ROBUST FINE-GRAINED OBJECT CLASSIFICATION IN VIDEOS DOMAIN

By

YOUSEF ALSAHAFI

B.S., University of King Abdulaziz, Computer Science Department, KSA, 2007

M.S., University of Colorado Colorado Springs, Computer Science Department, USA, 2015

A dissertation submitted to the Graduate Faculty of the

University of Colorado Colorado Springs

in partial fulfillment of the

requirement for the degree of

Doctor of Philosophy

Department of Computer Science

2020
This dissertation for Doctor of Philosophy degree by

Yousef Alsahafi

has been approved for the

Department of Computer Science

by

Terrance E. Boult, Chair

Jonathan Ventura, Chair

Richard White

Darshika G. Perera

Scott Kupferman

1/2/2020

Date
ABSTRACT

Fine-grained object classification task distinguishes between classes in a sub-category of objects, for instance, the particular species of bird or dog. It uses information from still images and ignores the information that can be exploited from the input video. Videos are more challenging than still images because they have many noisy frames that contain unfavorable poses, motion blur, problematic lighting, and occlusions. Over the years, several approaches and datasets have been proposed for fine-grained categorization in still images, but the enormous potential of video data to improve vehicle classification has largely been ignored. In this thesis, we make the following contributions. First, we introduce the CarVideo 100 (CV100) dataset, a novel dataset for fine-grained object classification of vehicle make, model, and year in which all frames were annotated with bounding boxes and labeled using the Amazon Mechanical Turk workforce. Second, we introduce a novel approach to fine-grained object classification in videos that combines a Single Shot Multibox Detector (SSD) with a single stream multi-region Convolutional Neural Network (CNN). The approach uses the SSD to extract the most important regions from each frame and sends them to the CNN that is a Residual Networks (ResNet) architecture for classification. The
ResNet is pre-trained on ImageNet and fine-tuned on the CV100. One of the most important steps in our process is the data augmentation technique in which each bounding box is cropped into four corner crops and a center crop. These crops relay additional information to the ResNet and assist us in reducing the chance for over-fitting during training. Finally, we present a novel approach to fine-grained car model categorization that utilizes a Scale-Aware Long Short-Term Memory (SA-LSTM) to address the fact that fine-grained car model classification for video has different characteristics and algorithmic requirements than the same process for still images. This model uses Single Shot MultiBox Detector (SSD) to localize objects, by establishing their scale (height and width) with bounding boxes, from the sequence of frames in which they appear. The CNN extracts the features from these objects and object parts. Our technique combines object scales with CNN features that acknowledges how, in video, objects will present with variations of size and scale in a sequence of frames. This information is fed to SA-LSTM, creating a fine-grained classification method that takes into account variations in size and scale that occur not only in video but that might, under certain circumstances, be an issue when classification involves still images.
DEDICATION

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Hamdah and Saleh.

This dissertation is dedicated to my wife (Reham Alsahafi) who encouraged me to pursue my dreams and finish my dissertation.
ACKNOWLEDGMENTS

First, thanks and praise be to God (Allah). Second, I wish to thank my committee members who were more than generous with their expertise and precious time. A special thanks to Drs. Terrance E. Boult and Jonathan Ventura, for their countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process. Thank you Drs. Richard White, Darshika G. Perera, and Scott Kupferman for agreeing to serve on my committee.
# TABLE OF CONTENTS

## CHAPTER

### I Introduction

1.1 Overview .................................................. 1

1.2 Research Questions ........................................ 5

1.3 Contributions ............................................ 7

1.4 Publications ............................................... 9

### II Literature Review

2.1 Feature Extraction ........................................ 11

2.1.1 Low-level Features ........................................ 12

2.1.2 Deep Learning ............................................ 12

2.2 Localizing the Discriminative Object Parts Models ......... 18

2.3 Video Classification ........................................ 23

2.3.1 Low-level features ........................................ 23

2.3.2 Deep Learning ............................................ 24

2.3.2.1 Two-Stream Network .................................. 26

2.3.2.2 3-D Convolutional Network ......................... 26

2.4 Video-based Fine-Grained Classification .................... 27

2.5 The Fine-grained Datasets .................................. 35

2.5.1 The Fine-grained Still Images Datasets .................. 35

2.5.2 The Fine-grained Video Datasets ......................... 38
LIST OF TABLES

TABLE

4.1  Accuracy and speed comparison of the SSD-CNN (25 frames) with baseline techniques for 10-fold cross-validation on CV100:CarVideo 100 dataset. Last column shows p-values from 10-folds cross-validation experiment; the SSD-CNN (25 frames) is statistically significantly better than the baselines.

5.1  Accuracy and speed comparison of the SA-LSTM with baseline techniques for 10-fold cross-validation on CV100:CarVideo 100 dataset. Last column shows p-values from 10-folds cross-validation experiment; the SA-LSTM is significantly better than the baselines.
LIST OF FIGURES

FIGURE

1.1 Object Classification .......................... 2
1.2 Object classification challenges .................... 3
1.3 Illustrates the difference between general object classification and fine-grained classification. In the left side, the general object classification should classify four classes from different different object categories such as a bird category and a dog category. While, in the right side, fine-grained classification classify four sub-classes belong to the same category. ........ 4
1.4 Each column presents a large appearance differences in their poses and the first raw illustrates the subtle difference between car models. ............ 5
1.5 Sequence of frames show large appearance changes of objects in a video. .... 6
2.1 Imagenet classification with deep convolutional neural networks ............ 11
2.2 Hand-crafted Feature Descriptors ........................ 12
2.3 The first part of CNNs is Convolutional layers attempts to extract features and the second part is fully-connected which classifies classes ............ 13
2.4 Four different CNN architectures .......................... 15
2.5 A Transfer Learning Approach ............................ 17
2.6 Poselets ........................................ 18
2.7 The Bridlet ........................................ 19
2.8 Symbiotic Segmentation and Part Localization for Fine-Grained Catego-
    rization ........................................ 21
2.9 Fine-grained pose prediction, normalization, and recognition ............ 22
2.10 3D SIFT descriptors ........................................... 24
2.11 Two-Stream Networks ......................................... 25
2.12 3D Convolutional Networks ................................... 26
2.13 Person spotting: video shot retrieval for face sets ....................... 27
2.14 Probabilistic Elastic Matching for Pose Variant Face Verification .... 28
2.15 Eigen-pep for video face recognition ................................ 29
2.16 Neural aggregation network for video face recognition ............... 30
2.17 Learning Discriminative Aggregation Network for Video-based Face Recognition .......................................................... 31
2.18 Coupling alignments with recognition for still-to-video face recognition ... 33
2.19 Fine-grained classification of pedestrians in video: Benchmark and state of the art .......................................................... 34
2.20 The caltech-ucsd birds-200-2011 dataset ................................. 36
2.21 A large-scale car dataset for fine-grained categorization and verification .. 37
2.22 Face recognition in unconstrained videos with matched background similarity 38
2.23 bc127: Video dataset for fine-grained bird classification .............. 40
2.24 Fine-grained video categorization with redundancy reduction attention ... 41
3.1 Each column presents a large appearance differences in their poses and the first raw illustrates the subtle difference between car models. ............ 44
3.2 Example of Toyota Camry in different years and models. ............... 45
3.3 Sequence of frames in the video presents the car on different views and varying angles. ................................................. 46
3.4 The differences between natural objects and unnatural objects when creating a dataset. ................................................................. 47

3.5 TRUECar has many car manufactures and each car manufacture has several of car models ................................................................. 48

3.6 We delete the first and fourth rows since they do not provide any car information. ................................................................. 49

3.7 Create a new project on MTurk. ................................................................. 50

3.8 Create Qualification Types. ................................................................. 51

3.9 User interface for MTurk workers. ................................................................. 52

3.10 Examples of our car models. ................................................................. 53

4.1 Each image shows bounding boxes predicted by the SSD network. ....... 56

4.2 The entirety of our pipeline, including the output from our SSD and then the following resize and pass to our CNN. ................................................................. 56

4.3 An example of the crops for each possible classification image which consists of four corner crops and a single center crop. The crops are then passed from this output into our CNN. ................................................................. 57

4.4 ........................................................................................................ 59

4.5 Comparison of precision-recall curves of the SSD-CNN (25 frames) with baselines techniques on CV100:CarVideo 100 dataset. ............... 62

4.6 Four test samples of classification and their prediction results. .......... 63

5.1 The SA-LSTM approach. ................................................................. 65

5.2 Sequence of frames in the video presents the car on different views and varying angles. ................................................................. 69

xii
5.3 Comparison of precision-recall curves of the SA-LSTM with baseline techniques on CV100:CarVideo 100 dataset. . . . . . . . . . . . . . . . . . . . 72
5.4 Sample classes that are mistakenly predicted as another class. . . . . . . . . . 73
CHAPTER I

INTRODUCTION

1.1 Overview

One of the most prominent computer vision tasks is object classification [1, 2]. It classifies the image depending on the object presented in the image. For example, an image showing an airplane is classified with an airplane label as shown in Figure 1.1. Object classification has been used in many real-world applications, such as image retrieval [3] and automatic image tagging [4]. These applications make everyday tasks easier by automatically analyzing image information. There are a large number of multi-media resources on the Internet and this number increases every second. For instance, the Flickr website has more than three billion images with approximately 2.5 million images being uploaded every day [5].

Object classification models have faced many challenges as shown in Figure 1.2. One of the challenges is the viewpoint variation where an object can present with different orientations or scales in an image. Some images have just a part of the object and not the whole object, which is called an occlusion. Some images are taken under different
Figure 1.1: Object Classification

Object classification systems are designed to detect and recognize an object in an image but cannot go to a deeper level. For example, they can detect and recognize a car in an image but cannot tell the car’s make, model, and year. Recognizing a sub-category of an object requires a classifier that can detect and classify a specific pattern between several visually similar objects. Figure 1.3 shows the difference between object classification and fine-grained classification. Recently, sub-category classification has become an interesting field in the computer vision and machine learning communities [6–8] and has become known as fine-grained classification.

Fine-grained object classification provides more detailed information about the objects present in images or videos. This information can help people build many helpful applications. For instance, car classification systems can help track a car over a multiple
camera network when a license plate cannot be seen. Clothing style recognition and retrieval, which uses an image of a product to find similar-looking products, could be another application. We also can build an application that provides information about bird species or types of flowers. In addition, we can create a food classification system, installed on mobile phones, that can help people measure calories in each meal.

Despite object classification achieving high performance in the past few years, fine-grained classification is still challenged when asked to perform accurate fine-grained classification of object sub-categories. Figure 3.1 presents an example of a fine-grained car classification problem and illustrates the challenges that fine-grained classification is facing. The first challenge is large appearance differences in positions and illumination. The car images or videos are taken from outdoor environments such as streets, highways, freeways, or parking lots. Furthermore, images or videos are taken under different scenarios with day or night light changes. The first row in Figure 3.1 shows the second challenge, inter-class variation, where four SUV models are very similar in their side views but have subtle differences between models.
Another challenge is collecting and annotating fine-grained datasets for images or videos as it is difficult and costs both time and money. Labeling natural objects, such as bird species [9] and dog breeds [10], needs people with expert knowledge of birds or dogs. For unnatural objects, such as cars where models are changed annually (and some years the differences between two models are very subtle), more effort is needed to investigate the make, model, and year.

The growing popularity of video sharing sites, such as YouTube, has caused researchers to address video fine-grained classification. This problem brings new questions, such as: How can we solve video fine-grained classification tasks? and Can we use existing methods for fine-grained in still images? Unfortunately, we cannot use them since large appearance changes of objects in videos, as shown in Figure 1.5, will affect the performance of the approach. Videos have frames, and each frame has different appearance, the frame
Figure 1.4: Each column presents a large appearance differences in their poses and the first raw illustrates the subtle difference between car models.

may suffer form motion blur, video defocused, part occlusion and bad pose as illustrated in Figure 1.5.

1.2 Research Questions

The goal of this thesis is to create a robust video fine-grained classification system. To do that we will try to answer these three questions, the answers of which should help us create a model.

1. Is there a large-scale benchmark for videos we can use to improve the robustness of fine-grained classification in videos?

In this thesis, we introduce a large-scale and comprehensive video dataset called CarVideo 100 (CV100) which contains 495,778 frames of 100 car models. Specifically, it is annotated with bounding boxes and attribute labels, which have not been provided by other fine-grained video datasets [11, 12]. CV100 can be used to vali-
date many computer vision algorithms, has numerous practical applications, as well as enormous potential for supporting future research.

2. Can we exploit temporal information available in videos and object detection plus utilize a Convolutional Neural Network (CNN) to improve robustness of fine-grained classification in videos?

Our novel approach uses a Single Shot Multibox Detector (SSD) [13] to extract the most important regions from each frame and send them to a CNN, which utilizes a ResNet architecture [14], for classification. The ResNet is pre-trained on ImageNet [15] and fine-tuned on our dataset. One of the most important steps in our process is the data augmentation technique from [14] in which each bounding box is cropped into four corner crops and a center crop. These crops relay additional information to the ResNet and assist us in reducing the chance for over-fitting during training.

3. Can we build a system that would be able to connect previous information to the present task, such as using previous video frames that might inform the understanding of the present frame, for fine-grained classification in videos?

In the second question, we use temporal information in videos by using multiple frames to classify each video in a fine-grained dataset. However, we did not analyze which frames were helped in the classification process, and even deeper, we did not determine why these frames were helped. In this section, we will present an approach
for fine-grained car model categorization that utilizes Scale-Aware Long Short-Term Memory (SA-LSTM). SA-LSTM should address the fact that fine-grained car model classification for video has different characteristics and algorithmic requirements than the same process for images. This model uses Single Shot MultiBox Detector (SSD) to localize objects by establishing their scale (height and width) with bounding boxes, from the sequence of frames in which they appear. The CNN extracts the features from these objects and object parts. Our technique combines object scales with CNN features that acknowledges how, in video, objects will present with variations of size and scale in a sequence of frames. This information is fed to SA-LSTM, creating a fine-grained classification method that takes into account variations in size and scale that occur not only in video but that might, under certain circumstances, be an issue when classification involves still images.

1.3 Contributions

The contributions of this thesis are twofold:

1. We introduce a large-scale and comprehensive video dataset called CarVideo 100 (CV100), which contains 495,778 frames of 100 car models. Specifically, it is annotated with bounding boxes and attribute labels, which have not been provided by other fine-grained video datasets [11, 12].

2. We provide a baseline performance evaluation of state-of-the-art video classification methods on our dataset. Our evaluation shows that, on this dataset, a fine-grained classification approach that combines an object detector with a classifier outperforms
both temporal segment networks and 3D convolutional networks, which are state-of-the-art methods for video classification.

3. We introduce a novel, SA-LSTM approach that learns information temporally across the video sequence and that is based on the object scale and LSTM deep neural architecture we developed for fine-grained video classification. It localizes the object’s location with scales (height and width) in the sequence of frames where objects present with variations of size and scale in a sequence of frames. Then we use the CNN to extract the features from the object and the object’s parts. Subsequently, we combine each object feature with its scale before sending it to the LSTM network. Our SA-LSTM uses each object’s features and bounding box size to learn sequence information extracted from video frames and ignores frames without important information.
1.4 Publications


CHAPTER II

LITERATURE REVIEW

Object classification has been studied for years and considerable progress has been made [2, 16–20]. Their approaches have achieved impressive results on some popular benchmarks such as the PASCAL VOC dataset [21], Caltech 256 [22] and ImageNet [2]. Many object classification approaches have been implemented over several decades, such as approaches based on object models [23], appearance-based methods [24], feature-based methods [25], Fisher Vectors (FV) [19] and Histogram Encoding [22]. The more recent approaches use CNNs [2], which have achieved the best performance on object classification when using ImageNet that contains millions of images and hundreds of object classes. The performance of these approaches using CNN on ImageNet is now close to that of humans. The example of CNN architecture [2] presents in Figure 2.1.

Fine-grained classification is a sub-field of object classification and is a highly studied area [6,9,26–29]. Finding and extracting discriminative and robust features is the most important part of fine-grained classification methods. Researchers started using hand-crafted feature descriptors such as the Scale Invariant Feature Transform (SIFT) [25], Histogram
Figure 2.1: Imagenet classification with deep convolutional neural networks

of Oriented Gradients (HoG) [30], and Color Histogram [31] which relies on color, texture, and edge information. However, Deep Convolutional Neural Network (DCNN) has achieved state-of-the-art performance for fine-grained classification by applying transfer learning [32, 33].

2.1 Feature Extraction

Many feature descriptors used for object classification tasks have been transferred for fine-grained classification. Object classification tasks were relying on low-level features such as Scale Invariant Feature Transform (SIFT) [25], Speeded-up Robust Features (SURF) [34], and Histogram of Oriented Gradients (HoG) [30]. These approaches could not handle inter-class variations in different positions and from different viewpoints. To tackle the problem of losing information, feature encoding, such as POOF proposed by Berg and Belhumeur [35] and Bags of Visual Words (BOVW) [36], have been used.
2.1.1 Low-level Features

During the last decade, SIFT and HoG were the famous descriptors and have been used a lot on general object classification. The SIFT descriptor is considered one of the best descriptors in computer vision. It can generate a large number of features and is also invariant to rotation and transformation. The SIFT method transforms an image into many local feature vectors where each vector is invariant to image translation, scaling, and rotation. While the HoG descriptor has feature descriptors used to detect the object at the image, it divides the image into small connected regions called cells and then computes a histogram of gradient direction for each cell. The descriptor is the combination of these histograms. Furthermore, SIFT and HoG have been applied to fine-grained classification tasks [37]. One of the drawbacks of low-level features is that they sometimes fail to catch the subtle differences between fine-grained sub-categories.

2.1.2 Deep Learning

The Convolutional Neural Network (CNN) model has become a trend in computer vision since 2012, when Krizhevsky et al. [2] exposed deep networks for image classification. It has improved state-of-the-art object detection and recognition, speech, and charac-
Figure 2.3: The first part of CNNs is Convolutional layers attempts to extract features and the second part is fully-connected which classifies classes.

CNNs are made up of layers (convolutional and fully-connected). Convolutional layers attempt to extract high-level features and provide them to fully-connected layers to classify classes as shown in Figure 2.3. These layers are made up of neurons that have learnable weights and biases. These neurons receive inputs from previous layers as a group of weighted inputs, applies an activation function, and returns an output. Fully-connected layers use loss functions, such as SVM or Softmax, to classify classes.

CNN results show that they are able to handle the limitation of low-level features which, along with their ability to learn multiple levels of abstraction, cause them to perform better. For fine-grained classification, Donahue et al. [38] achieved state-of-the-art performance on the CUB200-2011 bird dataset where they used a Deep Convolutional Activation Feature (DeCAF) for generic visual recognition. We will explain CNN layers and some case studies of various recent proposed network architectures.
There are many deep learning architecture modules, we show a few examples in Figure 2.4. We will describe the various neural network layers that are commonly used for classification tasks.

**Convolutional Layer (CONV):** The main purpose of a convolutional layer is to extract features from the input image. This is done by applying a filter or filters on the whole input image. The resulting matrix is called an Activation Map or Feature Map. This creates a clear filter which acts as a feature detector for the image. Most network architectures apply Rectified Linear Unit (ReLU) after every convolution operation to introduce non-linearity into the convolutional layer. ReLU replaces all negative pixel values in the Activation Map by zero.

**Pooling:** The goal of the pooling step is to reduce the dimensionality of each Activation Map and keep the most important information. Pooling can be performed via different techniques such as Max, Average, and Sum. For example, Max pooling creates a window e.g. (3x3), and then you slide the window over each Activation Map and take the maximum value in each patch. Reducing the Activation Map dimension will reduce the number of parameters and computations in the networks. Another benefit from the Pooling step is to generate statistical features over small regions.

**Fully Connected Layer (FCL):** The purpose of the Fully Connected Layer is to classify high-level features that are the output from convolutional and pooling layers. It uses an activation function such as Softmax and SVM to classify these features.
Example of Network Architectures

In this section we show some examples of popular CNN architectures. Some of these networks have produced state-of-the-art results in the ImageNet dataset.

**AlexNet:** We start with the AlexNet network which was presented by Krizhevsky et al. [2] in 2012. It achieved the state-of-the-art of image classification on the ImageNet dataset. The network has five CONV layers and three FCLs. It has more than one billion parameters and takes five to six days to train on two NVIDIA GTX 580 3GB GPUs. The architecture reduces over-fitting by data augmentation and dropout.

**VGGNet:** The idea of this network is to use a much deeper network and much smaller filters from beginning to end. VGGNet shows that by using a 3x3 filter for all layers in the network and adding between 16 to 19 layers, it can achieve 7.3 to error in the ImageNet dataset in 2014. Furthermore, Long et al. [39] showed VGGNet presented good generalization in multiple transfer learning tasks.
**GoogLeNet:** This network has 22 layers, which is more than AlexNet and VGGNet. The goal of this network is to solve the computation efficiency. GoogLeNet creates an inception model which applies several different filters (1x1, 3x3 and 5x5), in parallel, on top of the output of the previous layer. Also, they apply 3x3 max pooling on the same output of the previous layer. Then it concatenates all the filters and max pooling outputs together, depth-wise, and passes it on as input to the next layer. GoogLeNet is much deeper and has less parameters than AlexNet. It could achieve 6.5% top 5 error which was the state-of-the-art in the ImageNet dataset in 2014.

**ResNet:** This network was presented by He et al. [40] in 2015 and won the challenging of the ImageNet in that year. It achieved high performance by using residual blocks, which allows deep training of the neural network at 152 layers. The network used batch normalization to compensate for irrelevant variations in every layer. Furthermore, ResNet is trained to go much deeper than the other networks and requires a similar number of parameters.

**Transfer Learning**

Using a deep learning framework needs a large amount of data to obtain reasonable performance and avoid over-fitting. However, some tasks, such as fine-grained classification, has significantly smaller datasets that are unlike the ImageNet dataset which has 1000 classes and contains 1.2 million images. To solve this problem, Donahue et al. [38] presented a transfer learning approach. It trains a network on a large dataset, such as ImageNet, to learn general features and then fine-tunes the network parameters on the target dataset.
2.5. This approach works well on CNN because layers in the beginning of CNN are used to learn low- or mid-level features, which can be used to initialize other networks. Donahue et al., [38] achieved the state-of-the-art on fine-grained recognition, object recognition, and scene recognition by using this fine-tuning approach.

Another fine-grained classification method focuses on localizing the discriminative object parts in an image, such as the front bumper of a car, to distinguish between different models. Many parts-based methods [41–43] have been created and used in prior work. Some of these works [41,42,44] used datasets that have parts annotations to train a supervised parts detector to classify two similar classes. However, labeling parts of hundreds or thousands of objects is laborious and cost-prohibitive. Consequently, researchers have created approaches that do not rely on parts annotations [8,45,46].
2.2 Localizing the Discriminative Object Parts Models

The fine-grained image classification is very challenging since many sub-categories, such as birds or dogs, are highly deformable. To address this challenge, researchers use object parts to perform classification. Their methods localize these parts and then use them in training and testing stages, which helps to reduce the negative effects caused by position and viewpoint variations. Some of the object parts methods label datasets to learn and recognize these parts. Another object parts method is to use a dataset to learn discriminative object parts.

Poselets [47] use keypoints to localize discriminative parts of people in images as shown in Figure 2.6. A poselets describes parts of objects that are tightly clustered in configuration and appearance space, creating a set of examples that are close in 3D configuration space. Each example has a corresponding rectangular patch of a given anatomical part of a person, using an HoG algorithm to extract features from each patch. These features
are used for training a linear SVM classifier to perform classification. Farrell et al. [27] exploited the poselets idea to create Bridlet, which detects the basic-level category (e.g. bird) by a configuration of volumetric parts, using a Pose Normalized Appearance Descriptor (PNAD) to classify sub-category levels as shown in Figure 2.7. In 2012, Zhang et al. [48] extended [27] work. They used a poselets detection model to generate a set of poselets-style activations on a given image, computing a set of local descriptors at each poselets activation. These descriptors were used to classify a fine-grained category model.

Liu et al. [49] showed transferring parts or keypoints from training to testing helped to improve fine-grained classification. They proposed an exemplar-based method which
detects the eyes and noses of dogs by learning exemplar-based geometric and appearance models from the dog training dataset. Even though the approach could accurately localize eyes and noses, it is parametric-based and is sensitive to new samples. Goring et al. [45] showed that a parametric-based model could not handle the large variations present in fine-grained recognition tasks. Their method was a non-parametric parts transfer which located parts of the test image by finding visually similar objects of interest on a training dataset using HoG as features. This technique provided a number of parts locations from the annotations of K training images, which were scaled proportionally to the bounding box of the test image. The non-parametric approach copied parts with a high degree of position and view variations in unseen images where Deformable Part Model (DPM) [50] and exemplar-method fail.

DPM [50] is another approach that has been used to localize objects or parts of an object. It computes the image pyramid, and on each level it computes a feature map using the HoG approach. The model has a root filter, high resolution part filters, and a deformation model. The root filter is used to detect an entire object, using higher resolution part filters to detect smaller parts of the object. The deformable model approaches have been used in many applications on fine-grained classification tasks [43,44,51]

Parkhi et al. [43] used DPM to detect and localize the head of an animal in an image. The drawback of this approach was having limited pose variations for the object that relatively affect the classification. While Chai et al. [44] showed the combination of parts localization, DPM, and foreground segmentation, GrabCut, could boost the fine-grained classification as shown in Figure 2.8. Goring et al. [45] presented the DPM approach as a Gaussian model for the part location, but they analyzed the distribution of object parts for a
Figure 2.8: Symbiotic Segmentation and Part Localization for Fine-Grained Categorization

fine-grained dataset and found they are not Gaussian. Their analysis showed a single DPM model cannot handle the pose variation in a fine-grained dataset.

More recently, Krizhevsky et al. [2] exposed Deep Convolutional Neural Network (DCNN) model for image classification. It could achieve the highest accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). They trained a large DCNN on 1.2 million labeled images. Girshick et al. [52] used a CNN that was used in image classification to detect many objects on an image. They made a big improvement on object recognition performance by making a connection between image classification and object detection. In addition, CNN has been translated to the fine-grained domain. Zhang et al. [41] introduced Part-based r-cnns for fine-grained category detection that detects objects and localize their parts. The method improved the accuracy of fine-grained bird classifications and parts detection. One of the drawbacks of this approach was that if the regional proposal failed to detect the object, the performance would be affected negatively. To solve this problem, Zhang et al. [53] presented an end-to-end DCNN to detect semantic parts and classifications simultaneously see Figure 2.9.
Using a dataset that has object parts labeling provides impressive results in fine-grained classification tasks but requires laborious manual annotation of images. Approaches that do not use object parts labeling [46, 54–56] are a more realistic setting for real applications. Duan et al. [54] presented a method that discovered attributes that are detectable by machine and understandable by humans. These attributes were detected by a latent Conditional Random Field (CRF) model. Visual attributes are intermediate-level features, and they help to provide more details about objects. Yang et al. [55] presented a template model for fine-grained classification that aligned object parts in two different images. This method detected the common shape patterns of object parts and co-occurrence relation of the shape patterns. The model used two benchmark datasets, the Caltech-UCSD Bird200 and the Stanford Dogs, to prove the success of the approach. Krause et al. [56] showed fine-grained classification tasks can be solved and achieve good performance without using any
parts annotations. They generated poses and parts alignment based on co-segmentation. Also, Jaderberg et al. [46] achieved high performance on fine-grained bird classifications without using any part annotations. They applied a spatial transformer network to a CNN to detect and transfer the object into the correct position. Using multiple transformers in parallel allowed the method to localize the discriminative parts of the fine-grained bird classification. Also, Lin et al. [8] achieved similar performance [46] without using annotation parts. They presented a bilinear CNN model which applied two stream networks at each location of the image and then multiplied their result using outer product and pooled to get the bilinear vector, which was used to classify classes.

2.3 Video Classification

Video classification has been studied for many years in the computer vision community. Many problems have been explored under video classification, such as action recognition [57], video retrieval [58], video-based fine-grained object classification [29], and video-based face recognition [59]. To solve these problems, researchers used low-level features to describe videos. However, recently most of the video classification approaches have used deep learning techniques to describe videos.

2.3.1 Low-level features

Video classification can be obtained by using motion and appearance feature. Feature extraction is densely or sparsely extracted features from local appearance and motion [60–62]. For example, SIFT-3D [63], HoG-3D [64], or Histogram of Optical Flow
(HoF) [65] are dense feature descriptors. The drawback of dense features is that they come with a large computational cost and are not efficient to use for real-time applications. Using sparse features for video classification has solved this problem [66]. In 2013, Wand et al. [67] presented dense trajectories with low-level features and could achieve high performance in human action recognition. Afterwards, features were extracted, and the bag-of-words (BoW) was used to encode these features into a fixed-sized video-level description. Support-vector machines SVMs was then used to classify these descriptors into multiple classes.

2.3.2 Deep Learning

In the previous section we show CNN can achieve high performance with still image classification problems in large scale datasets [68]. To use CNN in video classification tasks, we need to use motion features that are extracted form each frame and then pool
all information across time. In 2014, Karpathy et al. [69] used a CNN to extract features from each frame and achieved high performance over several traditional video classification methods. Their experiments showed pooling multiple frames is only marginally better than the single frame-based method, which showed that using CNN to learn motion features is not simple. However, Simonyan and Zisserman [15] presented a two-stream network architecture that incorporates spatial and temporal networks. The first network, Spatial Stream ConvNet, operated on individual frames, and the second network, Temporal Stream ConvNet, used optical flow displacement fields between several consecutive frames as input. This input described the motion between video frames and improved recognition results. Another work for video classification was deep 3-Dimensional Convolutional Networks (3D ConvNets) [70]. They used 3D ConvNets to learn spatial-temporal features form frames of videos. Since these methods have addressed the video classification problem, we will briefly review them in this section.
2.3.2.1 Two-Stream Network

The two-stream network [15] consists of two independent spatial and temporal convolutional networks as shown in Figure 2.11. The Spatial Stream Network uses static images to make predictions for each frame. This network helps background and context information in the frame to improve action recognition since some actions have high correlations with certain objects. While the Temporal Stream Network uses stacked optical flow maps between consecutive frames as input. The optical flow has been used to describe the motion difference between frames. Finally, the fusion of the two Softmax outputs of these two networks are used to perform video classification.

2.3.2.2 3-D Convolutional Network

Tran et al. [70] presented the deep 3-Dimensional Convolutional Networks (3D ConvNets) for action recognition. They replaced all 2-D convolution and pooling operations in CNN architecture with 3-D. They used 3-dimensional convolutional kernels to model multiple frames of information simultaneously. Tran et al. [70] showed that 3D ConvNets can model appearance and motion information simultaneously and outperformed the two-streams ConvNet on different video analysis benchmarks. Also, 3D ConvNets can provide
generic features when applied to various tasks, such as action classification and sports classification without restriction of the number of frames.

2.4 Video-based Fine-Grained Classification

As we saw in the fine-grained image classification section, classifying sub-categories on images is difficult due to well known problems of illumination, position, and size variation. However, videos have more challenges, such as noise, image blurring, low object resolution, varying object position, occlusions, and illumination. Videos not only bring new challenges but also offer new opportunities. Videos provide many examples of the same sub-category that help to cover varied appearances of the same subject. Video-based fine-grained classification has received widespread attention for decades due to its wide-
applicability. However, most of that attention was focused on one example of fine-grained classification, which is face recognition [71–73]. Only recently has this interest spread into other examples such as birds [12, 28] and pedestrians [29].

Most of the face recognition methods include three steps, which are face detection, aligning, and matching. Researchers have created many designs for each step in order to improve face recognition in videos. Josef et al. [71] built sets of faces for individuals by using face detection in each frame and then used those sets to match between two different people in different videos as shown in Figure 2.13. Finally, they used face detection on several frames by using region tracks. There were five regions they used to discern whether two faces were from the same person, which were left and right eyes, tip of the nose, and center of the mouth. These regions were also used to describe each face in the sets. One drawback of their approach was that it just focused on the front view of faces as it could not handle a profile view. To solve this problem, Sivic et al. [74] added features that could be
Figure 2.15: Eigen-pep for video face recognition

extracted from profile views and added them to the features of frontal views. Their method found more features because they integrated both profile and frontal features. They also used point tracking instead of face tracking, which allowed them to track more frames than previous works [71]. Their approach improved the accuracy of identifying more characters from seven episodes of the TV series “Buffy the Vampire Slayer.”

Pose variation is a major challenge for real-world face recognition. Li et al. [75] came up with an idea to divide faces, or face tracks, on images or videos into patches with multiple scales, and using SIFT they extracted features from each patch as shown in Figure 2.14. The next step of their model was to add patch features to locations on faces or face tracks. Faces on images or frames were represented as bag of spatial appearance features. Gaussian Mixture Model (GMM) was used to train these features on faces in the training dataset. To match two face images, the GMM components were used to identify a pair of image patches for the same person or not. One of the drawbacks of their model was high dimensional, which needs vast memory. Another disadvantage was that the model repre-
sentation may lose some important information regarding position variation presented in
the video, because it used a part-based probabilistic max pooling, which keeps a small part
of the feature descriptors. Li et al. [76] handled these drawbacks by first taking the mean
of the all represented features over all patches on the frames to create an intermediate high-
dimensional part-based video-level representation. Then they reduced the dimensionality
of intermediate high-dimensional representations by applying Principle Component Anal-
ysis (PCA) see Figure 2.15. To classify faces on videos based on those representations,
they adopted the joint Bayesian classifier [77].

Like many other computer vision tasks, deep learning has improved the accuracy of
face recognition. Yang et al. [78] proposed a Neural Aggregation Network (NAN) for video
face recognition as shown in Figure 2.16. The module took a set of images or frames of
videos as inputs and then sent them to two network modules. The first module was called
the feature embedding module, which was a Convolutional Neural Network (CNN) that
Figure 2.17: Learning Discriminative Aggregation Network for Video-based Face Recognition

mapped each frame to a feature vector. The second module was called the aggregation module, which contained two attention blocks. They aggregated the feature vectors (results of CNN) to a single feature vector that represented the inputted face images. Then they used this feature vector for recognition. Their module learned to use high-quality face images while ignoring low-quality ones (e.g. blurred or occluded). Another approach presented by Rao et al. [59] was a Discriminative Aggregation Network (DAN), which was a model with three sub-networks as shown in Figure 2.17. The first network was the aggregation network which took video frames as input and produced one or more discriminative images. The goal of this network was to remove the low-quality frames that may affect the recognition performance in both speed and accuracy. The second network was the discrim-
inator network, which checked the aggregation network results and decided if they were selected from original video or generated by the aggregation network. The last network was the feature generator network, which extracted features from the aggregation network results and then classified them. Their methodology was different than [78] since they aggregated video frames directly before extracting their features. This approach helped to remove low-quality frames before features extraction, which can be hard to find in the feature space.

These methods have solved Video-to-Video face recognition problems that match a test video to a set of target videos. However, there are also some works that use the Still-to-Video face recognition system such as [79] see Figure 2.18. The challenge of this technique is matching faces in low-quality videos against high-quality still face images. Huang et al. [79] selected the best quality frames from videos and then aligned faces on those frames to faces on still images. Frame faces can be well-aligned with still faces and can be added to the gallery faces with the same identity. Their experiment showed that selecting high-quality frames and then adding well-aligned faces to gallery faces helped to improve the performance of face recognition systems.

All these papers were focused on faces to classify people from images or videos, while Hall et al. [29] used the entire human body to classify people Figure 2.19. They recognized pedestrians from real-world movies which they created with annotated data. They tested their movies with state-of-the-art algorithms for fine-grained classification and position estimation and reported the results as the baseline for their movies. They used [6] as the baseline for fine-grained classification. This method was proposed by Branson et al. [6] and used deep learning to extract features from image regions that were normalized
Figure 2.18: Coupling alignments with recognition for still-to-video face recognition

by position. For example, the object in a test image was divided into regions and then each region was warped to align with a set of prototypical models. These warped regions were then sent to the CNN to extract features from each warped region. In the last step of this model, they concatenated features that were extracted from each warped region to classify the object in that image. This model achieved the state-of-the art what on the bird species benchmark [9]. For the position estimation baseline, they used [80] which was presented by Chen and Yuille in 2014.
Another example of fine-grained classification on videos was IBC127 [12] which created a bird video benchmark. The goal of this example was to classify bird species that were present in videos. Tomoaki et al. [12] used image and motion features that were extracted from videos. To extract image features, they took the mean of the output of the hidden layers of this network [2]. Furthermore, for motion features, they took the mean of the improved Dense Trajectory (iDT) [57] encoded using Fisher Vectors (FV) [81] for all frames. So they have two classifiers, one for image features and the other for motion features, and the scores for both classifiers were aggregated during late fusion.
2.5 The Fine-grained Datasets

Creating fine-grained classification datasets is difficult since a label and bounding box for each sub-class needs to be created to locate different parts of the object in a dataset. The computer vision data collection community has created several tools, such as online crowd-sourcing technologies [82] and advanced methodologies [83], to help collect large-scale datasets. Collecting these datasets using those tools helps to accelerate the progress in various object recognition tasks [21, 68, 84].

2.5.1 The Fine-grained Still Images Datasets

The Machine Vision Group of Stanford University created the ImageNet dataset which contains 1000 categories and 1.4 million images. The structure of the ImageNet dataset is built based on WordNet hierarchy. ImageNet is a large dataset that enables training of computer vision models in order to achieve a high accuracy of image recognition. It cannot be used as a fine-grained dataset since 1000 classes do not belong to the same category. The fine-grained community has released various fine-grained datasets, such as bird datasets: CUB-200 [9], Cars dataset [85], the Comprehensive Cars (CompCars) dataset [7] and Labeled Faces in the Wild (LFW) dataset [86].

CUB-200 [9] contains 11,788 images of 200 bird species Figure 2.20. Each bird species category contains about 60 images. These images are downloaded from Flickr image search and annotated via Amazon Mechanical Turk. Each image is annotated with a bounding box around the bird and 15 parts of the bird are annotated by pixel location. The parts of the bird that is used are the back, beak, belly, breast, crown, forehead, left/right eye,
left/right leg, left/right wing, nape, tail, and throat. This dataset is considered a challenging dataset because it has a large variation of poses in each category.

CompCars [7] is another fine-grained dataset, and it contains 208,826 images of 1,716 car models, including images from web-nature and surveillance-nature Figure 2.21. The web-nature’s images are collected from car forums and public websites while the surveillance-nature’s images are collected by surveillance cameras. CompCars dataset contains 163 car makes with 1,716 car models. The web-nature data contains 136,727 images capturing the entire car and 27,618 images capturing car parts. Most of these images are labeled with attributes and viewpoints. There are 44,481 images in the surveillance category that captures the front view of the car. Each surveillance image is annotated with bounding box, model, and color of the car. The CompCars dataset provides more features than other existing car datasets. It provides namely car hierarchy, car attributes, viewpoints, and car parts.

Another dataset for fine-grained classification is LFW [86], which is a face recognition dataset. Face recognition is a striking example of fine-grained classification and there
are many questions that can be solved using a face recognition dataset. For example, if there are two images and each one contains a face, can the system recognize whether they are the same person? Another problem that can be studied using a face recognition dataset, is determining the name of the person present in the image. The LFW dataset, created in 2007, has 13,233 target face images. Some images may have more than one face, however, the defining face for the image is the one that contains the central pixel of the image. LFW has 5,749 people where 1,680 people have two or more images while 4,069 people have just a single image in the dataset. Each one of the 5,749 people have a unique name, such as George W Bush. Viola-Jones’ face detector [87] has been used to detect the face in an image and then crop and scale the face region that returned from the detector. The new images were saved in the JPEG 2.0 format.

In the next subsection we will present video fine-grained classification datasets. Videos usually provide far more information than single images [88, 89] and also have more challenges than images, such as motion blur, problematic lighting, position, and occlusions [89].
2.5.2 The Fine-grained Video Datasets

The YouTube Faces Database [89] is a large scale dataset that was designed to study the problem of unconstrained face recognition in videos see Figure 2.22. To create the dataset YouTube Faces Database used 5,749 classes in the LFW [86] to define elements of the dataset. They created queries to download any video that was related to the class name, then the top six results were downloaded, using the video’s name to delete duplicate videos. Each video was converted to frames at 24fps and then a VJ face detector [87] was used to detect faces in the frames. They automatically eliminated videos that were not long enough to provide useful information (less than 48 consecutive frames). This step reduced classes of the dataset from 5,749 to 1,595 and videos from 18,899 to 3,425. Each class had an average of 2.15 videos and the number of frames in each video was between 48 and 6,070 frames. Each bounding box around the face was expanded by 2.2 and then cropped from the image. The cropped image was resized to 200 x 200 pixels and cropped to 100 x 100 pixels centered on the face.
The YouTube Faces Database recognizes people using just their face while David and Pietro [29] created the Caltech Roadside Pedestrians (CRP) [29] dataset that recognizes people using the entire body. CRP contains multi-class sub-categories, such as age, gender, clothing style, and body shape. Furthermore, the dataset is annotated by bounding boxes, tracks, and 14 pose keypoints with occlusion information. The creators of the dataset used workers from Amazon’s Mechanical Turk (MTURK) since the dataset required a large number of annotations. CRP has seven videos that were captured by mounting a rightwards-pointing, GoPro Hero3 camera to the roof of a car. Each video was created at a resolution of 1280 x 720 pixels with a frame rate of 30fps. The average number of frames in each video is 37,000 while the total number of frames in the whole dataset is 261,645.

Drawing the bounding box around each pedestrian in each video is the first step in the annotation step. The instructions for labeling the CPR dataset is to send every tenth frame in the video to MTURK and then ask three workers to draw a bounding box around every pedestrian in the frame. The three results were combined into a single set of bounding box labels for each frame using clustering. The next step was tracking the pedestrians from the bounding box. The workers of MTURK were given a cropped frame X of a person and then were asked to select from the set of cropped people from frame X+5 that matched the frame X. The final step of CRP annotation was the pose annotation, which was also created by MTURK workers.

Recently, the fine-grained community has given video fine-grained tasks more attention [12, 28]. They have created video fine-grained datasets, such as Videos of 100 Bird species (VB100) [28] and the IBC127 dataset [12].
The VB100 dataset contains 1,416 video clips of 100 bird species taken by expert bird watchers. The dataset introduces many challenges such as large variations in scale, bird movement, camera movement, and considerable pose variations. Also, VB100 provides video clips, sound clips, taxonomy, and distribution location for each species of bird. Each class in this dataset has on average of 14 video clips with a median length of each clip at 32 seconds. VB100 contains 798 videos created by moving cameras and 618 videos created by static cameras.

IBC127 has more classes than VB100, at 127 fine-grained bird species Figure 2.23. Also, the number of videos is 8,014, which is substantially more than the 1,416 videos in the VB100 dataset. The dataset is collected from the Internet Bird Collection,\(^1\) which is an international website containing images, videos, and sounds of birds uploaded by users. The minimum number of videos in each class is 21 and the maximum number is 226.

\(^1\)https://www.hbw.com/ibc
Recently, the fine-grained community has given video fine-grained tasks more attention [12, 28]. They have created video fine-grained datasets, such as Videos of 100 Bird species (VB100) [28] and the IBC127 dataset [12].

Likewise, Chen [90] introduced two large-scale datasets that are YouTube Birds and YouTube Cars, see Figure 2.24. They used the taxonomy of the CUB-200-2011 and Stanford Cars to create YouTube Birds and YouTube Cars datasets. The YouTube Birds dataset has 200 different bird species, while YouTube Cars contains 196 different car models.

The YouTube Birds include more classes than IBC127. The minimum number of videos for each category in both datasets is 6, while the minimum number is 207. They use object detection to ensure videos have a bird or a car however, videos may have small objects and most object detection frameworks continue to struggle with small objects. Detecting small objects is difficult since they have lower resolution and larger influence of the surrounding environment. This approach may remove all videos have challenging scenarios that have small objects and keep videos that have large objects. They use a crowdsourcing system to annotate each video. They show part of the videos to the workers and asked them if these videos belonging to correct category. Also, they remove videos that
have more than multiple subjects from different categories which makes the dataset has less challenging.

Chen [90] presented Redundancy Reduction Attention (RRA) approach that aims to solve the redundancy problem that occurs in videos. Videos have a large numbers of information in the form of frames but they also contain a lot of redundant and irrelevant frames. Their approach reduce the redundancy by just focusing on discriminative patterns and blocking non-discriminative channels. The model divided a video into number of clips of the sam length. Then choose one frame or flow stack from each clip. Then the RRA iteratively updates the feature maps that are extract by the CNNs from the sampled frames. The soft attention mask is applied over each feature vector of input feature maps to weight-sum the feature maps into a summary feature vector and then each summary feature vector gives one classification score by the classifiers. Finally, they obtain the final prediction by averaged the scores that are provided by classifiers. Using Attention Mechanisms helped to extract salient regions from frames, while ignoring irrelevant parts and frames.
CHAPTER III

THE CV100: CARVIDEO 100 DATABASE

Automobiles are an essential aspect of modern life that provide flexibility and mobility to users. Given their necessity, and even their position as a status symbol, every year new and different models are designed and introduced while existing models remain in use on the roads. As a result, there is a critical need by law enforcement, and others, to utilize photographic and video surveillance to be able to quickly and easily identify vehicles captured through these methods. The ability to recognize or identify car models is an important task in computer vision models. In Figure 3.1, both the dramatic differences in appearance and perspective are shown as well as the very subtle differences between car models.

In general, car datasets greatly facilitate research in fine-grained classification. Due to the competitive retail nature of the auto industry, large amounts of pre-labeled data from a great number of models is readily available. Individual makes and models present significant differences in appearance when perspective and lighting change. Moreover, they have small inter-class variations because of other subtle differences, as shown by Figure 3.1.
Figure 3.1: Each column presents a large appearance differences in their poses and the first raw illustrates the subtle difference between car models.

The car category is divided into three levels: make, model, and year (see Figure 3.2). This tiered structure of the class label is used in many computer-driven identification methods, including fine-grained classification and verification of cars, and has several applications. For example, it can be used to identify a car with an unreadable or missing plate. While most applications are more relevant to law enforcement and surveillance agencies, this system might also be used by a consumer to compare car models of similar appearance for price, mileage, handling, environmental impact, and other important factors to be considered before purchase.

Fine-grained car classification in still images has attracted a lot of attention in the computer vision community. Despite being static, such images also have a multitude of characteristics that must be examined to achieve identification of the image. Several approaches have been proposed in the still image identification arena [7, 8, 85, 91], and a few datasets have been created to simplify fine-grained classification [7, 29, 92]. However, a still image presents only a limited number of object views and dimensions for fine-grained
classification. Utilizing video can address this lack as it provides far more information. In video, the object may be presented from different viewpoints and a variety of angles in many consecutive frames, as shown in Figure 3.3. Many existing methods have achieved high recognition performance by taking advantage of the fact that the object to be identified may present in a video in many frames [93–95]. Video is also useful in fine-grained object classification since it shows different features of objects. Although fine-grained car classification in still images has been studied for a decade in computer vision [7, 85], classification in video is still in its infancy. To expand the field of classification in video, Alsahafi [93] developed the CarVideos dataset for fine-grained object classification in video. The dataset contains hundreds of thousands of frames and ten different car model classes.

In this chapter, we introduce a large-scale and comprehensive video dataset called CarVideo 100(CV100), which contains 495,778 frames of 100 car models. Specifically, it is annotated with bounding boxes and attribute labels, which have not been provided by other fine-grained video datasets [11, 12]. Each frame is annotated with bounding boxes and attributes. CV100 can be used to validate many computer vision algorithms, and has numerous practical applications as well as enormous potential for supporting future research.
To collect such a dataset is difficult and has many challenges. The first challenge is the fact that car model designs change every year, unlike natural objects [9, 10, 96]. Even if we treat the car model images produced in different years as a single category, the classification model may treat them as different categories because their differences are relatively small as shown in Figure 3.4. The second challenge is that labeling a large-scale dataset is time consuming and can require a substantial monetary contribution.
3.0.1 Video Collection

To collect videos for the CV100 dataset, we initially considered solely using YouTube as the video-sharing site now has more than several billion videos available [97]. However, the task of labeling 100 classes of cars just from YouTube videos would have taken far too much time, so we adjusted our data collection method. We searched for a website that included the names of many car models and found the TRUECar\(^1\) site. This site lists each car manufacturer and also provides the year, make, and model of each vehicle available, as shown in Figure 3.5. We created a script to extract all manufacturers, along with year, make, and model of their offerings, and put the information into a text file. Our text file had hundreds of car model names, from which we choose 100 classes. We then created another

\(^1\)https://www.truecar.com
Figure 3.5: TRUECar has many car manufactures and each car manufacture has several of car models

script to read the text file and returned to YouTube to download 20 videos from the site for each category.

We manually checked each video to verify that it portrayed the correct car class. To narrow our results further, we removed videos that did not have a good variety of views of the car model. For example, some videos presented just one side of the car throughout the whole clip, which did not provide the kind of perspective needed. Also, some videos focused solely on interior features and never showed the exterior of the vehicle. Furthermore, we confirmed that the videos did not overlap between the training and testing sets. For example, we found some videos were repeated under many other titles so we did a manual examination to locate and delete all duplications. We then used MoviePy\textsuperscript{2}, a Python module for video editing to remove parts that do not have useful information for recognition algorithms. As an example, videos may start with a number of frames of a black screen and a company logo but no images of any car, as in the first row of Figure 3.6. We also removed frames that showed the inside of vehicles, as in the fourth row of Figure 3.6. Finally, we

\textsuperscript{2}https://zulko.github.io/moviepy
Figure 3.6: We delete the first and fourth rows since they do not provide any car information.

split each video into frames at 30fps. Figure 3.10 presents examples of the car models we chose.

3.1 Annotations

To annotate each frame on the dataset, we used workers from Amazon Mechanical Turk (MTurk). MTurk is an online crowdsourcing platform designed to help individuals and businesses to complete various tasks such as data collection, survey participation, content moderation, and more. MTurk assisted graduate student researchers in collecting datasets in a reasonable amount of time. Figure 3.7 shows an example of creating a new project on MTurk. MTurk has over 500,000 different workers from 190 countries who helped us complete tasks in a short time. In MTurk, requestors post the tasks as Human Intelligence Tasks (HITs). These HITs were completed by workers during a specific time for a set, usually small, fee. MTurk was a fast and inexpensive way to collect data, evaluate systems,
and provide categorical annotations for training data. While MTurk can help gather data in an abbreviated length of time, certain drawbacks should be considered when using this crowd-sourcing service. Unfortunately, some workers chose bad labels in order to maximize their pay [98]. They supplied quick answers that had nothing to do with the correct label. Hovy [98] referred to them as spammers. Usually, requestors collect multiple annotations of the same instance so they can choose the best label and reduce the effect of spammers. However, this approach will cost more money so instead of paying one worker for one instance, we paid multiple workers for the same instance. We used a different approach, we created the HIT that was an example of our data and submitted that to Workers. We chose workers that had labeled our data correctly. MTurk allowed requesters to create Qualification Types. System Qualification Types control who can and cannot work on our HITs. So we chose workers that labeled our data correctly, as shown in Figure 3.8. We did
that multiple times until we collected 245 workers that knew how to draw a bounding box around objects and which attributes we should choose for each object.

We created a user interface for MTurk workers to provide location information and attribute annotations, as seen in Figure 3.9. We collected three types of annotations: (a) bounding box, (b) identification of car model, if the frame included more than one car, and (c) attributes as described in Task 2 on Figure 3.9. Workers were asked to draw a bounding box, in order to create scale awareness, on each vehicle in the frame and then choose the car model we wanted to label. Likewise, workers annotated each frame with one or more of the following attributes: (a) the car is obvious, (b) the car is not obvious, (c) part of the car is obvious, (d) the car is not our target, (e) more than one car model, (f) no car in the frame, and/or (g) the car is occluded.
Figure 3.9: User interface for MTurk workers.

Task 1:

Draw a rectangle using mouse over each car.

If there is no car on the image skip this Task (Do not draw any box). You can go to next task and choose the attribute (There is no product in the image)

If the view inside the car do not draw any box, you can go to next task and choose the attribute (There is no product in the image)

Task 2: Choose one or more the attributes that are describe the target car (S[2015 Toyota Camry XSE]).

Tick any of the boxes below if they are true about the image.

- Product is occluded
- There is more than one product
- Product is truncated
- Detect and recognize the product was easy
- Detect and recognize the product was hard
- The brand of the product was obvious
- There is no product in the image
- There is a product but I can not tell what is the brand
Figure 3.10: Examples of our car models.

3.2 Conclusion

In this chapter, we introduced the CarVideo 100 (CV100) which was designed to study fine-grained classification of cars. This large-scale dataset contains a variety of viewpoints within its 100 categories of car models and 896 videos clips containing 495,778 frames. It was created with multi-label attributes (car is obvious, car is not obvious, part of car is obvious, car is not our target, more than one car model, no car in the frame, and car is occluded) and detailed annotations.
CHAPTER IV
FINE-GRAINED VIDEO CLASSIFICATION

4.1 Fine-Grained Video Classification

After we prepared our CV100 dataset, we start with a brief review of our chosen baseline methods that make use of spatial and temporal information: a two-stream network [99] and a 3-Dimensional Convolutional Network (3D ConvNet) [70]. These two methodologies are currently considered state-of-the-art on spatial-temporal feature learning for action recognition and represent a suitable baseline for our new dataset. We also evaluate the performance of the state-of-the-art of fine-grained classification for still images, Navigator-Teacher-Scrutinizer Network (NTS-Net) [91]. We then describe our approach, which combines an SSD [13] and a CNN [100].

4.1.1 Temporal Segment Networks & 3D ConvNets

Video-based recognition represents an interesting classification problem since it includes both temporal and spatial features. Several network architectures have been proposed for fine-grained object detection in videos such as TSNs [15,99], 3D ConvNets [70],
and others. The primary goal of these networks is to utilize both the temporal and spatial information available in videos, including motion between frames, to classify the object.

**Temporal Segment Networks:** TSNs are built to capture long-range temporal structure [99] and are an extension of the two-stream network architecture [15]. Two-stream networks have achieved great success in video-based recognition [14, 15, 99] by incorporating optical flow (temporal) and RGB color (spatial) information. The TSN approach divides a video into K-segments and then randomly selects one short snippet from each segment. TSNs take these short snippets as input and produce class scores for all the classes from each individual snippet. All of these results from the TSN are passed to a segment consensus function to obtain a video-level prediction.

**3D Convolutional Networks:** 3D ConvNets add an additional time dimension to a ConvNet to allow for both spatial and temporal information to be used for classification. 3D ConvNets use 3-D convolutional filters instead of 2-D filters and takes a segment of video as input. 3D ConvNets have been shown to outperform 2D ConvNets for video classification [70, 84, 101].

**Navigator-Teacher-Scrutinizer Network:** We also evaluated Navigator-Teacher-Scrutinizer Network (NTS-Net) [91], which is the state-of-the-art method for fine-grained classification of still images. The Navigator, the Teacher, and the Scrutinizer are agents that improve fine-grained classification. The Navigator agent detects the most important regions on an image and sends them to the Teacher agent. The Teacher agent evaluates each region and returns it to the Navigator if there is a probability that the region belongs to the ground-truth
SSD-CNN: Our novel fine-grained video object classification method uses a CNN [100] for classification but introduces an SSD object detector [13] as a pre-processing step. Using an object detector allows us to be much more precise when training and testing our network.
Figure 4.3: An example of the crops for each possible classification image which consists of four corner crops and a single center crop. The crops are then passed from this output into our CNN.

as we are able to ignore much of the non-essential information while still capturing and training on the most valuable parts of an image. The CNN can take either an RGB image or optical flow image as input. In addition, to capture multi-frame information, we compute an average prediction over a sequence of frames from the video.

Figures 4.1 and 4.2 give an overview of our SSD-CNN architecture. Objects or object part bounding boxes are detected by the SSD and then cropped and resized for input to the CNN. Each bounding box is separately classified by the CNN. Our architecture uses a ResNet-152 [100] for both the spatial and temporal (optical flow) streams.

4.1.2 Implementation Details

In this section we explain the implementation details for the baseline methods (TSN, 3D ConvNets, and SSD-CNN) we evaluated with the CV100 dataset.

TSN. Our first baseline divides the clip into three segments and a short snippet is randomly chosen from each segment. Each snippet is sent to a two-stream ConvNet, both streams of which use the BN-Inception architecture [102]. The BN-Inception architecture weights are initialized from models pre-trained on ImageNet [68]. The spatial stream utilizes a single RGB image while the temporal stream uses five consecutive optical flow images as input. To extract the optical flow for our CarVideos dataset frames we use the
TVL1 optical flow algorithm from [99]. The TSN approach uses the mini-batch stochastic gradient descent algorithm to learn the BN-Inception network parameters with a batch size of 256 and momentum of 0.9. We use the same TSN learning rate for spatial and temporal networks. For spatial networks, the learning rate starts at 0.001 and is multiplied by 0.1 every 2,000 iterations to a maximum of 4,500 iterations. For the temporal network, the learning network starts at 0.005 and is multiplied by 0.1 after 12,000 and 18,000 iterations before training ceases at 20,000 iterations. The TSN model is tested by taking 25 RGB images or optical flow stacks from the test video. Each frame is cropped to four corners and one center and flipped horizontally as well. Each clip is classified by taking the average score over the cropped and flipped frames.

**3D ConvNets.** The 3D ConvNet has eight convolutional layers, five pooling layers, two fully connected layers, and a softmax output layer. The network uses convolutional kernels of $3 \times 3 \times 3$ to model (16) frames of information simultaneously. We trained a 3D ConvNet fine-tuned from the model pre-trained on the Sports-1M. We follow the original paper’s approach [70] which takes five two-second long clips from each training video. Then we crop each two-second long clip into $16 \times 112 \times 112$ crops for spatial and temporal images. To obtain a video prediction, we randomly extract ten sub-clips from the video clip and take the average score over the ten clips.

**SSD-CNN Training.** Our implementation consists of two networks: an SSD [13] and a CNN [14]. The SSD network is used to locate objects in each frame and segment them using a bounding box and a label. Figure 4.4 shows the SSD network instructions. We only use the bounding boxes labeled “car.” The SSD network is an SSD300 that is pretrained and validated on the Visual Object Classes (VOC) train and validation datasets [14] provided
Figure 4.4

by https://github.com/amdegroot/ssd.pytorch. The SSD300 architecture [13] takes an input image size of size $300 \times 300$ and outputs a set of bounding boxes with confidence values in the range $[0, 1]$. Figure 4.1 shows examples of the SSD output.

Following the SSD detector stage, cars are cropped from the frame and converted into individual images before passing them to the CNN. Due to variations in the size of each bounding box, we scale each individual box to $224 \times 224$ before passing it to the CNN, as shown in Figure 4.2.

We prune and label the bounding boxes before providing them to the network as training data. We reject any bounding box with confidence less than 0.6. If a bounding box overlaps the target vehicle with Intersection-Over-Union (IOU) greater than 0.7, it is labeled as the target vehicle. Otherwise, it is labeled as not the target vehicle. If the image has no predicted bounding boxes, we pass the entire image to the network and label it as the target vehicle.

We consider data augmentation to be one of the important parts of training the SSD-CNN combination. Our data augmentation approach is adapted from [14]. After resizing each bounding box to $224 \times 224$, we apply four corner crops and a center crop, as shown
in Figure 4.3. This allows the network to train on parts of cars and thus be able to identify partially occluded cars by the part that is seen.

Following [14], we fine-tune a ResNet-152 [100] that has been pre-trained on ImageNet [68]. We fine-tune the network with stochastic gradient descent (SGD) momentum of 0.9 with a batch size of 25. We start with a learning rate of 0.001, divide it by 10 after every 4,000 iterations, and stop after 10,000 iterations.

**SSD-CNN Testing.** Our testing is similar to [14] and [15]. For testing we extract a series of 25 frames and apply our data augmentation approach to each frame. Each clip is classified by taking the average score over the cropped and flipped frames.

### 4.1.3 Results

We used mean classification accuracy to measure the performance of each method on the test set. A clip is counted as a true positive if the classifier correctly predicts the target car, if the target car is in the clip, or the classifier correctly predicts “no car.”

We evaluated the SSD-CNN (25 frames) and baseline methods of our new CV100 dataset. The evaluation was done using 10-fold cross-validation, which splits the dataset into training and testing sets. Here, \( k - 1 \) folds were employed for training and the remaining fold was used for validation, \( k \) times.

Table 4.1 presents the accuracy and speed comparison of the the SSD-CNN (25 frames) with baseline methods for 10-fold cross-validation of the CV100 dataset. The fine-grained object classification in still image NTS-Net [91] achieves higher performance than video classification methods TSN [99] and 3D [70], since NTS-Net localizes the discriminative object parts in frames. However, NTS-Net runs at 8fps which is slower than
Table 4.1: Accuracy and speed comparison of the SSD-CNN (25 frames) with baselines techniques for 10-fold cross-validation on CV100:CarVideo 100 dataset. Last column shows p-values from 10-folds cross-validation experiment; the SSD-CNN (25 frames) is statistically significantly better than the baselines.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Accuracy (in %)</th>
<th>Std. Deviation</th>
<th>FPS</th>
<th>p-value (25 frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3D [70]</td>
<td>44.7</td>
<td>6.037843618</td>
<td>27</td>
<td>1.22407E-12</td>
</tr>
<tr>
<td>TSN [99]</td>
<td>62.4</td>
<td>4.452215429</td>
<td>64</td>
<td>1.85287E-09</td>
</tr>
<tr>
<td>NTS-Net [91]</td>
<td>67</td>
<td>6.539622823</td>
<td>8</td>
<td>1.55453E-05</td>
</tr>
<tr>
<td>SSD-CNN (25 frames)</td>
<td>79</td>
<td>2.282785822</td>
<td>10</td>
<td>-</td>
</tr>
</tbody>
</table>

TSN and 3D. Conversely, SSD-CNN with (25 frames) outperforms NTS-Net in both accuracy and speed. It localizes the discriminative object in the sequence of frames and utilizes the temporal information in video. SSD-CNN classifies each video by taking the average score over multiple frames that do not share information features.

We use paired t-test to determine the accuracy of the SSD-CNN method compared to baseline methods. The last column on Table 4.1 presents p-values for 25 frames from the 10-fold cross-validation experiment. The SSD-CNN network (25 frames) performed significantly statistically better than the baseline.

In Figure 4.5, we show the Precision-Recall plot for each method. The areas under the Precision-Recall curve were obtained by the SSD-CNN (25 frames) and baseline methods. The SSD-CNN (25 frames) significantly outperforms all baseline methods across the CV100 dataset in term of Precision-Recall curve.

In Figure 4.6, we show four test samples of classification and their prediction results. The first row presents the correct classification while the rest shows the failure cases. Rows two and three present SSD network that provides the correct object (car) to CNN. However, CNN misclassified. The last row shows that the SSD network could not detect the object in the frame and then CNN misclassified the car.
4.2 Conclusions

In this paper, we present an evaluation of the performance of baseline methods for detection and recognition of fine-grained object categories in videos. Our evaluation includes a new detection and recognition approach using SSD object detection to identify objects or object parts on frames and a deep CNN with appropriate training to classify these inputs. Our SSD-CNN outperforms Temporal Segment Networks (TSN) and 3-D Convolutional Networks, which are state-of-the-art on human action recognition in videos.
Future work includes evaluating methods designed specifically for fine-grained object classification, such as bilinear pooling [8]. We also intend to add more videos and classes to the dataset in the future.
CHAPTER V

SCALE-AWARE LSTM NETWORK FOR VIDEO FINE-GRAINED CAR CLASSIFICATION

In the previous chapter, we showed how detecting the object in each frame helped to classify the frame correctly and also helped to classify the entire video correctly. However, we classified each frame by itself using classifier SVM and then using these results of several frames to classify the entire video. In this chapter we plan to aggregate features of multiple frames using A Recurrent Neural Network (RNN) to show the benefit of using temporal information in the video domain.

This chapter presents a novel approach to fine-grained car model categorization that utilizes a Scale-Aware Long Short-Term Memory (SA-LSTM) to address the fact that fine-grained car model classification for video has different characteristics and algorithmic requirements than the same process for still images. Over the years, several approaches and datasets have been proposed for fine-grained car model categorization in still images, but the enormous potential of video data to improve vehicle classification has largely been ignored.
Figure 5.1: The SA-LSTM approach.

This model uses Single Shot MultiBox Detector (SSD) to localize objects, by establishing their scale (height and width) with bounding boxes, from the sequence of frames in which they appear. The Convolutional Neural Network (CNN) extracts the features from these objects and object parts. Our technique combines object scales with CNN features that acknowledges how, in video, objects will present with many variations of size and scale in a sequence of frames. This information is fed to SA-LSTM, creating a fine-grained classification method that takes into account variations in size and scale that occur not only in video but that might, under certain circumstances, be an issue when classification involves still images. Our findings show that this novel SA-LSTM approach significantly outperforms existing video classification methods and also state-of-the-art fine-grained classification methods developed for still images, in terms of the accuracy of the dataset.
5.1 Baselines and Implementation Details

In the following, we evaluate the performance of several video classification methods and provide specific information on the selected structure and parameters of the baseline methods. We test fine-grained classification in both still image and video approaches. Moreover, we explain the implementation details for SA-LSTM using object scales plus LSTM for fine-grained object video classification.

Many network architectures have been proposed for video-based recognition, including TSNs [99] and 3D ConvNets [70]. TSNs are built as an extension of the two-stream network architecture [15] that incorporates temporal and spatial information present in video to improve video-based recognition. The TSN approach divides each video into three segments and then chooses one short snippet from each segment. Each snippet is sent to two networks (Spatial ConvNet and Temporal ConvNet) that use the BN-Inception architecture [102]. The first network (spatial) uses a single RGB image as input, while the second network (temporal) uses five consecutive optical flow images as input. These networks provide the class scores of different snippets that are fused by the segmental consensus function, which includes video prediction. TSNs allow ConvNets to share parameters on all snippets. TSN networks initialize with the mini-batch stochastic gradient descent algorithm to learn the BN-Inception network parameters and momentum of 0.9.

The second baseline method is 3D ConvNets [70], which presents a different approach to predicting video classes. This method can model temporal information better than conventional 2D ConvNets, which are used to process images. 3D ConvNets is used to process volume and is comprised of 16 layers that contain eight convolutional layers,
five pooling layers, two fully connected layers, and a softmax output layer. Essentially, 3D ConvNets added a time dimension to its predecessor. The time dimension is used to extract features from the sequence of frames simultaneously. The 3D ConvNets network pre-trained the model on Sports-1M. The approach takes five two-second clips from each training video and crops each clip into $16 \times 112 \times 112$ crops for spatial and temporal images. Video prediction is achieved by extracting ten sub-clips from the video clip and then taking the average score over all ten.

We also evaluated Navigator-Teacher-Scrutinizer Network (NTS-Net) [91], which is the state-of-the-art method for fine-grained classification of still images. The Navigator, the Teacher, and the Scrutinizer are agents that improve fine-grained classification. The Navigator agent detects the most important regions on the image and sends them to the Teacher agent. The Teacher agent evaluates each region and returns it to the Navigator if there is a probability that the region belongs to the ground-truth class. The Scrutinizer extracts features from the proposed regions and from the whole image to make the fine-grained classification. The approach resizes images to $448 \times 448$ and uses six regions to train the Navigator network to detect the most important regions in the image. The network uses ResNet-50 architecture [40] to extract features from images and Momentum SGD with an initial learning rate of 0.001, multiplied by 0.1 after 60 epochs. This approach allows agents to work together to improve fine-grained classification and to predict the most important parts of the image. However, the NTS-Net was designed to treat the fine-grained classification task as a still-image classification problem, and it ignores the useful temporal information present in a video.
To use the information present in video, Alsahafi [93] used multi-frame information to enable fine-grained car classification using video. The approach uses an SSD object detector [13] to find cars on each frame and then sends these to a deep Convolutional Neural Network (CNN) to classify them. Using object detection method allows this approach to remove the irrelevant information from each frame and to focus on the most valuable parts of the frame. As a result, this approach outperformed TSNs [99] and 3D ConvNets [70] methods of video classification.

5.2 The Proposed Approach

Our cutting edge, scale-aware approach, the SA-LSTM for fine-grained video object classification, allows processing of variable-length input sequences by localizing the most critical regions and providing a single prediction output as shown in Figure 5.1. Videos show the object from different perspectives and angles across the video sequence, which may help to increase recognition rates. We created a model that uses the Single Shot Multi-box Detection method (SSD) [13] to localize the essential areas of each frame. The SSD takes the frame as input and outputs a set of bounding boxes with confidence values in the range of [0 1]. Bounding boxes were only used to label the cars. The car model was cropped from a sequence of 30 frames and their original bounding box sizes (height and width) were saved. In video, the object size changes from frame to frame, and in some frames they are invisible, as shown in the first image of the first row of Figure 5.2. SA-LSTM uses the original bounding box scale to provide the SA-LSTM network more information about the
Figure 5.2: Sequence of frames in the video presents the car on different views and varying angles.

particular object, which trains the SA-LSTM to ignore unimportant frames, such as those where objects are too small and do not provide fine-grained detail.

Due to the variance of cropped images, we scaled each crop image to $300 \times 300$ before sending them to CNN. If SSD failed to detect the car in the frame or the frame did not contain any vehicle, we scaled the entire frame to $300 \times 300$. CNN was chosen to extract features from frames because it has helped to improve the performance of many image recognition tasks by providing stable features [103,104]. We trained the ResNet-152 network [40] on the CV100 dataset. We then chose the best ResNet-152 network trainable model to extract features for SA-LSTM. The pre-trained model provided the ResNet-152
network with a strong initialization that made training faster and reduced the chance of the need for over-fitting during the training of other datasets.

The CNN extracted 2,048 features from each cropped car image. Prior to cropping, we added the height and width of the original car image to the CNN features extracted, which increased the total extracted features to 2,050. Next, SA-LSTM was used to train the sequence of 2,050-dimensional feature vectors. The batch size for the model was set at 40 and the sequence of frames at 30 frames. So, SA-LSTM the input shape was $40 \times 30 \times 2050$.

LSTMs have achieved impressive results in recognizing patterns in sequences of data, such as language tasks, e.g. speech recognition [105] and machine translation [106]. For this project, a scale-aware LSTM was used to exploit the temporal information of car features for fine-grained car classification in video. Our SA-LSTM had 512 hidden units and one layer. We used the classification layer to classify SA-LSTM outputs. The SA-LSTM network used Cross-Entropy Loss as loss function and Adam as an optimizer to update the weight of the model during training. The learning rate for this model was 0.001.

### 5.3 Evaluation

We evaluated the SA-LSTM and baseline methods of our new CV100 dataset. The evaluation was done using 10-fold cross-validation, which split the dataset into training and testing sets. Here, $k - 1$ folds were employed for training and the remaining fold was used for validation, $k$ times. Table 5.1 presents the accuracy and speed comparison of the SA-LSTM with baseline methods for 10-fold cross-validation of the CV100 dataset. We also compared the SA-LSTM with the CNN-LSTM which did not use the object’s scale.
Table 5.1: Accuracy and speed comparison of the SA-LSTM with baseline techniques for 10-fold cross-validation on CV100:CarVideo 100 dataset. Last column shows p-values from 10-folds cross-validation experiment; the SA-LSTM is significantly better than the baselines.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Accuracy (in %)</th>
<th>Std. Deviation</th>
<th>FPS</th>
<th>p-value (30 frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3D [70]</td>
<td>44.7</td>
<td>6.037843618</td>
<td>27</td>
<td>8.78236E-14</td>
</tr>
<tr>
<td>TSN [99]</td>
<td>62.4</td>
<td>4.452215429</td>
<td>64</td>
<td>1.70203E-11</td>
</tr>
<tr>
<td>NTS-Net [91]</td>
<td>67</td>
<td>6.539622823</td>
<td>8</td>
<td>1.02348E-07</td>
</tr>
<tr>
<td>SSD-CNN (25 frames) [93]</td>
<td>79</td>
<td>2.282785822</td>
<td>10</td>
<td>2.07061E-05</td>
</tr>
<tr>
<td>CNN-LSTM (10 frames)</td>
<td>80.8</td>
<td>1.316561177</td>
<td>22</td>
<td>1.33367E-05</td>
</tr>
<tr>
<td>CNN-LSTM (30 frames)</td>
<td>84</td>
<td>2.263232693</td>
<td>23</td>
<td>0.042717721</td>
</tr>
<tr>
<td>SA-LSTM (10 frames)</td>
<td>83.1</td>
<td>2.378141198</td>
<td>18</td>
<td>0.022103532</td>
</tr>
<tr>
<td>SA-LSTM (30 frames)</td>
<td>86.2</td>
<td>2.22110833</td>
<td>21</td>
<td>-</td>
</tr>
</tbody>
</table>

on its training or testing. The fine-grained object classification in still image NTS-Net [91] achieved higher performance than video classification methods TSN [99] and C3D [70], since NTS-Net localized the discriminative object parts in frames. However, NTS-Net ran at 8fps, which was slower than TSN and C3D. Conversely, SSD-CNN [93] outperformed NTS-Net in both accuracy and speed. It localized the discriminative object in the sequence of frames and utilized the temporal information in video. SSD-CNN classified each video by taking the average score over multiple frames that did not share information features. The SA-LSTM network (30 frames) achieved the highest performance. It outperformed the CNN-LSTM that used LSTM to train an object’s features without including the object’s scale.

We use paired t-test to determine the accuracy of the SA-LSTM method compared to baseline methods. The last column on Table 5.1 presents p-values for 30 frames from the 10-fold cross-validation experiment. The SA-LSTM network (30 frames) performed significantly better than the baseline.
5.4 Analyzing Classification Performance

In Figure 5.3, we show the Precision-Recall plot for each method. The areas under the Precision-Recall curve were obtained by the SA-LSTM and baseline methods. The SA-LSTM network (30 frames) significantly outperforms all baseline methods across the CV100 dataset in terms of the Precision-Recall curve.

Fine-grained car classification is a challenging task, even when it is being done manually by a person. However, using just 10 or 30 frames, the SA-LSTM achieves remarkable classification accuracy. In this section, we will analyze the classification performance of the SA-LSTM and identify the classes the model had difficulty classifying. Figure 5.4
Figure 5.4: Sample classes that are mistakenly predicted as another class. shows some examples of misclassification our method encountered on the CV100 dataset; most of these were found to occur with vehicles of the same make (manufacturer). For example, this method classifies the 2017 Lexus RX as a 2019 Lexus ES-Hybrid, and it classifies the 2018 Audi S5 Coupe as a 2017 Audi S5 Cabriolet. Our model could not classify these classes correctly because the vehicle designs are so similar. We also found the SA-LSTM has difficulty classifying car models with similarly close designs but different manufacturers, such as the 2017 Audi TT Roadster and the 2017 Nissan GT-R.
5.5 Conclusion

In this paper, we introduced the CV100: CarVideo 100 dataset, that contains 100 categories of car models. CV100 contains 896 videos clips for a total of 495,778 frames. We used this dataset to evaluate the existing methods of video classification and fine-grained classification in still images and video. Additionally, we described how the SA-LSTM network used the object’s scale with CNN features to train and test the LSTM. The SA-LSTM network was employed to train those features and predict the video class. The SA-LSTM network (30 frames) outperformed other state-of-the-art video classification methods.
CHAPTER VI

CONCLUSION

Our objective for this thesis is to investigate a general and robust video fine-grained classification system to classify different sub-categories in challenging scenarios. In this thesis, we aim to answer this question: How can videos of sub-categories in challenging scenarios be robustly classified? To do that we divide this broad question into three sub-questions where each sub-question solves part of the main question. This chapter summarizes how we solve the three sub-questions, which are the main contributors of this thesis. Finally, we discuss future directions for our research.

6.1 Summary of Contributions

The three contributions made in this thesis are:

1. The CV100: CarVideo 100 dataset, a novel dataset for fine-grained object classification of vehicle make, model, and year. The first major contribution of this thesis is to create a large-scale benchmark for videos we can use to improve robustness of fine-grained classification in videos. Creating a fine-grained classification dataset is
difficult since it needs to create a label and bounding box for each sub-class and also needs to locate different parts of the object in the dataset. In this thesis we introduce the CV100: CarVideo 100 dataset, a novel dataset for fine-grained object classification of vehicle make, model, and year in which all frames were annotated with bounding boxes and attribute labels using the Amazon Mechanical Turk workforce. To collect videos for the CV100 dataset, we initially consider solely using YouTube, as the video-sharing site now has more than several billion videos available. However, the task of labeling 100 classes of cars just from YouTube videos would take far too much time, so we adjust our data collection method. We search for a website that includes the names of many car models, finding the TRUECar site. This site lists each car manufacturer and also provides a listing of the year, make, and model of each vehicle available. We use Amazon Mechanical Turk (MTurk) workers to annotate each frame on the dataset. We create a user interface for MTurk workers to provide location information and attribute annotations. We collect three types of annotations: (a) bounding box, (b) identification of car model, if the frame includes more than one car, and (c) attributes.

2. A novel approach to fine-grained object classification in videos that combines a Single Shot Multibox Detector (SSD) with a single stream multi-region Convolutional Neural Network (CNN). The second contribution is to propose a novel fine-grained video object classification method using a CNN for classification while introducing an SSD object detector as a pre-processing step. Using an object detector allows us to be much more precise when training and testing our network as we are able to ignore much of the non-essential information while still being able to capture
and train on the most valuable parts of the image. The CNN can take an RGB image. In addition, to capture multi-frame information, we compute an average prediction over a sequence of frames from the video. Objects or object parts bounding boxes are detected by the SSD and then cropped and resized for input to the CNN. Each bounding box is separately classified by the CNN. We consider data augmentation to be one of the important parts of training the SSD-CNN combination. Our data augmentation approach is adapted from [14]. After resizing each bounding box to $224 \times 224$, we apply four corner crops and a center crop. This allows the network to train on parts of cars and thus be robust to partially occluded cars. Our architecture uses a ResNet-152 for the spatial streams. Our SSD-CNN outperforms Temporal Segment Networks (TSN) and 3-Dimensional Convolutional Networks (3D ConvNets), which are state-of-the-art on human action recognition in videos. Also, it outperforms NTS-Net which is the state-of-the-art on fine-grained classification in the image domain.

3. A novel Scale-Aware LSTM Network for Video Fine-Grained Car Classification. The third major contribution of this thesis is to propose a novel Scale-Aware LSTM Network for video fine-grained car classification. The Scale-Aware Long Short-Term Memory (SA-LSTM) has been created to aggregate features of multiple frames with scale of objects in the sequence of frames. The SA-LSTM uses a Single Shot MultiBox Detector (SSD) to localize objects by establishing their scale (height and width) with bounding boxes from the sequence of frames in which they appear. The Convolutional Neural Network (CNN) extracts the features from these objects and object parts. Our technique combines object scales with CNN features that acknowledges how, in video, objects will present with large variations of size
and scale in a sequence of frames. This information is fed to the SA-LSTM, creating a fine-grained classification method that takes into account variations in size and scale that occur, not only in video, but that might, under certain circumstances, be an issue when classification involves still images. Our findings show that this novel SA-LSTM approach significantly outperforms existing video classification methods and also state-of-the-art fine-grained classification methods developed for still images, in terms of the accuracy of the dataset.

6.2 Future Work

In this thesis we covered multiple aspects of fine-grained classification but there are many aspects that can be explored in future work. This section summarizes the extension of this dissertation research.

1. We believe the SA-LSTM can be used for human action recognition and pedestrian classification in videos. In these two problem objects present with different scales and each scale will provide different features that may hurt the performance of the classification of these two problems. Our SA-LSTM network capable of learning which object scale that can help to improve action recognition by providing SA-LSTM by object scales and object features at the same time.

2. In the future we will do the comparison between our dataset that is CV100 and YouTube-Cars. We will compare the number of classes and annotations data. We will test YouTube-Cars on the SA-LSTM and test their method on CV100 dataset. We will draw the advantage and drawback of each dataset.
3. Chapter V shows a Scale-Aware Long Short-Term Memory (SA-LSTM) outperforms CNN-LSTM that does not use the object scale. In the future work, we will investigate frames weights that are given by LSTM. By analyzing frames weights, we will learn which frames have useful information for fine-grained classification and might help to improve the classification performance.

4. Also, chapter V shows the SA-LSTM misclassification car models that have similar designs such as the 2017 Lexus RX as a 2019 Lexus ES-Hybrid. We are planning to use the Attention Mechanism to localize the discriminative object parts from the sequence of frames. We believe cars have object parts that help to distinguish between different models such as the front bumper of a car.

5. Identifying the accurate landmarks within face images helped to improve the accuracy of face recognition task. We plan to extend chapter V to use car landmarks and investigate the differences between using car landmarks and without.
REFERENCES


