_Bus
- Archway
- Sign_Symbol
- Misc_Test
- TrafficLight
- Pedestrain
- Child
- Lane markings drivable
- Non-Drivable
- Shoulder
- Sky

- Tunnel
- Fence
- Sidewalk
- Parking block
- Column/Pole
- Bridge
- Traffic Cone
- CartLuggagePram
- MotorcycleScooter

4.1.9 WILDDASH (Zendel et al., 2018)

Wilddash is a widely known dataset containing semantic labels for the purpose of testing the robustness of already trained models. The labels are in accordance with the CityScapes Dataset.

4.1.10 KITTI (Dai et al., 2017)

KITTI dataset consists of 200 image-label pairs in the training data and 200 image-label pairs in validation data. It has the same class structure as the CityScapes Dataset.

4.1.11 SCANNET-20 (Geyer et al., 2020)

SCANNET is an RGB-D dataset, consisting of over 25000 frames. It provides semantically segmented images for these frames. The semantic labels are divided into the following classes:

- apartment
- bathroom
- bedroom / hotel
- bookstore / library
- classroom
- closet
- conference room
- dining room
- gym
- hallway
- kitchen
- laundry room
- living room / lounge
- lobby
- office
- stairs
- storage / basement / garage
- misc

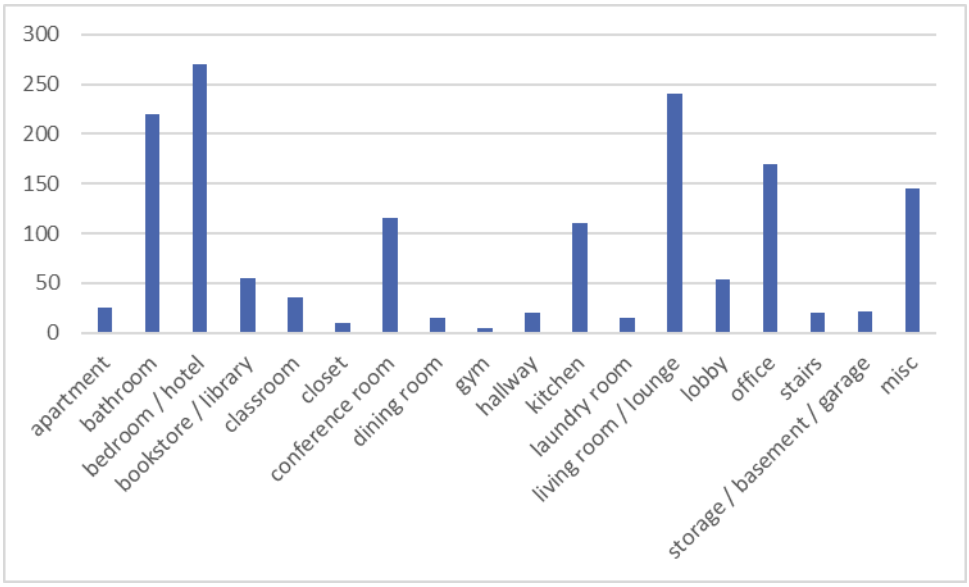


FIGURE 4.7: Classes distribution for SCANNET-20 dataset

4.1.12 Indian Driving Dataset

Indian Driving Dataset is a novel dataset that contains images obtained from a front-faced camera. It contains a total of 10,003 images, 6,993 of which are for training, 981 are for validation, and 2,029 for testing purposes.

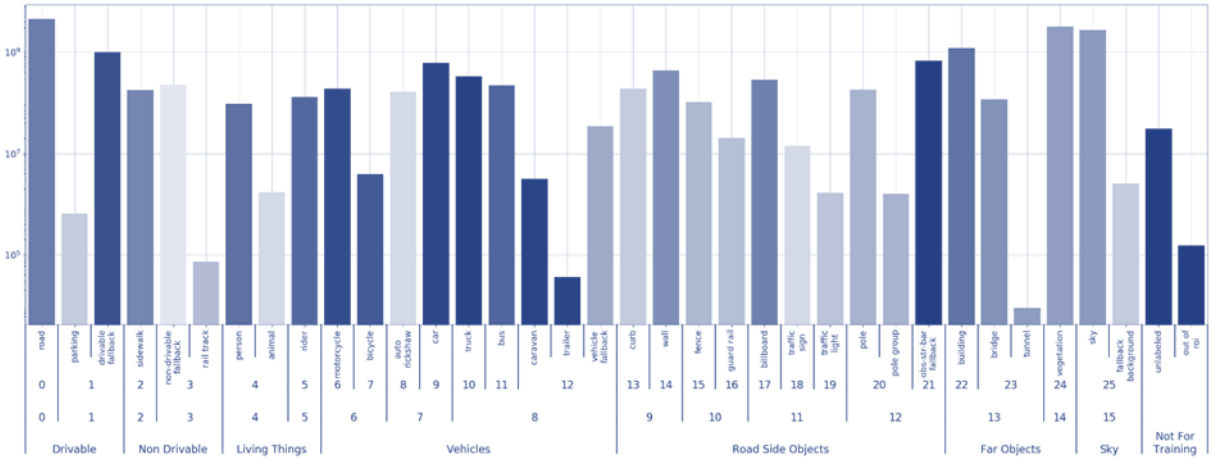


FIGURE 4.8: Classes distribution for Indian Driving Dataset

- road
- parking
- drivable

- fallback
- sidewalk
- non-drivable
- person
- animal
- rider
- motorcycle
- bicycle
- auto
- rickshaw
- car
- truck
- bus
- caravan
- vehicle
- fallback
- curb
- wall
- fence
- guard
- rail

- billboard
- traffic
- sign
- traffic
- light pole
- obs-str-bar fallback
- building
- bridge
- vegetation
- sky
- fallback
- background

4.1.13 BDD 100K

They provide a fine-grained, semantically segmented masks along with 60,000 car instances. The whole dataset is comprised of 3683 images for training, 500 for validation, and 1500 images for testing. The entire label set contains 40 classes, with the most predominant ones as follows:

- banner
- billboard
- lane divider
- parking

- sign
- pole
- polegroup
- street
- light
- traffic cone
- traffic device
- traffic light
- traffic sign
- sign frame
- person
- rider
- bicycle
- bus
- car
- caravan
- motorcycle
- trailer
- train
- truck

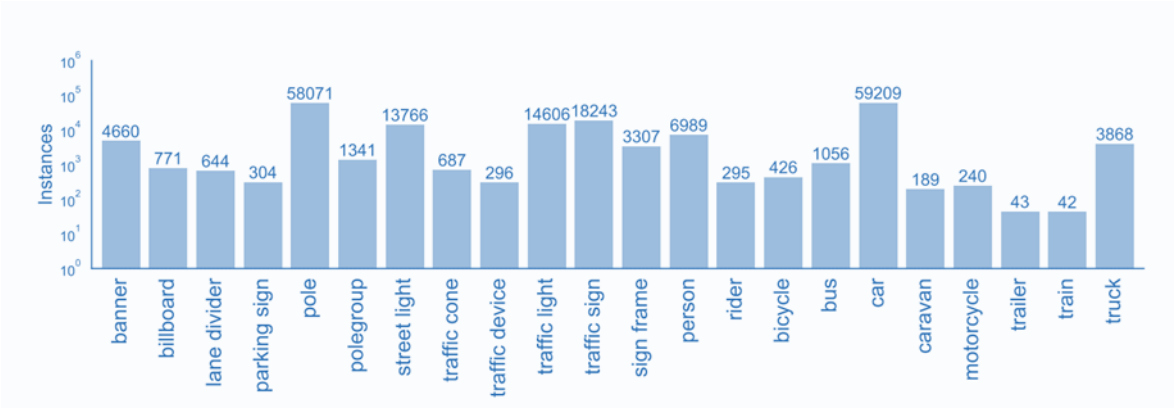


FIGURE 4.9: Classes distribution for BDD-100K

4.2 Dataset summary

In order to implement a domain generalized strategy, we took the classes from each of our training datasets and mapped them to a unified taxonomy.

4.2.1 Training and Testing Datasets

Our composite dataset is an amalgamation of fourteen datasets. Seven of them are used during training, while the remaining seven of them are used in the testing phase. The training datasets are as follows:

1. CityScapes
2. SUNGRBD
3. ADE20K
4. MAPILLARY
5. IDD
6. BDD
7. COCO and COCO Stuff

Whereas the testing datasets are as follows:

1. Pascal VOC
2. Pascal Context
3. Camvid
4. WildDash
5. KITTI

6. Scannet
7. A2D2

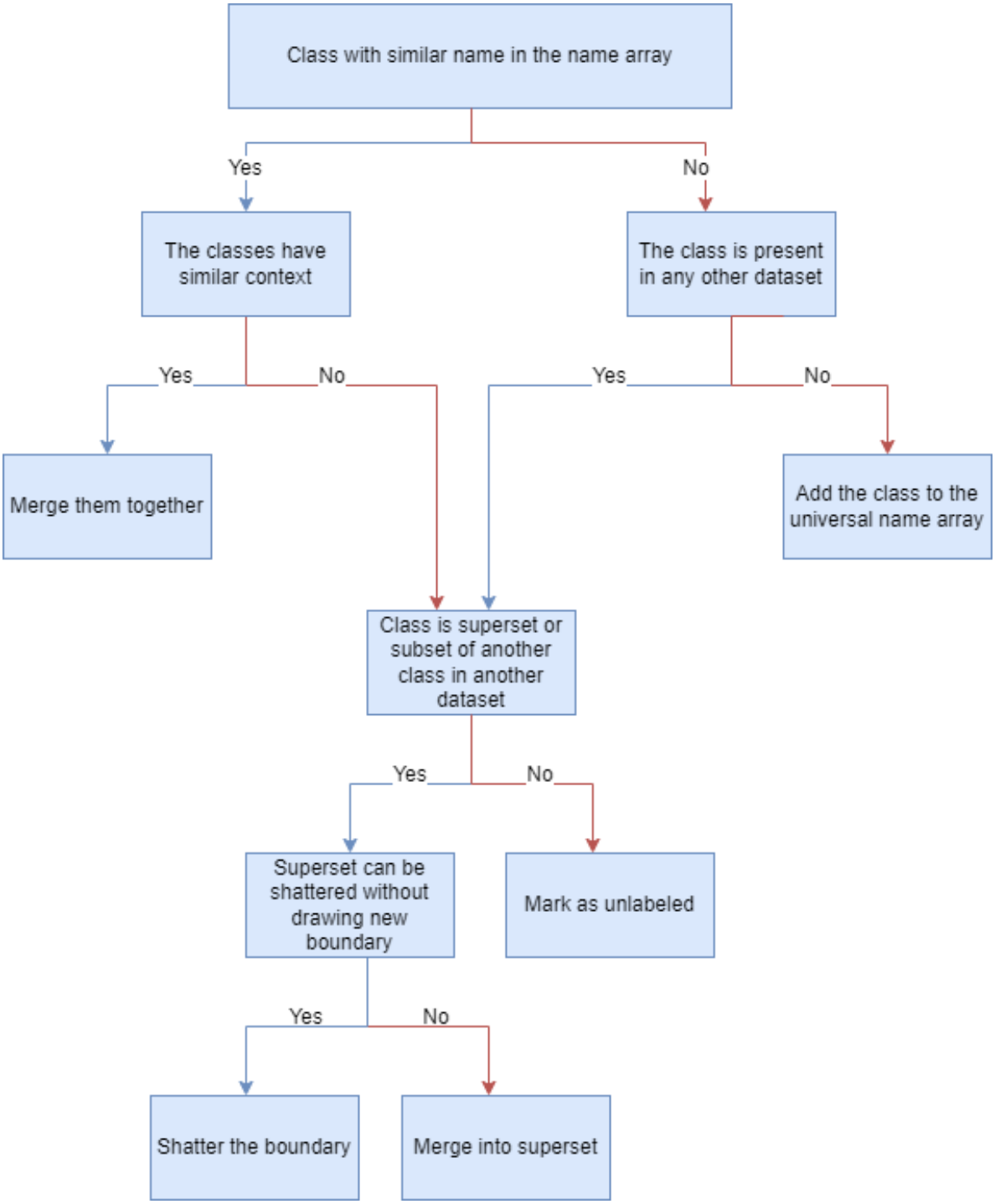
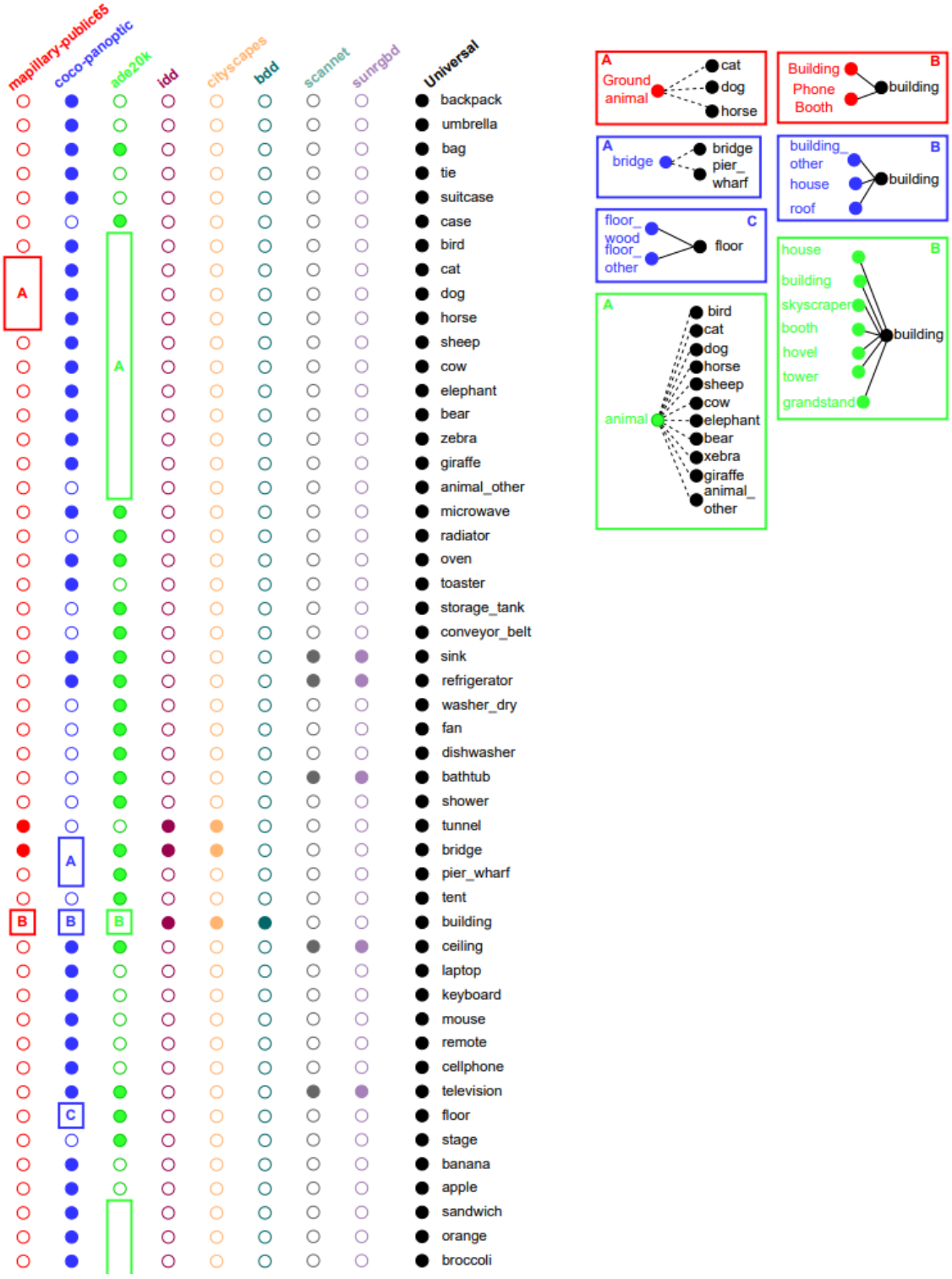
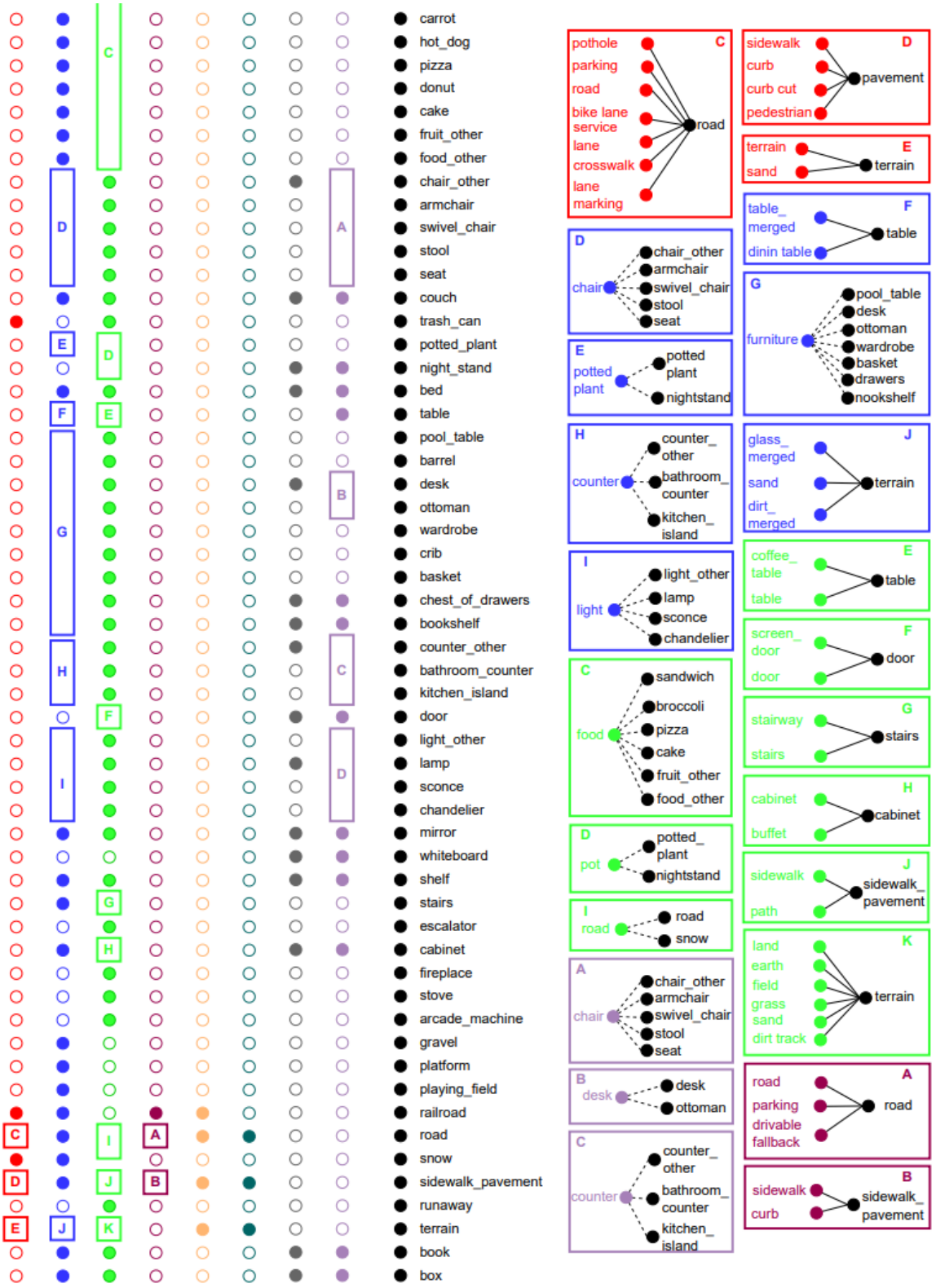
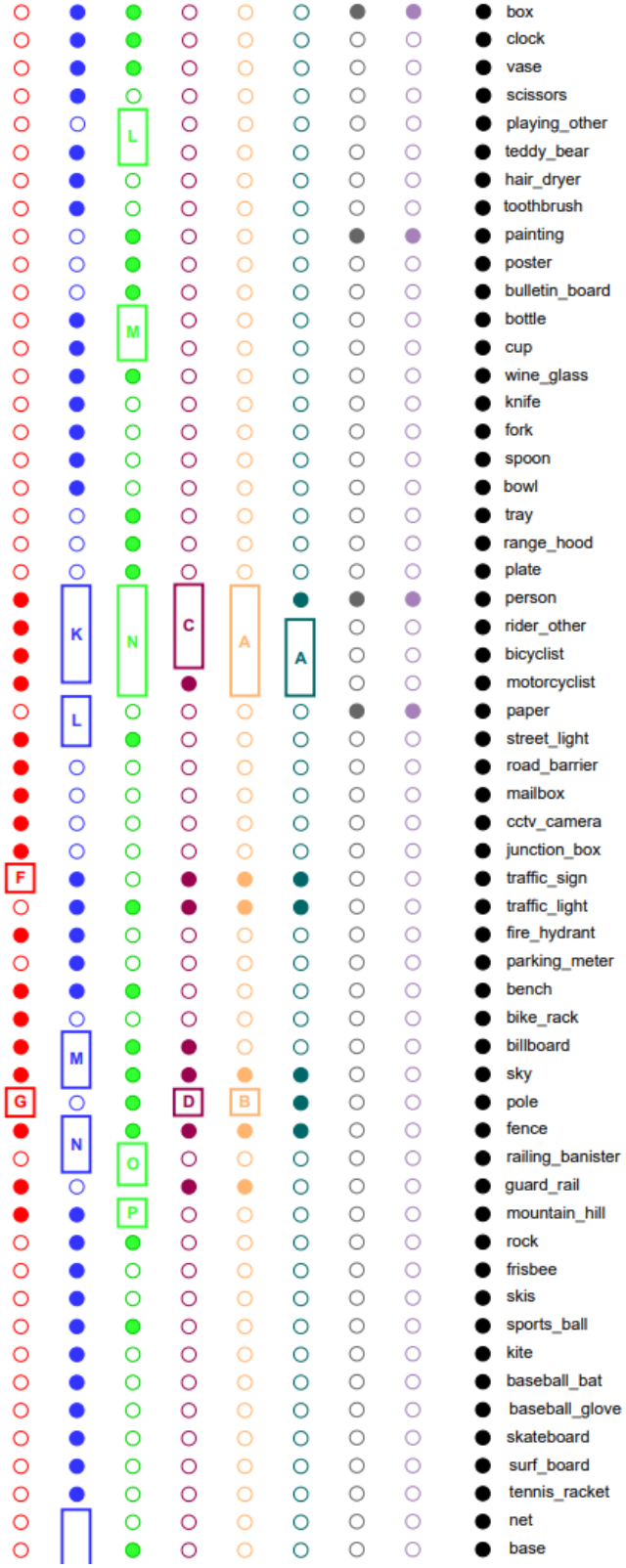


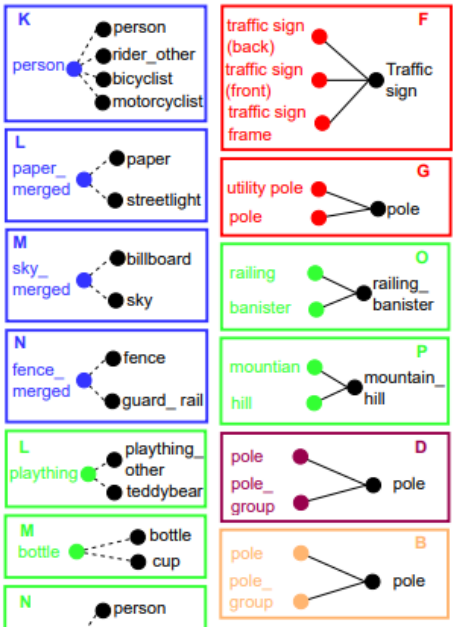
FIGURE 4.10: A Flowchart Depicting the Methodology to Merge, Shatter, or Create New Classes

By using the aforementioned methodology, we came up with the following results.




carrot
hot_dog
pizza
donut
cake
fruit_other
food_other
chair_other
armchair
swivel_chair
stool
seat
couch
trash_can
potted_plant
night_stand
bed
table
pool_table
barrel
desk
ottoman
wardrobe
crib
basket
chest_of_drawers
bookshelf
counter_other
bathroom_counter
kitchen_island
door
light_other
lamp
sconce
chandelier
mirror
whiteboard
shelf
stairs
escalator
cabinet
fireplace
stove
arcade_machine
gravel
platform
playing_field
railroad
road
snow
sidewalk_pavement
runaway
terrain
book
box
pothole
parking
road
bike lane
service lane
crosswalk
lane marking
sidewalk
curb
curb cut
pedestrian
pavement
terrain
sand
table_ merged
dinin table
table
chair_other
armchair
swivel_chair
stool
seat
chair
pool_table
desk
ottoman
wardrobe
basket
drawers
nookshelf
furniture
potted plant
nightstand
counter_ other
bathroom_ counter
kitchen_ island
counter
glass_ merged
sand
dirt_ merged
light_other
lamp
sconce
chandelier
light
coffee_ table
screen_ door
door
sandwich
broccoli
pizza
cake
fruit_other
food_other
food
stairway
stairs
potted_ plant
pot
cabinet
buffet
road
snow
sidewalk
path
sidewalk_ pavement
land
earth
field
grass
sand
dirt track
road
parking
drivable fallback
desk
ottoman
sidewalk
curb


box
clock
vase
scissors
playing_other
teddy_bear
hair_dryer
toothbrush
painting
poster
bulletin_board
bottle
cup
wine_glass
knife
fork
spoon
bowl
tray
range_hood
plate
person
rider_other
bicyclist
motorcyclist
paper
street_light
road_barrier
mailbox
cctv_camera
junction_box
traffic_sign
traffic_light
fire_hydrant
parking_meter
bench
bike_rack
billboard
sky
pole
fence
railing_banister
guard_rail
mountain_hill
rock
frisbee
skis
sports_ball
kite
baseball_bat
baseball_glove
skateboard
surf_board
tennis_racket
net
base


K
person
person
rider_other
bicyclist
motorcyclist
L
paper_ merged
paper
streetlight
M
sky_ merged
billboard
sky
N
fence_ merged
fence
guard_ rail
L
plaything
plaything_ other
teddybear
M
bottle
bottle
cup
N
person
person
rider_other
bicyclist
motorcyclist
C
person
person
rider_other
bicyclist
A
person
person
rider_other
bicyclist
motorcyclist
A
person
rider_other
bicyclist
motorcyclist
D
lamp
lamp_other
lamp
sconce
chandelier
F
traffic sign (back)
traffic sign (front)
traffic sign frame
Traffic sign
G
utility pole
pole
pole
O
railing
banister
railing_ banister
P
mountian
hill
mountain_ hill
D
pole
pole_ group
pole
B
pole
pole_ group
pole

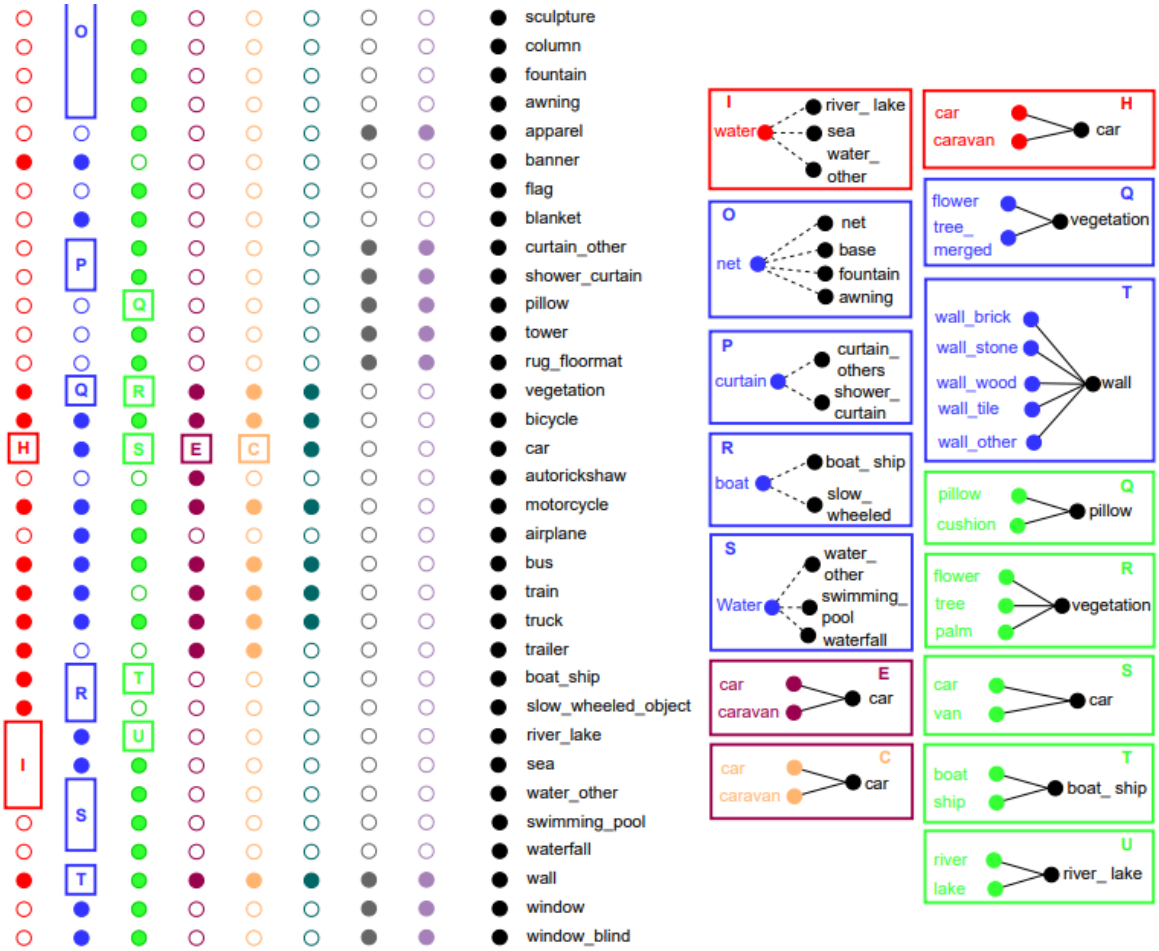


FIGURE 4.11: Our Universal Class and the subsequent classes in the training datasets. The merge and burst operation next to them indicate how these classes have been merged or split into different classes.

4.3 Reason

As stated earlier, the purpose of a composite, unified dataset was to train such a model that can train consistently along all domains. For the said purpose, we have utilized several datasets. When it comes to training, we have eight datasets, namely, CityScapes (Cordts et al., 2016), COCO (Lin, 2014) + COCO STUFF (Caesar, Uijlings, and Ferrari, 2018), ADE20K (Zhou et al., 2017), MAPILLARY dataset (Neuhold et al., 2017), IDD, BDD, SUNRGBD. When it comes to testing, we have PASCAL VOC (Everingham et al., 2010), PASCAL CONTEXT (Mottaghi et al., 2014), CAMVID (Richter et al., 2016),

WILDDASH (Zendel et al., 2018), (Geiger et al., 2013), SCANNET-20 (Dai et al., 2017), A2D2.

TABLE 4.1: The table depicts the datasets required for training of our model, the purpose for why they are used, and the number of images they contain.

Dataset	Place and Purpose	Number of Images
CityScapes	Driving/Germany	3,475
SUNRGBD	Indoor	5,285
ADE20K	Everyday Stuff	22,210
MAPILLARY	Driving/Worldwide	20,000
IDD	Driving/India	7,974
BDD	Driving/United States	8,000
COCO and COCO Stuff	Everyday Stuff	123,287

TABLE 4.2: The table depicts the datasets required for testing of our model, the purpose for why they are used, and the number of images they contain.

Dataset	Place and Purpose	Number of Images
PASCAL VOC	Everyday Stuff	1,449
PASCAL CONTEXT	Everyday Stuff	5,105
CAMVID	Driving/U.K.	101
WILDDASH	Driving/Worldwide	70
KITTI	Driving/Germany	400
SCANNET-20	Indoor	5,436
A2D2	Driving/Germany	5000

Chapter 5

Proposed methodology

In order to make our model generalized and adaptable across multiples domain, we present a composite dataset that contains 194 distinct labels or classes. Then, we train our model on the state-of-the-art segmentation models like UNet and FCN. This is done to justify our usage for using Generative Adversarial Networks like pix2pix. In order to further modify our methodology, we propose a Modified pix2pix, with an attention mechanism.

5.1 pix2pix

Pix2pix is a generative adversarial network (GAN) that is capable of Image-to-Image Translation. It was introduced in 2016 and has produced state of the art results in the field on image-to-image translation.

5.1.1 High Level Architecture

Pix2pix is a conditional GAN. Conditional GANs make use of images that are conditionally generated using a generator. Traditionally, GANs rely on generators to generate images while discriminators learn to differentiate between the synthetic images from the original ones. When using conditional GANs, they need to make use of some conditional information such as class labels, masks, e.t.c. Consequently, our model learns from the mappings from

the given inputs and the targets. Using conditional GANs, we have the following advantages:

- Convergence is achieved really fast. This is because the generated outputs will also have some pattern.
- It allows us to control the output according to the given input at test time.

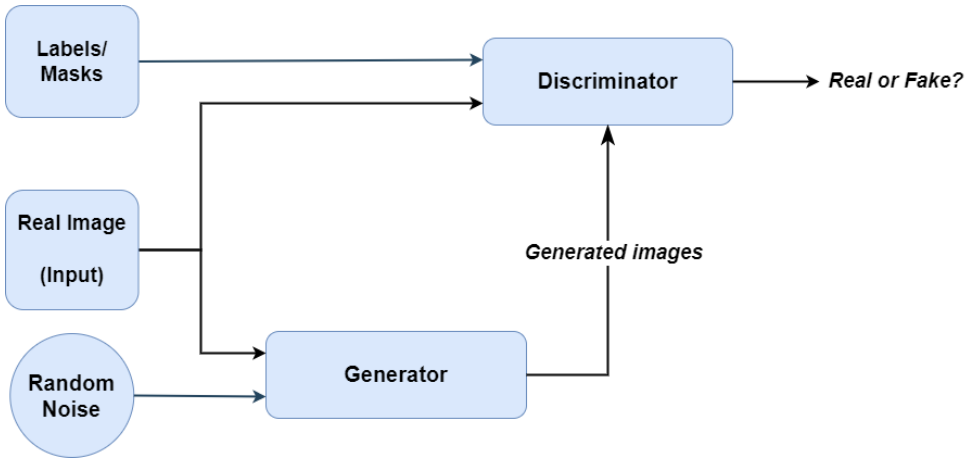


FIGURE 5.1: Using Generator, images are generated. The discriminator then compares the generated images to the target images/labels and determines whether it is real or fake.

When it comes to pix2pix, it uses a modified U-net architecture in its generator, while there is a PatchGAN inside the discriminator. For its input, it takes in 256*256*3 images with the 256*256*3 targeted real outputs. The generated output in the form of 256*256*3 image is fed to the PatchGAN, which classifies whether it is real or fake. The high-level architecture of this conditional GAN looks like as follows:

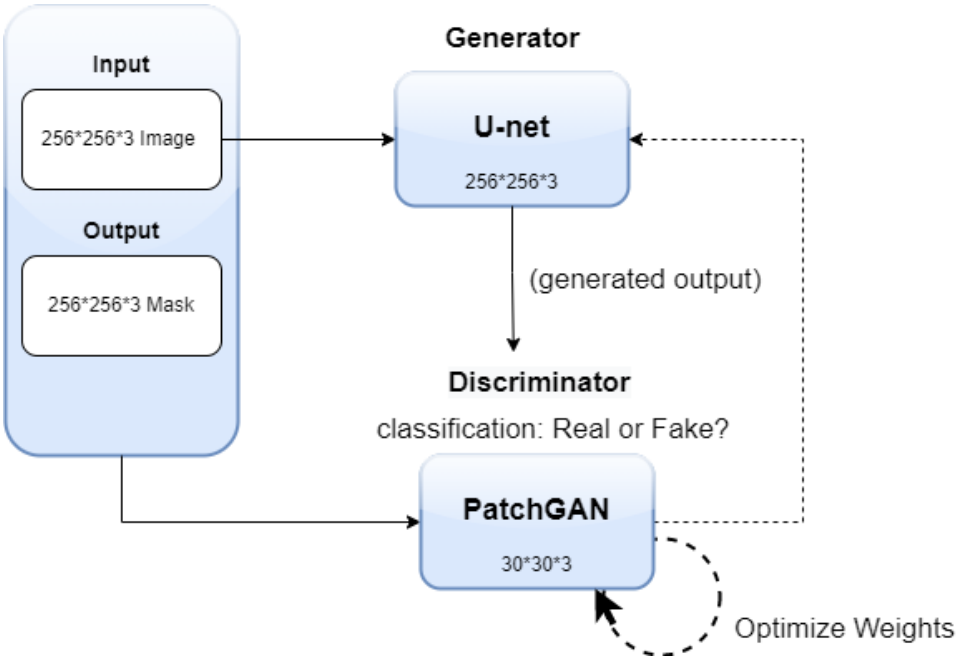


FIGURE 5.2: A pix2pix architecture containing a Generator, which is a modified U-Net and a Discriminator, which is a 30*30 PatchGAN.

Fig. A pix2pix architecture containing a Generator, which is a modified U-Net and a Discriminator, which is a 30*30 PatchGAN.

5.1.2 Generator

Pix2pix uses U-Net as its generator. It comes under the category of binarization architecture that do the pixel level classification. At its base, it utilizes an encoder-decoder architecture, while adding skip connections in downsampled and their corresponding upsampled layers.

5.1.2.1 U-Net Architecture

It is composed of three basic blocks:

- Convolutional Block: This block consists of two convolutional layers with a 3 by 3 kernel of stride 1. This is followed by an activation of ReLU activation and then a dropout layer that has a rate of 0.5.

- Downsampling Block: This block comprises of a single convolutional block which is then followed by a layer that does max-pooling with a stride of 2 to achieve downsampling of the input.

- Upsampling Block: This upsampling block is used to upsample the downsampled layers. In order to do that, there is a layer of transposed convolution having a stride of 2 and a kernel has a size of 3*3. This is then followed by a concatenation with the output from the convoluted block from the symmetrically similar downsampling layer, which is also called a skip connection. Thereafter, there is a convolution block.

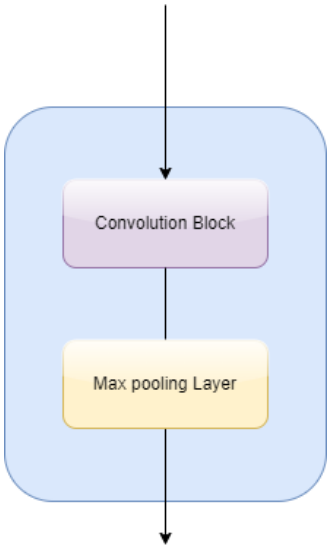


FIGURE 5.3: A Schematic Diagram of Downsampling Block

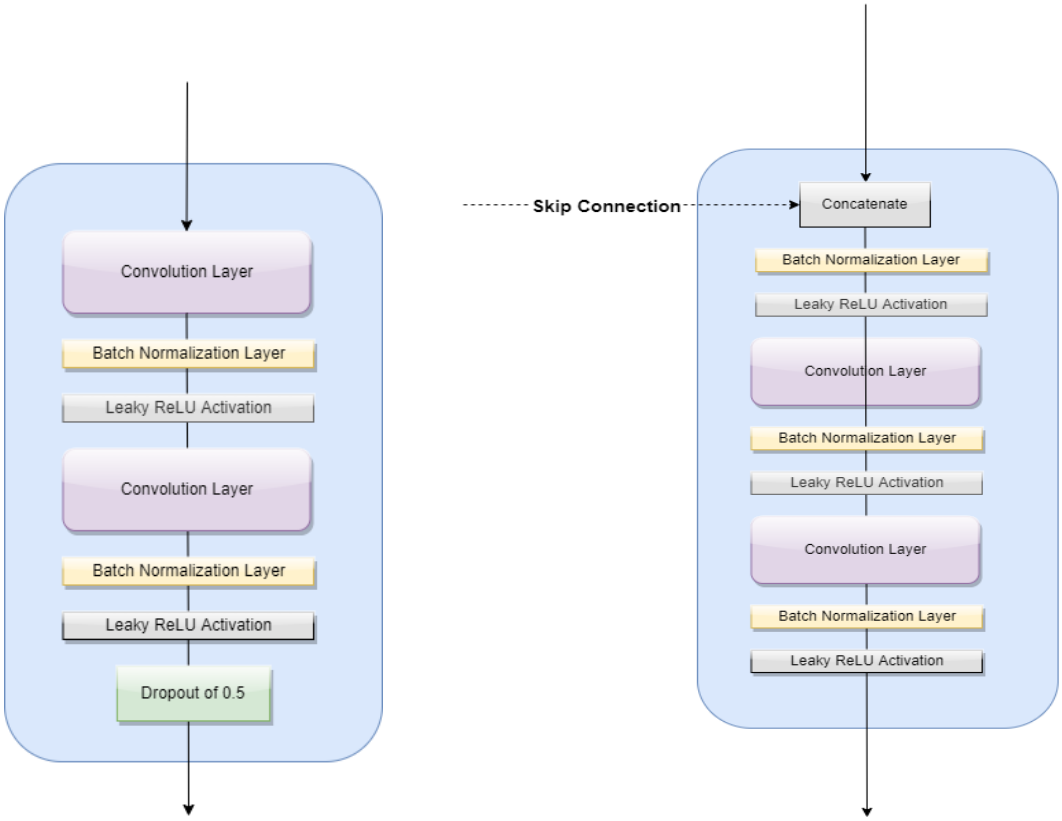


FIGURE 5.4: A schematic diagram of upsampling block on the left. A schematic diagram depicting a convolutional block on the right.

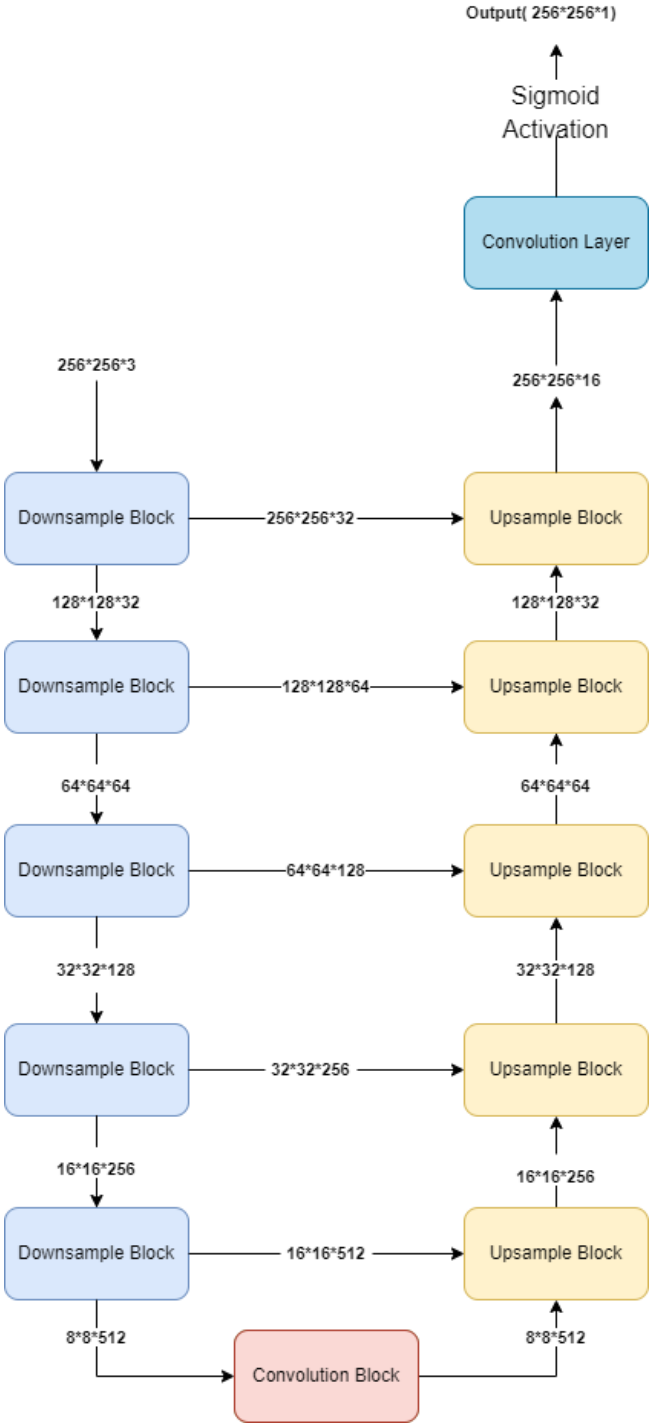


FIGURE 5.5: Flowchart showing association between the Downsampling block, Upsampling block and Convolution block

As discussed earlier, pix2pix utilizes a slightly modified version of U-Net. There are some key differences, which can be summarized as follows:

Pix2pix preserves the original image size and its dimensions. While downsampling, the use of only one convolutional layer block is used instead of the original two blocks. In U-Net, the network was scaled down to 32*32, whereas using pix2pix GAN, we scale it down all the way to 1*1. Here is how we have defined our upsampling block:

```
def upsample(filters, size, apply_dropout=False):
  initializer = tf.random_normal_initializer(0., 0.02)

  result = tf.keras.Sequential()
  result.add(
    tf.keras.layers.Conv2DTranspose(filters, size, strides=2,
                                    padding='same',
                                    kernel_initializer=initializer,
                                    use_bias=False))

  result.add(tf.keras.layers.BatchNormalization())

  if apply_dropout:
      result.add(tf.keras.layers.Dropout(0.5))

  result.add(tf.keras.layers.ReLU())

  return result
```

FIGURE 5.6: Code snippet showing the implementation of Upsampling block

Here is how we have defined our downsampling block:

```
def downsample(filters, size, apply_batchnorm=True):
  initializer = tf.random_normal_initializer(0., 0.02)

  result = tf.keras.Sequential()
  result.add(
      tf.keras.layers.Conv2D(filters, size, strides=2, padding='same',
                             kernel_initializer=initializer, use_bias=False))

  if apply_batchnorm:
    result.add(tf.keras.layers.BatchNormalization())

  result.add(tf.keras.layers.LeakyReLU())

  return result
```

FIGURE 5.7: Code snippet showing the implementation of Downsampling block

To define our overall generator, we have devised the following code:

```
def Generator():
  inputs = tf.keras.layers.Input(shape=[256, 256, 3])

  down_stack = [
    downsample(64, 4, apply_batchnorm=False),  # (batch_size, 128, 128, 64)
    downsample(128, 4),  # (batch_size, 64, 64, 128)
    downsample(256, 4),  # (batch_size, 32, 32, 256)
    downsample(512, 4),  # (batch_size, 16, 16, 512)
    downsample(512, 4),  # (batch_size, 8, 8, 512)
    downsample(512, 4),  # (batch_size, 4, 4, 512)
    downsample(512, 4),  # (batch_size, 2, 2, 512)
    downsample(512, 4),  # (batch_size, 1, 1, 512)
  ]

  up_stack = [
    upsample(512, 4, apply_dropout=True),  # (batch_size, 2, 2, 1024)
    upsample(512, 4, apply_dropout=True),  # (batch_size, 4, 4, 1024)
    upsample(512, 4, apply_dropout=True),  # (batch_size, 8, 8, 1024)
    upsample(512, 4),  # (batch_size, 16, 16, 1024)
    upsample(256, 4),  # (batch_size, 32, 32, 512)
    upsample(128, 4),  # (batch_size, 64, 64, 256)
    upsample(64, 4),  # (batch_size, 128, 128, 128)
  ]

  initializer = tf.random_normal_initializer(0., 0.02)
  last = tf.keras.layers.Conv2DTranspose(OUTPUT_CHANNELS, 4,
                                         strides=2,
                                         padding='same',
                                         kernel_initializer=initializer,
                                         activation='tanh')  # (batch_size, 256, 256, 3)

  x = inputs

  # Downsampling through the model
  skips = []
  for down in down_stack:
    x = down(x)
    skips.append(x)

  skips = reversed(skips[:-1])

  # Upsampling and establishing the skip connections
  for up, skip in zip(up_stack, skips):
    x = up(x)
    x = tf.keras.layers.Concatenate()([x, skip])

  x = last(x)

  return tf.keras.Model(inputs=inputs, outputs=x)
```

FIGURE 5.8: Code snippet showing the implementation of our generator

5.1.3 Discriminator

In GANs, generators are held responsible against an adversary: the discriminator. Discriminator's main job is to classify if the image is real or fake, and the weights are then optimized accordingly. In pix2pix, a discriminator named Patch GAN is used. It is a deep-CNN that utilizes both the mask and image as an input and subsequently produces an output of 30*30*1.

```
def Discriminator():
  initializer = tf.random_normal_initializer(0., 0.02)

  inp = tf.keras.layers.Input(shape=[256, 256, 3], name='input_image')
  tar = tf.keras.layers.Input(shape=[256, 256, 3], name='target_image')

  x = tf.keras.layers.concatenate([inp, tar])  # (batch_size, 256, 256, channels*2)

  down1 = downsample(64, 4, False)(x)  # (batch_size, 128, 128, 64)
  down2 = downsample(128, 4)(down1)  # (batch_size, 64, 64, 128)
  down3 = downsample(256, 4)(down2)  # (batch_size, 32, 32, 256)

  zero_pad1 = tf.keras.layers.ZeroPadding2D()(down3)  # (batch_size, 34, 34, 256)
  conv = tf.keras.layers.Conv2D(512, 4, strides=1,
                                kernel_initializer=initializer,
                                use_bias=False)(zero_pad1)  # (batch_size, 31, 31, 512)

  batchnorm1 = tf.keras.layers.BatchNormalization()(conv)

  leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)

  zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)  # (batch_size, 33, 33, 512)

  last = tf.keras.layers.Conv2D(1, 4, strides=1,
                                kernel_initializer=initializer)(zero_pad2)  # (batch_size, 30, 30, 1)

  return tf.keras.Model(inputs=[inp, tar], outputs=last)
```

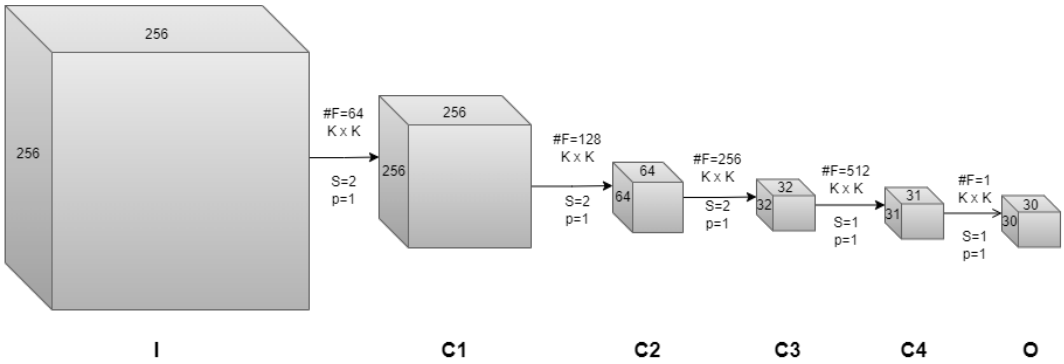


FIGURE 5.9: A 256 x 256 x 1 output from generated passes through four convoluted operations to form 30 x 30 tensors which are evaluated to be either fake or real.

5.2 Modified Pix2Pix

However, we have modified the pix2pix architecture to add attention to our network. This further reinforces the model to converge at an early stage. In order to do that, we have added the attention mechanism to the generator. We have made the following amendments to the code:

```
def attention(tensor, att_tensor, n_filters=512, kernel_size=[1, 1]):
    g1 = Conv2D(n_filters, kernel_size=kernel_size)(tensor)
    x1 = Conv2D(n_filters, kernel_size=kernel_size)(att_tensor)
    net = add(g1, x1)
    net = tf.nn.relu(net)
    net = Conv2D(1, kernel_size=kernel_size)(net)
    net = tf.nn.sigmoid(net)
    #net = tf.concat([att_tensor, net], axis=-1)
    net = net * att_tensor
    return net
```

FIGURE 5.10: Code snippet showing the implementation for attention mechanism to the generator

```
def decoder_block(layer_in, skip_in, n_filters, dropout=True):
    # weight initialization
    init = RandomNormal(stddev=0.02)
    # add upsampling layer
    g = Conv2DTranspose(n_filters, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(layer_in)
    # add batch normalization
    g = BatchNormalization()(g, training=True)
    # conditionally add dropout
    if dropout:
        g = Dropout(0.5)(g, training=True)
    else:
        g= attention(skip_in, g, n_filters)
    # merge with skip connection
    g = Concatenate()([g, skip_in])
    # relu activation
    g = Activation('relu')(g)
    return g
```

FIGURE 5.11: Code snippet showing the implementation of a decoder block

The attention is added to the U-Net architecture to highlight the salient features visible through skipping connections. Before concatenating the relevant activations, coarse scale information is generated. This information is then used to eliminate the unnecessary information and noisy data available in the skip connections. The neurons available in different layers of the U-Net model are activated using filter data from the attention gates during forward pass as well as during the backward pass. While activating the neurons during the backward pass gradients are generated and down weighted as per their relevancy to the given task. By down weighting the generated gradients, neurons referring to the spatial regions are expected to be effected the most as they belong to the shallower layers. Either way, the convolution parameters are updated using the equation below:

$$\frac{\partial(\hat{x}_i^l)}{\partial\left(\Phi^{l-1}\right)} = \frac{\partial\left(\alpha_i^l\, f(x_i^{l-1};\Phi^{l-1})\right)}{\partial\left(\Phi^{l-1}\right)} = \alpha_i^l \frac{\partial(f(x_i^{l-1};\Phi^{l-1}))}{\partial\left(\Phi^{l-1}\right)} + \frac{\partial(\alpha_i^l)}{\partial\left(\Phi^{l-1}\right)} x_i^l$$

FIGURE 5.12: Addition of Attention gates mechanism to the U-Net architecture

The above equations formulates the parameter updating of weights associated to the neurons available in the shallow layers of U-net model. The term ∂ depicts the first ever gradient value. When we talk about multi-dimensional Attention Gates mechanism, the first ever gradient value belongs to the vectors representing each grid scale. In a multi-dimensional Attention Gates scenario, each sub-attention gates is individually responsible for producing an output to generate skip connections. This output belongs to the complimentary information available during each phase. However, dealing with multi-dimensional attention the trainable parameters need to be reduced in order to reduce the complexity of Attention. This can be done by down sampling the extracted feature maps during the gating phase and be linear transformation. However, the linear transformation done in this case have no spatial support, which means that they have 1x1x1 convolutions. After the linear transformation and down sampling of feature maps, the dimensions of the gating operations is reduced deliberately that helps in reducing its complexity further. Most important thing to note here is that the gating functions do not the includes with low dimensional integrity especially the ones available in the first skip connection. To ensure the working of attention units at all possible scales, it is necessary to semantically differentiate between the selected images belonging to their respective feature maps. This type of technique is referred to as deep-supervision technique that allows the attention units to collect information within a large range and be able to make some dense predictions even with smaller subsets of skip connections.

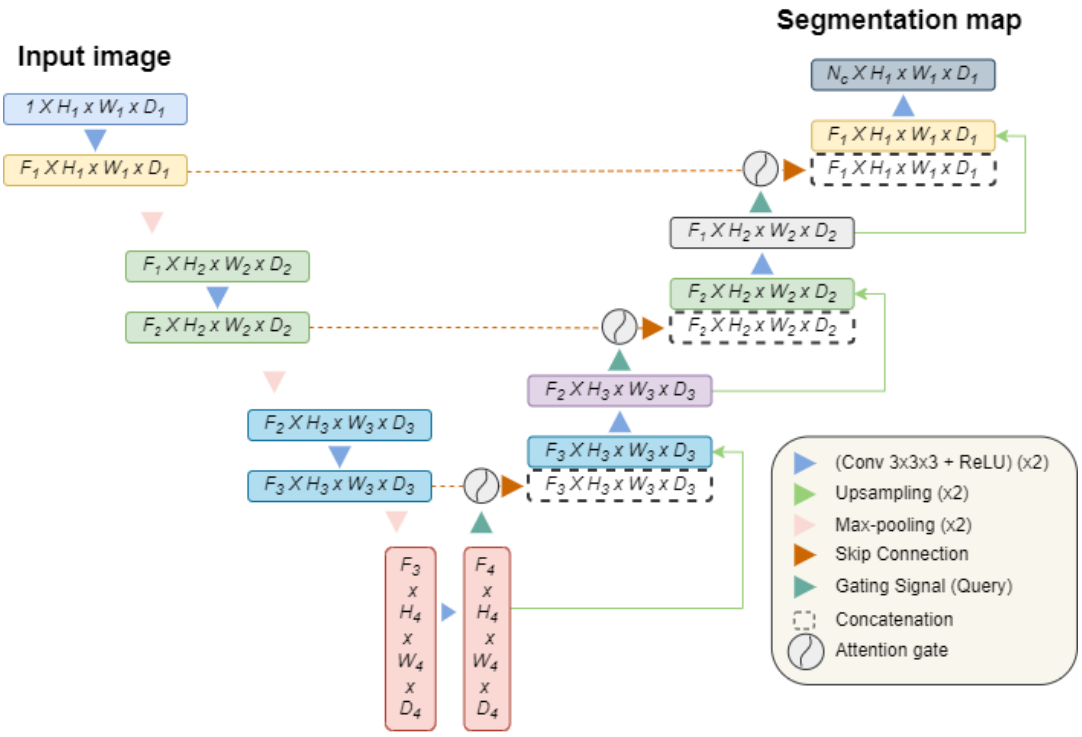


FIGURE 5.13: Adding Attention mechanism to U-Net model

Chapter 6

Implementation

6.1 Importing of Image Data

As the context of the project is huge, encapsulating over 1,60,000 images during the training. In order to make sure that the correct images along with their corresponding masks are being imported, we have a directory containing all the datasets with their corresponding images' names and labels' names. We ensure that each and every image is available with its mask. Here is how we import our dataset:

```
class SData(Dataset):
    def __init__(
        self, split: str = "train", data_root: str = None, data_list: str = None, transform: Optional[Callable] = None
    ) -> None:
        self.split = split
        self.data_list = make_dataset(split, data_root, data_list)
        self.transform = transform
    def __len__(self) -> int:
        return len(self.data_list)
    def get_imagepaths(self):
        image_paths= [i[0] for i in self.data_list]
        return image_paths
    def get_labelpaths(self):
        label_paths= [i[1] for i in self.data_list]
        return label_paths
    @delayed(nout=2)
    def getitem(self, index: int, size=(256,256)) -> Tuple[torch.Tensor, torch.Tensor]:
        image_path, label_path = self.data_list[index]
        image_path,label_path= os.path.abspath(image_path), os.path.abspath(label_path)
        image= tf.io.read_file(image_path)
        image = tf.image.decode_png(image, channels=3)
        image = tf.image.convert_image_dtype(image, tf.float32)
        image = tf.image.resize(image, (256, 256), method='nearest')
        label= tf.io.read_file(label_path)
        label = tf.image.decode_png(label, channels=3)
        label = tf.image.convert_image_dtype(label, tf.float32)
        label = tf.image.resize(label, (256, 256), method='nearest')
        if self.transform is not None:
            if self.split != "test":
                image, label = self.transform(image, label)
            else:
                image, label = self.transform(image, image[:, :, 0])
        return image, label
```

Using the dask library, we make of all the available clients, that is, processors. It parallelizes our code and makes it faster.

6.1.1 Cross Dataset Evaluation Results

In order to prove the efficacy of our technique, we provide a cross dataset evaluation results. In this technique, we use one dataset(on the right) for training. Then, we test the results on each of our testing data. As an evaluation parameter, we use mIoU scores. Then, for the overall performance against each dataset, we have evaluated the harmonic mean of the dataset's values. Similarly, in the last column, we use our composite dataset that inculcates all these datasets under a unified taxonomy. As it can be seen, we have been able to achieve reliable and consistent results across each dataset. The overall results are all mostly better than the ones garnered through training on a single dataset. This solidifies and furthers our claims regarding the validity of our technique.

TABLE 6.1: Cross Dataset Validation using Zero Shot Learning. The dataset on which the model has been trained is on the first column, and the first row represents the dataset on which it has been tested.

Train/Test	**VOC**	**Context**	**CamVid**	**WildDash**	**KITTI**	**ScanNet**	**A2D2**	**h.mean**
COCO	76.9	49.2	52.2	40.4	48.2	32.8	46.8	46.7
ADE20K	40.2	29.1	57.5	29.9	50.8	43.0	31.2	37.8
Mapillary	20.0	15.3	89.2	53.1	67.5	1.1	53.7	6.43
IDD	1.5	3.3	65.5	29.5	30.4	0.9	50.7	3.20
BDD	11.1	9.8	65.0	50.6	51.4	0.8	52.9	4.62
Cityscapes	12.1	6.5	71.3	34.5	58.1	1.7	62.3	7.77
SUNRGBD	9.2	3.1	0.6	1.2	0.6	40.0	5.7	1.46
Our Data	73.6	55.2	89.6	61.8	67.7	58.2	68.8	66.3

6.2 Results of pix2pix vs modified pix2pix

TABLE 6.2: To evaluate our model's performance, we made use of mIoU and Dice Coefficient Scores. The top row depicts whether the model used is pix2pix or its modified version.

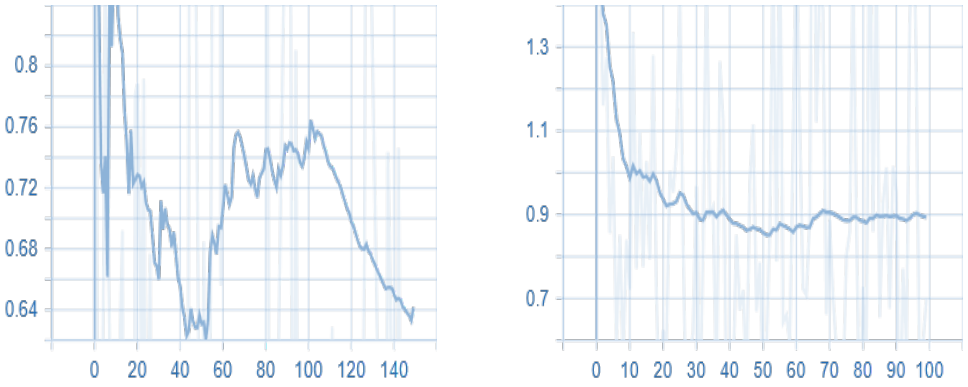
Performance Metrics	pix2pix	Modified pix2pix
mIoU	66.3	71.9
Dice Coefficient	68.9	79.9

6.3 Result comparison of GAN Loss, Discriminator Loss, and Generator L1 Loss

6.3.1 Baseline Paper and the subsequent comparison of our model

6.3.1.1 10,000 images with 1,000,000 steps

1. **Discriminator Loss**

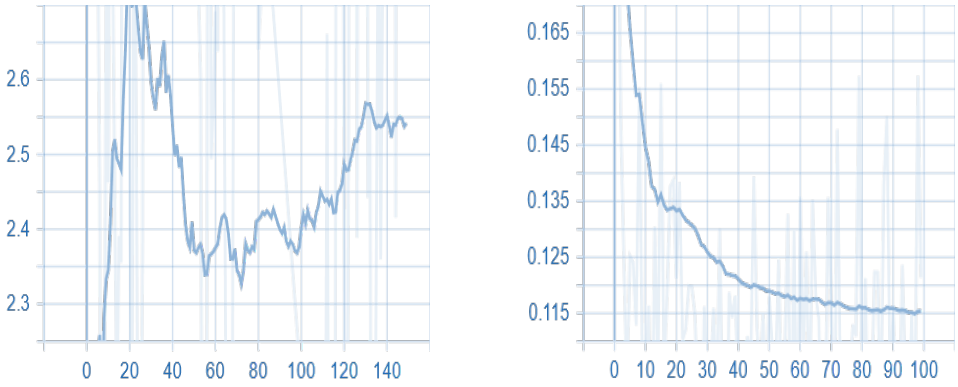


(A) Baseline pix2pix paper

(B) Our results

FIGURE 6.1: Baseline Paper and the subsequent comparison of our model for Discriminator loss

2. **Generator Loss**

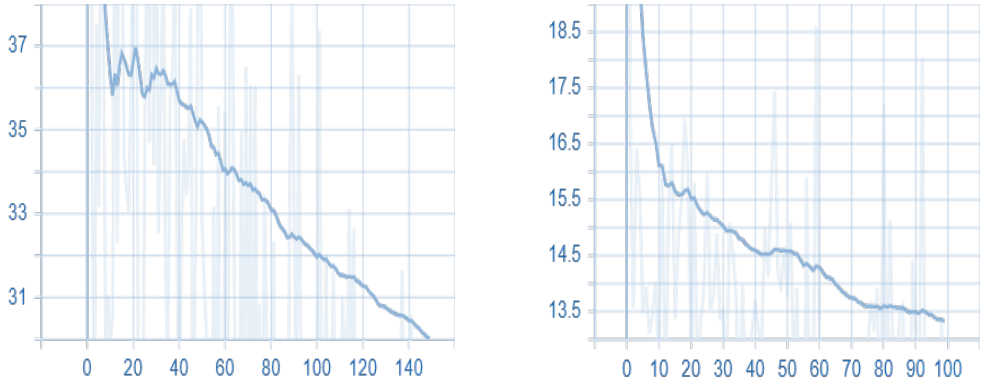


(A) Baseline pix2pix paper

(B) Our results

FIGURE 6.2: Baseline Paper and the subsequent comparison of our model for Generator loss

3. **Generator GAN Loss**



(A) Baseline pix2pix paper

(B) Our results

FIGURE 6.3: Baseline Paper and the subsequent comparison of our model for Generator GAN loss

6.3.1.2 Performance on entire dataset

1. **Discriminator Loss**

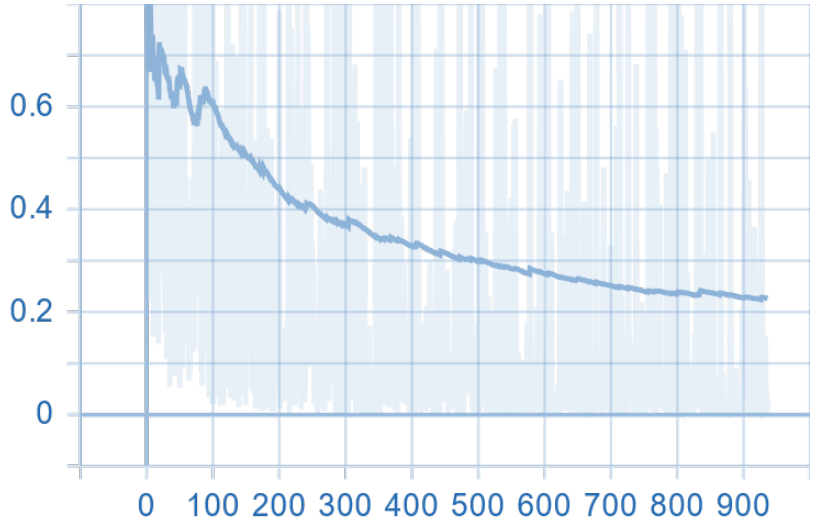


FIGURE 6.4: A graph showing the Discriminator loss for the entire dataset

2. **Generator Loss**

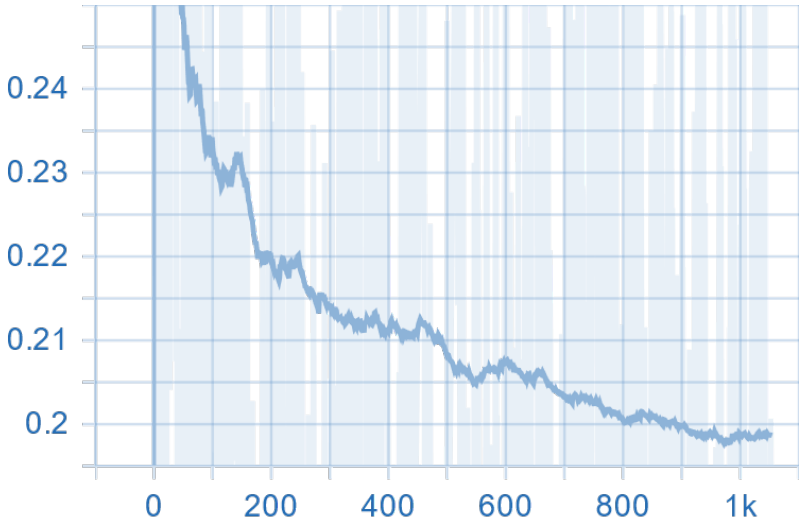


FIGURE 6.5: A graph showing the Generator loss for the entire dataset

3. **Total GAN Loss**

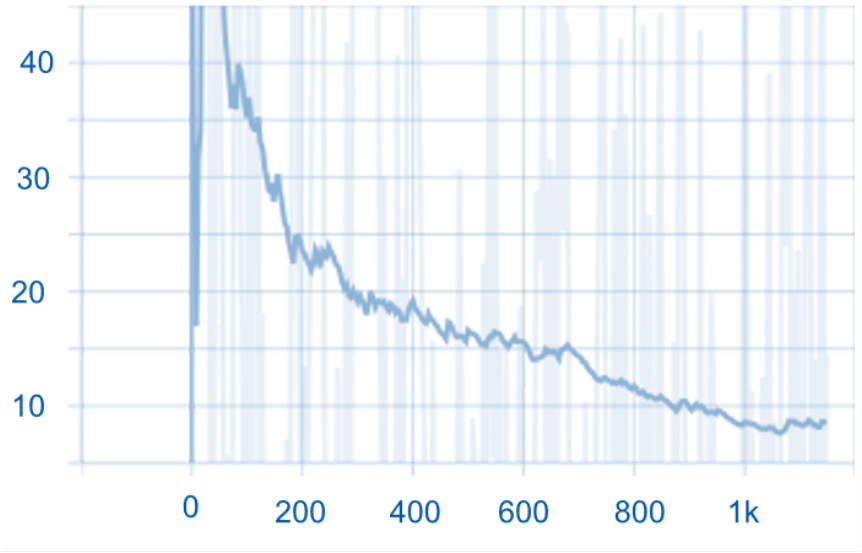


FIGURE 6.6: A graph showing the total GAN loss for the entire dataset

6.4 Qualitative Results

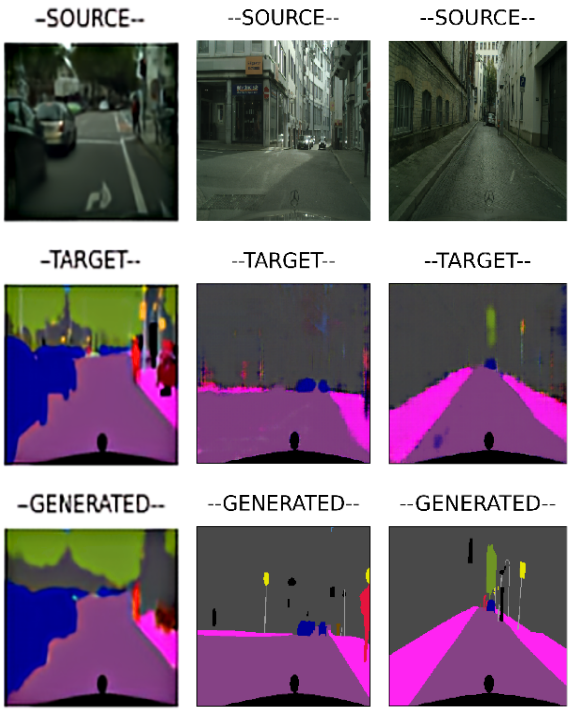


FIGURE 6.7: Comparison between the qualitative results produced in the baseline paper (left), after implementing Pix2pix (center), and after implementing Modified Pix2pix (right)

Chapter 7

Conclusion

This work answered the following questions:

- How to relabel and remap the dataset into unified taxonomy.
- How to semantically segment an image using GANs. This includes pix2pix and our modified pix2pix.
- Cross Dataset Evaluation and performance comparisons with baseline pix2pix paper.

Our main aim of this project was to propose a composite model that can classify a wide array of images in different taxonomies. For the same reason, we used such datasets belonging to different world areas. Using these datasets, we created a composite dataset containing 194 classes.

After successfully creating this dataset, we made a list that contains lists of these individual datasets and has the images' lists with their corresponding masks.

Afterward, we implemented pix2pix and achieved state-of-the-art results with our unified dataset. To prove the efficacy of our dataset, we utilized the method of cross-evaluation of the dataset, where we trained the model individually on each training dataset and tested it against each of our testing datasets. Then, we trained on our composite dataset and tested it against all the individual testing datasets. The model performed consistently better across all the different testing datasets.

There onwards, we proposed a modified pix2pix which adds an attention mechanism to the generator of our c-GAN. This significantly improved the mIoU scores by approximately 8 percent and the dice coefficient by approximately 16 percent.

Appendix A

A.1 Analysis of Classes in PASCAL VOC

Class Name	# of Images	Average Area (% of image)
empty	0	
accordion	2	5.26%
aeroplane	597	17.41%
air conditioner	5	2.68%
antenna	2	0.80%
artillery	1	3.36%
ashtray	18	1.11%
atrium	1	13.64%
baby carriage	13	12.28%
bag	319	3.18%
ball	25	2.12%
balloon	18	2.88%
bamboo weaving	1	12.78%
barrel	3	3.58%
baseball bat	1	1.05%
basket	89	5.70%
basketball backboard	6	5.37%

bathtub	5	36.08%
bed	120	28.08%
bedclothes	296	27.01%
beer	1	2.41%
bell	2	1.24%
bench	119	5.49%
bicycle	503	17.20%
binoculars	0	
bird	664	12.36%
bird cage	21	44.05%
bird feeder	14	18.47%
bird nest	6	21.69%
blackboard	2	12.70%
board	137	9.67%
boat	466	16.67%
bone	3	2.55%
book	416	3.80%

bottle	821	7.35%
bottle opener	1	0.61%
bowl	137	2.57%
box	434	5.16%
bracelet	1	0.76%
brick	9	4.38%
bridge	31	10.39%
broom	5	6.75%
brush	120	10.28%
bucket	175	2.73%
building	3064	20.22%
bus	399	32.86%
cabinet	536	13.62%
cabinet door	1	5.84%
cage	29	32.76%
cake	2	8.76%
calculator	1	1.30%

calendar	15	3.30%
camel	0	
camera	52	1.80%
camera lens	0	
can	78	2.24%
candle	43	1.04%
candle holder	29	0.93%
cap	16	3.45%
car	1273	19.45%
card	11	3.41%
cart	11	6.56%
case	91	4.12%
casette recorder	1	1.82%
cash register	1	0.88%
cat	1006	33.30%
cd	45	3.33%
cd player	1	1.35%

ceiling	586	10.00%
cell phone	43	0.49%
cello	4	15.31%
chain	10	1.52%
chair	1357	11.87%
chessboard	2	1.37%
chicken	0	
chopstick	8	1.12%
clip	1	0.47%
clippers	2	9.44%
clock	61	0.99%
closet	1	27.69%
cloth	877	8.96%
clothes tree	0	
coffee	4	0.59%
coffee machine	2	1.41%
comb	3	0.80%

computer	118	8.21%
concrete	9	6.79%
cone	30	1.32%
container	61	4.34%
control booth	0	
controller	3	4.67%
cooker	0	
copying machine	0	
coral	0	
cork	0	
corkscrew	0	
counter	31	11.19%
court	2	14.94%
cow	256	22.98%
crabstick	1	2.44%
crane	12	0.56%
crate	2	0.74%

cross	3	0.23%
crutch	2	0.52%
cup	432	2.46%
curtain	381	11.51%
cushion	75	12.60%
cutting board	3	2.50%
dais	1	10.10%
disc	2	2.49%
disc case	16	4.96%
dishwasher	0	
dock	3	10.26%
dog	1204	25.23%
dolphin	0	
door	455	10.01%
drainer	1	0.09%
dray	1	10.38%
drink dispenser	1	1.92%

equipment	27	7.52%
escalator	0	
exhibition booth	2	17.59%
extinguisher	6	0.95%
eyeglass	15	2.72%
fan	40	2.05%
faucet	4	2.27%
fax machine	1	2.58%
fence	1021	10.79%
ferris wheel	1	10.09%
fire extinguisher	2	0.75%
fire hydrant	2	4.74%
fire place	41	9.53%
fish	7	4.60%
fish tank	3	25.04%
fishbowl	1	18.00%
fishing net	1	7.74%

fishing pole	1	0.21%
flag	100	2.05%
flagstaff	3	1.58%
flame	0	
flashlight	2	0.38%
floor	1895	16.07%
flower	190	7.67%
fly	1	0.43%
foam	0	
food	285	5.16%
footbridge	1	10.82%
forceps	0	
fork	104	0.63%
forklift	0	
fountain	1	1.40%
fox	0	
frame	17	5.86%

fridge	69	10.07%
frog	1	0.70%
fruit	2	0.77%
funnel	1	0.37%
furnace	2	31.60%
game controller	4	0.66%
game machine	3	13.31%
gas cylinder	0	
gas hood	0	
gas stove	0	
gift box	2	10.96%
glass	21	5.24%
glass marble	1	8.14%
globe	0	
glove	6	4.34%
goal	4	10.04%
grandstand	30	18.55%

grass	2742	23.28%
gravestone	0	
ground	2987	20.98%
guardrail	35	6.09%
guitar	53	4.83%
gun	2	1.07%
hammer	1	0.08%
hand cart	11	4.87%
handle	2	4.16%
handrail	7	3.75%
hanger	3	0.66%
hard disk drive	0	
hat	27	4.07%
hay	3	21.08%
headphone	7	0.49%
heater	4	8.10%
helicopter	3	12.77%

helmet	62	2.32%
holder	2	2.23%
hook	2	0.61%
horse	429	22.31%
horse-drawn carriage	25	12.70%
hot-air balloon	1	84.05%
hydrovalve	0	
ice	6	31.35%
inflator pump	0	
ipod	15	0.75%
iron	2	2.15%
ironing board	1	9.63%
jar	18	2.80%
kart	1	16.90%
kettle	1	1.96%
key	4	1.86%
keyboard	140	4.69%

kitchen range	12	7.54%
kite	1	0.26%
knife	92	0.61%
knife block	2	0.30%
ladder	12	3.91%
ladder truck	1	0.84%
ladle	1	2.83%
laptop	75	10.55%
leaves	7	33.39%
lid	1	1.37%
life buoy	2	14.83%
light	908	1.18%
light bulb	1	0.68%
lighter	5	0.21%
line	1	0.14%
lion	0	
lobster	0	

lock	1	0.15%
machine	0	
mailbox	3	0.39%
mannequin	1	11.90%
map	3	32.42%
mask	0	
mat	35	11.11%
match book	1	2.29%
mattress	3	1.72%
menu	24	4.34%
metal	216	4.28%
meter box	1	3.43%
microphone	44	0.71%
microwave	26	3.61%
mirror	81	5.60%
missile	1	5.71%
model	1	27.81%

money	1	6.08%
monkey	1	2.75%
mop	3	0.66%
motorbike	471	24.63%
mountain	728	10.75%
mouse	122	0.43%
mouse pad	50	3.58%
musical instrument	7	11.16%
napkin	120	2.60%
net	3	6.81%
newspaper	30	3.99%
oar	10	5.48%
ornament	2	0.79%
outlet	26	0.48%
oven	8	10.24%
oxygen bottle	1	4.21%
pack	287	3.18%

pan	10	3.51%
paper	298	4.02%
paper box	4	0.99%
paper cutter	6	6.75%
parachute	4	3.45%
parasol	17	4.14%
parterre	1	14.77%
patio	1	23.16%
pelage	0	
pen	44	0.33%
pen container	8	1.14%
pencil	2	0.65%
person	3919	20.64%
photo	3	2.69%
piano	4	8.83%
picture	674	5.70%
pig	1	1.41%

pillar	8	3.16%
pillow	55	11.18%
pipe	16	2.01%
pitcher	3	3.36%
plant	106	14.52%
plastic	11	6.37%
plate	346	4.44%
platform	154	15.39%
player	24	2.37%
playground	1	5.70%
pliers	1	0.39%
plume	1	1.76%
poker	1	0.73%
poker chip	1	7.81%
pole	1555	1.53%
pool table	1	29.81%
postcard	1	1.72%

poster	185	6.82%
pot	310	3.08%
pottedplant	625	11.89%
printer	34	2.92%
projector	2	0.52%
pumpkin	2	7.72%
rabbit	1	0.14%
racket	1	4.75%
radiator	27	3.93%
radio	6	2.22%
rail	31	6.12%
rake	1	0.26%
ramp	2	3.15%
range hood	8	3.98%
receiver	13	1.12%
recorder	2	23.21%
recreational machines	1	61.90%

remote control	67	0.93%
road	1102	21.97%
robot	0	
rock	396	10.72%
rocket	0	
rocking horse	1	0.74%
rope	507	1.35%
rug	181	20.69%
ruler	1	1.61%
runway	2	11.72%
saddle	85	1.95%
sand	46	36.43%
saw	1	1.53%
scale	2	3.64%
scanner	1	1.43%
scissors	5	0.72%
scoop	1	0.04%

screen	4	13.73%
screwdriver	2	0.32%
sculpture	73	7.67%
scythe	0	
sewer	2	26.70%
sewing machine	3	1.06%
shed	2	2.25%
sheep	297	21.95%
shell	3	2.43%
shelves	448	7.64%
shoe	74	2.68%
shopping cart	2	4.54%
shovel	2	1.07%
sidecar	2	15.60%
sidewalk	382	8.65%
sign	592	2.32%
signal light	43	0.98%

sink	24	15.51%
skateboard	1	0.81%
ski	3	5.71%
sky	3231	26.77%
sled	2	4.28%
slippers	1	4.63%
smoke	56	10.42%
snail	0	
snake	3	15.25%
snow	173	28.71%
snowmobiles	1	16.52%
sofa	831	25.31%
spanner	0	
spatula	1	1.36%
speaker	118	3.48%
speed bump	0	
spice container	23	0.99%

spoon	75	0.51%
sprayer	0	
squirrel	1	6.33%
stage	16	49.30%
stair	36	21.55%
stapler	3	0.79%
stick	4	0.28%
sticky note	2	1.64%
stone	0	
stool	9	3.18%
stove	10	2.22%
straw	6	0.47%
stretcher	0	
sun	1	3.99%
sunglass	4	0.86%
sunshade	3	5.83%
surveillance camera	1	0.16%

swan	1	0.22%
sweeper	0	
swim ring	2	9.90%
swimming pool	1	3.01%
swing	1	2.09%
switch	5	0.51%
table	1566	9.76%
tableware	3	4.19%
tank	2	2.63%
tap	12	2.30%
tape	9	0.51%
tarp	2	43.12%
telephone	71	1.72%
telephone booth	0	
tent	36	12.26%
tire	35	3.29%
toaster	5	1.70%

toilet	2	12.68%
tong	1	1.80%
tool	3	3.06%
toothbrush	1	0.80%
towel	3	18.69%
toy	356	5.21%
toy car	2	30.91%
track	356	9.15%
train	465	28.46%
trampoline	1	15.35%
trash bin	91	3.03%
tray	9	5.46%
tree	3232	18.02%
tricycle	0	
tripod	2	8.27%
trophy	2	1.04%
truck	126	6.39%

tube	3	3.82%
turtle	1	0.98%
tvmonitor	536	15.32%
tweezers	0	
typewriter	1	3.40%
umbrella	13	4.32%
unknown	3734	8.99%
vacuum cleaner	6	4.15%
vending machine	2	23.66%
video camera	11	1.94%
video game console	11	1.77%
video player	37	2.97%
video tape	3	4.09%
violin	4	4.37%
wakeboard	0	
wall	2794	22.23%
wallet	7	1.26%

wardrobe	2	18.93%
washing machine	17	15.58%
watch	6	0.81%
water	899	32.68%
water dispenser	5	3.52%
water pipe	3	2.94%
water skate board	1	0.60%
watermelon	2	7.00%
whale	0	
wharf	0	
wheel	8	17.21%
wheelchair	5	3.78%
window	756	11.07%
window blinds	1	1.57%
wineglass	197	2.95%
wire	113	1.17%
wood	462	8.29%

Appendix B

B.1 Submitting the code on GPU

We used the Vienna Scientific Cluster for the execution of our code. In order to use any GPU system with SLURM Workload Manager, the following steps can be taken in the exact sequence for a seamless, hassle-free execution.

1. **salloc** commnd for allocation of GPU
2. **ssh** <node-name>
3. **module purge**
4. **module load** conda/9.1.85
5. **nvcc** (This is to ensure that nvidia drivers are working properly)
6. **conda activate** my-environment (my-environment is your conda environment having the required packages for your python code to run)
7. **python** pix2pix-run.py

Bibliography

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari (2018). "Coco-stuff: Thing and stuff classes in context". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218.

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello (2016). "An analysis of deep neural network models for practical applications". In: *arXiv preprint arXiv:1605.07678*.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289*.

Cordts, Marius et al. (2016). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

Dai, Angela et al. (2017). "Scannet: Richly-annotated 3d reconstructions of indoor scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839.

Everingham, Mark et al. (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Geiger, Andreas et al. (2013). "1Vision meets Robotics: The KITTI Dataset". In.

Geyer, Jakob et al. (2020). "A2d2: Audi autonomous driving dataset". In: *arXiv preprint arXiv:2004.06320*.

He, Kaiming et al. (2017). "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

Heckman, James J (1979). "Sample selection bias as a specification error". In: *Econometrica: Journal of the econometric society*, pp. 153–161.

Hoiem, Derek et al. (2015). "Guest editorial: Scene understanding". In: *International Journal of Computer Vision* 112.2, pp. 131–132.

Lambert, John et al. (2020). "MSeg: A composite dataset for multi-domain semantic segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2879–2888.

Lin, TY (2014). "Microsoft COCO: Common Objects in Context". In: *M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, L*, pp. 740–755.

Martnez-Daz, Margarita and Francesc Soriguera (2018). "Autonomous vehicles: theoretical and practical challenges". In: *Transportation Research Procedia* 33, pp. 275–282.

Mottaghi, Roozbeh et al. (2014). "The role of context for object detection and semantic segmentation in the wild". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898.

Neuhold, Gerhard et al. (2017). "The mapillary vistas dataset for semantic understanding of street scenes". In: *Proceedings of the IEEE international conference on computer vision*, pp. 4990–4999.

Paszke, Adam et al. (2016). "Enet: A deep neural network architecture for real-time semantic segmentation". In: *arXiv preprint arXiv:1606.02147*.

Richter, Stephan R et al. (2016). "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer, pp. 102–118.

Ros, German et al. (2016). "Training constrained deconvolutional networks for road scene semantic segmentation". In: *arXiv preprint arXiv:1604.01545*.

Shimodaira, Hidetoshi (2000). "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *Journal of statistical planning and inference* 90.2, pp. 227–244.

Treml, Michael et al. (2016). "Speeding up semantic segmentation for autonomous driving". In.

Triantafillou, Eleni et al. (2019). "Meta-dataset: A dataset of datasets for learning to learn from few examples". In: *arXiv preprint arXiv:1903.03096*.

Valada, Abhinav et al. (2017). "Adapnet: Adaptive semantic segmentation in adverse environmental conditions". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4644–4651.

Volpi, Riccardo et al. (2018). "Generalizing to unseen domains via adversarial data augmentation". In: *Advances in neural information processing systems* 31.

Zendel, Oliver et al. (2018). "Wilddash-creating hazard-aware benchmarks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 402–416.

Zhou, Bolei et al. (2017). "Scene parsing through ade20k dataset". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641.