Fraunhofer IOSB

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Cross-Domain Fine-Grained Classification

Masterthesis

Hochschule Karlsruhe
Technik und Wirtschaft

Ritu Ahmed

the 31st of August, 2022

Supervisor: Prof. Dipl.-Ing Jürgen Walter
Supervisor at Work: Stefan Wolf M.Sc.

Hochschule Karlsruhe
University of
Applied Sciences

HKA

Fakultät für Maschinenbau und Mechatronik

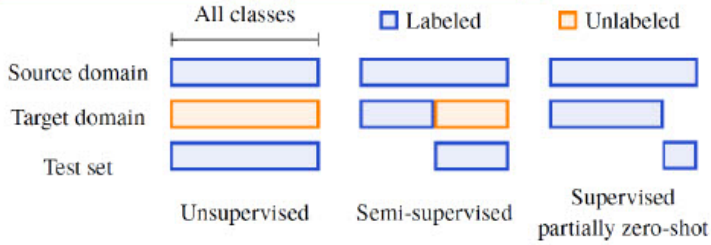
Master-Thesis	**Ritu Ahmed** **Mechatronic and Micro-Mechatronic Systems**
Arbeitsplatz	Fraunhofer IOSB: Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB Fraunhoferstraße 1, 76131 Karlsruhe
Betreuer am Arbeitsplatz	Stefan Wolf M. Sc.
Betreuender Dozent	Prof. Dipl. -Ing Jürgen Walter
Datum der Ausgabe	**04.04.2022** **Abgabetermin** **31.08.2022**

Kurzthema / Subject:

Domänenübergreifende feingranulare Klassifizierung
Cross-Domain Fine-Grained Classification

Ziele:

- Fine-grained classification of vehicles
- Training of new vehicle models with images from a different domain
- Supervised partially zero-shot setting
- Source domain: web images, target domain: surveillance images



Aufgabenstellung:

- Looking up SotA approaches for cross-domain fine-grained image classification
- Evaluate multiple approaches in the new setting
- Establish one approach as baseline
- Extend approach to optimize for new setting

Vorsitzender des Prüfungsausschusses	**Betreuender Dozent an der HSKA**	**Betreuender Dozent am Arbeitsplatz/extern**
Prof. Dr. Martin Kipfmüller	Prof. Dipl.-Ing. Jürgen Walter	Stefan Wolf, M.Sc.

Stefan Wolf
Digital unterschrieben von Stefan Wolf
Datum: 2022.03.23 10:56:52 +01'00'

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Masterthesis ohne unzulässige fremde Hilfe selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, the 31st of August, 2022

Ritu Ahmed

(Ritu Ahmed)

Declaration of Independent Work

Hiermit erkläre ich, diese Arbeit unabhängig geschrieben zu haben, ohne Quellen zu verwenden und andere Hilfe zu leisten. als die, die in dieser Arbeit angegeben sind. Ich habe Passagen markiert, die aus anderen Werke, ob wörtlich oder inhaltlich, als solche. Ich halte mich an die Regeln in Bezug auf die Sicherstellung korrekter wissenschaftlicher Praxis an der Hochschule Karlsruhe Technik und Wirtschaft in ihrer jetzigen Form.

Karlsruhe, the 31st of August, 2022

Ritu Ahmed

(Ritu Ahmed)

Acknowledgments

My sincere appreciation goes to Prof. Dipl.-Ing. Jürgen Walter and Stefan Wolf for leading this thesis. Both have been outstanding thesis advisers and mentors, offering direction and inspiration with the right amount of information and support. I'm glad we got to work together; it's an honor for me.

Abstract

Cross-domain fine grained visual classification is a comparatively new research area in Computer Vision - combining the two well-known areas of Domain Adaptation (DA) and Fine Grained Visual Classification (FGVC). In FGVC, the problem of classifying similar looking objects is tackled. The dissimilarity in appearance of such objects are minute - and can appear at different scale levels. In DA, images from different domains e.g. the web and surveillance footage are used to train and test an object detector. The work presented herein combines both of these ideas - in order to solve the challenge of applying a single stage end to end classification framework to a newly organized largely unlabeled automotive dataset.

To this end, a new architecture is proposed by combining (i) a recently published plug-in module using well-known feature extractor backbones and fusion mechanism for fine-grained classification; with (ii) a well-established adversarial domain adaptation method to generate domain invariant features. A particular model using this architecture is then trained and evaluated on a standard dataset; and a newly organized partially labeled dataset from a different domain designated as the target domain.

As evident from evaluation, performance of a model using the proposed architecture for fine grained classification specifically for target domain images is significantly improved (0.61 vs 0.51).

This thesis is further organized as follows: in chapter 1, the general problem and its application are introduced. A summary of the reasoning behind this work; and contributions are provided. Chapter 2 discusses in depth background technical information related to this thesis; including definition of used terminology. In chapter 3 related works are described - specifically the ones that have been used as building blocks in this work. Chapter 4 and 5 describes the proposed architecture in detail; including used tools and frameworks. Chapter 6 provides a comprehensive account of evaluations carried out in this work - including a discussion of results. Chapter 7 concludes this thesis by providing guidelines to the work - to be carried out next.

Contents

1 Introduction

A model trained on one or more "source domains" can be applied to a different but comparable "target domain" through a process known as ***Domain Adaptation(DA)***. Transfer learning and machine learning are related to this area. Solely Computer Vision (CV) is addressed in our program, and transformer-based design is only used to challenges involving image classification. With the right example, DA may be correctly understood. Assume there exist several standard datasets for various uses, such as the GTSRB for the recognition of German traffic signs, the LISA and LARA datasets for the detection of traffic lights, the COCO for object detection and segmentation, etc. However, if there is a collection of all different sorts of photographs of Bangladeshi roads, and then the classification for those images, which is a hard and time-consuming operation, a neural network works effectively for our purpose, for example, traffic sign detection on Bangladeshi roads. As the model can be trained on GTSRB (the source dataset) and tested on our photos of Bangladeshi traffic signs, domain adaptation may be applied in this situation (target dataset).

Finding datasets with all the necessary changes and variety to train a strong neural network may often be challenging. In this situation, it is easy to create sizable artificial datasets with all the necessary changes using various computer vision methods. The neural network is then tested on real-world data after being trained on an artificial dataset (source dataset) (target dataset).

Consider the classification of a vehicle picture. In order for the classifier to adapt from one domain to another, it must perform well on the features collected from both the source dataset and the target dataset. The classifier must perform well for the source dataset after the neural network has been trained on it. However, the features derived from the source and target datasets should be comparable in order for classifiers to perform effectively on the target dataset[7]. Therefore, it requires the feature extractor to extract comparable features for source and target domain photos during training. Fig.1.1 represents successful domain adaptation.

In domain adaptation, we want to achieve acceptable performance on another dataset (target) whose label or annotation is partially accessible by training a neural network on one dataset (source) for which label or annotation is available. The term "supervised partially zero shot settings" can also be used for it.

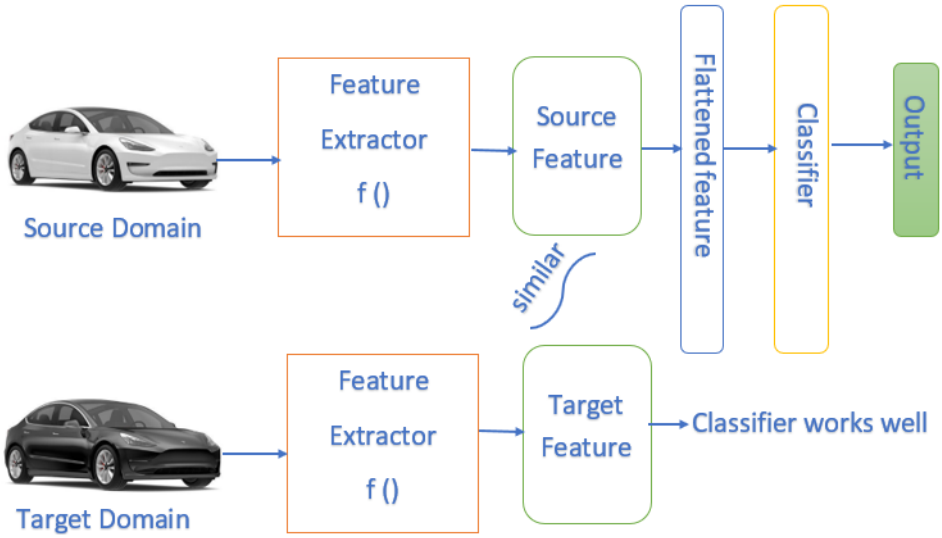


Figure 1.1: Domain Adaptation

An approach in computer vision known as ***Visual classification*** may categorize a picture based on its visual content [84]. It can be categorized as either coarse- or fine-grained [20].

1. Coarse-grained classification, such as the categorization of horses and cows, is the classification of categories with a high degree of dissimilarity.

2. Fine-Grained categorization refers to classifications with similarity, such as various animal species and diverse car brands and models. It alludes to more precise groupings within the taxonomy of common species. The fine-grained visual classification (FGVC) task has three obstacles [20]. First off, there is a lot of variance within a single category. Comparability between things belonging to several subcategories is the second point. Thirdly, unlike coarse-grained classification, fine-grained classification frequently necessitates the use of professional specialists to label the data, increasing the cost of the data.

On images of cars, a fine-grained classification method using a domain adaptation methodology is used. Automobiles have revolutionized convenience and mobility. Cars are a diverse object class since there are so many different car models and designs, which might lead to more advanced and reliable computer vision models and algorithms. Rich object is a term used to describe an item that cannot be fully defined or represented but about which claims may be made.

Figure 1.2: The two SUV models seem extremely similar from the side, but differently from the front.[114]

A variety of innovative study issues in object classification are made possible by the numerous special characteristics that cars possess that other items do not, which presents greater difficulties [114]. In particular, it has many more models than the majority of other categories, which makes fine-grained work more difficult. Additionally, when automobiles are in their natural postures, they produce significant visual variations (see Fig. 1.2). The manufacturer, model, and released year are the top-to-bottom levels of the automobile dataset. The hierarchical approach to tackling the fine-grained job is suggested by this structure [69]. The use of various design styles by several automobile manufacturers and in various years makes it more difficult to classify data at a finer level. Particularly, the characteristics of automobiles may be deduced from their outward look. These characteristics include car class, number of axles, distance, seating capacity, speed.

Car verification, which aims to confirm if two automobiles belong to the same model, is an interesting yet understudied subject compared to human face verification [91]. The unrestricted perspectives undoubtedly make automobile verification more challenging. In intelligent transportation systems, fine-grained automobile classification and verification may be utilized for a variety of tasks, such as regulation, description, and indexing. For instance, depending on differing prices for various sorts of cars, fine-grained automobile categorization may be used to quickly and affordably automate paying tolls from the lanes.

A sizable and thorough picture database dubbed ***"Comprehensive Cars"***, often known as "CompCars," is utilized for the project. Photos of cars from the web are included in the "CompCars" dataset, while images from security cameras are included in the "CompCarsSV" dataset.

There are many different research demands and application situations for fine-grained image

categorization in both business and academia. The identification of many species of birds, pets, flowers, automobiles, and aircraft are among the related study issues. There is a great need to distinguish between many subcategories in real life. Effective identification of various organism species, for instance, is a crucial condition for ecological study in ecological conservation [92]. Given the vast amount of data that exists for some domains, such as web photos, the usage of cross-domain adaptation is an intriguing element. Although fine-grained categorization is possible, there are only a few available methods, and relevant cross-domain settings are seldom taken into account. The use of computer vision technologies to more reliably classify fine-grained images at lower costs is significant for both academic and commercial circles.

1.1 Overview

The format of this report is as follows: We explain the issue and the specific research topics we focused on for this thesis in chapter 1.

The background information needed to comprehend the thesis and main ideas of this book is covered in Chapter 2. The explanation of a plug-in network that may be used with numerous models. This network combines feature fusion and background segmentation approaches, which can significantly increase the precision of fine-grained visual categorization. Additionally, domain adaption strategies are described in greater depth. Additionally, many model foundations and loss functions are examined in this chapter.

Based on this, the originality of the suggested scenario is understood, and chapter 3 defines the scope of the remaining work.

Chapter 4 discusses the methodology considerations and outlines the suggested architecture. The process is presented step by step. Chapter 5 explains how to put the model into practice. Here is a list of all the models' training environments as well as the implementation framework.

Chapter 6 goes into depth on how the final technique was evaluated. A baseline is first constructed after introducing the available metrics and statistics. The impact of significant design choices is explored, and ablation trials confirm the method's efficacy. The primary findings and contributions of this work are outlined in Chapter 7, along with potential future research topics.

1.2 Problem Statement

Small inter-class variation as opposed to huge inter-domain variance are two new issues brought on by cross-domain fine-grained categorization that must be carefully taken into account during adaptation. Furthermore, the high specificity of fine-grained classes raises

cross-domain possibilities, such as a supervised partly zero-shot scenario, which are unusual for conventional picture classification. When certain classes in the target domain have no photos at all but all other classes have plenty of images with annotations accessible in the target domain, this situation is known as supervised partly zero-shot. While all of the data in the target domain is unlabeled [33], conventional domain adaptation often assumes the availability of all classes in the target domain [104]. Due to the fact that previous research on cross-domain [104] or fine-grained categorization [105] [40] do not take into account works merging both disciplines . In this study, domain adaptability and fine-grained categorization are combined. Cross-domain settings are summarized in Figure 1.4.

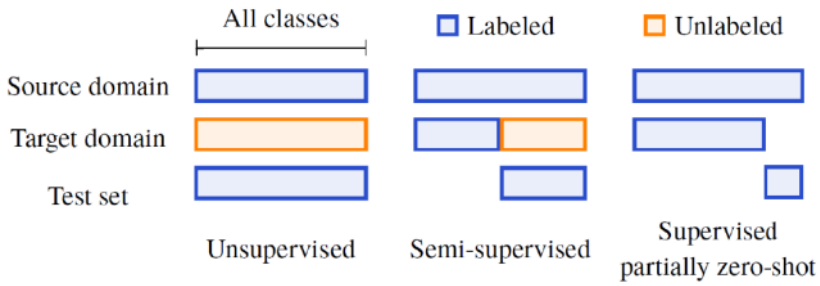


Figure 1.3: cross-domain setting[110]

1.3 Research Contribution

This thesis has the following main contributions:

1. Organization of a new image dataset for fine-grained classification tasks comprising source and target domain images. In the presented experimental setting, the source domain is taken as the standard CompCars dataset including only web Images, and the target domain is a new CompCarsSV dataset including only surveillance camera images.

2. A novel classification network where a fine grained visual classifier is supported by a domain classification network in a supervised partially zero shot setting, both networks jointly trained using back-propagation; in effect, combining both fine-grained and cross-domain classification in one unified architecture.

2 Background

This chapter describes the foundational ideas that the rest of this book builds upon. Before getting into the specifics of the implementation, a general overview of fine-grained visual categorization is provided. The formal definition of domain adaptation is followed by discussions of three particular examples. The model's training foundation and loss function's two key components are then explained.

2.1 Fine-Grained Visual Classification

It's a strategy for classifying data in which classes are quite similar and the classification model looks for minute variations to provide accurate predictions. Examples of these jobs include classifying various animal species or auto types [41].

2.1.1 Challenges

Fine-grained image classification is exceedingly challenging due to the extremely small classification granularity; even professionals struggle to discriminate between some groups. There are three primary causes.

1. There is simply a tiny change in one particular element, like a car's headlight, between subclasses.

2. Subclasses vary greatly from one another, for example, because of variances in background and posture.

3. The categorization was significantly impacted by the viewing aspect, ambient, occlusion, and other characteristics.[92]

2.1.2 Common methods

FGVC now mostly uses deep learning, which produces effective outcomes. It may be broken down into the following groups, specifically:

1. Deep Convolutional Neural Network (DCNN) in general Currently, it is not widely employed since it is challenging to capture the unique local characteristics while utilizing DCNN. [92].

2. The location-based technique involves first identifying the discriminative components, followed by feature extraction and classification in the manner of a standard classification algorithm. Strong supervision and weak supervision are two categories for the procedure;

3. The network-integration approach: To separate the comparable features in fine-grained recognition, it employs multiple DCNN;

4. High-order feature coding method: One of the crucial steps in picture classification is feature coding. In this stage, each local descriptor generates a coding vector by first activating a number of codewords [66]. Local features hold the discriminating visual data from across the image, however they rarely serve as the primary means of image categorization. Typically, a two-step feature coding and pooling procedure is used to condense the low level descriptors into an intermediate form [29]. High order feature coding approach based on FGVC first converted CNN features to high-order before doing classification; examples of this method include fisher vector, bilinear model, and kernel fusion [92].

When humans identify identical items, they often discover the distinctive area by quickly scanning the object, compare it, and then carefully recognize it. This is the basis for the location classification approach. The fine-grained image recognition approach based on location recognition is split into two components, much like how humans do it: discriminative area location and fine-grained feature learning in region. While features are taken from each region and merged for classification in fine-grained feature learning, convolution feature response of deep neural networks is typically employed in discriminative region localization.

Along with class labels, the strong supervision approach also requires components labeling and essential parts boxes. This approach has produced good results, but it has the drawback of requiring expensive manual labeling, and the manual labeling site may not always be the optimal place for differentiating features because it entirely depends on the cognitive ability of the tagger.

Numerous inadequate supervision techniques have been put out in recent years. This approach relies just on the classification label to finish the training without component labeling and leverages attention mechanisms, clustering, and other techniques to automatically identify the different areas.

Numerous approaches are now being used in this aim. Fine-grained image classification approach based on location classification has undergone evolution from manual feature engineering to multi-stage method, and finally to end to end, according to the basic emerging trends.

1. **Strong Supervision:** According to the definition of a "strong supervised fine-grained image classification model," this type of model incorporates extra human annotation data, such as Object Bounding Box and Part Annotation, in addition to the image assignable, to improve classification accuracy during model training [111]. This is seen in figure 2.1 below.

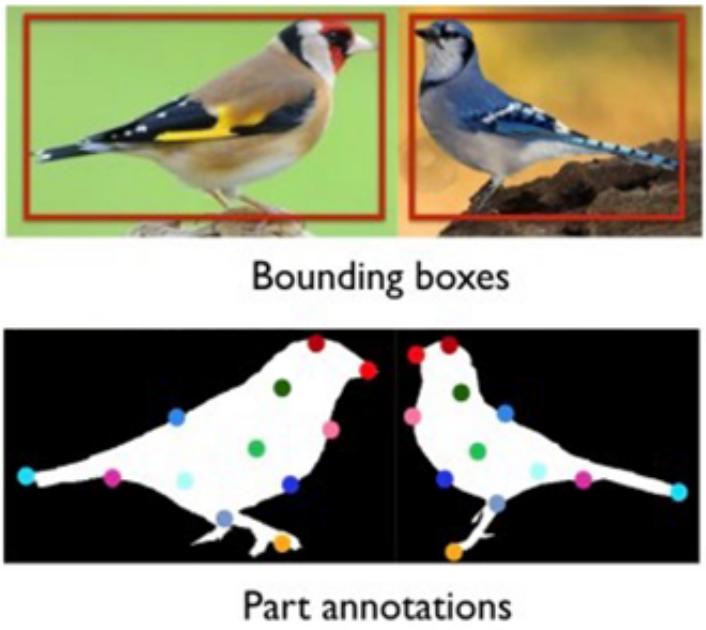


Figure 2.1: Strong Supervision [92]

2. **Weak Supervision:** Learning item positions from an image using only picture-level annotation is known as weakly supervised object localization. It is a technique for producing significantly bigger training sets much faster than would normally be possible with manual supervision. High-level and frequently noisier sources of monitoring are employed in this method (i.e. labeling examples manually, one by one)[112].

Recent years have seen a progressive movement in location classification method research toward weak supervised learning because of the demands of industrial applications. In this work, weak supervised learning is also employed.

2.1.3 State of the art

A Novel Plug-in Module for Fine-Grained Visual Classification by Po-Yung Chou is our choice for the state of the art[20]. The author of this study puts forth a brand-new plug-in module that can be plugged into a variety of widely used backbones, such as CNN- or Transformer-based networks, to produce regions with a high degree of discrimination. To improve fine-grained visual categorization, the plugin module may produce pixel-level feature maps and combine filtered features..

It contains a plug-in module that can be applied to popular backbone networks like ResNet, EfficientNet, and ViT in order to uncover strong discriminative areas for fine-grained classification problems. The general idea is to regard each pixel (or patch) on the feature map as a separate feature that might represent its region. It then categorizes these qualities and uses that capacity as a foundation for differentiating. Last but not least, end-to-end training can cover the whole network.

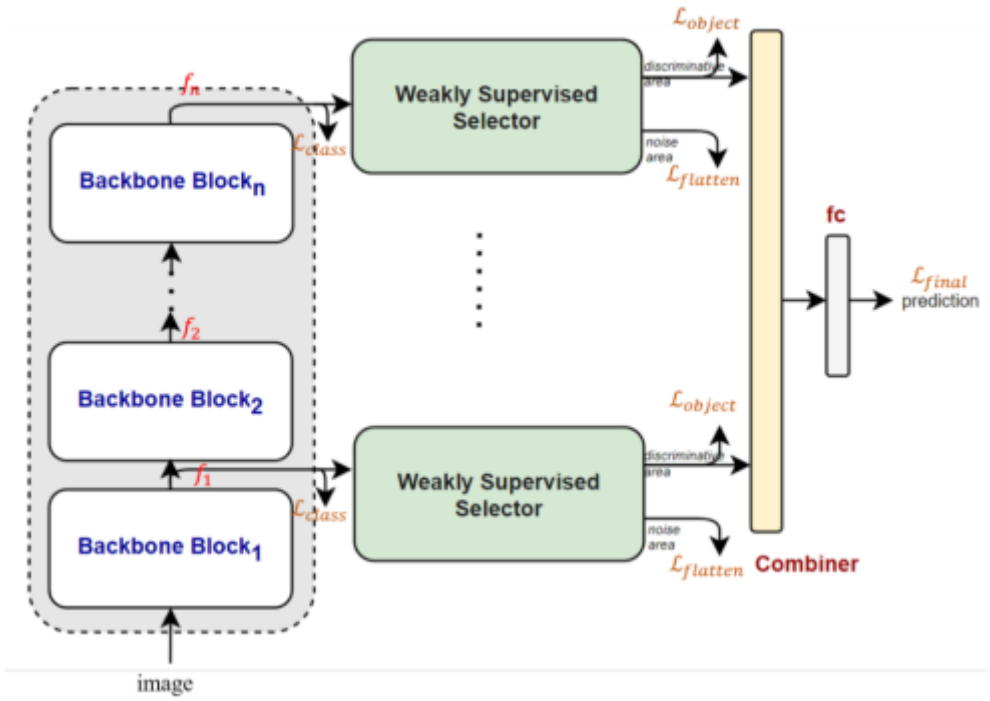


Figure 2.2: The plug-in module's flow diagram. The backbone network's kth block is represented by the Backbone Block. The Weakly Supervised Selector will screen away regions with high discrimination or regions that are less relevant to categorization when the picture is submitted to the network using the feature map generated by each block. In order to create the prediction results, a Combiner is employed to combine the characteristics of the selected results. Lfinal is a representation of the loss function.[20]

2.2 Domain Adaptation

In the discipline of domain adaptation, which is part of computer vision, the objective is to achieve excellent accuracy on a target dataset while training a neural network on data that is drastically/partially different from the source dataset.

Take the image categorization example from above as an example of domain adaptation. We need our classifier to perform effectively when applied to both the target dataset and the characteristics derived from the source dataset in order to adapt from one domain to another[104]. The classifier must perform well for the source dataset since we used the source dataset to train the neural network. But in order for classifiers to be effective on the target dataset, we would need the characteristics that are retrieved from the source and target datasets to be comparable. As a result, we require feature extractors to extract comparable features for

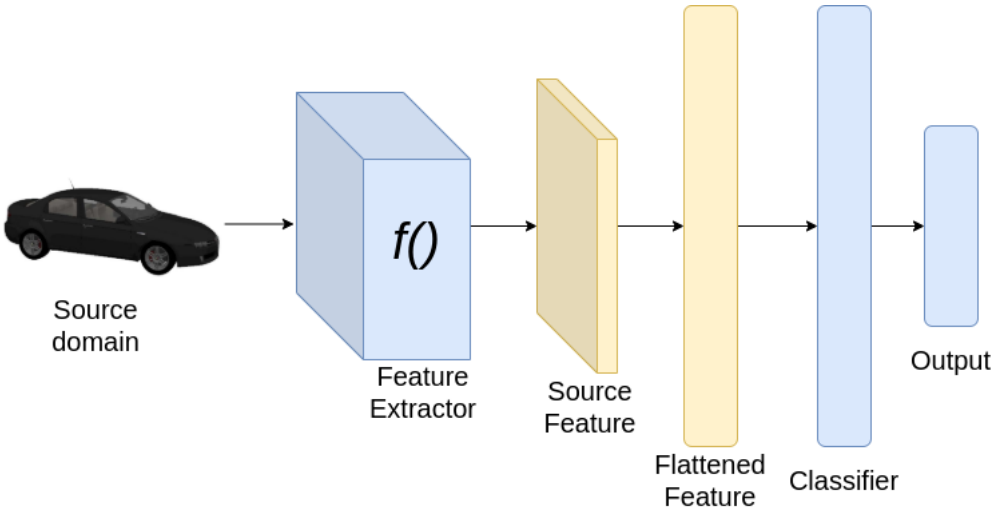


Figure 2.3: Classification Pipeline [68]

source domain and target domain photos when training.

2.2.1 Type of Domain adaptation based on Target domain

Domain adaptation may be grouped into the following groups depending on the type of data that is available from the target domain::

1. **Supervised**— The target dataset is substantially smaller than the source dataset since we have labeled data from the target domain.
2. **Semi-Supervised**— We have data from the target domain that is both labeled and unlabeled.
3. **Unsupervised**— There are many sample points in the target domain that are unlabeled.

2.2.2 Techniques for Domain Adaptation

Any domain adaptation method is typically implemented using three ways [28]. The three methods for domain adaptation are as follows:

1. Divergence based Domain Adaptation
2. Adversarial based Domain Adaptation
3. Reconstruction based Domain Adaptation

Divergence based Domain Adaptation

Divergence-based domain adaptation relies on the idea of minimizing a divergence-based criteria between the source and target distributions, producing characteristics that are domain invariant. Common divergence-based criteria include Contrastive Domain Discrepancy, Correlation Alignment, Maximum Mean Discrepancy (MMD), Wasserstein, etc. [28]

In Maximum Mean Discrepancy (MMD), we want to determine whether or not the two samples are members of the same distribution. We define the distance between two distributions as the difference between their mean feature embeddings. If we have two distribution say **P** and **Q** over set **X** [28]. Then MMD is defined by a feature map $\varphi : \mathbf{X} \rightarrow \mathbf{H}$, where **H** is called a reproducing Kernel Hilbert Space.

With the use of a linear transformation in MMD, we attempt to align correlation (second-order statistics) between the source and target domains in comparison to the mean.

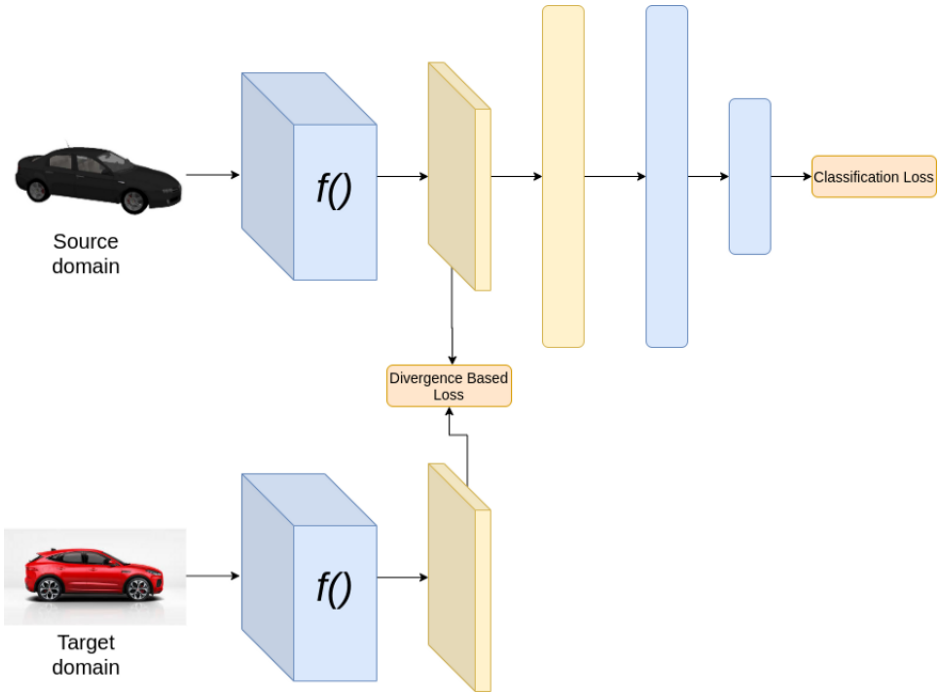


Figure 2.4: During Training of Divergence based Domain Adaptation [68]

The aforementioned design presupposes that the classes in the source and destination domains are same. In the aforementioned architecture, we minimize two losses during training: the classification loss and the divergence-based loss. The weights of the feature extractor and classifier are updated to ensure that classification performance is good [58]. By changing the weights of the feature extractor, divergence-based loss ensures that the features of the source

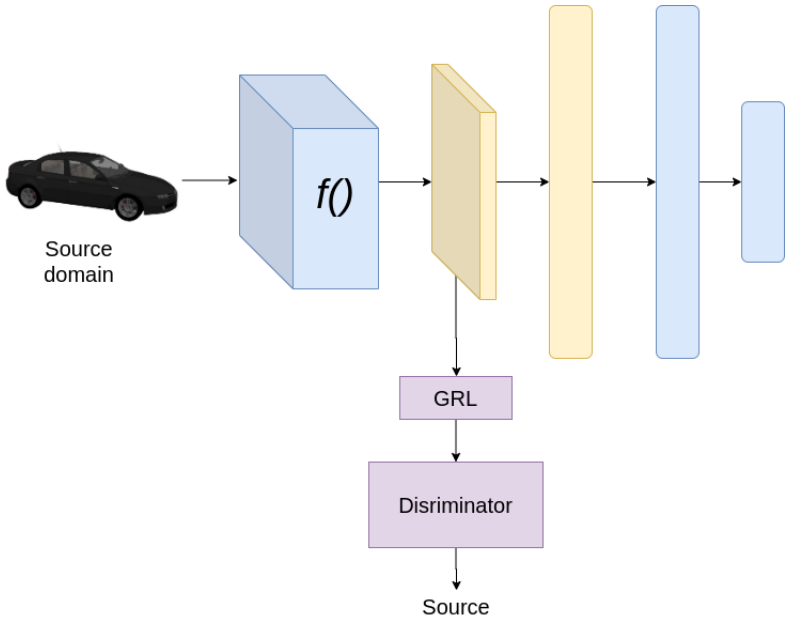


Figure 2.5: Adversarial based Domain Adaptation;During Training-for source [68]

and target domains are comparable. We just need to pass the target domain picture from our neural network during the inference phase.

The majority of the divergences are non-parametric, hand-crafted mathematical formulas that are not tailored to the dataset or our task, whether it be segmentation, object identification, or classification, for example. As a result, this formula-based strategy does not really cater to our situation[59]. As opposed to typical pre-defined divergences, it is anticipated that the divergence will perform better if it can be taught depending on the dataset or challenge.

Adversarial based Domain Adaptation

Generative Adversarial Networks (GANs) are used in this instance to implement adversarial-based domain adaptability . Here, our generator is only a feature extractor, to which we add additional discriminator networks that can recognize differences between features from the source and target domains. Given that there are two players in the game, the discriminator aids the generator in creating features that are identical for the source and target domains [96]. We have a learnable discriminator network, so we can train to extract features (unique to our problem and dataset) that might assist to differentiate between the source and target domain, aiding the generator in producing more robust features, or features that cannot be readily discriminated.

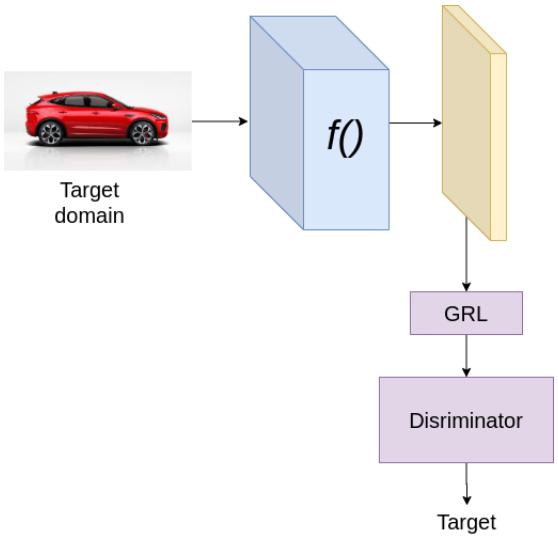


Figure 2.6: Adversarial based Domain Adaptation;During Training-for target [68]

The classification loss and the discriminator loss are the two losses we are employing, assuming the classification issue. Later, it is discussed why categorization loss occurs. Discriminator loss aids in the distinction between source and target domain characteristics that is made by discriminator. In this instance, adversarial training is implemented using the gradient reverse layer (GRL). Simple GRL blocks back-propagate a gradient while multiplying it by -1 or another negative number. Gradients from the discriminator and classifier, respectively, are used in tandem during training to update the generator[16]. Due to GRL, the gradients from the discriminator are multiplied by a negative number, which causes the training generator's impact to be the reverse of that of the discriminator. As an illustration, if the generator is updated with -2 if the computed gradient to optimize the loss function for the discriminator is 2 [16]. (assuming negative value to be -1). In doing so, we hope to train the generator such that it is unable to discern between the characteristics of the source domain and those of the target domain, even with the aid of the discriminator. In several works on domain adaptation, the GRL layer is frequently employed.

Reconstruction based Domain Adaptation

The concept of image-to-image translation is utilized here. One of the straightforward methods would be to train a classifier on the source domain while learning how to translate pictures from the target domain to the source domain. Using this concept [22], we may present a variety of ways. A discriminator might be used to drive an encoder-decoder network to create pictures

that are comparable to the source domain in the simplest model for image-to-image translation.

2.3 Backbone of the model

In DeepLab models and publications, the word "backbone" is used to describe the feature extractor network. The feature extractors take features from input, and then a straightforward decoder module of DeepLab models upsamples those features to produce segmented masks [26]. There are several types of backbones for various network and modeling types. Only the backbones we utilized for the experiment will be mentioned in this publication. ResNet50 and swin Transformer are those (Swin-T).

2.3.1 Resnet50

Training increasingly complex neural networks is more challenging. It is a residual learning framework designed to make it easier to train networks that are far deeper than those utilized in the past [50]. Instead of learning unreferenced functions, it explicitly reformulates the layers to learn residual functions with reference to the layer inputs.

With 48 Convolution layers, 1 MaxPool layer, and 1 Average Pool layer, ResNet50 is a variation of the ResNet model. It can operate on $3.8x10^9$ floating point numbers. It is a frequently used ResNet model, and we have extensively examined ResNet50 design [50].

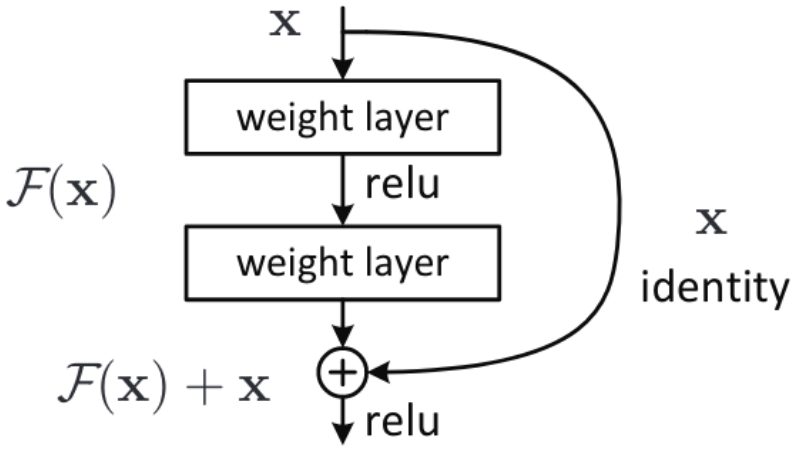


Figure 2.7: ResNet [50]

The shortcut connections used to skip two layers, but with ResNet 50 and upwards, they now skip three levels. Additionally, 1 X 1 convolution layers were added, which we will go over

in more depth with the ResNet 50 Architecture.

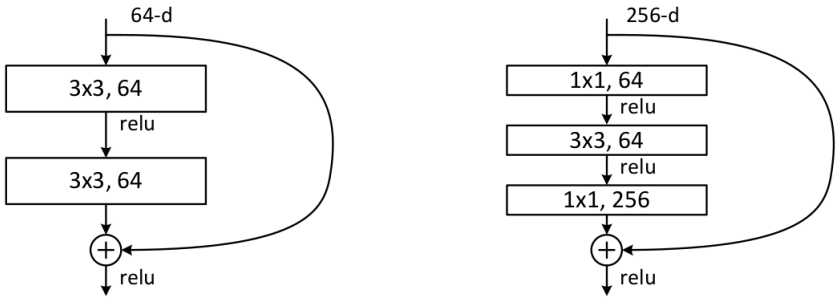


Figure 2.8: ResNet50 [50]

resnet50 architecture contains the following element [50]:

1. A convolution with 64 distinct kernels, each having a stride of size 2, and a kernel size of 7 X 7 gives us 1 layer.

2. Next there is max pooling with also a stride size of 2.

3. The next convolution consists of three layers: a 1 X 1,64 kernel, a 3 X 3,64 kernel, and finally a 1 X 1,256 kernel. These three levels are repeated a total of three times, giving us nine layers in this phase.

4. The kernel of 1 X 1,128 is shown next, followed by the kernel of 3 X 3,128 and, finally, the kernel of 1 X 1,512. We performed this procedure four times for a total of 12 layers.

5. Following that, we have a kernel of size 1 x 1,256, followed by two additional kernels of size 3 x 3,256 and size 1 X 1,1024. This is repeated six times, giving us a total of 18 layers.

6. Another 1 X 1,512 kernel was added, followed by two additional 3 X 3,512 and one 1 X 1,2048 kernels. This process was done three times, giving us a total of nine layers.

7. After that comes an average pool and a link layer with 1000 nodes, and a softmax function for 1 layer.

The activation operations and the maximum/average pooling layers are not included in the layer calculation.

All together, this is a 50-layer Deep Convolutional network with 1 + 9 + 12 + 18 + 9 + 1 [53].

2.3.2 Swin Transfromer

A capable general-purpose backbone for computer vision is Swin Transformer (the term Swin stands for Shifted window) [63]. This is a hierarchical Transformer where computation of representation uses shifted windows.

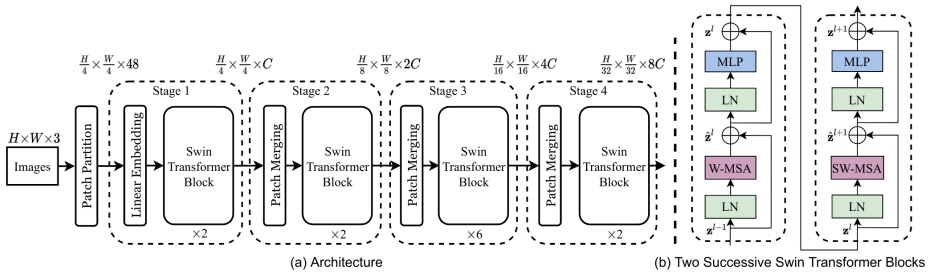


Figure 2.9: Swin Transformer [63]

Machine translation was the initial purpose for which transformers were designed. Due to the fact that speech is temporal data and does not have a spatial component as pictures do, traditional global self-attention-based transformers performed well on translation tasks since they were excellent at collecting long-range relationships.

Attending all pixels or patches (image crop) is useless when it comes to categorizing photographs or finding a cat in a corner of an image [77]. The ablation experiments of ViT provide evidence for this theory, since they calculated global attention across all patches and discovered that only the encoder layers were focusing on the small region that contained the item.

As a result, in order to adjust the network model for pictures, one also required to modify the learning algorithm inside the Transformer's Encoder block. — The creators of SWIN created a general-purpose Transformer backbone in computer vision by transforming global attention to local attention.

Swin transformer Architecture

Firstly the main components of SWIN is explained and then the final model will be explained.

Patch Partition

Similar to ViT, a SWIN transformer processes a picture by breaking it up into non-overlapping patches. Each patch, referred to as a token, is 4x4x3=48 pixels in size, with 3 representing the RGB channel and 4 representing the patch's height and breadth [44].

Patch Merging

As it introduces the inductive bias into SWIN that was absent from traditional ViT and DeIT, patch merging is a vital step in the construction of the SWIN transformer. Patch merging merges two 2x2 windows into one new window, downsampling the size of the feature map by two, and deepening each patch by two.

Self Attention in a Window (W-MSA)

SWIN transformer makes use of Encoder blocks from the original Transformer architecture. An array of self-attention modules with several heads and a feed-forward network make up each encoder block [44].

In SWIN, we start with a window of fixed size, with a specific amount of patches in each window. In their papers, writers use square windows that each have MxM patches, while our window in the left picture has 3x6 patches. Now we only pay attention to the patches in this window to compute the attention encoding of the top left patch. As opposed to taking care of all tokens for every token, this method is far more effective and scalable. Authors drew attention to this, Windowed multi-headed self-attention, W-MSA.

Self Attention Across "Shifted" Windows (SW-MSA)

A CNN's expanding receptive field informs us that correlations across windows are crucial for accomplishing visual tasks and would be lost if we only relied on window-based attention [77].The smart aprroach is :

1. Usin the result of W-MSA
2. Adjusting all windows' height and width by half
3. computing W-MSA in the repositioned windows

Shifted windowed multi-headed self-attention, or SW-MSA, is the name given to this attention.

Overall Architecture

Four phases have been created by the authors to represent the flow of an input picture from a SWIN transformer. **Stage 1:**

1. To divide an input picture into fixed-sized patches, it must first travel through a patch partition. If a patch is 4x4 and the picture is H x W, the patch partition gives us H x W patches [63].

2. It is 4x4x3 = 48 pixels as each patch has a channel dimension. We run each patch through a linear layer (48xC), which enlarges each patch up to C dim, in order to convert each patch from 48 pixels to the more practical size C. The output is a feature map of dimension H/4 x W/4 x C, consisting of H/4 x W/4 patches, each of size C [63].

3. SWIN transformer block is used to process this feature map. The size of the inputs and outputs doesn't change since the SWIN transformer block is made up of transformer encoder blocks. As a result, the stage-1 SWIN transformer block's output maintains a H/4 x W/4 x C feature map size comparable to that of the input feature map [63].

Stage 2:

1. The H/4 x W/4 x C-sized feature map is now run through a patch combining layer, which creates a new window by joining two adjacent ones. This reduces the resolution by two times and deepens the feature map by two. H/4 x W/4 x C thus becomes H/8 x W/8 x 2C.

2. Another SWIN transformer block is used to pass this extracted features, maintaining its dimensions.

Stage 3 and 4:

1. After moving beyond each patch merging layer in Stages 3 and 4, which also follow the same process as Stage 2, the resolution of the feature map is cut in half.

The size of feature space at each level is shown in the above picture, which illustrates how, similar to CNN, as we travel further into the network, the feature map's resolution decreases and its depth increases.

2.4 Loss function

The cost function in machine learning is the average of the loss functions for all training instances, while the loss function in machine learning is defined as the difference between the actual output and the anticipated output from the model for a single training example [97]. The error value is the computed difference between the actual value and the anticipated value from the loss functions Regression loss, Multi-class Classification loss, Binary Classification loss.

2.4.1 Loss Functions: How Do They Operate?

Loss denotes the cost of failing to provide the anticipated results. The loss function produces a greater number if there is a considerable discrepancy between the projected value and the anticipated value predicted by our model and a minor deviation that is substantially closer to the expected value.

It is crucial to remember that the degree of variance is irrelevant in this situation. Depending on the problem statement, different loss functions will apply[**common**]. Another name for the loss function that is frequently used interchangeably is the cost function, which has a somewhat different definition . A cost function is an average loss over the whole train dataset, whereas a loss function is for a single training case.

2.4.2 Loss Functions in Machine Learning and Their Types

Below are the different types of the loss function in machine learning which are as follows:

1. **Regression loss functions:**A key idea in this function is linear regression. We attempt to fit the best line in space on these variables because regression loss functions produce a linear connection between a dependent variable (Y) and an independent variable (X). These loss functions were developed to gauge how well the categorization model performed [97].

2. **Binary Classification Loss Functions:**These loss functions were developed to gauge how well the categorization model performed. In this, each data point is given one of the labels, 0 or 1, as shown.

3. **Multi-class Classification Loss Functions:**Data points are allocated to more than two classes in multi-class classification prediction models. Each class is given a distinct value between 0 and (Number of classes − 1). It is highly advised for situations involving the categorization of text or images, when a single document may cover several subjects.

In our case we need binary classification loss function, so we will explain it in more details.

2.4.3 Binary Classification Loss Functions

They can be classified as:

Binary Cross-Entropy

This loss function serves as the default one for binary classification issues. A classification model's effectiveness is calculated using the cross-entropy loss, which outputs a probability

value between 0 and 1. As the anticipated probability value deviates from the actual label, the cross-entropy loss grows [15].

Hinge loss

Cross-entropy, which was first created to be utilized with a support vector machine algorithm, may be replaced by hinge loss. The classification issue benefits from hinge loss the most since the goal values fall into the range of -1,1. When there is a difference in sign between the actual and anticipated numbers, it enables the assignment of greater error. Consequently, it performs better than cross-entropy[97].

2.4.4 Importance of Loss Function in Image Classification

Let's say we are instructed to put 10 kg of sand in a bag. You must fill it up completely until the scale reads exactly 10 kilograms, or you must remove the sand if the reading is higher.

Our loss function will provide a greater value if our forecasts are incorrect, much like that weighing machine. If they're decent, it will provide a lower number. Our loss function will let us know whether we're making any progress as we test your method to see if we can make your model better.

A loss function, at its heart, measures how well your prediction model performs in terms of being able to forecast the anticipated result (or value)[48]. The learning issue is transformed into an optimization problem, a loss function is defined, and the method is then optimized to minimize the loss function.

Loss functions provide as more than just a static illustration of how well your model is doing; they also serve as the foundation upon which your algorithms fit data. When choosing the optimum parameters (weights) for your data, the majority of machine learning algorithms employ some form of loss function.

Importantly, the activation function that is employed in the output layer of your neural network has a direct impact on the loss function that you choose. These two design components go together[48].

Consider the output layer configuration as a decision for how to frame your prediction issue, and the loss function selection as the method for calculating the error for a certain framing of your problem.

3 Related Work

In This section, related work on fine-grained image classification and domain adaptation is reviewed. Additionally a comparison of the difference among methods is presented.

3.1 Fine-grained Classification

Fine-grained visual classification has grown to be a common issue in computer vision nowadays. Some techniques incorporated extra labels such part-annotations and visual features to increase fine-grained recognition since it takes skill to distinguish the tiny distinctions between the subsidiary categories within the same root category [118, 14, 117, 101].

Finding highly discriminating zones is the fundamental goal of FGVC. Numerous techniques, including NTS-Net [115], FDL [120] , use the Region Proposal Network (RPN) to find strong discriminative preselected boxes, resize the preselected boxes to a fixed size, and then use these local strong features to identify [46]. In other words, these techniques actually split the identification task into two stages.

Some research tried to find alternate approaches to boost the fine-grained recognition performance in place of the expensive part-annotations or extra characteristics[35]. By employing co-segmentation and alignment to create components, Krause et al[54]. attempted to address the issue of fine-grained recognition. A bilinear pooling-based two-stream CNN model that is also trained using category labels was proposed by Lin et al.[61]. In order to reduce computing complexity while maintaining comparable accuracy, Gao et al. [60] developed the compact bilinear pooling approach as an adaptation of [52].

Soon after, new iterations of the basic bilinear pooling technique were put forth and used on neural network models for fine-grained recognition [55, 35] . Confusion was added to the activations [113] and MaximumEntropy was reviewed [62] by Dubey et al. Some techniques to enable the fine-grained recognition models to profit from the vast but noisy online data have been proposed [34, 57], helping to further reduce the difficulties of manually compiling expert-level annotations. As a result of the ViT architecture's impressive image recognition performance since the publication of Vision Transformer (ViT) [13], several methods for using this architecture to complete the FGVC problem have been offered. For instance, FFVT [25],

AFTrans [119], RAMS-Trans [47], and TransFG [42] use ViT as backbone to search for strong discriminative areas. To finish the recognition task, the characteristics of these areas are processed. This method is comparable to the one using a convolutional network that was previously stated. Instead of using the reaction of the feature map, the strength of the attention map is employed.

[82] uses the attention map to find highly discriminative areas. The map is then additionally used to complete mixed reinforcement. Through these regional characteristics, CCFR [90] may detect global features and learn improved local area features using triplet loss and scale-separated NMS.

In FGVC tasks like MACN [121], WS-DAN [13], and CAL [25], attention mechanisms are also frequently utilized to learn object locations and to extract certain location-specific properties from feature maps using the attention-map. The attention mechanism not only aids in location learning but also enhances the expressiveness of characteristics.

By training the differences between two comparable pictures concurrently using the attention mechanism, for instance, MAMC [103] and API-Net [119] learn unique features, while CAP [47] combines several local characteristics using the same attention process. All of the strategies mentioned above demonstrate how effective the attention mechanism is.

The job of unsupervised domain adaptation is based on the assumption that both the source domain's large labeled dataset and the destination domain's large unlabeled dataset include examples for all categories. Gebru et al. [35] are the first to investigate such a situation with fine-grained categories. By adding a second classification head for each auxiliary attribute and using an attribute consistency loss to force the projected attributes to match the primary classification category, the authors take use of the auxiliary attributes frequently present in fine-grained datasets. Due to the attribute consistency loss, the number of training samples per attribute category is larger than the number of training samples per main category, resulting in more reliable attribute prediction across domains. Additionally, Tzeng et [96] .'s suggested domain confusion loss is implemented when using auxiliary characteristics. The method is tested for vehicle classification using Google Street View (GSV) photographs instead of web-crawled marketing photos.

The aforementioned approaches managed to perform rather well even without part-annotations. However the lack of fine-grained annotations for the numerous subordinate categories in the actual world limits their potential to scale.

3.2 Domain Adaptation

In order to avoid the expense of human annotations, domain adaptation involves transferring information from the source domain to the target domain [75]. The major barriers to knowledge transfer are differences between the source domain and the target domain. Some research offered various adaption layers based on deep networks to learn domain-invariant, transferrable characteristics [96, 65, 16] . Recent research on domain-adversarial approaches examined how adversarial learning may be implemented into the domain adaption framework [96, 33].

By attempting to trick the domain discriminator, these models were able to align the feature distributions of several domains. In order to solve the issue of fine-grained cross-domain recognition, CDAN [64] further uses the adversarial adaptation models to predict class labels. In order to choose simple samples from the target domain and match these pseudo-labeled samples with their associated source categories, PFAN [17] use a "Easy-to-Hard Transfer Strategy". A key contrast between PFAN and our work is that our technique learns from progressive granularity diametrically whereas PFAN learns from progressive samples without analyzing the granularity information.

By gradually changing the training distribution, some techniques execute a smooth transition from the source to the target domain (Gopalan et al [39]; Gong et al.[38]). One of these techniques, (S. Chopra and Gopalan [19]), does this in a "deep" manner by layerwise training a series of deep autoencoders and gradually swapping out samples from the source domain for samples from the target domain. This is an improvement over a related strategy used by Glorot et al.[37], which only trained one deep autoencoder for each domains. In both methods, the feature representation learnt by the autoencoder is used to train the actual classifier or predictor in a separate phase(s); accomplishing feature learning, domain adaptation, and classifier learning in a unified architecture, and utilizing a single learning algorithm, in contrast to (Glorot et al.[37]; S. Chopra & Gopalan [19]). (backpropagation).

While the aforementioned methods of domain adaptation are unsupervised, there exist methods that use labeled data from the target domain to conduct supervised domain adaptation. Such information may be utilized to "fine-tune" the network trained on the source domain in the context of deep feed-forward architectures (Zeiler and Fergus [116]; Oquab et al. [74]; Babenko et al.[12]). Labeled target-domain data is not required. At the same time, when such information becomes available, it may quickly include it. Goodfellow et al.[**Goodfellow**] is also relevant in this case. The minimization of the difference between the distribution of the training data and the distribution of the synthesized data remains on similar in both approaches.

Domain adaptation for feed-forward neural networks has garnered a lot of attention in

recent months. 2014's (Ajakan et al.[32]) On a task involving natural language, their system is assessed (sentiment analysis). Long Wang [65] and Tzeng et al.[96] both recently published publications that address domain adaptability in feed-forward networks by calculating and reducing the separation of data distribution means across domains.

A portion of the target domain samples are labeled in the context of semi-supervised domain adaptation. The subsets of labeled and unlabeled data are divided by classes in the case of fine-grained classification [35]. In a semi-supervised scenario, Gebru et al.[35] and Wang et al. [104] also assess their methods as described above. Wang et al. [34] suggest a contrastive loss for category-level alignment using the labeled target samples, in contrast to Gebru et al. who use a cross entropy loss for the labeled samples in the target domain. Before the final classification layer, the author suggest integrating a residual correction block that is trained to reduce the discrepancy between the distributions of features in the source and target domains. Based on the realization that early features are more task- and domain-invariant than late features [104], it was decided to include the residual correction block.

Usuyama et al.'s [99] proposal is the first of its kind to present a fine-grained domain adaptation setting that is distinct from unsupervised or semi-supervised; specifically, they forbid the use of unlabeled data in a semi-supervised situation. Thus, the scenario is even more challenging since certain classes have no examples at all accessible in the target domain. The difficulty in collecting samples for certain classes or guaranteeing that a given class is included in a collection of samples that have been randomly obtained is a result of the high specificity of fine-grained classes, which gives this scenario a significant practical significance. The classes in this case that lack examples in the target domain are the ones that are evaluated. In the words of Ishii, Takenouchi, and Sugiyama [49], such a situation is referred to as supervised partly zero-shot. Usuyama et al. [99] suggest a new fine-grained dataset called ePillID that includes pictures of pills in two domains: reference pictures with specific lighting and vantage points and masked backgrounds, and consumer pictures with more intra-class variation. Using metric learning in a cross-domain environment, the authors assess a baseline method.

These approaches are smart ones. However none of them investigated the label and none of them were explicitly created for fine-grained cross-domain visual classification.

3.3 Fine-grained Domain Adaptation

As it already mentioned first research on fine-grained domain adaptation was done by Gebru et al. [35]. Using the domain adaptation technique suggested in [47], they offered a model that was trained using annotated online photos and tested with real-world data, necessitating additional attribute annotations. Its semi-supervised adaptive loss, which is just a specialized

design for fine-grained domain adaptation, can only be accomplished when labeled pictures on the target domain are available.

With a novel strategy for utilizing strong supervision, Xu et al. [113] made use of regular image-level labels together with specific annotations like object bounding boxes and component landmarks. Weakly supervised web pictures were used to transfer as much information as feasible from existing strongly trained datasets. Additionally, Cui et al. [23] were able to significantly enhance a number of fine-grained visual benchmarks by fine-tuning CNNs that had already been trained on the large iNaturalist2017 dataset [100] . They suggested a metric to gauge domain similarity and chose a portion of the source domain that is closer to the target domain. All of the aforementioned strategies produced positive results.

The issue configurations, nevertheless, are very unlike from the presented work. This method depends on the hierarchical labels that are simpler to get in fine-grained tasks rather than characteristics, bounding boxes, or component landmarks.

4 Methodology

This chapter introduces the proposed architecture which is a combination of Fine-grained visual Classification with semi-supervised domain adaptation technique to improve mainly unlabeled class prediction.Figure 4.1 shows the complete model of our propose method. This chapter is divided into several section;shows step by step procedure.

Firstly, we have introduced the fine grained classification technique,later the cross-domain setting. After that Domain adaptation technique is combined with the previous step.

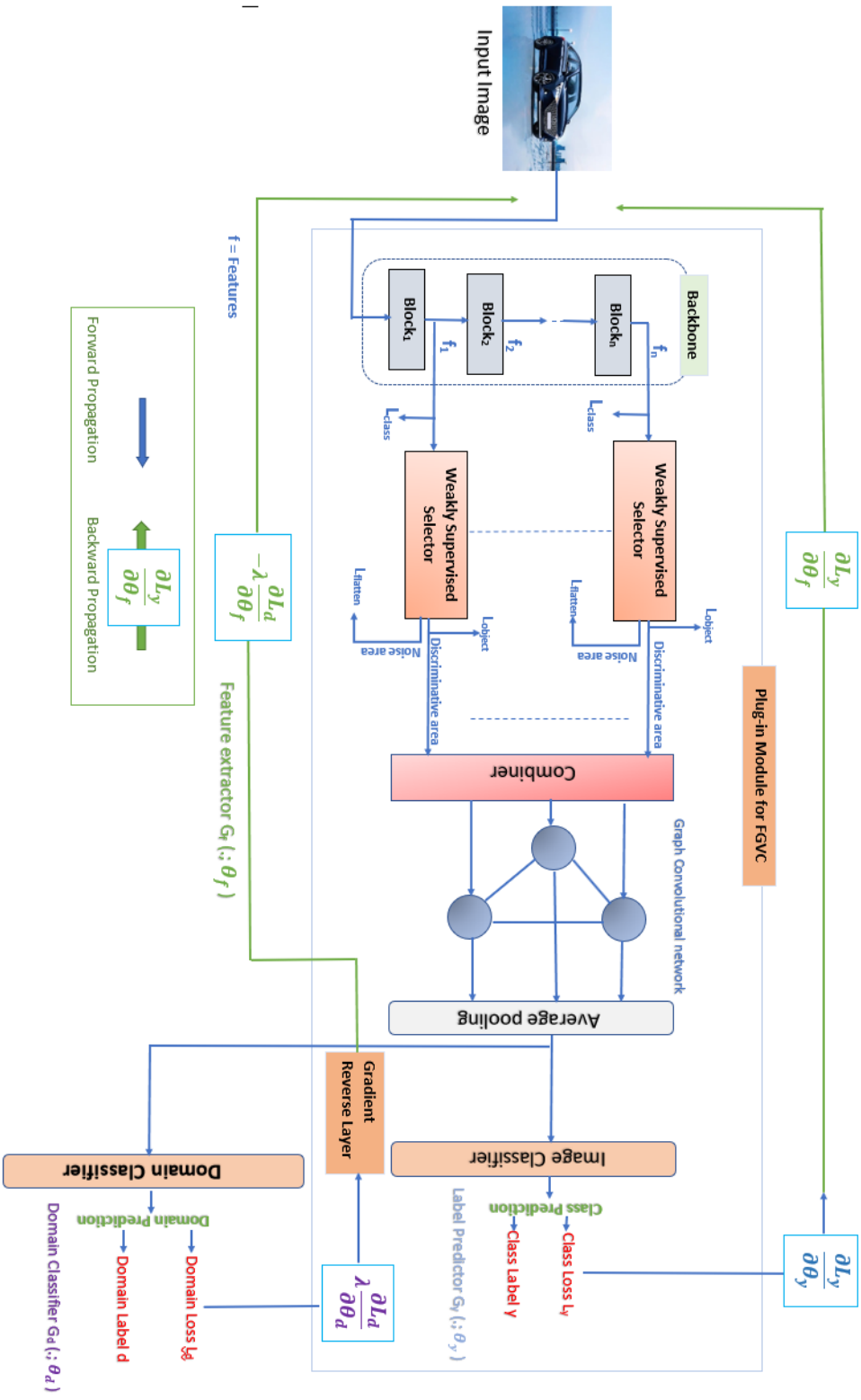


Figure 4.1: Complete model of Cross-Domain Fine-Grained classification

4.1 Fine-Grained Classification on CompCars Dataset

Plug-in Module for Fine-Grained Visual Classification is what we used [20]. This section's content is entirely drawn from this publication. The idea behind this method is to regard every single pixel on the feature map as a separate feature that can be used to refer a particular location. After that, it classifies these qualities and uses the ability to classify as a foundation for differentiating. Finally, end-to-end training may be used to finish the network as a whole. This chapter will provide a detailed introduction to the architecture of the model, the application of the loss function, and the integration with each framework.

4.1.1 Architecture

Finding an area with strong discrimination is a goal shared by the FGVC and Weakly Supervised Object Detection (WSOD) approaches [20]. For more precise localization, FGVC reduces this region or enhances focus while The second part of training is ongoing , whereas WSOD frameworks frequently employ multiple instance learning (MIL). Although the placement objectives for these two tasks are still significantly different, these structures show the characteristics of pixels in the feature space may adequately convey the significance of this region to the categorization job. The value present in each pixel of the feature map is referred to as the feature point in the text that follows.

The size of this feature point is R^C , where C stands for the block's output feature dimension size. It uses a relatively straightforward approach that predicts the category by running each feature point through a fully linked layer.The feature point is regarded as a useful feature and will be reserved for later fusion when the maximum likelihood of the anticipated outcome after softmax is bigger than a specific amount. The feature point, on the other hand, is viewed as being less useful for fine-grained categorization. It creates an architecture that can be taught end-to-end based on the aforementioned ideas, as shown in Fig. 4.1. The backbone model is shown in the region enclosed by the dotted line. It utilizes $f_i \in R^{C\times H\times W}$ to express the feature map generated by the backbone network's i^{th} block , Here, H stands for the feature map's height, W for its width, and C for its size. Then, a weakly supervised selection receives this feature map as input, and a linear classifier will categorize each feature point. Following this stage, the feature map is designated as $f_i \in R^{C'\times H\times W}$ with C' is identical to the amount of target classes there are. After each feature point's class prediction probability has been calculated using softmax, the initial few feature elements with a high score will be picked in the weakly supervised selector.

It features two distinct architectures that work together to achieve the feature fusion. The

first is put into practice using a completely linked layer. assuming that the overall number of feature points selected is N. The feature maps are first concatenated for the feature dimension before being fed to the fully linked layer. As a consequence, after passing through the fully linked layer, the $R^{N\times C}$ dimesional feature map generates a dimension $R^{C'}$ based prediction result. This architecture enables the combination of specific local properties to produce global features that can describe the entire image.

Graph convolution, a method used to create the second architecture, interprets all feature points as a graph structure with nodes that represent features at various spatial scales and locations. The graph convolutional network receives the input graph and uses it to learn the connections between the nodes. The pooling layer then divides the feature points into a number of super nodes, and after averaging the characteristics of these super nodes.The prediction is completed by using a linear classifier model. This method has the benefit of allowing for more effective integration of each point's attributes without affecting the backbone model's output of results. This leads us to our final feature fusion approach, graph convolution.

Furthermore, the model uses a feature pyramid network (FPN) as its backbone network to effectively combine features of various sizes and produce more accurate identification results, enabling it to more successfully extract the properties of tiny regions.

4.1.2 The process of forward and backward propagation

This architecture uses just image-level annotations as training targets because its primary function is fine-grained categorization. No additional artificial labels are used. The initial training objective is to provide each feature map's f_i features the capacity to categorize. As demonstrated in the accompanying Eq 4.1 [20], it first averages the prediction output of all feature points before calculating the total loss (4.1) where $f_{l,s} \in R^{C'}$ The size of this block is H × W, and R denotes the feature point in the location s in the l^{th} block of the feature map. S stands for the output feature map space of this block. Next, Cross Entropy is used to determine the block's overall class loss, as indicated in Eq (4.2) [20].

$$Z_l = \frac{1}{H \times W} \sum_{s \in S} f_{l,s} \tag{4.1}$$

$$L_b = -\sum_{l \in L} \log(Z_l^{cls}) \tag{4.2}$$

The selected position of the feature map is indicated by $Mask \in R^{H\times W}$, in along with training the overall classification. The binary data pattern of this *Mask* designates the selected region as 1 and the dropped area as 0. As a result, $S \odot (Mask)$ indicates lost feature points whereas

$S \odot (\sim Mask)$ indicates feature points with high discrimination. The characteristics of the chosen region in the l^{th} block are represented by the sum in equation (4.3) with the notation [20]". Eq. (4.4) [20] indicates the loss function of the chosen category, and Eq. (4.5) [20] denotes the total of the characteristics that were not chosen. The dropped area's flattening loss function is L_s defined by Eq. (4.6) [20].

$$h_l = \sum_{s \in S \odot (Mask)} f_{l,s} \tag{4.3}$$

$$L_s = -\sum_{l \in L} \sum_{i \in C} \log(h_l^i) \tag{4.4}$$

$$n_l = \sum_{s \in S \odot (\sim Mask)} f_{l,s} \tag{4.5}$$

$$L_s = -\sum_{l \in L} \sum_{i \in C'} \log(1 - n_l^i) \tag{4.6}$$

It is less useful for categorization when the output is flattened. This method is comparable to determining the foreground or background's "score" in an object identification framework. The score in this technique is calculated using the softmax's highest probability value.

Now, the choosen feature $f_s \in R^{N \times C}$ pass through the Combiner to produce mixed-scale prediction outcomes, the output feature is $f_{comb} \in R^{N' \times C}$, where N' represents how many super nodes remain after condensing the input nodes. In order to generate predictions, a linear classifier receives the averaged information from these super nodes. Cross entropy is used to compute the Combiner category prediction loss, and the loss function is denoted by L_c. The definition of the total loss function is Eq.(4.7) [20]. The weighted average of the previous loss functions makes up this loss function, where λ_b, λ_s,λ_n and λ_c are the weights of L_b, L_s, L_n, and L_c, respectively:

$$L = \lambda_b L_b + \lambda_s L_s + \lambda_n L_n + \lambda_c L_c \tag{4.7}$$

To anticipate classes, this approach makes advantage of local characteristics. The projected values of local background areas or locally comparable elements of distinct classes are therefore difficult to "distinguish." Maximum anticipated probability is a powerful selection criteria in this situation. To finish the final forecast, the chosen local features are combined with the global data. The dataset has to be fixed so that fine-grained categorization may be used.

4.2 Cross-domain setting

Arranging the datasets is required before performing domain adaptation. In this experiment, both the source domain and the target domain are used. The target domain "CompCarsSV" is for security camera photos of automobiles, whereas the source domain "CompCars" is for online photographs of cars. Both domains are thought to feature two different kinds of classes: base classes and novel classes. Both classes in the source domain feature photos with labels. The target domain, on the other hand, has only tagged pictures for the base class. It is a supervised partially zero-shot scenario, and the objective is for the new class of the target domain to learn from the source domain and partially also from its own domain. Consequently, a novel kind of target domain serves as the experiment's primary test domain.

4.3 Adversarial Domain Adaptation

A gradient reversal layer connects the feature extractor and domain classifier and multiplies the gradient during backpropagation-based training by a specific negative constant to achieve domain adaptation. The symbol for this constant is *lambda*. In all other cases, training proceeds normally and minimizes both the label prediction loss and the domain classification loss. Gradient reversal layer ensures that the feature distributions over the two domains are equal (as similar as possible to each other for the domain classifier), resulting in the domain-invariant features. When discrimination performance is subpar, coupling the effect to the loss lessens the impact on the main network, resulting in more stable training [109]. The coupling is implemented as

$$\lambda(i) = \exp(L_d(i-1)) \tag{4.8}$$

with an exponential function mapping the discriminator's negative log-likelihood loss $L_d(i) \in [0,\infty)$ to the influence factor $\lambda \in (0,1]$ [109]. with an exponential function mapping the discriminator's negative log-likelihood loss Ld(i)[o,)to the influence factor (o,1]. The *gradient reversal layer (GRL)*, described as follows, is a key concept for this domain adaption. No parameters are connected to the gradient reversal layer. The GRL serves as an identity transformation while the signal is being sent. To alter its sign, or to multiply it by 1, the GRL takes the gradient from the following level and applies it to the prior layer during backpropagation. By defining simply the forward propagation and backpropagation operations [51], it is easy to implement such a layer using the current object-oriented deep learning tools. The layer doesn't need its parameters updated. The architecture shown in Figure 4.1 is created by inserting the GRL as described above between the domain classifier G_d and feature extractor G_d.

The whole cross-domain fine-grained classification model is depicted in Figure 4.1. To summarize, A backbone, in this case SWin-T, is used to transform input images into feature maps, which are then sent to a weakly supervised selector where the desired discriminative area is found, followed by a combiner, and the noise area is flattened. Once the feature map has passed through the backbone, the input image is ready for further processing. The local feature that is chosen is merged with the global feature to represent the entire image in the combiner. Following the completion of the graph convolution, the feature points are combined into a number of super nodes through the pooling layer, with all the chosen features being handled as graph structure. It must be noted that FPN is also present in the backbone network in order to extract features more precisely. The necessary characteristics are then processed by an image classifier and a domain classifier. Instead, a gradient reversal layer that increases the gradient by a specific negative constant connects the domain classifier and the feature extractor during backpropagation-based training. This approach categorizes class labels and domain labels more successfully and effectively.

5 Implementation

This chapter explains implementation details required to produce the results of this work. Section 5.1 details the implementation environment and used frameworks. Section 5.2 explains inputs of the model. Finally, section 5.3 lists the training settings.

5.1 Implementation Environment

In an application concerning machine learning where data driven model is used to capture pattern , to record data,or to predict some future events normally a special computing setup is required, to handle the peculiarities of this application.For this type of application, one side it has the recorded data, and it try to learn the model from this data, on the other side the model tries to capture the pattern inside the data. At the end when the model is trained or learned we try to use this model.First we check the quality of the model ,this process is basically called evaluation.When the evaluation is done and we are happy with the performance of the model then we want to use it in the real environment in a real application. In summary there are four steps to be considered for preparing a machine learning model. First, preporcessing input data, second training the model,third step is to store the trained deep learning model and fourth is deployment of the model. So, it is already cleared that we need a special computing setup. Just like any other computing setups we can easily distinguish the characteristics of our required setup in some minimum requirement.We can divide this requirement into hardware and software.

The application concerning neural network generally we have dataset, inside the dataset we have multiple data threads or data samples that we would like to process to generate some sort of outcome. This outcome is basically the result of the model. Normally in a neural network model , we end up applying the same mathematical function to different parts of the same dataframe. This happen both during the training process also in evaluation and testing.Because its a very similar mathematical function, which is applied to different parts of the same data; it would be helpful to have some sort of hardware where there are multiple parallel hardware threads or processors available which can execute same mathematical functions.Nowadays there are different hardware that can do such processing. For example we can have multi core

processor/CPU(Central Processing Unit) from Intel which has 128 cores or 256 cores. Also we can have an FPGA(Field Programmable Gate Array) based setup where this little functions are actually implemented directly in hardware, we can replicated the same process multiple times in hardware and then it is possible to process all the data in parallel in this replicated hardware units.

And last but not the least the most interesting hardware component we can use for such processing is *GPU*. GPU stands for Graphics Processing unit.The architecture of the GPU includes several thousands of hardware processors that are connected in parallel which can execute exactly the same function but of different types of data. GPU is basically developed for graphics but nowadays it is being used more and more for neural network applications.Another part of the hardware which also must be considered the storage capacity. There are two types of data storage capacity we think of.One type of data storage capacity is basically mass storage where we have dataset,this is normally done using Flash or SSD. This is only for the data storage. Another type of storage that we have to considered is the memory needed for processors to load and store the result of each operation, so this type of memory is generally known as RAM-random access memory.

So from the hardware point of view we can make a list, for storage we need enough storage to store the dataset and for memory we need enough memory to load the part of the dataset and also store intermediate results, for processing units the minimum requirement is that we need enough processing cores available so we can train the complete model in a reasonable time.Those are the minimum requirements for hardware.

The minimum requirement for software is we need a framework or some tools which is general enough to train different kinds of machine learning models,which provides us with tools to evaluate performance of the model and provides us with reusable functions ,load the data and process them using standardized mathematical functions. Those are minimum requirements for software. In the sub section the distinct part of the environment is explained.

5.1.1 Hardware

CPU

CPUs are built to handle sophisticated logic and serial operations. Their architecture, which has fewer cores and greater cache memory to swiftly acquire complicated instructions, reflects this. However, compared to CPUs, GPUs offer a greater throughput due to their large number of tiny cores for straightforward computing. Even with a laptop lacking a GPU, it is feasible to study the basics of machine learning [87]. In that scenario, the CPU would handle every operation. It's OK to carry out ML activities on a CPU whilst studying. However, as the datasets

grow, a high-quality GPU is required to support our activities. With the aid of an analogy, it's simple to comprehend how CPU and GPU operations differ from one another.

Let's say we have 10 really quick drones, and we want to deliver pizza. We can deliver up to 10 items at once since each drone can only deliver one order at a time. However, someone will have to wait for one of the drones to serve a pizza first if there are more than ten orders. What happens if we receive 500 orders? Despite the speed of our drones, it will take a while to deliver all the pizzas.

CPUs enable targeted and quick sequential processing in this way. It might be modified to deliver more than one pizza at once. Even so, CPU activities will always be sequential—for example, delivering one order before moving on to the next.

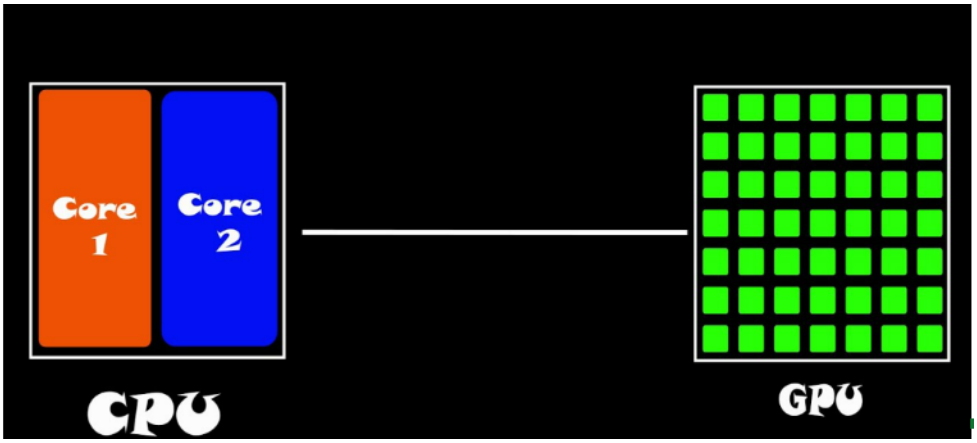


Figure 5.1: CPU vs GPU [21]

Contrarily, a GPU is comparable to 1000 drones flying at an average speed. We'll be able to ship 500 orders significantly quicker even though they're slower. This is how we can deliver them all at once. Pizzas may be delivered in large numbers simultaneously, making up for the slower speed. In conclusion, CPUs are quicker than GPUs and have the ability to handle several tasks simultaneously. The higher number of cores on GPUs, however, cancels out the quicker performance of CPUs [24]. GPUs perform better than CPUs thanks to their tens of thousands of cores, which enable them to conduct several tasks at once. They are thus more appropriate for tasks like deep learning, machine learning, or mining for virtual currencies.

GPU

Applications that require a lot of data, like artificial intelligence and machine learning, benefit from GPU-accelerated computing. Businesses may leverage this processing capacity without

having to install a GPU on each user's local device thanks to a *Cloud Graphics Processing Unit* . Machine learning algorithms train themselves by consuming enormous volumes of data and processing it repeatedly through their own models, figuring out optimal practices and tactics connected to certain data sets – operations best suited for massively parallel processing. For example, hyper specialization, cost reduction, energy reduction, and scalability [24] are just a few of the many advantages cloud GPUs provide to the cloud computing industry.

Nvidia GPUs are more suited for machine learning than other GPUs. They have a firmly established CUDA software development kit (SDK) which covers all important machine learning frameworks [86]. The usefulness of Nvidia's GPUs has also improved thanks to its programming frameworks and tools. The OpenCL SDK, which supports AMD GPUs, is a substitute for CUDA. The major ML frameworks aren't supported out of the box though. Hopefully soon, AMD will emerge as a desirable option for machine learning GPUs and OpenCL will support ML frameworks [87]. Market prices for some really good cards are really reasonable. Nvidia GPUs are the best option right now if you want to conduct machine learning without experiencing any serious issues. Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation's GPU server was utilized in this work (IOSB). They have many types of servers, including those with the names Pallas, Taurus, Titan, Deneb, Sirius-a, Sirius-b, and so on. There are 8 Tesla V100 GPUs utilized in this work, with a video memory size per GPU of 32 gigabytes, and Sirius-a and Sirius-b are employed in alternate configurations. for our computer configuration, which is enough. The NVIDIA Tesla V100 Tensor Core is the most sophisticated data center GPU ever created to enhance high performance computing (HPC), data science, graphics, and artificial intelligence (AI). It has a single GPU that can perform as well as 100 CPUs according to NVIDIA Volta architecture and is available in 16 and 32 GB variants [30].

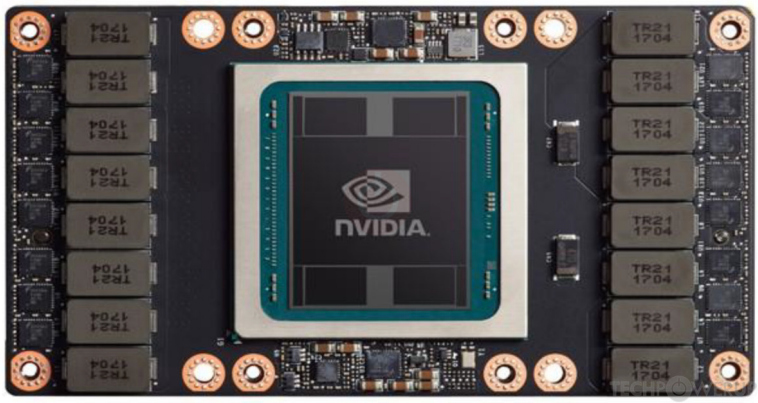


Figure 5.2: NVIDIA Tesla V100 SXM2 32 GB [73]

Built using a 12 nm technology and based on the GV100 graphics engine, the NVIDIA Tesla V100 SXM2 32 GB is a professional graphics card that supports DirectX 12. The GV100 graphics processor is a large device with a die area of 815 mm2 and 21,100 million transistors [73]. It has 5120 shading units, 320 texture mapping units, and 128 ROPs. 640 tensor cores are also incorporated, which accelerate machine learning applications. The Tesla V100 SXM2 32 GB and 32 GB of HBM2 memory from NVIDIA are coupled through a 4096-bit memory interface. Memory runs at 876 MHz while the GPU operates at a frequency of 1290 MHz, which may be increased to 1530 MHz [73]. The NVIDIA Tesla V100 SXM2 32 GB is a dual-slot card, thus it

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.44       Driver Version: 440.44       CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:06:00.0 Off |                    0 |
| N/A   31C    P0    42W / 300W |      0MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  Off  | 00000000:07:00.0 Off |                    0 |
| N/A   33C    P0    42W / 300W |      0MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  Off  | 00000000:0A:00.0 Off |                    0 |
| N/A   73C    P0   293W / 300W |  14361MiB / 32510MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  Off  | 00000000:0B:00.0 Off |                    0 |
| N/A   38C    P0    67W / 300W |  14343MiB / 32510MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla V100-SXM2...  Off  | 00000000:85:00.0 Off |                    0 |
| N/A   36C    P0    57W / 300W |  18801MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla V100-SXM2...  Off  | 00000000:86:00.0 Off |                    0 |
| N/A   39C    P0    56W / 300W |  18801MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla V100-SXM2...  Off  | 00000000:89:00.0 Off |                    0 |
| N/A   68C    P0   279W / 300W |  11299MiB / 32510MiB |     80%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla V100-SXM2...  Off  | 00000000:8A:00.0 Off |                    0 |
| N/A   57C    P0   267W / 300W |  11367MiB / 32510MiB |     78%      Default |
+-------------------------------+----------------------+----------------------+
```

Figure 5.3: sirius-b GPU server status

doesn't need a separate power connector; its maximum power usage is 250 W [73]. As it is not intended to have displays connected to it, this device lacks display connection. Using a PCI-Express 3.0 x16 interface, the Tesla V100 SXM2 32 GB is linked to the remaining hardware.

Before using the GPU we must know if its free or not. With the nvidia-smi command we can see the status of the GPU before using it as the fig. 5.3 shows.

From the figure we can see on that time only GPU id 2 and 3 are busy, so we can use other GPU. Its necessary to know the status to avoid overloading, overloading can make process very slow.

FPGA

As has previously been mentioned, an FPGA-based configuration is also an option for a computer environment. It can create compute circuits with thousands of memory units that function similarly to GPUs and their CUDA threads (Compute Unified Devie Architecture) [86]. Because of its customizable design, FPGAs may be further optimized for higher throughput. FPGAs are a feasible alternative to GPUs as a result of the potential amount of calculations. FPGAs might be the best choice for embedded applications since they use less power in comparison. In safety-critical activities like vehicle ADAS (Advanced Driver Assistance Systems), they are also a recognized standard.

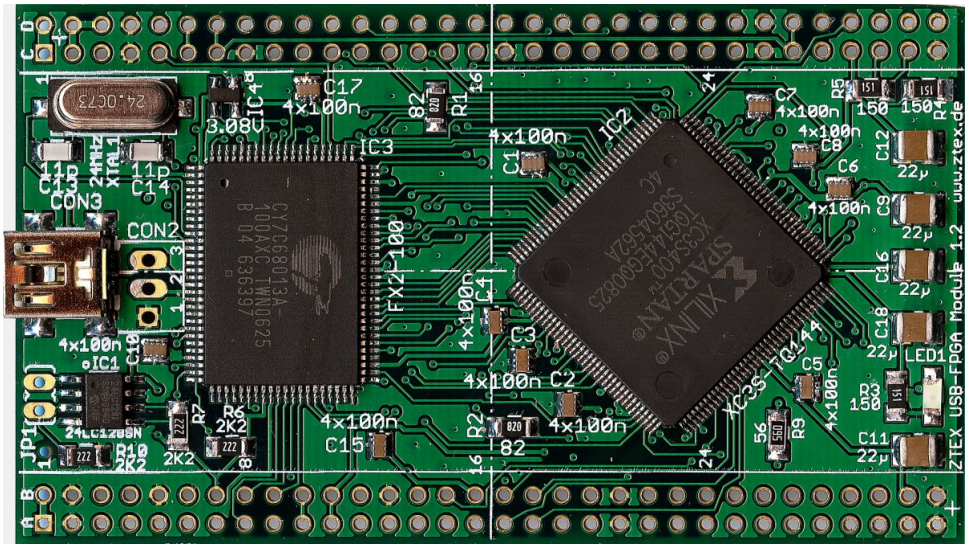


Figure 5.4: FPGA [98]

In contrast to GPUs, which are constrained by design, FPGAs may implement bespoke data types. It is advantageous to have the adaptability that FPGAs provide as neural networks change in numerous ways and penetrate additional sectors. A piece of hardware with customizability is an FPGA [98]. You may see it as a sea of floating logic gates. A hardware description language (HDL), like Verilog or VHDL, is used by a designer to create a program. The connections created

and the manner in which they are carried out utilizing digital components are determined by that software. HDL is also known as RTL (register-transfer level) [86] language.

The challenging aspect of using ML frameworks developed in higher level languages like Python on FPGAs is their implementation. Although it is written in code to specify hardware elements like registers and counters, HDL is not by nature a programming environment. Verilog and VHDL are only a few HDL languages. Additionally, the cost of such a setup is quite high. At the moment, models are trained on a GPU before being put on an FPGA for real-time processing when money is not an issue. For us, it is preferable to continue using GPUs for academic research.

5.1.2 Software

Through the use of machine learning, which is a form of artificial intelligence, software programs may anticipate outcomes more accurately without being explicitly instructed.Machine learning algorithms use past data as input to predict the new output value. The ML code works wonderfully when executed through software. There are various software programs and applications available. TensorFlow, , Amazon Machine Learning , Hogun Google Cloud ML Engine, PyTorch, and Keras are a few of the most well-known frameworks among them. If we begin by introducing each software, there will be a significant discussion. Therefore, it would be best to go straight to what we really used in our job. For this Linux server, the operating system comes first. The many tools we employed in this investigation are shown in Figure 5.5. The next section will provide a quick overview of each tool.

5.1.3 Linux Server

Open source software includes Linux (OS). The computer program that directly controls a system's hardware and resources, such as its CPU, memory, and storage, is called an operating system [107]. All of your software is connected to the working physical resources through the OS, which stands between apps and hardware. Imagine an OS as the engine of a vehicle. Even while an engine may operate on its own, a car only becomes useful when it is coupled with a transmission, axles, and wheels. The remainder of the automobile won't function if the engine isn't operating properly. Originally intended to function similarly to UNIX, Linux has developed to run on a broad range of hardware, including everything from smartphones to supercomputers [83]. The Linux kernel, which controls hardware resources, and a collection of software packages that make up the remainder of the operating system are components of any Linux-based OS.

Figure 5.5: Different tools and library used in this work [1], [2], [3], [4], [5], [78], [8], [94], [106], [9], [76]

For machine learning and artificial intelligence applications, Linux is increasingly the preferred operating system [11]. Without a doubt, Linux's lack of a licence charge is one of its benefits. Large firms like TensorFlow and PyTorch use Linux to create systems with tens of thousands of processors without needing to purchase licenses for those processors. Most ML servers run Linux. Linux is therefore the obvious choice if you wish to develop for the most popular server OS. There isn't much Windows support for PyTorch and TensorFlow. As a result, a developer working on PyTorch may find it difficult to get the application running on Windows because doing so needs specific software compilation, and instead favor Linux OS.

For data-intensive operations like AI and ML, Linux also provides great performance. Devel-

opers may modify the stack to function better for their data centers because the Linux kernel is fairly easy to utilize. Linux is used by DagsHub [83], a pioneer in machine learning initiatives, for a wide range of projects.

A server that runs the Linux open-source operating system is referred to as a Linux server [11]. It gives businesses a cheap way to provide their customers with information, apps, and services. Due to Linux's open-source nature, users also have access to a large community of supporters and resources. The finest environment for Python machine learning development may be found here. The tools are simple and easy to install. Direct development and execution of big models is feasible.

5.1.4 CUDA 10.2

To use NVIDIA GPUs for general-purpose computing right now, we require CUDA. According to NVIDIA, "CUDA is a parallel computing platform and programming style that makes utilizing a GPU for general purpose computation straightforward and beautiful." Nvidia created the parallel computing platform and programming style known as CUDA for use with its own GPUs in general computing (graphics processing units)[43]. By using GPU hardware for the parallelizable portion of the calculation, CUDA enables developers to accelerate computationally expensive applications. While there have been other proposed GPU APIs, like OpenCL, and there are competitor GPUs from other manufacturers [9], like AMD, the combination of CUDA with Nvidia GPUs dominates various application areas, including deep learning, and is the basis for some of the fastest computers in the world. In our job, we employ CUDA 10.2.

5.1.5 Python 3.9.12

A high-level, interactive, interpreted, and object-oriented scripting language is Python [79]. Python is made to be extremely readable. It typically employs English terms instead of punctuation, and it has fewer syntactical structures than other languages. It is a fantastic language for novices since it can be interpreted, is interactive, and object-oriented. Simpleness and consistency, availability of excellent libraries and frameworks for AI and machine learning (ML), flexibility, platform freedom, and a large community are features that make Python the ideal choice for projects based on these technologies [108]. These raise the language's general level of acceptance. Python 3.9.0 is used in our work.

5.1.6 Visual Studio Code 1.60

Microsoft created the source-code editor known as Visual Studio Code, often known as VS Code, for Windows, Linux, and macOS [6]. Among the features are debugging assistance, syntax highlighting, intelligent code completion, snippets, code refactoring, and integrated Git. The theme, keyboard shortcuts, settings, and the installation of extensions that offer new functionality are all customizable by users. Python is fully supported by Visual Studio Code, a free source code editor with helpful options including real-time collaboration [70]. It may be easily modified to match your teaching style and classroom environment. Python with VS Code: a Perfect Match for Data Science [81]. In order to make the work of Python data scientists simpler, companies like Visual Studio Code, GitHub, Codespaces, and Azure Machine Learning have made significant investments in tools and platforms.

5.1.7 Jupyter Notebook

It all started with the Jupyter Notebook, an online tool for generating and sharing computational documents [78]. It delivers a user-friendly, efficient, document-focused experience. For the development of data science applications, Jupyter notebooks offer an interactive computing environment. Jupyter notebooks are documents that include software code, computational output, explanatory prose, and rich content. With the use of notebooks, code may be edited, executed, and the results of computations are shown. An extension of.ipynb is used to preserve notebooks [102]. With its name indicating support for Julia (Ju), Python (Py), and R, the Jupyter Notebook project offers support for dozens of computer languages [102].

5.1.8 Git

Git is a distributed version control system that is free and open source and is intended to manage any project, no matter how big or little, quickly and effectively [36].

In order to remember particular versions later, version control is a system that tracks changes made to a file or collection of files over time [56]. Let's take the scenario where two data scientists are working on the identical task to develop a machine learning model. In the event that one of them modifies the function and uploads it to a remote repository, where it is then merged with the master branch, the model becomes version 1.1. (just an example). Into version 1.1, a different data scientist also makes some adjustments to the same function, and the updated changes have now been merged with the main branch. Version 1.2 of the model is now available. If the development team discovers flaws in version 1.2 at any time after it has been released, they may easily go back to version 1.1. And version control is wonderful in that regard.

5.1.9 PyTorch 1.11.0

Deep learning models based on neural networks are created and trained using PyTorch, an open source machine learning library. The Facebook AI research team is principally responsible for its development [89]. Both Python and C++ may be utilized with PyTorch. Of course, the Python interface is better designed. Pytorch is very well-liked in research laboratories and is supported by major corporations like Facebook, Microsoft, SalesForce, and Uber. Pytorch is gaining popularity quickly, however it isn't yet widely used on production systems [89], which are dominated by TensorFlow (backed by Google) and other similar frameworks. PyTorch employs dynamic computation instead of static computation graphs like the majority of other well-liked deep learning frameworks, such as TensorFlow, giving it more flexibility for creating complicated designs [89]. Pytorch leverages fundamental Python constructs like classes, structures, and conditional loops, which are far more recognizable to humans and easier to comprehend. Compared to other frameworks that incorporate their own programming style, like TensorFlow, this makes it more simpler.

5.1.10 Anaconda 3

For scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), Anaconda is a Python and R distribution that seeks to make package management and deployment easier[10]. The Anaconda package offers the installation of Python together with a number of IDEs, including Spyder, Jupyter Notebook, Anaconda prompt, etc. Therefore, it is a highly practical bundled solution that can be downloaded and installed on a computer with ease. Python, along with certain fundamental IDEs and libraries, will be installed automatically [67].Data science in the present day is a very active field. Astonishing advancements and brand-new difficulties are both present every day. Organizations must equip their teams with the most flexible, potent data science and machine learning technologies possible to succeed in this environment. This will allow teams to innovate quickly without compromising security and governance. Anaconda has arrived. Data scientists created Anaconda, and it was created for data scientists [94].

5.1.11 NumPy

Numerical Python is referred to as NumPy [72]. The Python package NumPy is used to manipulate arrays. Additionally, it contains matrices, fourier transform, and functions for working in the area of linear algebra [72]. In the year 2005, Travis Oliphant developed NumPy. The project is open source.

5.1.12 OpenCv

A computer vision and machine learning software library called OpenCV (Open Source Computer Vision Library) is available for free use [8]. A standard infrastructure for computer vision applications was created with OpenCV in order to speed up the incorporation of artificial intelligence into products. Along with well-known firms like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, and Toyota that utilize the library, several startups also heavily rely on OpenCV, including Applied Minds, VideoSurf, and Zeitera. OpenCV has been used for a variety of purposes, including stitching together streetview images, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, assisting robots in navigating and picking up objects at Willow Garage, detecting swimming pool drowning accidents in Europe, operating interactive art in Spain and New York, checking runways for debris in Turkey, inspecting product labels in factories all over the world, and performing rapid face detection in Japan [8].

5.1.13 Timm

Timm is an acronym for Python Image Models. In order to compile the most advanced picture classification models from the most recent articles [95], Ross Wightman wrote the Python library in 2019. He would then put them into practice and train them, building the greatest library of computer vision models that would allow users to swiftly assess and contrast their practical outcomes.

5.1.14 Wandb

The machine learning platform called Weights Biases [**weight**] allows programmers to create better models more quickly. Utilize the simple, interoperable tools from WB to monitor experiments, modify and iterate datasets, assess model performance, replicate models, see results and identify regressions, and communicate findings to colleagues.

5.1.15 pandas

For the purpose of manipulating and analyzing data, the Python programming language has a software package called pandas [76]. It includes specific data structures and procedures for working with time series and mathematical tables. The program is free.

5.1.16 Scikit-learn

Scikit-learn, originally known as scikits.learn and frequently referred to as sklearn, is a free machine learning package for Python [85]. It includes different methods for classification, regression, and clustering, such as support-vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is made to work with Python's NumPy and SciPy libraries [85].

5.1.17 PIL

The Python Imaging Library (PIL) gives the Python interpreter access to image editing features [80]. A class with the same name that is used to represent a PIL image is offered by the Image module. The module offers a variety of factory tasks as well, including the ability to load photos from files and generate brand-new images [80].

5.2 Model Parameters and Implementation details

We have everything set up for computation and are almost ready to execute our model. It's time to introduce the model parameters at this point. For this work, the Comprehensive

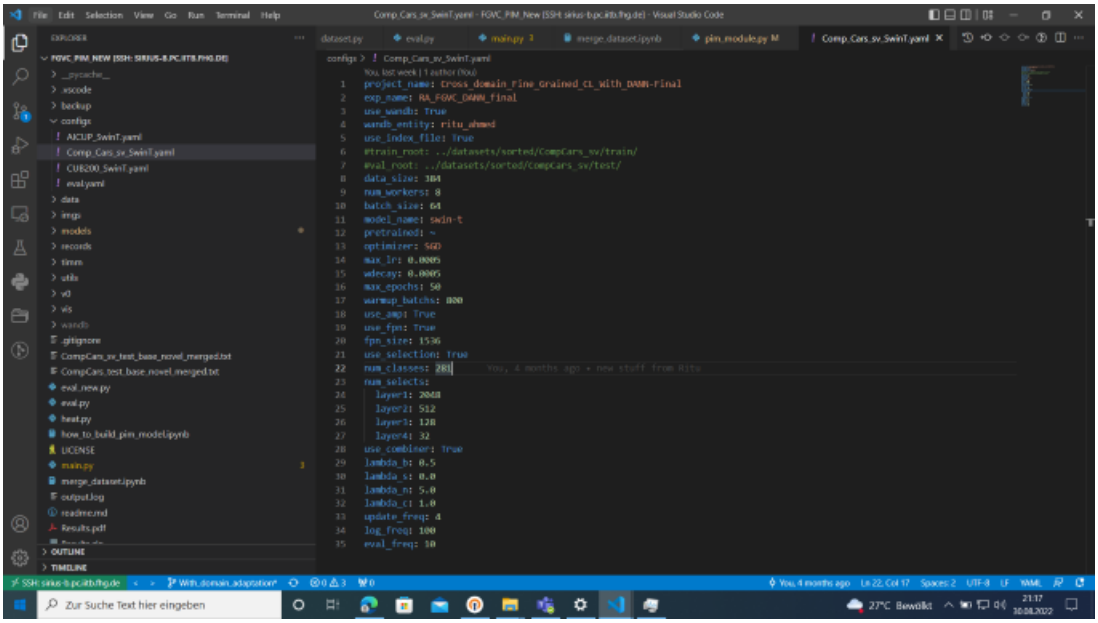


Figure 5.6: Parameters of the model

cars[114] dataset is utilized. As was previously defined, CompCars refers to online pictures and ComCarsSV to surveillance photos. For this experiment, there are 181 classifications of

automobiles in the dataset. Only the base class of CompCarsSV and the base and novel classes of CompCars are combined for training, totaling 28,007 pictures. However, the test set is separated into two types, one of which just includes the CompCars dataset and has 7,301 test photos, and the other of which includes the CompCarsSv Domain and has 9,659 test images. 16,960 photos in total will be tested and evaluated. Despite the fact that the datasets include keypoint locations and image-level annotations, only image-level annotations will be utilized in this work. The input picture is a 384×384 color image, and Swin-T[63] is employed as the network's backbone. The following are the techniques for data augmentation. When the input picture size is 384 by 384, the image is initially scaled to 510 by 510, and when it is 448 by 448, it is scaled to 600 by 600. Randon Crop, Random HorizontalFlip, and Random GaussianBlur are used to supplement the data in the training phrase, whereas Center Crop is utilized in the testing phrase.

SGD is used as the optimizer, and the learning rate is set to 0.0005, the cosine decay is employed, and the weight decay is set to 0.0005. A straightforward yet very effective method for fitting linear classifiers and regressors under convex loss functions, such as (linear) Support Vector Machines and Logistic Regression, is stochastic gradient descent (SGD) [7]. SGD has been present in the machine learning field for a while, but in the context of large-scale learning, it has just lately attracted a lot of interest. and the batch size is set to 64. In machine learning, the phrase "batch size"[71] refers to the number of training samples used in a single iteration. 50 epochs in total are trained.All tests are finished on a Fraunhofer Sirius-a or Sirius-b server with four Nvidia Tesla V100 [73] GPUs. The primary substrate for implementation is the Pytorch [89] toolset. The course takes roughly 11 hours to complete.

Firstly, The model is evaluated for 1 epoch then after every 10 epochs until 40 epochs. After 40 Epochs each epoch is evaluated until the maximum epochs 50 to see the variation of the results. The Evaluation techniques and results are explained in the next section.

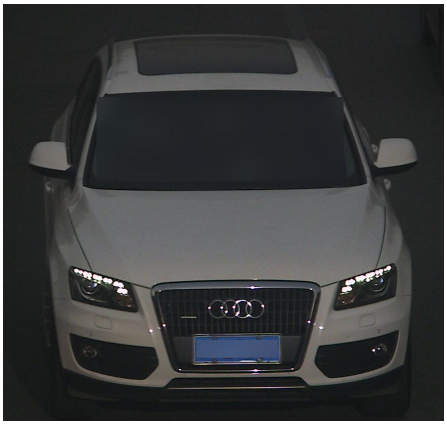
6 Evaluation

The performance of the proposed models across two domain dataset named CompCars [114] is evaluated in this chapter. The results are used to discuss the merits and drawbacks of adversarial domain adaptation.Our main concern is the performance of novel classes of CompCarsSV which is not used during training. Before the results can be analyzed, section 6.1 introduces the datasets used throughout experiments. Section 6.2 additionally explains the metrics used for evaluation. The baseline used for comparisons is introduced in section 6.3. Next, the performance of the FGVC-PIM[20] without domain adaptation is analyzed in section 6.4, while the performance of FGVC-PIM with domain adaptation are discussed in section 6.5. Lastly, the two models performance is compared in section 6.6.

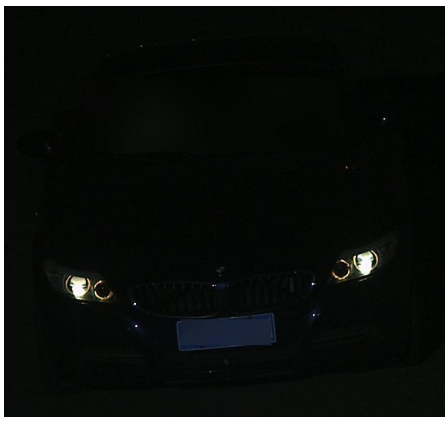
6.1 Dataset

Images from web-nature and surveillance-nature are among the data from two scenarios that are included in the CompCars dataset. Car forums, public websites, and search engines are where web-nature photographs are gathered [114]. Surveillance cameras gather pictures of a surveillance-related nature. The information from these two situations is frequently applied in real-world settings. The web-nature data in particular comprises 181 kinds of automobiles, which represent the majority of the commercial car models produced during the last 10 years [114]. 28,007 photos in all show the whole collection of automobiles. 16,960 front-view photos of cars were taken for the 16,960 surveillance-nature photographs used in this investigation. The bounding box, model, and color of the automobile are tagged on each image in the surveillance-nature division. The fact that the data from the surveillance-nature and the web-nature in Fig. 6.1 and Fig. 6.2 are noticeably dissimilar suggests the considerable difficulties in cross-scenario automobile analysis. The dataset is accesible with index file. The text file of novel class and base class of web images are merged together with surveillance images. This data set is use to train the model. Fig. 6.3 shows how the train file looks.

The domain class is represented by the first word, followed by the address of the pictures and, finally, the class label, which is represented by the numerical value. Other details, such as the brand name, model name, and make year, are included in the image's path. The index



(a) Surveillance Image- Audi Q5



(b) Surveillance Image- BMW Z4

Figure 6.1: Images from CompCarsSV Dataset

(a) Surveillance Image- BMW X6

(b) Surveillance Image- Audi Q7

Figure 6.2: Images from CompCars Dataset

file for our test dataset is similarly defined. The outcomes of the novel class and base class are separated during assessment after obtaining F1 Score Accuracy from each test dataset, and the average accuracy score is then determined for each class. The output of novel classes label correctness of the target domain may then be focused on by separating the findings from other classes in this manner.

CompCars_train_index_shared_base_0_shared_novel_0_CompCarsSv_train_index_shared_base_0.txt

deeplearning > users > rit18165 > datasets > sorted > CompCars > CompCars_train_index_shared_b

```
1  CompCars/train/Audi Audi A1 2012/5db36b188866e7.jpg=0
2  CompCars/train/Audi Audi A1 2012/a4b3a51386941c.jpg=0
3  CompCars/train/Audi Audi A1 2012/8b15602bb6d261.jpg=0
4  CompCars/train/Audi Audi A1 2012/964427b79f4b5c.jpg=0
5  CompCars/train/Audi Audi A1 2012/79fecc390c4242.jpg=0
6  CompCars/train/Audi Audi A1 2012/b8f5b83a2a411e.jpg=0
7  CompCars/train/Audi Audi A1 2012/55108a053ffb0d.jpg=0
8  CompCars/train/Audi Audi A1 2012/8903f2aa0b09a3.jpg=0
9  CompCars/train/Audi Audi A1 2012/cda8a04583387e.jpg=0
10 CompCars/train/Audi Audi A1 2012/82b5c278e248b7.jpg=0
11 CompCars/train/Audi Audi A1 2012/8439bd5aeaab9e.jpg=0
12 CompCars/train/Audi Audi A1 2012/dda00c3509dacd.jpg=0
13 CompCars/train/Audi Audi A1 2012/6320279a22a583.jpg=0
14 CompCars/train/Audi Audi A1 2012/0667782d5b94c4.jpg=0
15 CompCars/train/Audi Audi A1 2012/7d94743633ef18.jpg=0
16 CompCars/train/Audi Audi A1 2013/707b5318136b69.jpg=0
17 CompCars/train/Audi Audi A1 2013/b359a4cca003e0.jpg=0
18 CompCars/train/Audi Audi A1 2013/a4bfc503836385.jpg=0
19 CompCars/train/Audi Audi A1 2013/6e74f043ed712b.jpg=0
20 CompCars/train/Audi Audi A1 2013/631b48461cb79b.jpg=0
21 CompCars/train/Audi Audi A1 2013/e0dcb00d37d125.jpg=0
22 CompCars/train/Audi Audi A1 2013/2b9e486c0d5fd6.jpg=0
23 CompCars/train/Audi Audi A1 2013/12aba01afe082c.jpg=0
24 CompCars/train/Audi Audi A1 2013/bdfc20cec33bd8.jpg=0
25 CompCars/train/Audi Audi A1 2013/c18f454077ad55.jpg=0
26 CompCars/train/Audi Audi A1 2013/a19fcbd7310fbd.jpg=0
27 CompCars/train/Audi Audi A1 2013/91af2d45787eb3.jpg=0
28 CompCars/train/Audi Audi A1 2013/f315f198f5570b.jpg=0
29 CompCars/train/Audi Audi A1 2013/fbf6d2f6ff4b11.jpg=0
30 CompCars/train/Audi Audi A1 2013/f7b9208de0ecd1.jpg=0
31 CompCars/train/Audi Audi A1 2013/a68e5a9b9fbcad.jpg=0
32 CompCars/train/Audi Audi A1 2013/c8440f1632f4c3.jpg=0
33 CompCars/train/Audi Audi A1 2014/ce7de135f85a6e.jpg=0
34 CompCars/train/Audi Audi A1 2014/01cb542f5e08fc.jpg=0
35 CompCars/train/Audi Audi A1 2014/cec1585562dc74.jpg=0
```

Figure 6.3: Snapshot from index file of training dataset

6.2 Metrics

Two significant Metrics The model's performance is assessed using the Top-k accuracy and F1 score. The frequency with which the right label appears among the top k anticipated labels is determined by this statistic (ranked by predicted scores). It is the standard version of accuracy; for instance, the top 1 accuracy only takes into account the class with the highest likelihood [88]. When two or more labels are given equal expected scores, the labels with the highest indices are picked first. This might affect the outcome if the right label is below the threshold [88].

By calculating the harmonic mean of a classifier's accuracy and recall, the F1-score integrates both into a single metric [93]. It mainly used to compare the effectiveness of two classifiers. Assume classifiers A and B have greater recall and accuracy, respectively. The F1-scores for both classifiers in this situation may be used to assess which one yields superior results. The following formula [93] is used to determine a classification model's F1-score:

$$F_1 = \frac{2(P * R)}{P + R} \tag{6.1}$$

where P represents the precision of the model classifier and R refers to Recall of the classifier.In order to calculate precision, divide the total number of true positives by the sum of true positives and false positives [27].Recall is calculated by dividing the total number of true positives by the sum of all true positives and false negatives [27].

6.3 Baseline

Since we combine fine-grained classification with domain adaptation in our work; we can take either an experimental setup from FGVC or DA as our baseline; and compare the performance of our combined methods to theirs. In this way we can investigate the improvements from our method. We choose FGVC-PIM as our baseline method. In this experiment, Swin-T, which consists of four blocks, serves as the backbone. The number of regions chosen by the Weakly Supervised Selector is output by each block. In It has been noted that the accuracy rate is not significantly affected by the number of selected locations. However, the number of regions that are chosen has a significant impact on how much the Combiner operates since the number of input nodes directly impacts how many model parameters are needed for further mixing [20].

The final number of regions chosen by them takes into account the trade-off between the number of operations and the accuracy rate and the final number of region [256, 128, 64, 32] is selcted. A single layer Graph Convolution Network(GCN) is used as combiner. In their case,

adding FPN can enhance accuracy , adding Weakly Supervised Selector also can increase some accuracy , and adding Combiner can increase accuracy a little bit than those . These findings demonstrate that, following two feature fusions, the backbone model's accuracy may be significantly enhanced. Accuracy can greatly increase by fusing different scale characteristics using a GCN-Combiner. Experimental results show that the plugin module outperforms state-of the-art approaches and significantly improves the accuracy to 92.77% and 92.83% on CUB200-2011 and NABirds,respectively [20].

In the adversarial domain adaptation field the first popular approach was proposed by Ganin et al. [31]. The reason we did not choose the Ganin et [31] as a baseline is it takes one input image a at time, it doesn't have any mini batch also, so it will not be comparable if we use this as a baseline,

6.4 Results of Fine-Grained Classification Without Domain Adaptation

This section represents Fine-Grained visual Classification on CompCars dataset. The implementation is done following the methodology in section 4. The output of the experiment is shown with graphical representation From Fig 6.4 to Fig. 6.20. Fig 6.4 represents the accuracy for novel class of Target domain and we get the F1 score accuracy .51.

Table table:1 is the summary of Results of Fine-Grained Classification of CompCars dataset without domain adaptation.

Key	Value
CompcarsSV avg f1 score base:	0.96
Compcars SV avg f1 score novel	0.51
Compcars web avg f1 score base	0.94
Compcars web avg f1 score novel	0.94
Maximum number of epoch	50
Learning rate	0.00000180851192292985 83
combiner accuracy	100
combiner loss	0.006229400634765625
total loss	6.957401275634766

Table 6.1: Summary of solo FGVC Model Accuracy

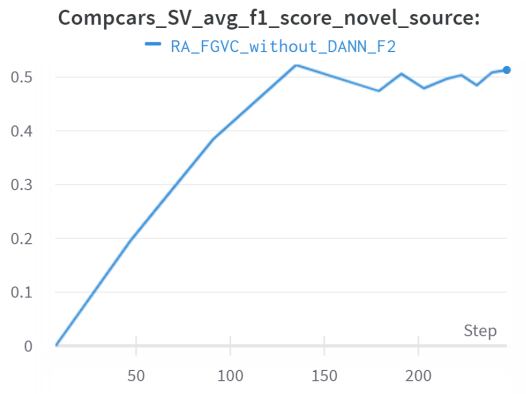


Figure 6.4: Average F1 Score for novel class of Target Domain

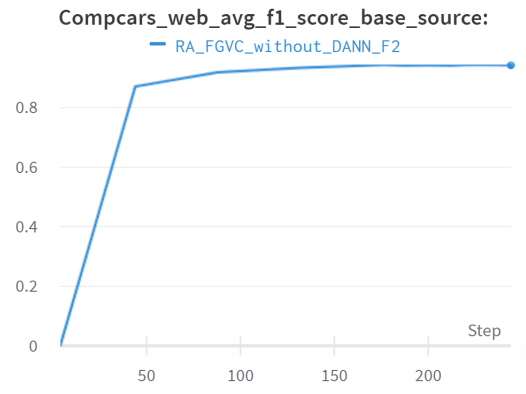


Figure 6.5: Average F1 Score for novel class of Source Domain

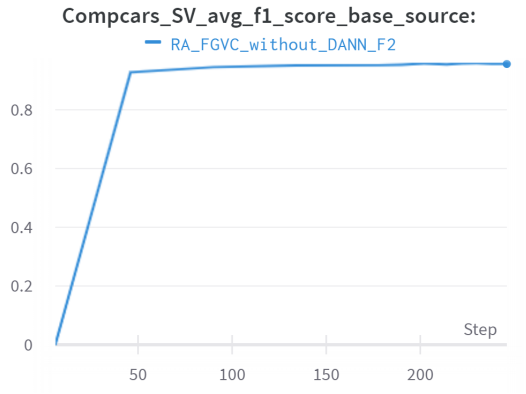


Figure 6.6: Average F1 Score for base class of Source Domain

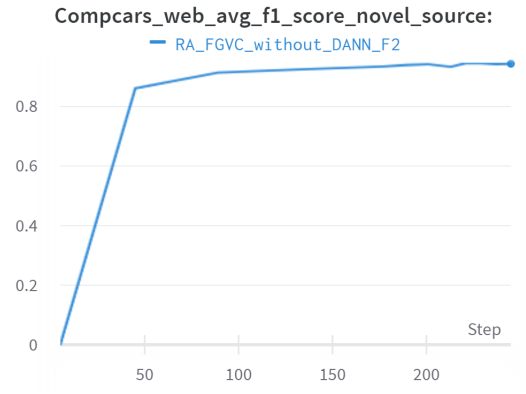


Figure 6.7: Average F1 Score for base class of Target Domain

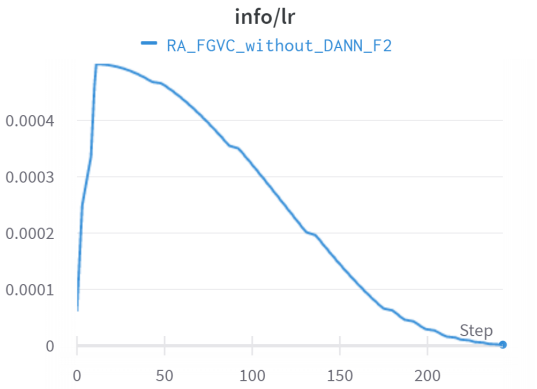


Figure 6.8: Learning Rate

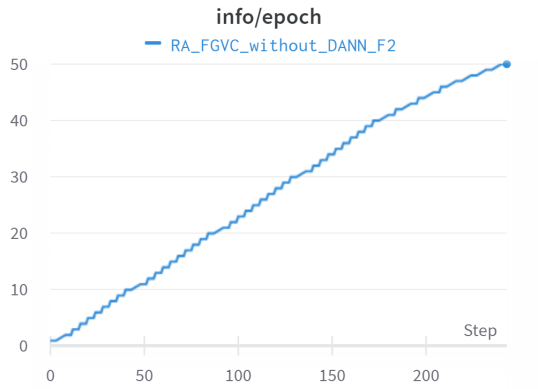


Figure 6.9: Epoch

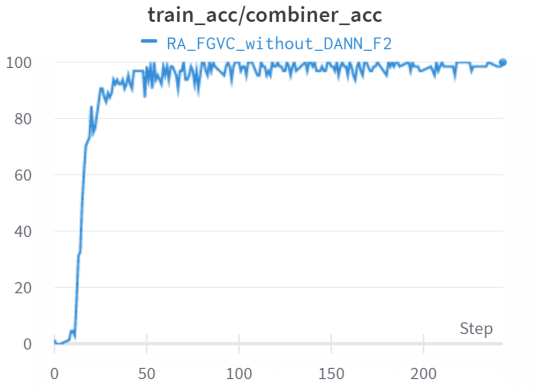


Figure 6.10: Combiner Accuracy

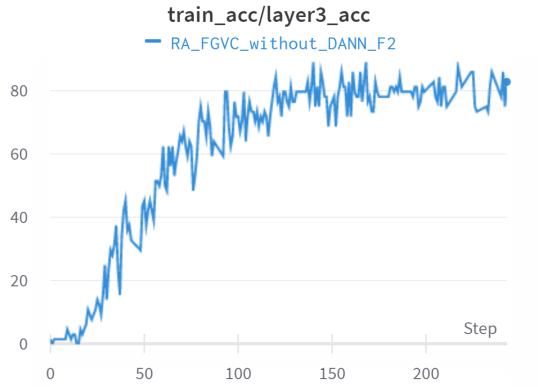


Figure 6.11: Layer2 Accuracy

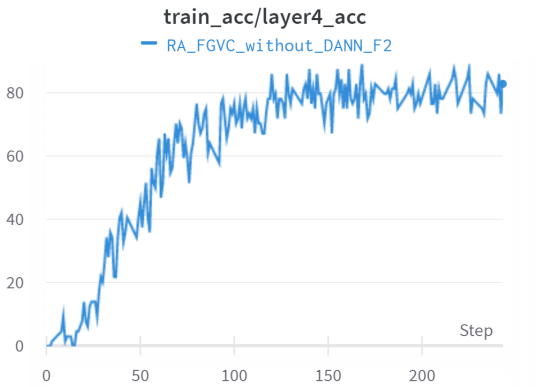


Figure 6.12: Layer3 Accuracy

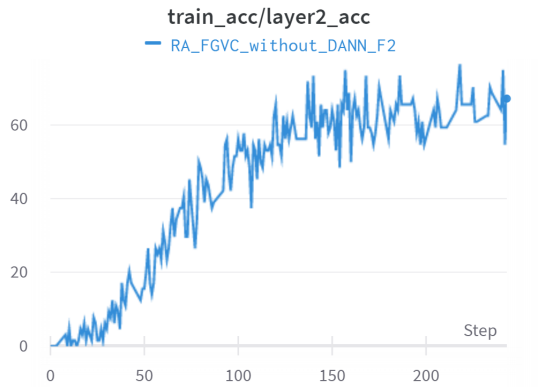


Figure 6.13: Layer1 Accuracy

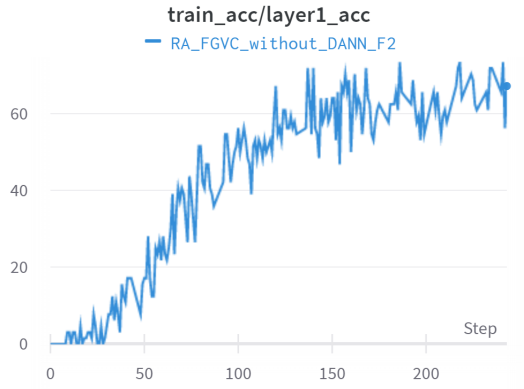


Figure 6.14: Layer4 Accuracy

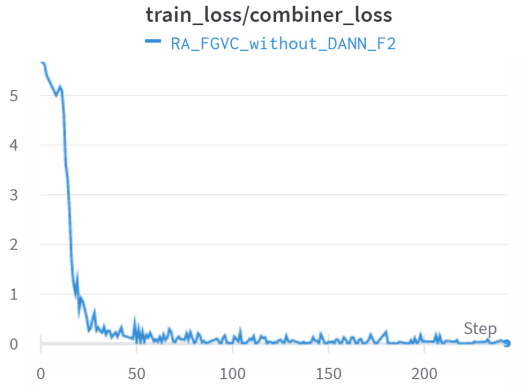


Figure 6.15: Layer3 Loss

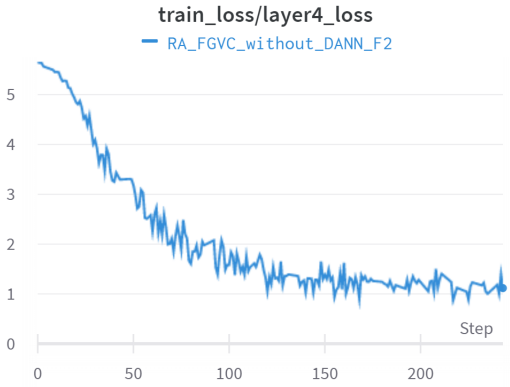


Figure 6.16: Layer1 Loss

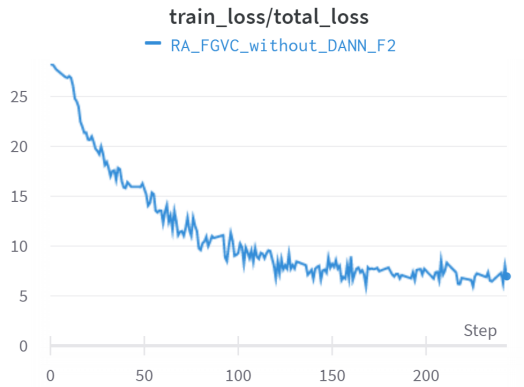


Figure 6.17: Combiner Loss

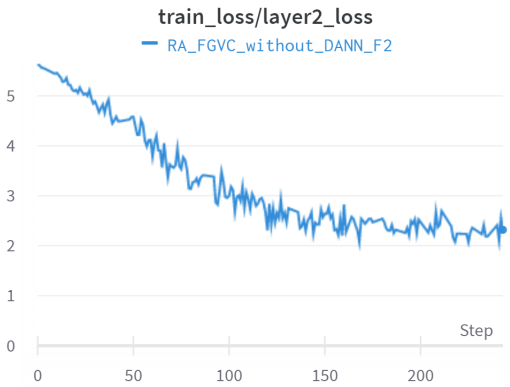


Figure 6.18: Layer4 Loss

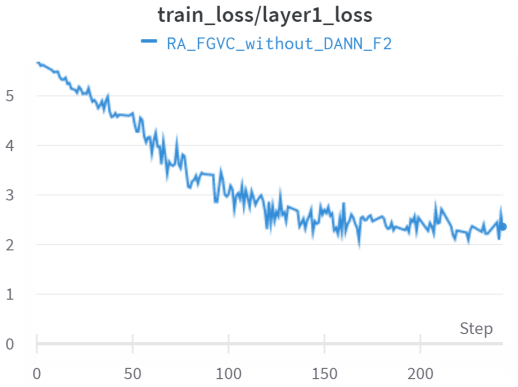


Figure 6.19: Layer2 Loss

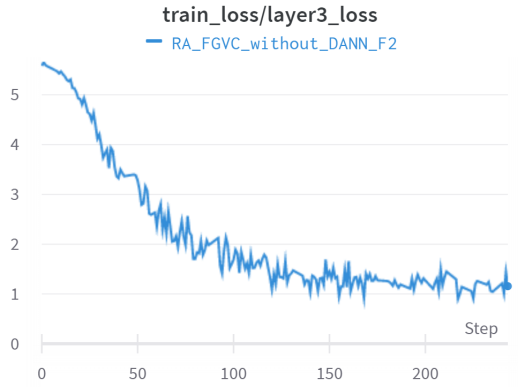


Figure 6.20: Total Loss

6.5 Results of Cross-Domain Fine-Grained Classification

This result is the combination of Fine-Grained visual Classification and supervised partially zero shot domain adaptation technique. The implementation is done following the section 5. The output of the experiment is shown with graphical representation From Fig 6.21 to Fig. 6.37. Fig 6.21 refers to the accuracy for novel class of Target domain which is our main consideration in this experiment-As it depicts we get the F1 score accuracy .61.
Table table:2 is the summary of Results of Cross-Domain Fine-Grained Classification

Key	Value
CompcarsSV avg f1 score base:	0.96
Compcars SV avg f1 score novel	0.61
Compcars web avg f1 score base	0.94
Compcars web avg f1 score novel	0.93
Maximum number of epoch	50
Learning rate	0.00000180851192292985 83
combiner accuracy	98.44
combiner loss	0.0244293212890625
total loss	7.1689605712890625

Table 6.2: Summary of Proposed Model Accuracy

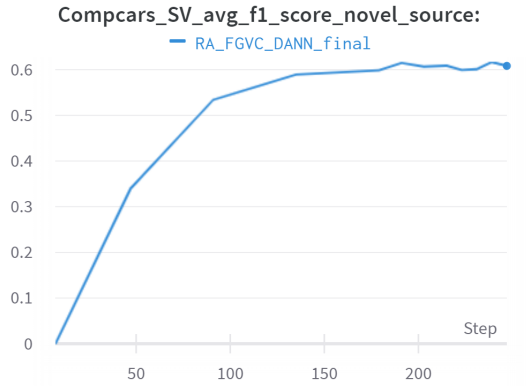


Figure 6.21: Average F1 Score for novel class of Target Domain

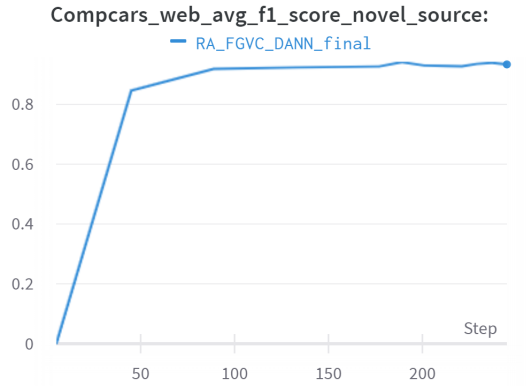


Figure 6.22: Average F1 Score for novel class of Source Domain

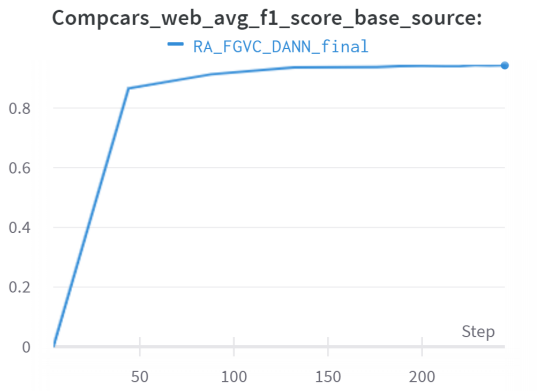


Figure 6.23: Average F1 Score for base class of Source Domain

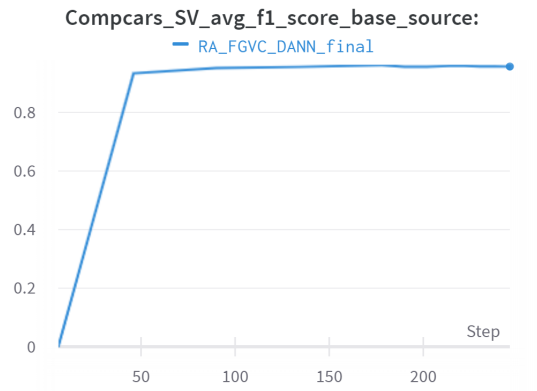


Figure 6.24: Average F1 Score for base class of Target Domain

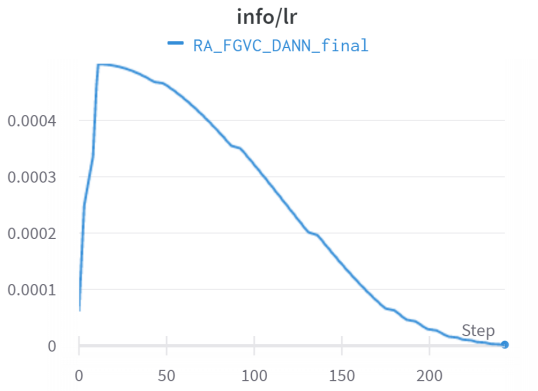


Figure 6.25: Learning Rate

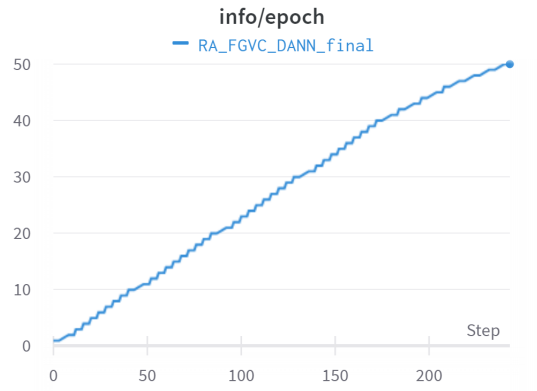


Figure 6.26: Epoch

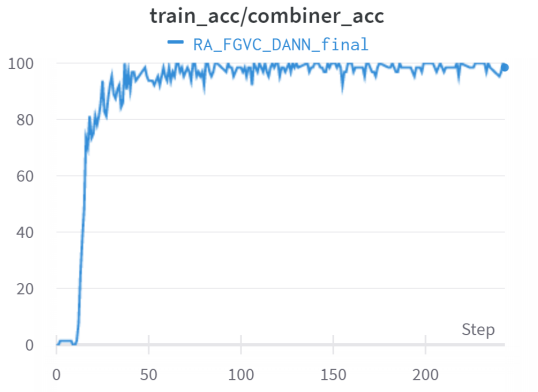


Figure 6.27: Combiner Accuracy

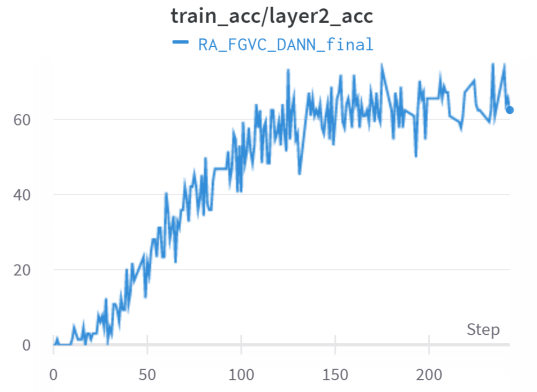


Figure 6.28: Layer2 Accuracy

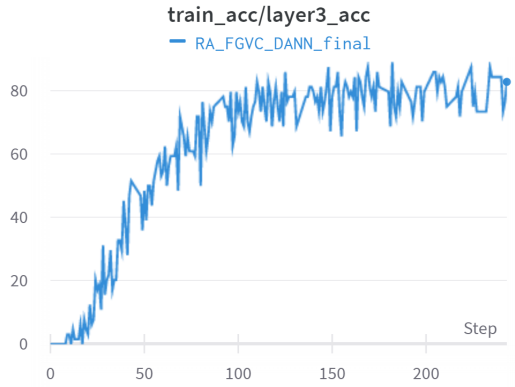


Figure 6.29: Layer3 Accuracy

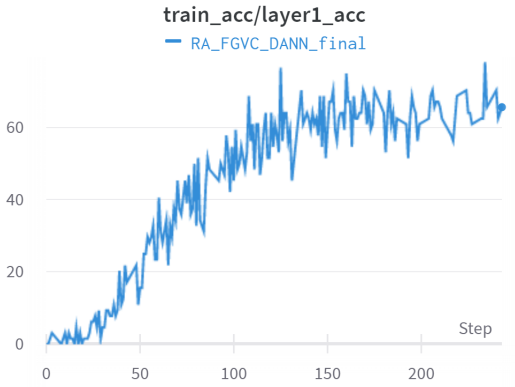


Figure 6.30: Layer1 Accuracy

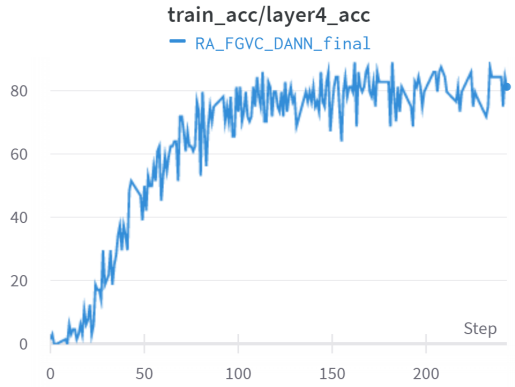


Figure 6.31: Layer4 Accuracy

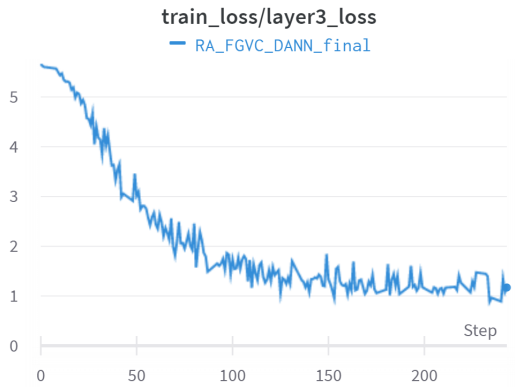


Figure 6.32: Layer3 Loss

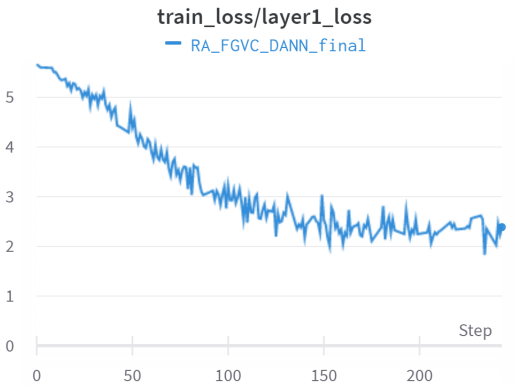


Figure 6.33: Layer1 Loss

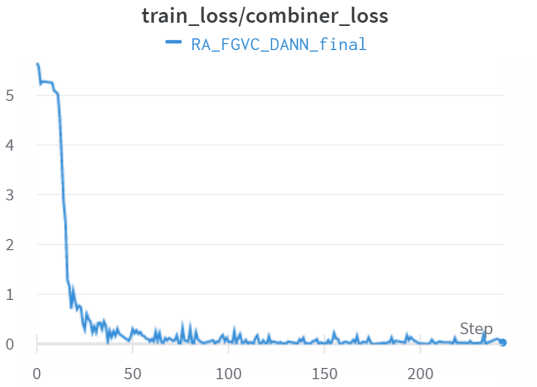


Figure 6.34: Combiner Loss

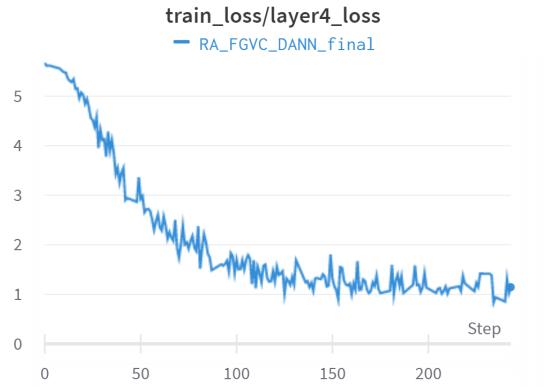


Figure 6.35: Layer4 Loss

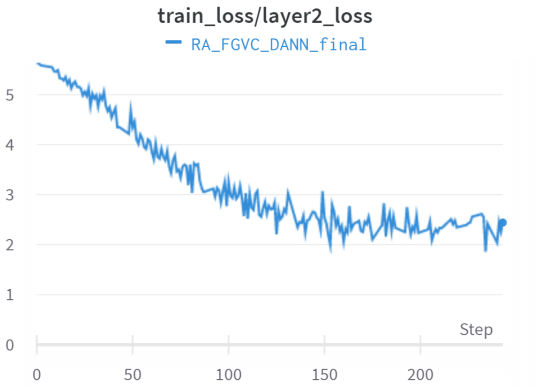


Figure 6.36: Layer2 Loss

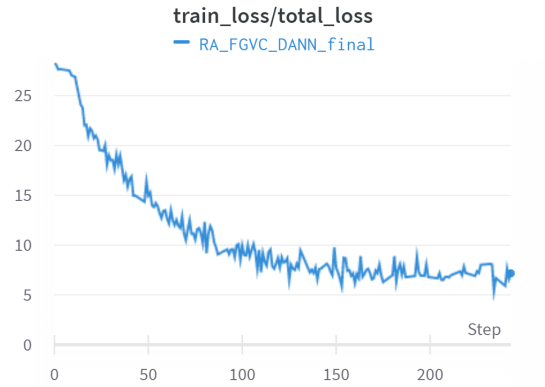


Figure 6.37: Total Loss

6.6 Comparison

From the section 6.5 we can see the differences between two model with experimental result. we wanted to achieve acceptable performance on novel class of CompCarsSV whose label or annotation is partially accessible by training a neural network on CompCars plus base clas of CompCarsSV dataset. for which label or annotation is available.The goal of this work meets, we get a significant improved in the accuracy of novel class from target domain, as we can see without domain adaptation the average f1 score is .51 on the other hand With domain adaptation is .61. Other results are quite similar except the total loss and combiner loss and accuracy, it is slightly increased in our proposed model. The proposed model is also trained with the resnet50 Backbone network. Table table:3 is the summary of Results of Cross-Domain Fine-Grained Classification with Resnet50

Key	Value
CompcarsSV avg f1 score base:	0.93
Compcars SV avg f1 score novel	0.20
Compcars web avg f1 score base	0.84
Compcars web avg f1 score novel	0.83
Maximum number of epoch	50
Learning rate	0.00000180851192292985 83
combiner accuracy	98.44
combiner loss	0.02824401855468 75
total loss	16.03800964355468 8

Table 6.3: Summary of Proposed Model Accuracy with resnet50

As we can see from the table the accuracy is not so good in this case, its quite unacceptable,some changes in hyper parameters or input size may improve the result. The main experiment was with Swin-T backbone and we get a noticeable improvement in classification using our proposed model as we can see in Table 6.2.

7 Conclusion

The central theme of this work is a data-driven model that concerns the particular applicative area of cross-domain fine grained visual classification. To this end, as presented in the preceding text, a neural network based model architecture has been developed. The resulting model has been trained and evaluated on an automotive dataset capturing the pecularities of both fine grained classification and domain adaption. The presented results and improvements contained therein; while seemingly modest; remain encouraging and motivate continued investigation into this field of work.

In the following distinct possibilities of improvements; and directions of future work are described.

7.1 Future Work

This work was motivated primarily because there was a lack of methods or architectures combining both fine grained visual classification and domain adaptation under one hood. More concretely, for a particular application; a model was required that can successfully classify images contained in the *CompCarsSV* dataset (see [**chap_evaluation_sec_dataset**]). The *CompCarsSV* dataset contains surveillance camera still images from cars of different make, model, company and year. Unfortunately, most of these images are unlabeled (except for the *novel* classes); and as such cannot be used to train a data-driven model e.g. a standard CNN based classification model in a supervised setting. On the other hand, the high level of visual similarity amongst the categories (labels) of the images makes the standard classification task also challenging.

Therefore, a central hypothesis of this work was that a combination of fine-grained classification approaches with domain adaptation techniques may lead to better performance on a challenging dataset such as *CompCarsSV*. In consequence, an architecture was developed that combines the fine-grained supervised classification approach as contained in [20]; with an unsupervised domain adaptation technique; such as the one contained in [31].

7.1.1 Domain Adaptation

Domain adaptation is a rapidly changing area of work with new papers and techniques being published regularly. Recall that DA approaches can be loosely classified into the following methodologies; when domain alignment is the goal:

1. Divergence based
2. Adversarial based
3. Reconstruction based

Instead of domain alignment, domain adaptation can also be achieved through converting one domain (source) to the other (target). This conversion or mapping can also be learned in an adversarial setting - typically using a conditional generative adversarial network (CGAN). An interesting approach is detailed in [18].
One future direction of work can be to integrate domain adaptation through the use of domain mapping techniques; as potentially this may lead to better performance.

7.1.2 Fine Grained Classification

The area of fine grained visual classification remains an active area of research; with interesting approaches published regularly. The approach used in our current work is detailed in [20]. In contrast to other methods; this work distinguishes itself by the use of a novel plug-in architecture; enabling single stage training of the complete network.
Similar to our chosen method of [20]; the approach detailed in [45] is a single stage approach; using a novel spatially weighted pooling layer to combine features from different scales. Additionally, it has state-of-the-art performance on the *CompCars* dataset (97.6%); making it an interesting candidate for further integration into our architecture.
Domain adaptation (DA) in a fine-grained visual classification (FGVC) task is a new research area; relevant for challenges associated with new applications in autonomous driving, surveillance etc. It is expected that DA-FGVC will stay relevant and may even become one of the mainstays of modern computer vision approaches employing deep learning techniques; taking a right step in the direction of general object detection. Further work remains necessary to improve performance; and also to evaluate and compare with other approaches in this area.

Bibliography

[1] 2022. URL: https://git-scm.com/.

[2] 2022. URL: https://pytorch.org/.

[3] 2022. URL: https://numpy.org/.

[4] 2022. URL: https://wandb.ai/site.

[5] 2022. URL: https://code.visualstudio.com/.

[6] 2022. URL: https://en.wikipedia.org/wiki/Linux.

[7] *1.5. stochastic gradient descent.* URL: https://scikit-learn.org/stable/modules/sgd.html.

[8] *About.* Nov. 2020. URL: https://opencv.org/about/.

[9] *An even easier introduction to Cuda.* Aug. 2022. URL: https://developer.nvidia.com/blog/even-easier-introduction-cuda/.

[10] *Anaconda (python distribution).* Aug. 2022. URL: https://en.wikipedia.org/wiki/Anaconda_(Python_distribution).

[11] Janus Atienza, By, and Janus Atienza. *Why is linux popular for machine learning? top distributions to use.* Feb. 2022. URL: https://www.unixmen.com/why-is-linux-popular-for-machine-learning-top-distributions-to-use/.

[12] Artem Babenko et al. "Neural codes for image retrieval." In: *European conference on computer vision.* Springer. 2014, pp. 584–599.

[13] Ardhendu Behera et al. "Context-aware attentional pooling (cap) for fine-grained visual classification." In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 35. 2. 2021, pp. 929–937.

[14] Steve Branson et al. "Bird species categorization using pose normalized deep convolutional nets." In: *arXiv preprint arXiv:1406.2952* (2014).

[15] Andreas Buja, Werner Stuetzle, and Yi Shen. "Loss functions for binary class probability estimation and classification: Structure and applications." In: *Working draft, November* 3 (2005), p. 13.

[16] Manliang Cao et al. "Adversarial domain adaptation with semantic consistency for cross-domain image classification." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 259–268.

[17] Chaoqi Chen et al. "Progressive feature alignment for unsupervised domain adaptation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 627–636.

[18] Yunjey Choi et al. "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8789–8797.

[19] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. "Dlid: Deep learning for domain adaptation by interpolating between domains." In: *ICML workshop on challenges in representation learning*. Vol. 2. 6. Citeseer. 2013.

[20] Po-Yung Chou, Cheng-Hung Lin, and Wen-Chung Kao. "A Novel Plug-in Module for Fine-Grained Visual Classification." In: *arXiv preprint arXiv:2202.03822* (2022).

[21] *cpuvsgpu*. URL: https://www.bing.com/images/search?view=detailV2&ccid=LR1Uszq0&id=2038A2EA5D337767DA99CDF85344AC5D83517ECB&thid=OIP.LR1Uszq0NWB1TvIseBWgHAHaEK&mediaurl=https%3A%2F%2Fi.ytimg.com%2Fvi%2Fzr2-ofVpnWU%2Fmaxresdefault.jpg&cdnurl=https%3A%2F%2Fth.bing.com%2Fth%2Fid%2FR.2d1d54b33ab43560754ef22c7815a01c%3Frik%3Dy35Rg12sRFP4zQ%26pid%3DImgRaw%26r%3D0&exph=720&expw=1280&q=CPU%2Bvs%2BGPU%2Bin%2Bmachine%2Blearning&simid=608036265692576008&FORM=IRPRST&ck=3BDC1E6F606DC1F02EA50E75166A21E1&selectedIndex=17&ajaxhist=0&ajaxserp=0.

[22] Gabriela Csurka. "Domain adaptation for visual applications: A comprehensive survey." In: *arXiv preprint arXiv:1702.05374* (2017).

[23] Yin Cui et al. "Large scale fine-grained categorization and domain-specific transfer learning." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00432.

[24] Daisy. *Do you need a good GPU for machine learning?* May 2022. URL: https://datasciencenerd.com/do-you-need-a-good-gpu-for-machine-learning/#:~:text=A%20good%20GPU%20is%20indispensable%20for%20machine%20learning.,learning%20tasks%2C%20thanks%20to%20their%20several%20thousand%20cores..

[25] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale." In: *arXiv preprint arXiv:2010.11929* (2020).

[26] Omar Elharrouss et al. "Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches." In: *arXiv preprint arXiv:2206.08016* (2022).

[27] *F-score*. May 2019. URL: https://deepai.org/machine-learning-glossary-and-terms/f-score.

[28] Abolfazl Farahani et al. "A brief review of domain adaptation." In: *Advances in data science and information engineering* (2021), pp. 877–894.

[29] *Feature coding and pooling - state of the art*. URL: https://123dok.net/article/feature-coding-pooling-state-art.zpngx5ro.

[30] *Fraunhofer Institute of Optronics, System Technologies and image exploitation IOSB*. Aug. 2022. URL: https://www.iosb.fraunhofer.de/en.html.

[31] Yaroslav Ganin and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." In: *International conference on machine learning*. PMLR. 2015, pp. 1180–1189.

[32] Yaroslav Ganin et al. "Domain-adversarial training of Neural Networks." In: *Domain Adaptation in Computer Vision Applications* (2017), pp. 189–209. DOI: 10.1007/978-3-319-58347-1_10.

[33] Yaroslav Ganin et al. "Domain-adversarial training of neural networks." In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[34] Weifeng Ge, Xiangru Lin, and Yizhou Yu. "Weakly supervised complementary parts models for fine-grained image classification from the bottom up." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3034–3043.

[35] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. "Fine-grained recognition in the wild: A multi-task domain adaptation approach." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1349–1358.

[36] *Git*. Aug. 2022. URL: https://en.wikipedia.org/wiki/Git.

[37] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Domain adaptation for large-scale sentiment classification: A deep learning approach." In: *ICML*. 2011.

[38] Boqing Gong et al. "Geodesic flow kernel for unsupervised domain adaptation." In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2066–2073.

[39] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. "Domain adaptation for object recognition: An unsupervised approach." In: *2011 international conference on computer vision*. IEEE. 2011, pp. 999–1006.

[40] Chun-feng GUO et al. *A survey of fine-grained image classification based on Deep Learning*. URL: http://www.dpi-journals.com/index.php/dtcse/article/view/30636/29251.

[41] Harald Hanselmann and Hermann Ney. "Fine-grained visual classification with efficient end-to-end localization." In: *arXiv preprint arXiv:2005.05123* (2020).

[42] Ju He et al. "Transfg: A transformer architecture for fine-grained recognition." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 852–860.

[43] Martin Heller. *What is Cuda? Parallel Programming for gpus*. Aug. 2018. URL: https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html#:~:text=CUDA%20is%20a%20parallel%20computing,parallelizable%20part%20of%20the%20computation.

[44] *How swin transformer is made!* URL: https://deepaie.com/how-swin-transformer-is-made.html.

[45] Qichang Hu et al. "Deep CNNs with spatially weighted pooling for fine-grained car recognition." In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), pp. 3147–3156.

[46] Tao Hu et al. "See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification." In: *arXiv preprint arXiv:1901.09891* (2019).

[47] Yunqing Hu et al. "Rams-trans: Recurrent attention multi-scale transformer for fine-grained image recognition." In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 4239–4248.

[48] *Introduction to loss functions*. June 2022. URL: https://www.datarobot.com/blog/introduction-to-loss-functions/.

[49] Masato Ishii, Takashi Takenouchi, and Masashi Sugiyama. "Partially Zero-shot domain adaptation from incomplete target data with missing classes." In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2020). DOI: 10.1109/wacv45572.2020.9093298.

[50] Aakash Kaushik. *Understanding Resnet50 architecture*. July 2020. URL: https://iq.opengenus.org/resnet50-architecture/.

[51] Lingdong Kong. "Unsupervised." In: *Unsupervised domain adaptation for Lidar segmentation* (). DOI: 10.32657/10356/158401.

[52] Shu Kong and Charless Fowlkes. "Low-rank bilinear pooling for fine-grained classification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 365–374.

[53] Brett Koonce. "ResNet 50." In: *Convolutional neural networks with swift for tensorflow*. Springer, 2021, pp. 63–72.

[54] Jonathan Krause et al. "Fine-grained recognition without part annotations." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5546–5555.

[55] Jonathan Krause et al. "The unreasonable effectiveness of noisy data for fine-grained recognition." In: *European Conference on Computer Vision*. Springer. 2016, pp. 301–320.

[56] Admond Lee. *Why git and how to use git as a data scientist*. Aug. 2022. URL: https://towardsdatascience.com/why-git-and-how-to-use-git-as-a-data-scientist-4fa2d3bdc197.

[57] Hao Li et al. "Attribute mix: Semantic data augmentation for fine grained recognition." In: *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE. 2020, pp. 243–246.

[58] Jingjing Li et al. "Maximum density divergence for domain adaptation." In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3918–3930.

[59] Qi Li. "Literature survey: domain adaptation algorithms for natural language processing." In: *Department of Computer Science The Graduate Center, The City University of New York* (2012), pp. 8–10.

[60] Tsung-Yu Lin and Subhransu Maji. "Improved bilinear pooling with cnns." In: *arXiv preprint arXiv:1707.06772* (2017).

[61] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN models for fine-grained visual recognition." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1449–1457.

[62] Chuanbin Liu et al. "Filtration and distillation: Enhancing region attention for fine-grained visual categorization." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11555–11562.

[63] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[64] Mingsheng Long et al. "Conditional adversarial domain adaptation." In: *Advances in neural information processing systems* 31 (2018).

[65] Mingsheng Long et al. "Learning transferable features with deep adaptation networks." In: *International conference on machine learning*. PMLR. 2015, pp. 97–105.

[66] Xiao Lv et al. "Feature coding for image classification based on saliency detection and fuzzy reasoning and its application in elevator videos." In: *WSEAS Trans. Comput* 13.1 (2014), pp. 266–276.

[67] *Machine learning: Installing anaconda and Python - Javatpoint*. URL: https://www.javatpoint.com/machine-learning-installing-anaconda-and-python#:~:text=Anaconda%20distribution%20is%20a%20free,learning%20and%20deep%20learning%20models..

[68] Harsh Maheshwari. *Understanding domain adaptation*. Sept. 2021. URL: https://towardsdatascience.com/understanding-domain-adaptation-5baa723ac71f.

[69] Subhransu Maji et al. "Fine-grained visual classification of aircraft." In: *arXiv preprint arXiv:1306.5151* (2013).

[70] Microsoft. *Python in visual studio code*. Nov. 2021. URL: https://code.visualstudio.com/learn/educators/python.

[71] Andrew Murphy. *Batch size (machine learning): Radiology reference article*. May 2019. URL: https://radiopaedia.org/articles/batch-size-machine-learning?lang=us.

[72] *Numpy introduction*. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp.

[73] *Nvidia Tesla v100 sxm2 32 GB Specs*. Aug. 2022. URL: https://www.techpowerup.com/gpu-specs/tesla-v100-sxm2-32-gb.c3185.

[74] Maxime Oquab et al. "Learning and transferring mid-level image representations using convolutional neural networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.

[75] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[76] *Pandas (software)*. Aug. 2022. URL: https://en.wikipedia.org/wiki/Pandas_(software).

[77] Sieun Park. *Swin transformers: The most powerful tool in computer vision*. May 2022. URL: https://sieunpark77.medium.com/swin-transformers-the-most-powerful-tool-in-computer-vision-659f78744871.

[78] *Project jupyter*. URL: https://jupyter.org/.

[79] *Python - overview*. URL: https://www.tutorialspoint.com/python/python_overview.htm#:~:text=Python%20is%20a%20high%2Dlevel,syntactical%20constructions%20than%20other%20languages.

[80] *Python PIL: Image.open() method*. July 2019. URL: https://www.geeksforgeeks.org/python-pil-image-open-method/.

[81] David Ramel07/12/2022. *VS code and python: A natural fit for data science*. URL: https://visualstudiomagazine.com/articles/2022/07/12/python-vs-code.aspx.

[82] Yongming Rao et al. "Counterfactual attention learning for fine-grained visual categorization and re-identification." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1025–1034.

[83] Navid Rezaei. *Linux server for AI: The step-by-step guide*. Nov. 2020. URL: https://becominghuman.ai/linux-server-for-ai-the-step-by-step-guide-9dccb0bb660.

[84] Kavish Sanghvi. *Image classification techniques*. Mar. 2022. URL: https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac.

[85] *Scikit-Learn*. Jan. 2022. URL: https://en.wikipedia.org/wiki/Scikit-learn.

[86] Ashwin Singh. *Deep Learning Hardware: Know Your Options*. Mar. 2020. URL: https://towardsdatascience.com/deep-learning-hardware-know-your-options-9e95026b5d5e.

[87] Himanshu Singh. *Hardware requirements for machine learning*. Dec. 2019. URL: https://www.einfochips.com/blog/everything-you-need-to-know-about-hardware-requirements-for-machine-learning/.

[88] *Sklearn.metrics.top$_k$accuracy$_s$core*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.top_k_accuracy_score.html.

[89] Vikas Solegaonkar. *Introduction to PY torch*. June 2021. URL: https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3.

[90] Ming Sun et al. "Multi-attention multi-class constraint for fine-grained image recognition." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 805–821.

[91] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1891–1898.

[92] Supper. *FGVC*. Feb. 2021. URL: https://www.fatalerrors.org/a/fine-grained-image-classification-fgvc-a-survey.html.

[93] Educative Answers Team. *What is the F1-score?* Aug. 2022. URL: https://www.educative.io/answers/what-is-the-f1-score.

[94] *The world's most popular data science platform*. URL: https://www.anaconda.com/.

[95] *Timm with FASTAI - the largest computer vision models library*. July 2022. URL: https://appsilon.com/timm-with-fastai/.

[96] Eric Tzeng et al. "Adversarial discriminative domain adaptation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[97] *Understanding loss functions in machine learning*. URL: https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/.

[98] *USB FPGA module 1.2*. URL: https://opencores.org/projects/usb_fpga_1_2.

[99] Naoto Usuyama et al. "EPillID dataset: A low-shot fine-grained benchmark for pill identification." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2020). DOI: 10.1109/cvprw50498.2020.00463.

[100] Grant Van Horn et al. "The iNaturalist Species Classification and Detection Dataset." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018). DOI: 10.1109/cvpr.2018.00914.

[101] Andrea Vedaldi et al. "Understanding objects in detail with fine-grained attributes." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3622–3629.

[102] Ishu Verma. *Introduction to machine learning with Jupyter Notebooks*. Aug. 2022. URL: https://developers.redhat.com/articles/2021/05/21/introduction-machine-learning-jupyter-notebooks#the_jupyter_notebook_dashboard.

[103] Jun Wang, Xiaohan Yu, and Yongsheng Gao. "Feature fusion vision transformer for fine-grained visual categorization." In: *arXiv preprint arXiv:2107.02341* (2021).

[104] Mei Wang and Weihong Deng. "Deep visual domain adaptation: A survey." In: *Neurocomputing* 312 (2018), pp. 135–153.

[105] Xiu-Shen Wei et al. "Fine-grained image analysis with deep learning: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[106] *Weights amp; Biases - documentation.* URL: https://docs.wandb.ai/.

[107] *What is linux?* URL: https://www.redhat.com/en/topics/linux/what-is-linux#:~:text=Linux%C2%AE%20is%20an%20open,resources%20that%20do%20the%20work.

[108] *Why use python for AI and machine learning?* Dec. 2021. URL: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/#:~:text=Benefits%20that%20make%20Python%20the,overall%20popularity%20of%20the%20language.

[109] Thaddaus Wiedemer et al. "Few-shot supervised prototype alignment for pedestrian detection on Fisheye images." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2022). DOI: 10.1109/cvprw56347.2022.00459.

[110] Stefan Wolf. "Cross-Domain Fine-Grained Classification: A Review." In: *Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. 2022, p. 189.

[111] Zhe Xu et al. "Augmenting strong supervision using web data for fine-grained categorization." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2524–2532.

[112] Zhe Xu et al. "Friend or foe: Fine-grained categorization with weak supervision." In: *IEEE Transactions on Image Processing* 26.1 (2016), pp. 135–146.

[113] Zhe Xu et al. "Webly-supervised fine-grained visual categorization via deep domain adaptation." In: *IEEE transactions on pattern analysis and machine intelligence* 40.5 (2016), pp. 1100–1113.

[114] Linjie Yang et al. "A large-scale car dataset for fine-grained categorization and verification." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3973–3981.

[115] Shaokang Yang et al. "Re-rank coarse classification with local region enhanced features for fine-grained image recognition." In: *arXiv preprint arXiv:2102.09875* (2021).

[116] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[117] Ning Zhang et al. "Fine-grained pose prediction, normalization, and recognition." In: *arXiv preprint arXiv:1511.07063* (2015).

[118] Ning Zhang et al. "Part-based R-CNNs for fine-grained category detection." In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[119] Yuan Zhang et al. "A free lunch from ViT: adaptive attention multi-scale fusion Transformer for fine-grained visual recognition." In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 3234–3238.

[120] Heliang Zheng et al. "Learning multi-attention convolutional neural network for fine-grained image recognition." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5209–5217.

[121] Peiqin Zhuang, Yali Wang, and Yu Qiao. "Learning attentive pairwise interaction for fine-grained classification." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13130–13137.

List of Tables

List of Figures