Spatio-Temporal Multimedia Big Data Analytics Using Deep Neural Networks

Samira Pouyanfar

Florida International University, spouy001@cs.fiu.edu

Follow this and additional works at: https://digitalcommons.fiu.edu/etd

Part of the Databases and Information Systems Commons, Numerical Analysis and Scientific Computing Commons, and the Other Computer Sciences Commons

Recommended Citation

https://digitalcommons.fiu.edu/etd/4265

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.
FLORIDA INTERNATIONAL UNIVERSITY
Miami, Florida

SPATIO-TEMPORAL MULTIMEDIA BIG DATA ANALYTICS USING DEEP NEURAL NETWORKS

A dissertation submitted in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
COMPUTER SCIENCE
by
Samira Pouyanfar

2019
To: Dean John L. Volakis  
   College of Engineering and Computing

This dissertation, written by Samira Pouyanfar, and entitled Spatio-Temporal Multimedia Big Data Analytics using Deep Neural Networks, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

__________________________________________
Xudong He

__________________________________________
Sitharama S. Iyengar

__________________________________________
Jainendra K Navlakha

__________________________________________
Keqi Zhang

__________________________________________
Shu-Ching Chen, Major Professor

Date of Defense: June 6, 2019

The dissertation of Samira Pouyanfar is approved.

__________________________________________
Dean John L. Volakis  
   College of Engineering and Computing

__________________________________________
Andres G. Gil  
   Vice President for Research and Economic Development  
   and Dean of the University Graduate School

Florida International University, 2019
DEDICATION

I dedicate this work to my parents, Maman Behjat and Baba Abbas, and my husband, Ramin, who offered unconditional love and support and have always been there for me. Thank you so much.
ACKNOWLEDGMENTS

This research is supported by the Dissertation Year Fellowship award from Florida International University’s Graduate School, NSF HRD-0833093, NSF HRD-1547798, NSF CNS-1126619, NSF CNS-1461926, DHSs VACCINE Center under Award Number 2009-ST-061-CI0001, Florida Public Hurricane Loss Model (FPHLM), Volkswagen Group of America Electronics Research Lab, and FIU NSF Center of Research Excellence in Science and Technology (CREST).

First of all, I would like to express my utmost gratitude to my advisor Professor Shu-Ching Chen, for his invaluable guidance, encouragement, patience, and support throughout so many years of research. In addition, I would also like to thank Professor Mei-Ling Shyu of the Department of Electrical and Computer Engineering at University of Miami (UM), professors Sitharama S. Iyengar, Jainendra K Navlakha, and Xudong He of the School of Computing and Information Sciences, and Professor Keqi Zhang of the Department of Environmental Studies and International Hurricane Research Center for the suggestions they provided. Secondly, my thanks go to the friends and colleagues from the Distributed Multimedia Information Systems (DMIS) Laboratory at FIU and the Data Mining, Database & Multimedia (DDM) Research Group at UM, in particular Yudong Tao, Haiman Tian, Maria Presa Reyes, Tianyi Wang, Hector Cen, Yilin Yan, Saad Sadiq, Hsin-Yu Ha, Yimin Yang, Yingxin Li, Diana Machado, Raul Garcia, Yuexuan Tu, and Daniel E. Martinez. I would also thank my dear friend Leila Zahedi. Last, but not least, I am extremely grateful for the deep love from my kind parents, sisters, and my supportive husband, Ramin. I would never have been able to finish my dissertation without their support and encouragement.
ABSTRACT OF THE DISSERTATION

SPATIO-TEMPORAL MULTIMEDIA BIG DATA ANALYTICS USING DEEP NEURAL NETWORKS

by

Samira Pouyanfar

Florida International University, 2019

Miami, Florida

Professor Shu-Ching Chen, Major Professor

With the proliferation of online services and mobile technologies, the world has stepped into a multimedia big data era, where new opportunities and challenges appear with the high diversity multimedia data together with the huge amount of social data. Nowadays, multimedia data consisting of audio, text, image, and video has grown tremendously. With such an increase in the amount of multimedia data, the main question raised is how one can analyze this high volume and variety of data in an efficient and effective way. A vast amount of research work has been done in the multimedia area, targeting different aspects of big data analytics, such as the capture, storage, indexing, mining, and retrieval of multimedia big data. However, there is insufficient research that provides a comprehensive framework for multimedia big data analytics and management.

To address the major challenges in this area, a new framework is proposed based on deep neural networks for multimedia semantic concept detection with a focus on spatio-temporal information analysis and rare event detection. The proposed framework is able to discover the pattern and knowledge of multimedia data using both static deep data representation and temporal semantics. Specifically, it is designed to handle data with skewed distributions. The proposed framework includes the following components: (1) a synthetic data generation component based on simula-
tion and adversarial networks for data augmentation and deep learning training, (2) an automatic sampling model to overcome the imbalanced data issue in multimedia data, (3) a deep representation learning model leveraging novel deep learning techniques to generate the most discriminative static features from multimedia data, (4) an automatic hyper-parameter learning component for faster training and convergence of the learning models, (5) a spatio-temporal deep learning model to analyze dynamic features from multimedia data, and finally (6) a multimodal deep learning fusion model to integrate different data modalities. The whole framework has been evaluated using various large-scale multimedia datasets that include the newly collected disaster-events video dataset and other public datasets.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background and Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Proposed Solutions</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Scope and Limitations</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Outline</td>
<td>9</td>
</tr>
<tr>
<td><strong>2. RELATED WORK</strong></td>
<td>10</td>
</tr>
<tr>
<td>2.1 Domain Adaptation and Randomization</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Imbalanced Data Classification</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Deep Learning</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Visual Data Processing</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2 Multimodal Learning</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Hyper-parameter Learning for Neural Networks</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Spatio-Temporal Data Analytics</td>
<td>21</td>
</tr>
<tr>
<td><strong>3. OVERVIEW OF THE PROPOSED FRAMEWORK</strong></td>
<td>23</td>
</tr>
<tr>
<td>3.1 Synthetic Data Generation</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Automatic Sampling for Imbalanced Data</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Deep Static Representation Learning</td>
<td>26</td>
</tr>
<tr>
<td>3.4 Automatic Hyper-parameter Learning</td>
<td>27</td>
</tr>
<tr>
<td>3.5 Deep Spatio-Temporal Learning</td>
<td>28</td>
</tr>
<tr>
<td>3.6 Multimodal Deep Learning Fusion</td>
<td>28</td>
</tr>
<tr>
<td><strong>4. SYNTHETIC DATA GENERATION</strong></td>
<td>30</td>
</tr>
<tr>
<td>4.1 Data Generation using Generative Adversarial Networks</td>
<td>30</td>
</tr>
<tr>
<td>4.1.1 Adversarial Data Augmentation</td>
<td>32</td>
</tr>
<tr>
<td>4.1.2 Experimental Analysis</td>
<td>34</td>
</tr>
<tr>
<td>4.2 Data Generation using Simulation</td>
<td>38</td>
</tr>
<tr>
<td>4.2.1 Domain Randomization for Bridging the Reality Gap</td>
<td>39</td>
</tr>
<tr>
<td>4.2.2 Data Preparation and Model Training</td>
<td>45</td>
</tr>
<tr>
<td>4.2.3 Experimental Analysis</td>
<td>47</td>
</tr>
<tr>
<td>4.3 Conclusion</td>
<td>53</td>
</tr>
<tr>
<td><strong>5. AUTOMATIC SAMPLING FOR IMBALANCED DATA CLASSIFICATION</strong></td>
<td>56</td>
</tr>
<tr>
<td>5.1 Spatio-Temporal Synthetic Oversampling</td>
<td>56</td>
</tr>
<tr>
<td>5.1.1 Experimental Analysis</td>
<td>59</td>
</tr>
<tr>
<td>5.2 Dynamic Sampling</td>
<td>63</td>
</tr>
<tr>
<td>5.2.1 Real-time Data Augmentation</td>
<td>64</td>
</tr>
<tr>
<td>5.2.2 Dynamic Sampling in CNNs</td>
<td>65</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2.3 Experimental Analysis</td>
<td>66</td>
</tr>
<tr>
<td>5.3 Conclusion</td>
<td>70</td>
</tr>
<tr>
<td>6. DEEP REPRESENTATION LEARNING</td>
<td>71</td>
</tr>
<tr>
<td>6.1 Ensemble Deep Learning</td>
<td>71</td>
</tr>
<tr>
<td>6.1.1 Preprocessing</td>
<td>73</td>
</tr>
<tr>
<td>6.1.2 Deep Feature Extraction</td>
<td>73</td>
</tr>
<tr>
<td>6.1.3 Classification</td>
<td>76</td>
</tr>
<tr>
<td>6.1.4 Experimental Analysis</td>
<td>79</td>
</tr>
<tr>
<td>6.2 Efficient Deep Residual-Inception Network</td>
<td>90</td>
</tr>
<tr>
<td>6.2.1 Residual Module</td>
<td>91</td>
</tr>
<tr>
<td>6.2.2 Inception Module</td>
<td>92</td>
</tr>
<tr>
<td>6.2.3 Network Architecture</td>
<td>93</td>
</tr>
<tr>
<td>6.2.4 Experimental Analysis</td>
<td>96</td>
</tr>
<tr>
<td>6.3 Conclusion</td>
<td>98</td>
</tr>
<tr>
<td>7. AUTOMATIC HYPER-PARAMETER LEARNING</td>
<td>101</td>
</tr>
<tr>
<td>7.1 Trend-based Learning Rate Annealing</td>
<td>101</td>
</tr>
<tr>
<td>7.1.1 Stochastic Gradient Descent</td>
<td>102</td>
</tr>
<tr>
<td>7.1.2 Non-Parametric Trend Analysis</td>
<td>104</td>
</tr>
<tr>
<td>7.1.3 Online Learning Rate Schedule</td>
<td>106</td>
</tr>
<tr>
<td>7.2 Experimental Analysis</td>
<td>108</td>
</tr>
<tr>
<td>7.3 Conclusion</td>
<td>114</td>
</tr>
<tr>
<td>8. DEEP SPATIO-TEMPORAL LEARNING</td>
<td>116</td>
</tr>
<tr>
<td>8.1 Spatio-Temporal Representation Learning</td>
<td>117</td>
</tr>
<tr>
<td>8.1.1 Transfer Learning with Deep CNNs</td>
<td>117</td>
</tr>
<tr>
<td>8.1.2 LSTM</td>
<td>117</td>
</tr>
<tr>
<td>8.1.3 Bidirectional LSTM</td>
<td>118</td>
</tr>
<tr>
<td>8.1.4 CNN-Residual Bidirectional LSTM</td>
<td>119</td>
</tr>
<tr>
<td>8.1.5 Experimental Analysis</td>
<td>122</td>
</tr>
<tr>
<td>8.2 An Integrated Spatio-Temporal Sampling and Deep Learning Model</td>
<td>125</td>
</tr>
<tr>
<td>8.2.1 Experimental Analysis</td>
<td>128</td>
</tr>
<tr>
<td>8.3 Conclusion</td>
<td>128</td>
</tr>
<tr>
<td>9. MULTIMODAL DEEP REPRESENTATION LEARNING</td>
<td>132</td>
</tr>
<tr>
<td>9.1 Static Feature Extraction Module</td>
<td>133</td>
</tr>
<tr>
<td>9.1.1 Visual Feature Extraction</td>
<td>133</td>
</tr>
<tr>
<td>9.1.2 Audio Feature Extraction</td>
<td>136</td>
</tr>
<tr>
<td>9.1.3 Textual Feature Extraction</td>
<td>136</td>
</tr>
<tr>
<td>9.2 Temporal Feature Extraction Module</td>
<td>137</td>
</tr>
<tr>
<td>9.3 Fusion Module</td>
<td>138</td>
</tr>
<tr>
<td>9.4 Experimental Analysis</td>
<td>139</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>9.5 Conclusion</td>
<td>144</td>
</tr>
<tr>
<td>10. CONCLUSIONS AND FUTURE WORK</td>
<td>146</td>
</tr>
<tr>
<td>10.1 Conclusions</td>
<td>146</td>
</tr>
<tr>
<td>10.2 Future Work</td>
<td>148</td>
</tr>
<tr>
<td>10.2.1 Automatic Hyper-Parameter Learning</td>
<td>148</td>
</tr>
<tr>
<td>10.2.2 Integrated Synthetic Data Generation</td>
<td>150</td>
</tr>
<tr>
<td>10.2.3 Other Future Work</td>
<td>151</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>156</td>
</tr>
<tr>
<td>VITA</td>
<td>178</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 The size of the collected flood image datasets</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Recall scores on the flood dataset separated by style</td>
<td>37</td>
</tr>
<tr>
<td>4.3 Comparison results between different style transfer techniques and the baseline</td>
<td>38</td>
</tr>
<tr>
<td>4.4 Comparison results on simulation (Domain 2)</td>
<td>52</td>
</tr>
<tr>
<td>4.5 Comparison results on real world (Domain 3)</td>
<td>53</td>
</tr>
<tr>
<td>5.1 Performance evaluation results of the proposed spatio-temporal synthetic oversampling algorithm on disaster dataset</td>
<td>61</td>
</tr>
<tr>
<td>5.2 Performance evaluation results of the proposed spatio-temporal synthetic oversampling algorithm on imbalanced UCF101</td>
<td>61</td>
</tr>
<tr>
<td>5.3 Performance evaluation of the proposed dynamic sampling</td>
<td>69</td>
</tr>
<tr>
<td>6.1 Pre-trained reference models for feature extraction</td>
<td>76</td>
</tr>
<tr>
<td>6.2 Average performance of various feature sets and classifiers on the disaster dataset</td>
<td>83</td>
</tr>
<tr>
<td>6.3 Statistic summary of selected concepts in TRECVID</td>
<td>87</td>
</tr>
<tr>
<td>6.4 Performance evaluation for different concepts on the TRECVID dataset</td>
<td>88</td>
</tr>
<tr>
<td>6.5 Deep Residual-Inception architecture</td>
<td>94</td>
</tr>
<tr>
<td>6.6 Performance evaluation for different concepts on the disaster dataset</td>
<td>99</td>
</tr>
<tr>
<td>7.1 Accuracy comparison on CIFAR-10 for different iterations and two different values for the significance factor ($\varphi_0$)</td>
<td>113</td>
</tr>
<tr>
<td>8.1 Classification results on UCF101. The proposed CNN-ResBiLSTM compared with baselines and state-of-the-art models.</td>
<td>124</td>
</tr>
<tr>
<td>8.2 Performance evaluation results on UCF101 using different techniques based on our experimental setup.</td>
<td>125</td>
</tr>
<tr>
<td>8.3 Performance evaluation results on Disaster dataset.</td>
<td>130</td>
</tr>
<tr>
<td>8.4 Performance evaluation results on imbalanced UCF101.</td>
<td>130</td>
</tr>
<tr>
<td>9.1 The statistical information of the disaster dataset</td>
<td>140</td>
</tr>
<tr>
<td>9.2 Performance evaluation results on the disaster dataset</td>
<td>143</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>2.1</td>
<td>The network top-5 errors (%) and layers in the ImageNet classification over the time</td>
</tr>
<tr>
<td>3.1</td>
<td>The proposed framework for multimedia big data analytics</td>
</tr>
<tr>
<td>4.1</td>
<td>Samples of (a) noisy and (b) normal flooding images</td>
</tr>
<tr>
<td>4.2</td>
<td>The proposed adversarial data augmentation model</td>
</tr>
<tr>
<td>4.3</td>
<td>Cycle-GAN style transfer samples on the flood dataset</td>
</tr>
<tr>
<td>4.4</td>
<td>Correctly classified samples by the proposed framework for each style category on the flood dataset</td>
</tr>
<tr>
<td>4.5</td>
<td>Transferring the knowledge to the real world (center) from a photorealistic simulation (left) vs a primitive simulation (right)</td>
</tr>
<tr>
<td>4.6</td>
<td>The proposed model for collision-free autonomous driving based on domain randomization</td>
</tr>
<tr>
<td>4.7</td>
<td>Samples of three domains including Domain 1 or primitive simulation (a), Domain 2 or photorealistic simulation (b) and Domain 3 or the real world (c)</td>
</tr>
<tr>
<td>4.8</td>
<td>Examples of various randomizations applied to Domain 1</td>
</tr>
<tr>
<td>4.9</td>
<td>Impact of adding individual randomization to the model on autonomy</td>
</tr>
<tr>
<td>4.10</td>
<td>Examples of results on parking and Kitti datasets</td>
</tr>
<tr>
<td>5.1</td>
<td>The proposed dynamic sampling model</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of F1-scores for each concept in the network camera dataset</td>
</tr>
<tr>
<td>6.1</td>
<td>The proposed ensemble deep learning model</td>
</tr>
<tr>
<td>6.2</td>
<td>Sample keyframes for each concept in the disaster dataset</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance evaluation for different concepts on the disaster dataset</td>
</tr>
<tr>
<td>6.4</td>
<td>Sample keyframes for each concept in TRECVID dataset</td>
</tr>
<tr>
<td>6.5</td>
<td>Performance evaluation on the TRECVID dataset</td>
</tr>
<tr>
<td>6.6</td>
<td>A comparison of True Positive value on the TRECVID dataset</td>
</tr>
<tr>
<td>6.7</td>
<td>A residual building block</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Background and Introduction

The explosive growth and widespread accessibility of digital data have led to a surge of research activity in the big data, and data sciences fields. The conventional approaches for data management have achieved limited success as they are incapable of handling the huge amount of complex data with high volume, high velocity, and high variety [6, 7, 8, 9, 10].

In the last few years, the fast and widespread use of multimedia data, including image, audio, video, and text, as well as the ease of access and availability of multimedia sources, have resulted in a big data revolution in multimedia management systems [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Multimedia data analytics addresses the issue of manipulating, managing, mining, understanding, and visualizing different types of data in effective and efficient manners to solve real-world challenges. The solutions include but are not limited to text analysis, image/video processing, computer vision, audio/speech processing, and database management for a variety of applications such as healthcare, education, entertainment, and mobile devices [27, 28, 29, 30, 31, 32, 33, 34, 35].

For decades, machine learning and data mining researchers have tried to discover the patterns and data representations from the raw data [36, 37]. The field of machine learning is witnessing its golden era as deep learning slowly becomes the leader in this domain. Deep learning uses multiple layers to represent the abstractions of data to build computational models. Some key enabler deep learning algorithms such as Generative Adversarial Networks (GANs) [38], Convolutional Neural Networks (CNNs) [39, 40], and model transfers have completely changed our
perception of information processing. With the acute development in deep learning and its research venues being in the top limelight, deep learning has gained extraordinary momentum in speech, language, and visual detection systems. However, several domains are practically still untouched by Deep Neural Networks (DNNs) either due to their challenging nature or the lack of data availability for the general public. This creates significant opportunities and fertile ground for rewarding future research avenues.

The scope of this dissertation is to provide a systematic and comprehensive framework for multimedia big data analytics using deep neural networks. This study aims to address some of the main challenges in this area and provide solutions to manage and analyze multimedia big data effectively. There exist many challenges to analyze such data which can be summarized as follows:

- The success of many machine learning and deep learning algorithms heavily depends on the existence of clean, large-scale, and annotated datasets. Collecting such datasets is challenging and time-consuming. This issue is even more daunting in real-world applications such as disaster information management [41, 42, 43, 44] or autonomous driving cars [45] in which the dataset should represent different locations, noise, lighting, etc., as well as rare scenarios (e.g., accidents, bad weather, unusual movements). Automatic data generation using simulators and generative models is an efficient solution to synthetically generate large-scale datasets with sufficient varieties. However, bridging the gap between real-world and synthetic data is still an open question.

- Another main challenge faced by the multimedia community is the non-uniform distribution of real-world datasets [46]. This problem is known as “data imbalance problem”, in which some of the classes contain much fewer samples
than the others. Examples of the imbalanced data problem include rare disease identification, fraud detection, and natural disaster recognition. It has been widely shown in the literature that techniques such as data resampling (oversampling and under-sampling) can enhance the prediction results of rare classes, especially for the binary classification tasks (e.g., cancer detection). However, it is challenging to employ such techniques on a multi-class or multi-label imbalanced data problem while maintaining the temporal information on the multimedia data.

- Discovering the patterns and discriminative features from raw data has been a challenging task in machine learning area. This method is called representation learning. Unlike conventional machine learning and data mining techniques, deep learning is able to generate very high-level data representations from massive volumes of raw data. Therefore, it has provided a solution to many real-world applications. However, deep learning usually requires large-scale datasets with manually labeled data. The most relevant future machine learning problems will not have sufficient training samples with labels [47]. Current deep learning models will also need to adapt to the rising issues such as data sparsity, missing data, and messy data in order to capture the approximated information through observations.

- Many real-world problems are characterized as time series (e.g., human activity recognition, stock prediction, and sentiment analysis), and it is critical to discover the temporal patterns in a time series problem [48, 49]. For example, video data consisting of sequences of image frames can be considered as a time series problem in which both static and motion information need to be extracted and analyzed [50, 51]. However, most existing multimedia classification techniques either ignore temporal information or utilize very complex
engineering features for spatio-temporal data analysis \[52, 53\] to model the temporal features which are not very efficient in practice.

- Multimedia data usually contains various types of modalities such as image, audio, and text. For example, a video may contain sequences of frames, audio clips, as well as text descriptions and meta data. These data modalities are usually complementary, which can be integrated to enhance the final decisions. However, many existing studies only focus on one or two data modalities due to the complexity and difficulty of multimodal data collection, analysis, and fusion \[54\]. In addition, despite the great success of deep learning models on processing of single data modalities, there are a few studies to propose a comprehensive deep learning framework for multimodal data analysis.

1.2 Proposed Solutions

To address these challenges, this dissertation presents a new multimedia big data framework that effectively handles the multi-class multi-label data imbalance problems using DNNs. The proposed solutions include:

- **Synthetic data generation:** To address the big data collection and generation issues, in this dissertation, novel synthetic data generation techniques are proposed that are later integrated to our proposed deep learning models. In particular, deep adversarial networks and simulators are leveraged to generate synthetic data for deep learning training automatically. These models are specifically applied to real-world applications such as disaster information management and autonomous driving. This study extends the idea of domain adaptation and randomization to bridge the reality gap between simulation and the real world. Also, it utilizes generative models to transfer various styles
to regular data, which increases the generalization capability of deep learning models while reducing the need to have large-scale annotated datasets.

- **Automatic sampling for imbalanced data:** To address the data imbalance problem, in this study, an automatic sampling method is proposed to be integrated with CNNs. First, an early spatio-temporal oversampling is presented specifically for video data, that contains spatial and temporal information. The proposed spatio-temporal oversampling model utilizes random data augmentation techniques to generate new synthetic data for minority classes in the imbalanced dataset. Second, the dynamic sampling model is integrated with existing CNNs to automatically modifies the samples of classes in each training iteration. This model is proposed to further enhances the classification performance especially for those classes with minor or complex samples.

- **Deep static representation learning:** Successful machine learning models generally rely on rich data representations since various hidden characteristics and patterns can be derived from original data. Thus, in this dissertation, efforts have been devoted to generate the most discriminative data representation using unsupervised transfer learning. As a single model may not be able to handle large datasets with multiple feature sources, an integrated model is utilized to enhance the data representation by taking advantages of multiple pre-trained deep learning models. This study proposes a mixture of deep learning feature extractors integrated with an enhanced ensemble algorithm. Also, a new deep learning structure is designed based on Inception and Residual modules to efficiently extract significant and robust features from the raw data.
• **Deep spatio-temporal learning:** Upon the proposed spatio-temporal synthetic sampling schema and multimedia deep representation learning, a new deep learning structure is proposed to classify data with spatio-temporal information. In particular, deep Recurrent Neural Networks (RNNs) with bidirectional and residual connections are integrated to capture past and future temporal information from time-series data which remembers the information for a longer period of time compared to the conventional machine learning algorithms. Specifically, the model is employed for a large-scale video classification task to discover the dynamics and motions using DNNs without utilizing complex engineering features.

• **Multimodal deep learning fusion:** DNNs have been successfully applied for single modality feature extraction and classification. To further extend the proposed framework, we propose a novel deep learning model to extract unsupervised features from multiple modalities (e.g., image, text, audio) and train a new fusion model to combine different data modalities in an effective manner. Particularly, a multi-label multimodal deep learning model is proposed to integrate the deep spatio-temporal features obtained for each data modality. This model will efficiently discover the correlation between different modalities and the final classes in the dataset.

### 1.3 Contributions

This dissertation has several major contributions as listed below:

• An automatic sampling is proposed for imbalanced data classification. This method includes spatio-temporal synthetic oversampling and dynamic sampling. Both methods are designed to overcome the bias in deep neural net-
works especially in datasets with skewed distribution. The first method is used as a preprocessing step to balance the dataset and generate synthetic video samples for minority classes. While there might be still some complex classes in the dataset which can be distinguished using the dynamic sampling combined with CNNs.

- A new hyper-parameter learning is proposed for DNNs namely “Trend-based Learning Rate Annealing (T-LRA)”. T-LRA is a drop-based learning rate scheduling that improves the SGD algorithm in deep learning. It reduces the task of selecting an appropriate learning rate using the statistical trends of the training process. Specifically, T-LRA automatically determines how and when to modify the learning rate based on the previous losses during the training. In particular, it is applied to the CNNs. To the best of our knowledge, this is the first algorithm that schedules the learning rate using statistical trend analysis.

- A new deep spatio-temporal model is proposed namely “CNN-ResBiLSTM” for large-scale multimedia classification. This model integrates the pre-trained CNN deep representations with a new temporal structure. The temporal component contains two-layers of Residual Bidirectional Long Short Term Memory (LSTM) to capture dynamics, motion, and temporal information from video datasets. This component does not require any handcrafted features for motion analysis and automatically combines space and temporal features from the video.

- A novel multimodal multi-label fusion which considers the correlations between different data modalities and final classes is proposed. It bridges the gap between low-level data representations and high-level abstractions. This
multimodal data fusion model is proposed to enhance the final classification performance and reduce the complexity of multiple modalities.

- Two new applications are used in this dissertation to evaluate the proposed framework. Specifically, autonomous driving in simulation is used to assess the proposed synthetic data generation and domain randomization techniques. This is the first time domain randomization is used for the application of “deep driving” which can avoid obstacles. Disaster information management is another new application used in this study to evaluate the whole multimedia deep learning framework.

1.4 Scope and Limitations

The proposed framework has still several limitations and assumptions as follows:

- Without loss of generality, the proposed framework is evaluated on video and image datasets. However, proposed models such as automatic sampling, hyper-parameter learning, spatio-temporal deep learning, and multimodal fusion can be extended to cover other data types.

- The hyper-parameter learning model concentrates on learning rate scheduling as one of the main hyper-parameters in DNNs. However, there are other parameters such as momentum, kernel size, number of layers, and number of epochs that can be automated in the future.

- Although the unsupervised deep representation is used in this dissertation, the proposed framework mainly focused on supervised learning and multimedia classification tasks. Some of the proposed techniques can be easily extended for unsupervised learning, however, it is out of the scope of this dissertation.
• The proposed framework alleviates the big data challenges in multimedia data using efficient algorithms and techniques which speed up the training and testing process. However, the distributed processing and other big data solutions (such as GPU programming, Hadoop, Spark) are not taken into considerations in this framework.

1.5 Outline

This dissertation is organized as follows. Chapter 2 reviews the literature in the area of multimedia big data analytics including domain adaptation, imbalanced data classification, multimodal deep learning, and spatio-temporal data analytics. The proposed framework and its main components are presented in chapter 3. In chapter 4, two new techniques are proposed for synthetic data generation using simulators and adversarial networks. Chapter 5 discusses the proposed sampling techniques including spatio-temporal synthetic oversampling and dynamic sampling. In chapter 6, several techniques are discussed for multimedia static deep representation learning including ensemble deep learning and deep Residual-Inception network. Chapter 7 presents a new algorithm for automatic hyper-parameter learning in deep learning. Chapter 8 describes the proposed spatio-temporal deep learning model and its components. In chapter 9, a new multi-label multimodal deep learning fusion is presented. Finally in chapter 10, the conclusions and several future directions are given.
Chapter 2

Related Work

In this chapter, the literature in the area of domain adaptation and randomization, imbalanced data classification, visual and multimodal deep learning, and spatio-temporal data analytics are presented.

2.1 Domain Adaptation and Randomization

Deep learning achievements heavily depend on the existence of large-scale datasets [55, 56]. Collecting such datasets in real-world is expensive and laborious. In particular, for different real-world applications, it is challenging to collect large-scale data for a diverse set of scenarios (e.g., day and night, various lighting conditions, users movements, etc.).

An alternative technique to collect a large amount of data is data augmentation with label-preservation that is commonly used in various computer vision applications [57, 58]. The goal of data augmentation is to enhance generalization and overcome the overfitting problem [55]. However, data augmentation cannot generate a high variability in the environment, since it is usually limited to image brightness, translation, color, and cropping rather than generating completely new scenes and scenarios.

In recent years, GANs have shown promising results in domain adaptation and deep learning generalization [59, 60, 61]. In addition, recent advances in deep learning enable the transformation of styles from one domain (source) to another domain (target) [59, 62, 63, 64, 65, 66]. In [62], separation and recombination of content with neural representations were used to transfer the style. A generalized framework that combines untied weight sharing, discriminative modeling, and a GAN loss was proposed for visual style transformation [63]. Also, an unsupervised pixel-level
domain adaptation method without the need of source and target domain pairs was developed to learn the style transformation [64]. Hoffman et al. [59] proposed a model based on cycle-GAN that is able to capture both pixel-level and image-level domain shifts. Shrivastava et al. [65] developed a method to improve the performance of unsupervised adversarial domain adaptation by combining techniques such as local adversarial loss, periodically updated discriminator using refined image history and self-regulation term. The Multi-Style Generative Network (MSGNet) [66] uses a CoMatch Layer approach that learns to match the lower order statistics of content image with the style images. Long et al. [67] extends the conditional GANs by studying the cross-covariance dependency between the domain-specific feature representations and classifier predictions. Nevertheless, GAN models still need real-world data for training which is usually expensive.

Another practical approach for synthetic data generation and style transfer is utilizing simulators. Collecting data from game engines and generating synthetic data for deep learning training has attracted significant attention in recent years. It is used for car and pedestrian detection [68, 69, 70] as well as robotic grasping and motion control [71, 72, 73]. Synthetic data is also utilized in different tasks such as optical flow and geometric problems [74, 75]. Flying Chairs dataset, for instance, is a popular example of synthetic data generated for optical flow learning with CNNs [76]. However, few deep learning models are generated based on only synthetic data due to the reality gap challenge. In other words, the networks that are purely trained on synthetic data may not generalize well to the real world. Conventional approaches usually use synthetic data to train the network and then fine tune it on the real-world data [77]. Moreover, some studies used photo-realistic synthetic images to bridge the reality gap [69, 75]. However, generating photo-realistic images is expensive and often requires laborious manual designing.
Compared to the aforementioned approaches, DR is a relatively new topic among deep learning training methods. DR is originally used in various robotics applications to transfer deep learning from simulation to the real world \cite{73, 78}. CAD$^2$RL \cite{73} is one of the first applications of DR that flies a quadcopter through indoor environments using reinforcement learning. Although CAD$^2$RL is based on only simulation data, it is still leveraging realistic scenes (e.g., chairs, doors, etc.) in the simulation which represent the real world quite well. Tobin \textit{et al.} \cite{78} proposed DR for object localization and robot manipulation. Using data from simulator and non-realistic textures, they are able to train a deep learning model for object detection that is accurate to 1.5cm in the real world. Similarly, Bousmalis \textit{et al.} \cite{72} leveraged both domain adaptation and randomization to transfer simulation to the real world for robotic grasping systems. In that work, a GAN is employed to make synthetic images more realistic. In a recent work by Tremblay \textit{et al.} \cite{68}, it is shown how DR can be used for object (cars) detection and it also illustrates the benefit of fine-tuning deep neural networks on real data after training on simulated data.

2.2 Imbalanced Data Classification

A critical challenge in multimedia data is how to process data with skewed distributions or in other words, the imbalanced datasets. This can be seen commonly in real world multimedia applications where the classes are not distributed uniformly \cite{80, 81, 82, 83, 84}. There are usually two classes: the major classes (or called the negative classes) and the minor one (or called the positive class), where we are more interested in detecting the minor class. For instance, in medical lab results, cancer instances are rare but more important than those instances for regular diseases. Other applications of imbalanced data are fraud activities detection,
bomb detection, failure predictions of technical equipment, etc. In such conditions, conventional machine learning and data mining algorithms often fail to detect the minor class, and they are biased toward the negative classes, which may have serious effects. Suppose an instance of a medical lab result is predicted as non-cancer (a negative class), while in reality the patient has the cancer. This error is called false negative, which can cause very serious harm.

Regarding the data imbalance issue, conventional approaches can be mainly categorized into the following groups: Sampling methods, cost sensitivities, learning, and hybrid algorithms. Typically, sampling methods modify the data distribution in order to balance the dataset and improve the classification results. There are two main re-sampling approaches in the literature: over-sampling the minority (positive) class or under-sampling the majority (negative) class. In other words, the techniques in this group either decrease the frequency of the majority class (under-sampling) or increase the frequency of the minority class (oversampling). Either way can be used in any machine learning algorithm as a preprocessing phase. Although these techniques can address the data imbalance problem, they may discard potentially important information or increase the likelihood of overfitting. More advanced techniques such as Synthetic Minority Over-sampling Technique are proposed to avoid overfitting and information loss.

The solutions of the latter group are algorithmic techniques in which the classifiers are designed to naturally handle the imbalanced datasets. For example, Cost Sensitive Learning (CSL) modifies the learning process by incorporating the misclassification costs of the different classes. Currently, CSL has been applied in various learning algorithms such as decision trees, AdaBoost, and Naive Bayes. Also, ensemble techniques such as bagging and boosting can improve the performance of classification and overcome the overfitting problem. Recently,
various hybrid methods have been proposed, which combine the traditional solutions for data imbalance subject \[98\].

Existing work on imbalanced data classification is mainly limited to binary classification since multi-class imbalanced data classification has more complicated relations between its classes. An intuitive strategy to handle multi-class imbalanced data is to apply decomposition methods to turn the problem into a set of binary classification problems \[99\]. However, this method needs careful combination strategies to reconstruct the original multi-class dataset.

In the deep learning literature, the challenges of imbalanced data classification have not been thoroughly investigated. Few recent studies have focused on this problem by generating synthetic data \[100\] or changing the loss function to improve the detection performance of minority classes \[101\]. To the best of our knowledge, there is no framework for multi-class and multi-label multimodal imbalanced data classification using deep neural networks.

### 2.3 Deep Learning

Deep learning techniques have become the main parts of various state-of-the-art multimedia systems and computer vision \[102\]. More specifically, CNNs have shown significant results in different real-world tasks, including image processing, object detection, video processing, etc. This section discusses more details about the most recent deep learning models and algorithms proposed over the past few years for multimedia data processing.
2.3.1 Visual Data Processing

In 1998, LeCun et al. presented the first version of LeNet-5. LeNet-5 is a conventional CNN that includes two convolutional layers along with a subsampling layer and finally ended with a full connection in the last layer. Although, since the early 2000s, LeNet-5 and other CNN techniques were greatly leveraged in different problems, including the segmentation, detection, and classification of images, they were almost forsaken by the data mining and machine learning research groups. More than one decade later, CNN algorithm has started its prosperity in computer vision communities. Specifically, AlexNet is considered the first CNN model that substantially improved the image classification results on a very large dataset (e.g., ImageNet). It was the winner of the ILSVRC 2012 and improved the best results from the previous years by almost 10% regarding the top 5 test error. To improve the efficiency and the speed of training, a GPU implementation of the CNN is utilized in this network. Data augmentation and dropout techniques are also used to substantially reduce the overfitting problem.

Since then, a variety of CNN methods have been developed and submitted to the ILSVRC competition. Among them, ZFNet demonstrates its supremacy and could achieve the lowest top 5 error 11.7 in the ILSVRC 2013. Although ZFNet is very similar to AlexNet and can be considered a fine tuning of that network, it still provides some key ideas and modifications. In 2014, two influential but different models are presented which mostly focused on the depth of neural networks. The first one, known as VGGNet, includes a very simple 19 layer CNN. In this network, at each layer, the spatial size of the input is reduced, while the depth of the network is increased to achieve a more effective and efficient model. Although VGGNet was not the winner of the ILSVRC 2014, it still shows a significant improvement (7.3% top 5 error) over the previous top models which came from its two major
specifications: the simplicity and depth. In contrast to VGGNet, GoogleNet [2], the winner of this competition (6.7% error), proposed a new complex module named “Inception”, allowing several operations (pooling, convolutional, etc.) to work in parallel. This network is basically inspired by the Network in Network model [4], which provides dimensionality reduction using micron neural network (1 × 1 convolutions).

Microsoft deep residual network (known as ResNet) [1] took the lead in the 2015 competitions including ILSVRC 2015 and COCO detection and segmentation tasks by introducing the residual connections in CNNs and designing an ultra deep learning model (50-152 layers). This model achieved an incredible performance (3.6% top 5 error) which means, for the first time, a computer model could beat human brains (with 5-10% error) in image classification. On the contrary of the extremely deep representation of ResNet, it can handle the vanishing gradients [106] as well as the degradation problem (saturated accuracy) in deep networks by utilizing residual blocks.

In the last few years, several variations of ResNet have been proposed. The first group of methods has tried to increase the number of layers more and more. Current CNN models may include more than 1000 layers [107]. Finally, in 2017, ResNeXT [108] is proposed as an extension of ResNet and VGGNet. This simple model includes several branches in a residual block, each performing a transformation which is finally aggregated by a summation operation. This general model can be further reshaped by other techniques such as AlexNet. ResNeXT outperforms its original version (ResNet) using half of the layers and also improves the Inception-v3 as well as Inception-ResNet networks on the ImageNet dataset. Figure 2.1 demonstrates the revolution of depth and performance in image classification (e.g., ImageNet) over the time. The problem of supervised image classification is

On the other hand, video analytics has attracted considerable attention in the computer vision community and is considered as a challenging task since it includes both spatial and temporal information. In an early work, large scale YouTube videos containing 487 sport classes are used to train a CNN model [109]. The model includes a multi-resolution architecture which utilizes the local motion information in videos and includes context stream (for low-resolution image modeling) and fovea-stream (for high-resolution image processing) modules to classify videos. An event detection from sport videos using deep learning is presented in [110]. In that work, both spatial and temporal information are encoded using CNNs and feature fusion via regularized Autoencoders.

3-Dimensional CNN (C3D) [111] has demonstrated a better performance on video analysis tasks over the traditional 2D CNNs. It automatically learns spatiotemporal
features from video inputs and models the appearance and motions at the same time. Two-stream networks \cite{112} are another set of video analysis techniques that model spatial (RGB frame) and temporal information (optical flow) separately and average the predictions in the last few layers of the network. This network is extended in a recent work called Inflated 3D ConvNet (I3D), utilizing the idea of C3D. It is also pre-trained on Kinetics dataset \cite{113}.

Deep Long Short Term Memory (LSTM) networks have been widely utilized in different applications such as NLP, speech processing, and time-series that require long-term temporal information. Specifically, it is used for video classification tasks in recent few years \cite{114, 115}. Deep residual networks (ResNet) \cite{1} were originally proposed by Microsoft Research (MSR) for an image competition task (ILSVRC 2015). This idea was later applied to many different applications and also video classification tasks \cite{116}.

### 2.3.2 Multimodal Learning

Multimodal contents provide a vast amount of data from social media websites such as Facebook and YouTube in a daily manner. Multimedia data is not restricted to a single modality and usually contains multimodal data, such as textual, visual and acoustic \cite{117}. An intelligent multimodal analysis framework is proposed in \cite{118} to extract information and aggregate the semantic information efficiently. Multimedia data represents features from different media sources. In \cite{119}, the fusion techniques such as PCA and ICA that are popularly applied to multimedia data are discussed. Feature extraction and feature fusion are two crucial improvements for multimodal data analysis. Multimodal fusion methods, especially bimodal, have been proposed in numerous research studies, but optimal solutions remain elusive. A multimodal
analysis is frequently applied to facial expression and face recognition tasks [120]. Several fusion models [121, 122, 123] target the facial expressions by fusing the speech models with the visual models. Specifically, in [123], bi-directional Long Short-Term Memory (LSTM) networks are utilized to improve the classification performance compared to the traditional Hidden Markov Model (HMM) and Support Vector Machine (SVM) classification frameworks. There has been a surge of progress in the field of deep multimodal representation learning in the past several years. Most of the models present improvement in bi-modal learning with specific tasks, such as facial expression, emotion recognition [124]. EmoNets [124] explored multiple combination methods to fuse modalities into one classifier by using SVM and won the EmotiW challenge in 2013. In [125], it is shown that finding the relationship between the image data and the audio data in the early stage is not straightforward since there is no direct connection between the raw pixels and audio waveforms or spectrograms. Thus, a cross-modality feature learning is proposed for speech recognition which generates the shared representation features utilizing the videos of the lip movements. Finally, in [126], Gibbs sampling is applied to generate the fused representation for bi-modal feature learning.

Despite the fast growth of deep learning and its applications (e.g., NLP, computer vision, and speech processing), current research in multimedia big data analysis using deep learning is still in its initial stage. Multimodal deep learning techniques are needed to analyze different modalities of data [124]. Moreover, handling high-dimensional, heterogeneous, and unlabeled multimedia data is a great potential for the future deep learning research. However, computational efficiency still remains a big challenge in multimedia deep learning since deep learning requires a great amount of resources and (usually) more training time in comparison to the traditional machine learning approaches.
2.3.3 Hyper-parameter Learning for Neural Networks

Artificial neural networks have been widely used in a wide range of applications. Specifically in recent years, it has been extended to deep learning, which has shown its strength in dealing with real-world problems [55]. Till now, several algorithms have been proposed to train neural networks and minimize the loss functions, including Stochastic Gradient Descent (SGD) [127], AdaDelta [128], AdaGrad [129], Adam [130], to name a few. In a gradient descent process, selecting the appropriate parameters such as the learning rate (step size) is crucial for a better and faster learning. It also needs expert knowledge and differs for each problem. LeCun et al. [131] proposed an online estimation of principal eigenvalues and eigenvectors of the loss function’s Hessian or second derivative matrix. Generally, the optimal learning rate can be selected as the inverse of the largest eigenvalue of the Hessian matrix. However, since computing the Hessian matrix for large learning algorithms (e.g., backpropagation) with thousands of parameters is computationally expensive, an online version of this algorithm was proposed by LeCun et al. In another work [132], the learning rate is changed in each epoch based on the weights and gradient values of the previous epoch in order to minimize the loss function. This method is inspired by the learning rate adaptation proposed in [133], which derives two-point step sizes approximation to the secant equation.

AdaGrad [129] has shown promising results on the large learning tasks. This method utilizes the first order information but relies on some second order features and annealing. In this method, small gradients have large learning rates and vice versa. However, it is very sensitive to the initial conditions and network hyperparameters. AdaDelta [128], an extension of AdaGrad, is a dynamic method adapting the learning rate over time using only the first order information. This approach overcomes some of the AdaGrad problems such as continual decay of the learning
rate during the training process and the need for manually selecting the global learning rate. Adam \cite{130} is another algorithm for the first-order optimization of gradient descent. It combines the advantage of AdaGrad and RMSProp \cite{134}, an optimization for online and non-stationary environments. Currently, Adam has been applied in popular deep learning architectures \cite{1} and has shown its effectiveness.

### 2.4 Spatio-Temporal Data Analytics

Video classification is challenging due to its multimodality and spatio-temporal nature \cite{25}. Traditional methods combined several modality representations to enhance the classification performance. Chen et al. \cite{135} proposed a multimodal data mining framework for semantic event detection from sports videos. Despite the great capability of the framework, it still needs human efforts for temporal analysis and also uses handcrafted features. In computer vision, several techniques have been proposed to detect motion and temporal information from videos. Among them, optical flow \cite{52} and iDT \cite{53} are able to generate discriminative motion features from the data. However, using engineering techniques for temporal analysis is a computationally expensive task.

Deep learning has been applied greatly in recent years to overcome the challenges of traditional methods and generate general-purpose models for feature analysis, either static or temporal \cite{136,137}. Spatio-temporal deep learning techniques can be divided into two groups: 1) Those generating separate models for each modality and fusing the information in the final layers \cite{112}, and 2) Those designing a comprehensive model to handle spatio-temporal information and their connections in one single model \cite{111}. The 3D convolutional neural networks (called C3D) \cite{111} fall under the second category that inherently applies both pooling and convolutional layers in
the 3D space. In that work, the third dimension is time. This network requires very large-scale datasets to converge and very powerful and parallel machines including GPUs with high memory to train the deep 3D networks.

LSTM was originally proposed in 1997 [138] which is a variant of RNNs. Deep LSTM networks have been widely utilized in different applications such as NLP, speech processing, and time-series that require long-term temporal information. Specifically, it is used for video classification tasks in recent few years [114, 115]. Deep residual networks (ResNet) [1] were originally proposed by Microsoft Research (MSR) for an image competition task (ILSVRC 2015). This idea was later applied to many different applications and also video classification tasks [116].

All the aforementioned methods employ complex and computationally intensive handcrafted features such as optical flow [52] or iDT [53] for video classification and usually fuse several models to capture the spatio-temporal information. Moreover, these techniques usually ignore the imbalanced distribution of real-world data and are only evaluated on very balanced datasets.
CHAPTER 3
OVERVIEW OF THE PROPOSED FRAMEWORK

Multimedia semantic concept detection is an emerging research area in recent years. Many multimedia search engines often use textual descriptions and metadata to retrieve data such as image or video. However, due to the limitation and subjectivity of multimedia metadata, such engines may not provide accurate results. Thus, automatic concept detection is crucial in multimedia analysis. Therefore, the goal of this work is to address existing challenges in multimedia big data analysis and support multimedia semantic representation.

Currently, many research studies have been done in multimedia data management. However, very few research provides a comprehensive framework to address the multimedia challenges such as imbalanced data problem, large-scale data preparation, multimodal data representation learning, and spatio-temporal information extraction. For this purpose, in this dissertation, an integrated framework is proposed for multimedia big data analytics especially multimedia semantic concept detection. The whole framework is shown in Figure [3.1] which consists of five major components: synthetic data generation, automatic sampling for imbalanced data classification, deep static representation learning using transfer learning, deep spatio-temporal learning, and multimodal deep learning fusion. These components are coherently integrated to address the challenges in multimedia big data and support different functionalities in this area. In this framework, synthetic data are automatically generated using generative models and simulators to overcome lack of real-world data. A novel sampling technique is proposed to handle imbalanced data problem. Also, DNNs are utilized as the main learning algorithm due to their great capability in multimedia data analytics. Specifically, an ensemble of deep representation learning together with an efficient Residual-Inception model are utilized.
for static feature analysis of visual data. This framework also utilizes an automatic parameter learning to further enhance the efficiency of model training. A deep spatio-temporal model is proposed to handle both static and temporal features in multimedia data, then different modalities (e.g., audio, image, etc.) are combined using a novel fusion model.

3.1 Synthetic Data Generation

Existing deep learning models require very large-scale datasets with enough variety that represent various scenarios and conditions in the real world. However, these kinds of datasets may not be available for many multimedia applications. An efficient solution to address this challenge is automatically generating synthetic data for training deep learning models. This dissertation presents two novel synthetic data generation methods for two challenging real-world applications:

- The first method addresses the problem of flood event detection from images with real-world conditions [139]. This work proposes a new image classification model based on adversarial data generation and augmentation to overcome the unavailability of real-world data in rare scenarios.

- Another method based on simulation data is proposed for the application of autonomous driving and obstacle avoidance [140]. This dissertation proposes new domain and scenario randomization techniques to reduce the gap between simulation and the real world.

The experimental results of both flood event detection and autonomous driving models demonstrate the effectiveness of the proposed models compared to the conventional methods. These technique increases the generalization in deep learning,
Figure 3.1: The proposed framework for multimedia big data analytics
improves the model performance, and reduces the need to have a large-scale annotated dataset. Both models can be extended for other applications and domains.

3.2 Automatic Sampling for Imbalanced Data

Current deep learning techniques do not automatically consider the data imbalance problem. As mentioned earlier, the existing machine learning techniques that handle the imbalanced datasets can be divided into sampling and algorithmic methods. In this dissertation, we proposed new methods to overcome the datasets with non-uniform distributions in multi-class classification. This method mainly contains two steps:

- We propose a new spatio-temporal synthetic oversampling to automatically resample the data using both spatial and temporal information.
- A dynamic sampling method is presented which modifies the data samples of each class during the training using the evaluation score for that class in the reference set.

3.3 Deep Static Representation Learning

Finding the best attributes or features which represent each data modality and discover the knowledge and relationship between them, is an essential phase of data mining. Thus, we propose a new multimedia representation learning using transfer learning and pre-trained deep learning models. This component contains two main parts:

- First, multimedia representation learning using an ensemble deep learning which handles the imbalanced data problem and overfitting. Inspired
by the great success of transfer learning and deep learning, the most discrimi-
native deep features are extracted from various pre-trained models and then a
new ensemble technique based on Support Vector Machine (SVM) is designed
to enhance the semantic event detection in imbalanced datasets.

• We further enhance this model by proposing a new deep learning model based
on traditional CNNs integrated into a two levels of Residual-Inception com-
bination [146]. The proposed model is able to automatically detect semantic
events from multimedia data.

In summary, based on the experiments on different multimedia datasets, the pro-
posed deep static representation learning model has shown its superiority and effec-
tiveness while maintaining low computational costs in multimedia semantic event
detection.

3.4 Automatic Hyper-parameter Learning

In this dissertation, we conduct an algorithm to automatically adjust the hyper-
parameters (especially learning rate) in deep neural networks [147]. In particular,
an automatic drop-based learning rate scheduling is proposed to improve the SGD
algorithm in deep learning. This work alleviates the task of selecting an appropriate
learning rate by analyzing the statistical trends of the training process in an online
manner. It automatically decides when to drop the learning rate based on the
losses in the previous training iterations. Specifically, this algorithm is applied on
CNNs and send the feedback to the learning model in each training iteration. It
not only improves the performance but also significantly speeds up the training and
convergence processes.
3.5 Deep Spatio-Temporal Learning

Retrieving valuable information from large-scale multimedia data is yet another big challenge. In recent years, video classification has attracted significant attention in the multimedia and deep learning community. It is one of the most challenging tasks since both visual and temporal information should be processed effectively. Existing techniques either disregard temporal information between video sequences or generate very complex and computationally expensive models to integrate the spatio-temporal data. In this dissertation, we propose an effective deep learning model for imbalanced video classification by utilizing both spatial and temporal information [141]. The model includes a series of residual bidirectional LSTM to capture temporal knowledge in video datasets. Experimental results on two imbalanced video datasets demonstrate the superiority of the proposed spatio-temporal model compared to the state-of-the-art approaches.

3.6 Multimodal Deep Learning Fusion

When the volume of multimedia data increases exponentially, so do the complication and connection between the data. As multimedia data contains various media types, extracting multi-modal discriminative information from the data instances is imperative. In general, multi-modal data can be categorized into visual (e.g., image and video), audio, and textual modalities. It is critical how to effectively integrate the information from different data modalities to better manage multimedia systems. In this dissertation, we address this challenge by proposing a new multimodal deep learning model [148] [149] [150] [151]. The main purpose of this model is to bridge the gap between multimedia data low-level characteristics and its high-level semantic content. This model combines the information generated from previous models
(multimedia static representation learning and deep spatio-temporal learning) using a new fusion model which considers the correlations between data modalities and final classes. The results on a collected disaster-events video dataset demonstrate the effectiveness of both visual model and fusion model compared to the baseline approaches.
Deep learning achievements heavily depend on the existence of clean, annotated, and large-scale datasets \cite{55,56}. Collecting such datasets is expensive and laborious. Moreover, many existing visual data classification techniques use datasets that usually include high-resolution images without considering real-world noise. However, in many real-world applications, it is challenging to collect large-scale clean data for a diverse set of scenarios and conditions (e.g., day and night, various lighting or weather conditions, users movements, etc.) that would allow the systems to work robustly. Therefore, in this dissertation, new techniques are proposed for synthetic data generation for deep learning training. In particular, two challenging applications are used to evaluate the proposed solutions. The first application is a flood event detection using GANs while the second one is an autonomous driving and obstacle avoidance in simulation using domain randomization techniques.

### 4.1 Data Generation using Generative Adversarial Networks

Although machine learning and deep learning have achieved substantial progresses in image classification, there are only very few methods that leverage deep learning for real-world disaster detection and management \cite{152}. This is mainly due to the limited annotated data available in this domain. Existing work usually collects the data from Web/social media and annotates them manually. Nevertheless, the variability of images in such datasets may not be sufficient to create a robust model which can be used in different real-world situations. For example, many flood images in social media were taken during the day, most users posted clear images without significant noise, etc. The model trained on such data cannot easily detect a specific
disaster from the real-world noisy images (e.g., blurry or night images). Figure 4.1(a) shows several noisy flood images that cannot be detected by a deep learning model trained on normal flood images (Figure 4.1(b)).

In this study, three sets of image styles including “night”, “blurry”, and “rainy” is used. Since collecting and annotating these sets of images are difficult and tedious, recent photorealistic style transfer techniques are utilized to transfer images between two different domains (normal to style) in an unsupervised manner. Specifically, a new data augmentation method based on Cycle-Consistent Generative Adversarial Networks (CycleGANs) is proposed. For each set of styles, a CycleGAN is trained to transfer the images from regular flood to the styled flood (e.g., to night-flood, rainy-flood, and blurry-flood.). These images are later utilized in the data augmentation step to train a Convolutional Neural Network (CNN). To the best of our knowledge, this is the first work that applies style transfer to flood event detection. In addition, this is the first flood detection framework that can detect unusual flood images without seeing such irregular images in the training set.
4.1.1 Adversarial Data Augmentation

The proposed method aims to train a CNN model for image classification which is robust to various contexts (styles), denoted as $Y_1, Y_2, \ldots, Y_N$. For the application of flood event detection, the training images are classified into two categories: flood and non-flood. Therefore, a training dataset $\mathcal{I} = \{I^0_1, I^0_2, \ldots, I^0_M\}$, containing both flood-related and non-flood images, is collected to train the model. Figure 4.2 depicts the proposed adversarial data augmentation method in which CycleGAN is utilized as data augmentation to enhance the CNN classifier for flood event detection.

Data augmentation is a common way to enhance the training dataset and improve the performance of the CNN models and its generalization capability. The conventional approaches performing data augmentation include flipping, scaling, cropping, rotation, etc., which manipulate the pixel values in a simple manner. However, the patterns of the images can significantly change in various contexts.
with a complicated transformation. In the case of flood event detection, the images show different visual characteristics in day or night, in rainy or sunny weather, and when the camera is moving or not. GAN has shown powerful performance to learn the patterns/styles of contexts regardless of the objects in the images, and thus in this work, we propose a novel approach of data augmentation by utilizing Cycle-GAN \cite{Xu15} to perform carefully curated style transfer for flood in different contexts.

We first define the most common context as the regular context $X$. All the original images in the training dataset are from $X$. Then, for each stylized target context $Y_i$, a CycleGAN model is trained to translate a given flood image from $X$ to $Y_i$ without any paired image samples. The goal is to learn a set of functions $G_i : X \rightarrow Y_i, \forall i$ so that the learned transformation of images after applying $G_i(X)$ are indistinguishable from the style references $Y_i$ by using an adversarial loss. The adversarial loss ($L_{Adv}$) is applied to the mapping functions $G_i(X)$ as follows.

$$L_{Adv}(G_i, D_{Y_i}, X, Y_i) = \mathbb{E}_{y_i \sim p(y_i)} [\log D_{Y_i}(y_i)] + \mathbb{E}_{x \sim p(x)} [\log(1 - D_{Y_i}(G_i(x)))]$$  \hspace{1cm} (4.1)

where $G_i$ generates images $G_i(x)$ or $\hat{y}_i$, and $D_{Y_i}$ discriminates the training sample $G_i(x)$ from the real target $y_i$. A similar loss is applied to the inverse mapping $F_i : Y_i \rightarrow X$ and its discriminator $D_X$. Since these adversarial mapping functions are under-determined and prone to overfitting, further reduction of mapping functions is achieved through cycle-consistency, i.e., $x \rightarrow G_i(x) \rightarrow F_i(G_i(x)) \approx x$. Similarly, another inverse cycle-consistency is introduced that learns the transformation back $y_i \rightarrow F_i(y_i) \rightarrow G_i(F_i(y_i)) \approx y_i$. This is achieved by using a cycle consistency loss, defined as:

$$L_{cyc}(G_i, F_i) = \mathbb{E}_{x \sim p(x)} [||F_i(G_i(x)) - x||_1] + \mathbb{E}_{y_i \sim p(y_i)} [||G_i(F_i(y_i)) - y_i||_1]$$  \hspace{1cm} (4.2)

The aforementioned generative model is trained with images of regular floods as well as other stylized contexts. At the end, the reconstructed images $F_i(G_i(x))$
closely match the input images $x$. These stylized images $G_i(x)$ are then used in the training of the CNN flood detection model.

Given all the CycleGAN models, each training image $I_i$ can be transferred into $N$ types of contexts. The synthetic images can be represented by $I^1_i, I^2_i, \ldots, I^N_i$, respectively. In each training epoch, the switch randomly selects one of the transferred or original contexts for each image in the training dataset. Then, it feeds the selected images into the CNN model and updates the model parameters accordingly. In other words, in each epoch, a proxy dataset $\mathcal{I}_k = \{I^{n_{1,k}}_1, I^{n_{2,k}}_2, \ldots, I^{n_{M,k}}_M\}$ is generated to train the CNN model, where $k$ is the epoch number and $n_{i,k} \in \{0, 1, 2, \ldots, N\}$ is the selected context of image $I_i$, randomly generated by the uniform distribution.

For the image classification, ResNet50 is applied, where the last layer is replaced by a fully connected layer with sigmoid activation. After the CNN model is trained, the test images are directly used to compute the prediction results, without using any CycleGAN model to transfer the style.

### 4.1.2 Experimental Analysis

**Datasets.** We collected flood-related and non-flood images from YouTube and Twitter with the corresponding keywords and tags. First, we used the keyword “Harvey” which was a major hurricane that occurred in the United States in 2017 with a severe inland flooding to search flood-related videos on YouTube. Meanwhile, Twitter is leveraged to collect flood-related visual data via Twitter API [154]. Both images and videos are collected from the tweets with hashtags “flooding” and “flood”. The training set included all the data collected from YouTube, however it randomly selected 30% of Twitter’s data. The remaining data from Twitter are used as the testing set. All the images are manually labeled as flood and non-flood
Table 4.1: The size of the collected flood image datasets

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Test</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Flood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1364</td>
<td>1866</td>
<td></td>
</tr>
<tr>
<td>Flood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>5072</td>
<td>8645</td>
<td></td>
</tr>
<tr>
<td>Regular</td>
<td>-</td>
<td>3627</td>
<td></td>
</tr>
<tr>
<td>Night</td>
<td>-</td>
<td>294</td>
<td>7000</td>
</tr>
<tr>
<td>Rainy</td>
<td>-</td>
<td>799</td>
<td>7000</td>
</tr>
<tr>
<td>Blurry</td>
<td>-</td>
<td>434</td>
<td>7000</td>
</tr>
</tbody>
</table>

for training and evaluation purposes, while the flood-related images are tagged as “night”, “rainy”, and “blurry” for evaluation purposes only, i.e., the proposed model is blind to these tags. Each image, if applicable, can have more than one tag. Meanwhile, we also collected style images from Google Images to train CycleGAN models with the corresponding keywords. The number of images of each context is shown in Table 4.1.

**Computing Environment.** An NVIDIA Tesla P100 GPU with 16GB of GPU device memory is used to deploy the proposed model, including ResNet50 and all the CycleGAN models in the experiment.

**Hyperparameters.** The ResNet50 model [1] pre-trained on ImageNet [155] is used as the image classifier. Adam solver [130] with learning rate=1e-3 and decay=1e-6 is applied to train the image classifier for 100 epochs. We implement CycleGAN to transfer the image style and train the models with Adam solver with learning rate=2e-4 for 150 epochs.

Figure 4.3 shows several samples generated by Cycle-GAN style transfer model on our dataset. The first row of images includes original images from the collected dataset, while the subsequent rows demonstrate the synthetic images generated from each Cycle-GAN for blurry, rainy, and night contexts, respectively. Although some of the generated images are not realistic (e.g., blurry), it can still help the model to generalize well and detect real-world flood images.
Figure 4.3: Cycle-GAN style transfer samples on the flood dataset
Table 4.2: Recall scores on the flood dataset separated by style

<table>
<thead>
<tr>
<th>Method</th>
<th>night</th>
<th>rainy</th>
<th>blurry</th>
<th>avg. noisy</th>
<th>flood (total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.785</td>
<td>0.795</td>
<td>0.791</td>
<td>0.790</td>
<td>0.898</td>
</tr>
<tr>
<td>Proposed work</td>
<td>0.831</td>
<td>0.927</td>
<td>0.873</td>
<td>0.877</td>
<td>0.936</td>
</tr>
</tbody>
</table>

To demonstrate the effectiveness of the proposed model in transferring the style for each context (night, rainy, and blurry), its performance (Recall) is compared with the original CNN without style transfer (please refer to Table 4.2). Recall or true positive rate is selected to show the number of correctly classified images for each context. It can be seen from the table that the proposed work significantly enhances the performance in all categories (the average recall in three categories is increased by more than 8% and the total flood recall reaches 0.94). In other words, the proposed model is able to accurately detect noisy and abnormal flooding samples compared to the conventional CNNs.

Now the question is why Cycle-GAN is utilized in this work rather than other style transfer techniques. To answer this question, we compare the proposed work with two other relevant style transferred methods, namely MSGNet [66] and neural style transfer by Gatys [62]. The MSGNet uses a CoMatch Layer approach that learns to match the lower order statistics of content image with the style images. While the Gatys uses separation and recombination of content with neural representations to transfer the style. Table 4.3 shows the comparison results between these three style transfer techniques and the CNN model. As can be inferred from the table, CNN has the highest precision compared to other techniques, meaning it can detect non-flood images better than other methods. However, its recall value is the lowest among the others. On the contrary, both style transfer methods can achieve very high recall but significantly lower precision. Conclusively, the proposed work
Table 4.3: Comparison results between different style transfer techniques and the baseline

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.916</td>
<td>0.898</td>
<td>0.907</td>
<td>0.855</td>
</tr>
<tr>
<td>MSGNet [66]</td>
<td>0.855</td>
<td>0.970</td>
<td>0.909</td>
<td>0.847</td>
</tr>
<tr>
<td>Style transfer [62]</td>
<td>0.853</td>
<td>0.978</td>
<td>0.911</td>
<td>0.849</td>
</tr>
<tr>
<td>Proposed work</td>
<td>0.896</td>
<td>0.936</td>
<td>0.916</td>
<td>0.864</td>
</tr>
</tbody>
</table>

beats all the benchmarks regarding the F1 score (the weighted average of precision and recall) and accuracy.

Finally, Figure 4.4 depicts several noisy flood samples that are correctly classified by our model, whereas the regular CNN cannot detect any of them. These results are evident that the proposed adversarial augmentation model can significantly improve the existing disaster management systems.

### 4.2 Data Generation using Simulation

Although GANs have shown promising results in domain adaptation and deep learning generalization, they still need real-world data for training which is usually expensive. In particular, for some applications such as autonomous driving, the dataset should represent different locations, obstacles, movements, lighting, etc. Also, it is sometimes impossible to collect data for rare scenarios (e.g., accidents, bad weather, and unusual driver behavior). To alleviate this problem, simulators can be used to quickly generate a huge amount of synthetic data. Until now, simulators have been widely used in various computer vision and autonomous driving applications [72, 76, 77, 156, 157, 158]. However, the question is how to effectively make the network trained on synthetic data operate on real-world data, in other
words, how to bridge the reality gap.

4.2.1 Domain Randomization for Bridging the Reality Gap

The existing solutions to overcome the reality gap include generating photo-realistic worlds [158], Generative Adversarial Networks (GANs) for image-to-image translation [153], and Domain Randomization (DR) [78]. The latter is the most inexpensive, yet effective technique recently proposed by researchers to manage this challenge, mostly in the field of robotics [71, 73]. DR aims to expose the network to simulation’s data with a wide range of variability (e.g., lighting, texture, objects, etc.) during the training to address the reality gap. This simple technique is able to re-
Figure 4.5: Transferring the knowledge to the real world (center) from a photo-realistic simulation (left) vs a primitive simulation (right)

duce (or eliminate) the need for large-scale real-world data since it forces the model to generate the representation invariant to the appearance of the object and environment [73]. In other words, the models trained on a wide variety of object meshes and scenes can generalize to the realistic scenes that may be completely different from the renderings generated for training [71]. DR has been recently investigated in very few specialized object localization and detection tasks [68, 78, 159].

This work explores the DR potential for the application of autonomous driving which can be later extended for other applications. More specifically, we investigate whether networks trained using non-realistic simulation data can be used for collision-free driving in photo-realistic simulators and to generate collision-free driving paths in the real world. This is the first application of DR for collision-free autonomous driving using an end-to-end deep neural network. Different from some existing work on DR, the collected synthetic images are not photo-realistic and do not need to reflect real complex objects such as cars, pedestrians, and traffic signs. In other words, the goal is to transfer the knowledge from a primitive simulator with simple randomization techniques to a complex world instead of using expensive photo-realistic simulation as shown in Figure 4.5. This technique significantly reduces the time and cost of collecting realistic synthetic data.

---

1The left photo is taken from http://sonify.psych.gatech.edu/research/driving/index.html

40
Moreover, we extended DR to generate dynamic randomized scenarios during the training. More specifically, a wide range of random scenarios (events) is generated each representing a completely new world. Each world contains a new terrain, texture, road, light, shadow, and multiple objects with different sizes, randomly moving in various directions. We call this approach “Scenario Randomization” (SR) which not only includes randomization for static objects and their texture but also it randomizes the dynamic of objects (e.g., moving direction of obstacles). After training the network on those primitive simulation worlds, it is tested on the existing realistic simulation worlds for autonomous driving as well as two real-world image datasets including a self-collected parking-lot dataset and Kitti. To be able to evaluate the network in the real world without really driving a car, the future driving path of the car is also predicted. The network receives a single image and predicts the few next steering angles that are later translated to an estimated path. The extensive experiments on simulators’ data demonstrate the effectiveness of DR in training deep neural networks for collision-free autonomous driving in simulation and also show interesting performance on real-world images.

The contributions of this method include: (1) Applying DR to a new and complex application (collision-free deep driving); (2) Extending the idea of DR to SR with leveraging dynamic objects with random movements in addition to the static domain randomization; (3) Demonstrating that obstacle avoidance can be learned with simple geometric shapes rather than expensive photo-realistic objects; (4) Conducting comprehensive experiments to show the importance of various randomization factors in making deep driving work in the simulation that also reveals interesting results in the real-world.

The proposed model is shown in Figure 4.6. As can be seen from the figure, we generated a very simple simulation world consisting of primitive object shapes and a
Figure 4.6: The proposed model for collision-free autonomous driving based on domain randomization road using the Unity 3D game engine\(^2\). Thereafter, we apply various SR techniques on this world and collect both images and the corresponding steering angles by driving the car in simulation. These data are later used to train the neural network and finally predict the future steering angles path on a realistic image.

In this work, we proposed three different domains (worlds) as shown in Figure 4.7. The first domain is designed for the training purpose while the other two are used only for testing the model. We need these three domains to see how the model trained on Domain 1 can drive on more realistic simulated worlds (Domain 2) and how it predicts the path in the real-world images (Domain 3).

**Domain 1:** The first domain is designed using a simulator (i.e., Unity game engine). It contains a simple road and some basic primitives that the ego-car tries to avoid as shown in Figure 4.7a. This domain does not include real texture (e.g., roads with lane marking) or any real objects such as vehicles, pedestrians, tree, and bridge. In Domain 1, we apply various DR and SR techniques and use the collected data from this domain for model training. To collect the data, human users control the ego-car in this domain to keep the car on the road while avoiding the obstacles.

\(^2\)https://unity.com
**Domain 2:** The next domain is also designed in the simulator but it includes more photo-realistic elements similar to the real world. In Figure 4.7b, two samples of Domain 2 are shown that include realistic objects such as trees, a lake, cars (static and dynamic), real road texture, etc. This domain is only utilized for the validation phase.

**Domain 3:** Finally, the last domain includes real-world images/video of outdoor environments (e.g. highways, urban, and parking lots). Two image samples of this domain are shown in Figure 4.7c. It must be noted that Domain 3 is only used for final testing of the deep driving network and never used during training.

It is worth mentioning that Domains 1 and 2 are both designed based on the Lake Track scene of the Udacity’s self-driving car simulator. For Domain 1, we only used the basic road track and removed all the realistic components. We also modified the road size, shapes, and curves.

This work aims to add a variety of randomization to the original simulated domain (Domain 1) which helps the model generalize to more realistic domains (Domain 2 and Domain 3). In fact, the model is trained to see the new domain merely as one more randomized flavor of the original domain. The randomization factors include:

- **Terrain:** To cover various surroundings in the real world, we utilize several unrealistic terrains (grounds) with various textures selected from a small set of terrain textures (20 textures) which can help the model to drive in different real-world environments (e.g., mountains, jungle, parking, highway, etc.)

- **Road texture:** for each scenario, a new road texture is randomly selected from a set of road textures. This texture can be unrealistic and does not include any lane marking or other realistic road texture. However, its variation

---

3 [https://github.com/udacity/self-driving-car-sim](https://github.com/udacity/self-driving-car-sim)
Figure 4.7: Samples of three domains including Domain 1 or primitive simulation (a), Domain 2 or photo-realistic simulation (b) and Domain 3 or the real world (c) can help the model to generalize to real-world unseen textures. In particular, we use 20 unrealistic road textures for training and 10 realistic textures for testing.

- **Novel objects (types, size, color, texture):** the 3D objects in Domain 1 include three simple primitives (cube, cylinder, sphere). In each scenario, the program selects a random number (between 10 to 40) of objects from a list of random object types with random scale, color, and texture and place them in random positions \((x, y, z)\) on the road using 3D waypoints in Unity. Different from existing work [68], we use simple pattern-based textures instead of using realistic textures from large-scale datasets.

- **Light:** in each scenario, a random intensity of light (between 0 to 2) is selected which represents different time (day, night) and conditions of the real world.
• **Shadow**: for shadow randomization, a special terrain with various heights is designed. This terrain generates shadows in different parts of the road and helps the model to learn object’s and the environment’s shadows.

• **Dynamic objects**: one of the major novelties of this work is handling both static and dynamic objects with various movement trajectories. For this purpose, a random path (right or left lane) with a random velocity (0 for static objects, negative numbers for reverse driving, and positive numbers for normal driving) is assigned to each object. Thus, the object can automatically follow the waypoints in the assigned lane.

Through extensive research and experiments, it is shown that the aforementioned factors play important roles in generalizing the model to the real worlds. Examples of different randomizations are shown in Figure 4.8.

### 4.2.2 Data Preparation and Model Training

In this work, the goal is to steer the car on a road while avoiding the static and dynamic obstacles. Therefore, the speed of the car is fixed using the auto-cruise component in the simulator and only the steering angle for each frame is collected. For training, the video frames from three cameras, placed in left, center, and right positions of the ego-car, together with the corresponding steering angle applied by the user are collected from Domain 1 with different domain and scenario randomization. Specifically, ten frames and steering angles are selected per second from the simulation video at 40 frames per second (fps) rate. As mentioned in [160], images from left and right cameras are required to train the agent on how to recover from non-optimal positions, which is essential because of the cascading errors that occur due to behavior cloning based imitation learning. Specifically, this method helps
Figure 4.8: Examples of various randomizations applied to Domain 1

(a) Samples of terrain randomization

(b) Samples of road texture randomization

(c) Samples of object color/textured/scale randomization

(d) Samples of light intensity randomization

(e) Samples of shadow randomization

the model to avoid drifting off the road by augmenting with images that are shifted laterally relative to the longitudinal axis of the car.

After collecting the data from the simulator, it is necessary to handle data outliers and smooth the steering angles since human are not always able to drive smooth trajectories. Outliers are replaced by mean and the steering angle curve is smoothed with a moving average with a window size of 20 based on our empirical study.

In this work, rather than generating steering angles as traditionally done \[160\], we chose to generate a path to be able to test our model in the real world. To do so, we collect the current and $N - 1$ future steering angles. Thus, the network
can predict the path while driving on the road. The generated path can help us to evaluate the model on real-world images and videos in an open-loop manner without driving a real car. After the preprocessing step, the steering angles of the left and right cameras are set by shifting the center cameras’ steering angle by a factor of $\gamma$ (to avoid driving off the road). In this work, the correction factor for the next $N$ consecutive steering angles is calculated as:

$$\alpha_{L(R)} = \alpha_C \pm (\gamma \ast (N - i^2)/N)$$

(4.3)

where $\alpha$ refers to the steering angle, $L$, $C$, and $R$ refer to the left, center, and right cameras, respectively, and $i \in \{0...N - 1\}$ ($i = 0$ refers to the first steering angle).

The neural network architecture of the proposed end-to-end deep driving model is shown in Figure 6.9. It takes a single image which goes through several convolutional layers followed by four dense (fully-connected) layers. The last dense layer generates ten outputs which correspond to the $N$ steering angles. In other words, given a single image, this regression model is able to predict the next $N$ steering angles in an end-to-end manner. The model outputs are later converted to the path in order to visualize the performance of the model in the real world data.

### 4.2.3 Experimental Analysis

**Datasets.** Multiple datasets are used for training and evaluation, with a focus on covering diverse simulated and real-world scenarios. The details of each dataset is described below:

- Simulation dataset: As described before, our training data is collected from Domain 1 (refer to Figure 4.7a). For each scenario, a new combination of all randomization factors are automatically generated. To collect the images
and the corresponding steering angles, we had test subjects drive the car in the simulated world akin to playing a computer game (each played between 10-30 minutes). In total, the combined DR dataset contains around 200k images. We also collected 30K-100K images for each flavor of randomization (e.g. road texture, object, terrain, light, and shadow). For the baseline model (“No Rand”), we used Domain 1 data without applying any randomization technique for training, while keeping the total number of images fixed. For evaluation, we utilized two different photo-realistic simulation worlds (please refer to Domain 2 in Figure 4.7b). The first version is the lake track from Udacity in which several obstacles (cars, objects, etc.) are added on the road and the second simulation world designed by us includes moving cars, and also photo-realistic road/terrain textures.

- Parking dataset: A dataset is collected by driving a car around our corporate campus in California. This dataset is useful in evaluating the behavior of our model in a complex real-world environment. Presence of a large number of stationary parked vehicles is an additional benefit of this dataset for evaluation of our obstacle avoidance model.

- Kitti: Finally, we downloaded several sets of Kitti raw data\(^4\) including city, residential, and road categories to further evaluate our model on real-world images.

**Experimental Settings.** In this work, the steering angle correction is empirically set to +0.25 and −0.25 for the left and right shifts, respectively. \(N\) is set to 10 consecutive steering angles. The image size is set to 160*320px. The car speed is fixed to 30mph during both data collocation and testing. For preprocessing, the

moving average window is set to 20. The sequence of steering angles and the camera
calibration are used to place a 2D path on the image for visualization purposes. This
projection assumes a flat ground plane for the road surface.

The deep learning model includes preprocessing layers to normalize the image
(mean centered) and crop the top parts of the image (remove the sky and only focus
on the road and obstacles). Similarly, the real-world images from parking and Kitti
datasets are preprocessed to follow the same format of the simulation data.

For deep learning training, the following settings are used: Number of epochs=5,
Optimizer=Adam, learning rate=0.0005, batch size=32, Loss function= Mean Squared
Error (MSE), activation functions=RELU, dropout=20%. The images collected
from 2/3 of the road from Domain 1 is used for the training and the remaining is
used for the validation. In all the experiments, the same deep learning model is used
for training and all the training images are obtained from Domain 1 without using
any real images.

We evaluate the performance of the deep learning model using Nvidia’s autonomy
metric \[160\] for simulation. This metric counts the number of human interventions
to retake the control of the car. In our case, there are two types of errors while
the model is driving the car: (1) collisions with an object ($\# Collisions$), (2) events
where the car ends up outside of the road boundary ($\# Off - roads$). We assign
the same penalty as \[160\] (6 seconds) when an error happens. Thus, autonomy is
calculated as:

\[
autonomy = (1 - \frac{\# Collisions + \# Off - roads}{\text{total time}}} \times 6 \times 100)
\]

When a collision or off-road event happens, we programmatically count the num-
ber of errors and reset the car’s location to the next waypoint on the road. This
reduces human intervention while testing the model.
Ablation Study for Simulation Environment

The goal of this experiment is to see how the model trained on Domain 1 can drive in a photo-realistic simulation environment. The first set of experiments is executed in a new simulation environment (Domain 2) which is never seen during the training. To test the impact of each randomization factor, we trained a model for every single randomization and compared them with no randomization (No Rand) and our model (DR), which is trained on all the randomizations together. Specifically, a fixed amount of image data (30K) are collected for each randomization model (e.g., terrain, road texture, light, object scale/color/texture, and shadow). After training, each model is tested on four different scenarios as follows: (1) Our designed Domain 2 including 7 static cars, (2) Our designed Domain 2 including 8 dynamic cars moving in various directions, (3) Our designed Domain 2 including 6 static cars in the middle of the road and 16 cars on the side (simulating a narrow parking space) (4) Domain 2 from Udacity (lake track) including 4 static cars and 2 cubes. In total, each model is tested for 20 minutes on these four fixed scenarios. The result of this experiment is shown in Table 4.4. This table shows total number of collisions, off-roads, and autonomy. As can be seen from the table, with adding terrain randomization (R1), the number of collisions and driving off the road decreases. With road texture randomization (R2) the model confuses objects with the road textures causing more collisions. However, this greatly helps the ego-car stay on the road. Similarly, light (R3) and shadow (R4) are important factors for avoiding off the road driving. Object randomization factors (R5 & R6) are obviously the best parameters for reducing collisions. This is powerful as the object randomization appears to teach the model the concept of avoiding obstacles. This is indicated by the fact that despite the training set only containing simple geometric objects, the model avoids more complex obstacles like cars and pedestrians. Finally, the combined domain
randomization model can reduce the collision with a great margin while staying on the road all the times. The autonomy of our DR model reaches 0.98 in this set of experiments which is 11% higher than the one from the “No Rand” model.

To further investigate the impact of DR on simulation, randomization components (R1 to R6) are added one by one to the dataset (while keeping the size of the dataset fixed) and a model is trained for that specific combination. Although the previous experiment shows our model is able to avoid dynamic objects without seeing them during the training, we also use “object movement randomization” (R7) to further enhance the model reaction to the moving objects. Each model is tested in our designed Domain 2 for 30 minutes while changing the environment components (e.g., light, shadow, terrain, static and dynamic objects, etc.) after each cycle. More specifically, each model continuously tested over multiple cycles where each cycle used a different environment (to have a fair comparison, we keep these changes fixed for all the models). Figure 4.9 depicts the results of this experiment regarding the autonomy metric. As can be inferred from this plot, R1-2 (terrain+road) can extensively enhance the performance (especially decreasing the off-road) and adding other randomization factors can gradually increase the model’s generality to realistic environments. The full randomization model (also includes object movement randomization) achieves 99% autonomy.

**Ablation Study for Realistic Environment**

To evaluate the performance of our model in the real world, we utilized our collected parking dataset and the public Kitty dataset as explained before. To do so, the model receives a single image and generates a path demonstrating the future direction of the car. Figure 4.10 shows several samples from both datasets with DR and without DR. It can be clearly seen that when there are objects in its path the
<table>
<thead>
<tr>
<th>Method</th>
<th># collision</th>
<th># off-roads</th>
<th>autonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Rand</td>
<td>16</td>
<td>11</td>
<td>87</td>
</tr>
<tr>
<td>Terrain (R1)</td>
<td>6</td>
<td>8</td>
<td>94</td>
</tr>
<tr>
<td>Road (R2)</td>
<td>20</td>
<td>1</td>
<td>90</td>
</tr>
<tr>
<td>Light (R3)</td>
<td>13</td>
<td>1</td>
<td>93</td>
</tr>
<tr>
<td>Shadow (R4)</td>
<td>23</td>
<td>1</td>
<td>88</td>
</tr>
<tr>
<td>Obj Scale (R5)</td>
<td>13</td>
<td>4</td>
<td>92</td>
</tr>
<tr>
<td>Obj color &amp; texture (R6)</td>
<td>6</td>
<td>4</td>
<td>95</td>
</tr>
<tr>
<td>DR (Ours)</td>
<td>5</td>
<td>0</td>
<td>98</td>
</tr>
</tbody>
</table>

Table 4.4: Comparison results on simulation (Domain 2)

Figure 4.9: Impact of adding individual randomization to the model on autonomy
<table>
<thead>
<tr>
<th>Method</th>
<th>Parking</th>
<th>Kitti</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Rand</td>
<td>18.60</td>
<td>12.44</td>
</tr>
<tr>
<td>DR (Ours)</td>
<td>58.14</td>
<td>73.42</td>
</tr>
</tbody>
</table>

Table 4.5: Comparison results on real world (Domain 3)

DR model changes its path to avoid the objects, while the “No Rand” model can barely stay on the road (it can be seen from the sharp trajectory to the either left or right) or goes directly towards the object. For the Kitti dataset, the “No Rand” model is showing a sharp turn to the left in almost all the images, while our DR model smoothly changes its direction when observing an object close to it (e.g., first and second rows in Figure 4.10 (b)). These results also show that our model can deal with extra shadows (e.g., fourth rows in Figure 4.10 (a) and (b)) and detect obstacles in noisy images (e.g., last row in Figure 4.10 (a) which is an image taken through the windshield of the car using a cellphone camera with reflections from the dash).

Finally, the accuracy of these two models (No Rand and ours) is compared in Table 4.5. Accuracy is calculated by \( \frac{\text{number of images with correct trajectories}}{\text{total number of images}} \times 100 \). It can be seen from the table that DR can greatly enhance the performance of our obstacle avoidance model on the real-world images. These results show the effectiveness of DR in bridging the reality gap for this application.

### 4.3 Conclusion

One of the main challenges in multimedia and deep learning is having enough training data which represent various conditions and scenarios in the real world. Synthetic data generation is a practical technique to efficiently overcome this challenge.
In this chapter, two methods for synthetic data generation is presented each applied to a new application. The first method is a data generation and augmentation method based on Cycle-GAN for real-world flood event detection. While the second method is based on the simulation’s data for autonomous driving in which enough domain randomization is applied to bridge the reality gap between simulation and the real world. The proposed techniques increase the generalization of deep learning models in handling the unseen real-world scenarios while reducing the need to have a large-scale annotated dataset. The experimental results on these two real-world applications illustrate the effectiveness of the proposed methods in synthetic data generation.
Figure 4.10: Examples of results on parking and Kitti datasets
CHAPTER 5

AUTOMATIC SAMPLING FOR IMBALANCED DATA CLASSIFICATION

Most real-world data has a long tail distribution. In other words, some of the concepts are very scarce while others are abundant. This phenomenon is widely seen in different applications such as medical, object classification, and surveillance systems. The problem is to classify the minority cases from the overwhelming majority cases correctly.

To address this challenge, this chapter presents a new automatic sampling method that effectively handles the multi-class data imbalance problem. This component first introduces a new sampling technique which generates synthetic videos for minority classes using a spatio-temporal oversampling approach. Also, it is extended using a dynamic sampling model which automatically resamples the data during the training of CNNs.

5.1 Spatio-Temporal Synthetic Oversampling

Studies have shown that the use of sampling methods consisting the modification of the data distribution in an imbalanced dataset can help improve the classification performance. Thus, a new video oversampling method is proposed which includes two main components: random frame selection (temporal) and random augmentation (spatial). Suppose the multi-class training video dataset $V$ includes $N$ video samples and $M$ classes ($V = \{v_{i,j}|i = 1, \cdots, N; j = 1, \cdots, M\}$, where $v_{i,j}$ refers to the $i^{th}$ video sample belonging to the class $j$). The class set is $CL = \{cl_j|j = 1, \cdots, M\}$ where $cl_j$ refers to the $j^{th}$ class, that includes a different number of video
samples $nv_j$. The maximum number of samples in a class set is $\delta$ and each video includes $frm_{i,j}$ frames.

Algorithm 1 illustrates the steps of the proposed spatio-temporal synthetic oversampling method which gets the video dataset $V$, the class list $CL$, $\delta$, and $\alpha$ (sequence size) as the inputs and outputs the oversampled video dataset $\hat{V} = \{\hat{v}_{i,j,fr}|i = 1, \ldots, N; j = 1, \ldots, M; fr = 1, \ldots, freq\}$, where $\hat{v}_{i,j,fr}$ is the oversampled video related to the $i$th video, $j$th class, and $fr$th frequency. First, the frequency of oversampling for each class $cl_j$ is calculated as $freq_j \leftarrow \lceil \frac{\delta}{nv_j} \rceil$, where $\lceil \rceil$ is the ceiling function. In other words, the lower the number of samples in each class is, the higher the number of oversampling frequency will be. For example, if the maximum number of samples in all classes is $\delta = 100$ and the number of videos in the class $j$ is $nv_j = 20$, then $freq_j = 5$. Therefore, this video is oversampled five times. Next, for each video $v_{i,j}$, the function $GetFrames()$ generates all the frames $frm_{i,j}$ in the video $v_{i,j}$.

Since different videos have different numbers of frames, we turn each video into $\alpha$-frames sequences. So, for each frequency (e.g., $\{1, \cdots, 5\}$), we either randomly downsample the frames to $\alpha$-frames using $RandDown(.)$ function or upsample it to $\alpha$-frames using $UpSample(.)$ function. If the number of frames in a video is higher than the specified sequence size ($\alpha$), $RandDown(.)$ will return a random rescaled list of frames by getting a number to skip between iterations ($skip = \frac{\left|frm_{i,j}\right|}{\alpha}$) and then generating a random number for each $skip$. For example, if $\alpha = 5$ and $\left|frm_{i,j}\right| = 25$, then $skip = 5$ and a random number between one to five is selected in each iteration to generate the new rescaled frames.

The random frame selection process leads to a temporally oversampled dataset which can generate synthetic video samples from the original dataset. Although different frames are selected from each video in every iteration, they are spatially
Algorithm 1 The proposed spatio-temporal synthetic oversampling algorithm for an imbalanced video dataset

**Input:** Original training video dataset \( V = \{ v_{i,j} | i = 1, \ldots, N; j = 1, \ldots, M \} \), class list \( CL = \{ cl_j | j = 1, \ldots, M \} \), maximum number of video samples \( \delta \), and sequence size \( \alpha \).

**Output:** Oversampled video dataset \( \hat{V} = \{ \hat{v}_{i,j,fr} | i = 1, \ldots, N; j = 1, \ldots, M; fr = 1, \ldots, freq_j \} \).

1: \( \hat{V} \leftarrow \{ \}; \)
2: for all class \( cl_j \in CL \) do
3: \( freq_j \leftarrow \lceil \frac{\delta}{mv_j} \rceil; \)
4: for all video \( v_{i,j} \in V \) do
5: \( frm_{ij} \leftarrow \text{GetFrames}(v_{i,j}); \)
6: for \( fr = 1, \ldots, freq_j \) do
7: if \( \text{SizeOf(frames}_{i,j}) > \alpha \) then
8: \( frm_{ij} \leftarrow \text{RandDown}(frm_{ij}); \)
9: else
10: \( frm_{ij} \leftarrow \text{UpSample}(frm_{ij}); \)
11: \( Seq_{i,j,fr} \leftarrow \{ \}; \)
12: \( \hat{v}_{i,j,fr} \leftarrow \{ \}; \)
13: for all \( Img \in frm_{i,j} \) do
14: \( Img \leftarrow \text{RandAug}(Img); \)
15: \( \hat{v}_{i,j,fr} \leftarrow \hat{v}_{i,j,fr} + Img; \)
16: end for
17: \( \hat{V} \leftarrow \hat{V} + \hat{v}_{i,j,fr}; \)
18: end for
19: end for
20: end for
21: return \( \hat{V} \)
similar to each other, which may cause overfitting during the training. This is one of the main disadvantages of the oversampling techniques for imbalanced data.

To overcome this issue, we utilize augmentation techniques for image samples. Essentially, we propose a random augmentation method $\text{RandAug}(.)$ which applies various image transformation to each oversampled video using random parameters. In other words, a random uniform distribution is used to generate different parameters for image transformation. Specifically, the image transformation function includes random rotation, translation, shear, and brightness. Finally, the new augmented image ($\hat{\text{Img}}$) is added as the frames of the new oversampled video $\hat{v}_{i,j,fr}$. Finally, the new video set $\hat{V}$ is returned and used for the final classification.

5.1.1 Experimental Analysis

The proposed sampling method is applied to two video datasets to evaluate its performance, namely, the disaster video dataset introduced in [149] and public UCF101 action recognition dataset [49]. The disaster dataset was collected during two significant hurricanes (Irma and Harvey) and is naturally imbalanced. It includes seven classes (demo, emergency response, flood/storm, human relief, damage, victim, and speak) and the number of instances of each class varies from 40 to 400. On the other hands, UCF101 with 101 action categories is selected, which is one of the most challenging datasets due to its diversity in terms of actions, views, background, camera motion, and so on. However, different from the existing work on this dataset, the training set is resampled to serve for imbalanced video classification. To do so, a random number between 10 to the maximum number of instances in each class is generated and then those numbers of samples are randomly selected from each class. This means that each class contains at least 10 samples but may not include all of
its original samples for training. The goal is to show how the proposed model can enhance the multi-class classification on a large-scale dataset with skewed distributions. The first train/test split of this dataset suggested by the reference website is used in this experiment.

In the preprocessing step, we first extract all the frames form each video. Thereafter, we extract the features of every video frame through the last pooling layer of InceptionV3, resulting in a feature set with 2048 dimensions. These extracted features are later grouped into sequences. For the sake of simplicity and similar to the experiments in [161], α is selected as 40. In other words, we turn each video into a 40-frame sequence. For temporal analysis, a two-layer Residual Bidirectional LSTM with 1024-wide followed by a 1024 fully connected layer and 50% dropout is used. This relatively shallow network outperforms other deep stacked Residual Bidirectional LSTM models. We use Adam stochastic optimization with an aggressively small learning rate 0.000001 and L2 regularization with λ = 0.0003.

Tables 5.1 and 5.2 summarize the experimental evaluation with the comparison against no-sampling models on the disaster dataset and imbalanced UCF101, respectively. The tables include: (1) a model based on the CNN features and a simple LSTM. Although this model utilizes the temporal information using LSTM cells, it does not include any oversampling to handle the data imbalance problem; (2) the same CNN-LSTM architecture as the previous baseline, but in this model, the class weighting is added to automatically assign higher weights to the minority classes in the learning process; (3) the same CNN-LSTM architecture which also includes the proposed video oversampling; and (4) the same CNN-LSTM architecture which includes both video oversampling and class weighting.

As shown in Table 5.1 in the first group, no video oversampling is applied and it is assumed that deep learning can automatically handle the imbalanced data. It can
Table 5.1: Performance evaluation results of the proposed spatio-temporal synthetic oversampling algorithm on disaster dataset.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Acc</th>
<th>F1</th>
<th>Weighted F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No video oversampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.589</td>
<td>0.339</td>
<td>0.526</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.663</td>
<td>0.428</td>
<td>0.654</td>
</tr>
<tr>
<td>With video oversampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.671</td>
<td>0.456</td>
<td>0.662</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td><strong>0.678</strong></td>
<td><strong>0.477</strong></td>
<td><strong>0.688</strong></td>
</tr>
</tbody>
</table>

Table 5.2: Performance evaluation results of the proposed spatio-temporal synthetic oversampling algorithm on imbalanced UCF101.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Acc</th>
<th>F1</th>
<th>Weighted F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No video oversampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.685</td>
<td>0.655</td>
<td>0.670</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.680</td>
<td>0.660</td>
<td>0.670</td>
</tr>
<tr>
<td>With spatio-temporal video oversampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td><strong>0.706</strong></td>
<td><strong>0.684</strong></td>
<td><strong>0.696</strong></td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.690</td>
<td>0.669</td>
<td>0.679</td>
</tr>
</tbody>
</table>
be seen that both accuracy and F1 measures are significantly improved with a simple
class weighting. This shows when the data samples of some of the classes are limited,
it is necessary to assign a higher weight to these classes so that the learning algorithm
will not bias toward the majority ones. In the second group, similar experiments are
conducted plus applying the proposed spatio-temporal synthetic oversampling. It
can be inferred from this set of results that the accuracy is boosted using the video
oversampling. More importantly, the F1 measure is significantly improved, which
shows the importance of this sampling technique over the weighting approaches. It
is worth mentioning that the combination of oversampling and class weighting can
enhance the performance results on this dataset since it is highly imbalanced.

Similar experiments are conducted on the UCF101 with imbalanced distributions
to further show the ability of the proposed work on a large dataset. The results are
shown in Table 5.2 which includes two sets of results: CNN-LSTM with no video
oversampling, and CNN-LSTM with video oversampling. Each set includes the
results with and without class weighting. Similar to the disaster dataset, data over-
sampling can improve the performance regarding both accuracy and F1 measures
in a multi-class classification task. More specifically, the accuracy and F1 metric
are improved by 1.5% and 0.3, respectively. Different from the disaster dataset, the
results are decreased when video oversampling is combined with the class weighting
technique. Based on our observations, more overfitting happens for this dataset,
which is a common disadvantage of class weighting and oversampling techniques. It
is also due to the fact that the disaster dataset is much more imbalanced than the
UCF-101 and needs more balancing strategies.
5.2 Dynamic Sampling

The existing deep neural networks such as CNNs can achieve very high performance using a balanced dataset (e.g., CIFAR, MNIST, Caltech, etc.) compared to the conventional classifiers. However, based on our empirical study, they perform worse in imbalanced datasets since they were not originally designed to address this problem. In addition, in current studies, few evaluation metrics have been utilized to accurately measure the performance of the deep learning models on the minority concepts.

These challenges motivate us to propose a new deep learning model to tackle the class imbalance problem in real-world data. This model modifies the existing CNNs to handle imbalanced data for multi-class classification in an effective manner. For this purpose, the proposed model dynamically modifies the samples of each class in each iteration based on the F1-score of that class in the reference set. We propose to integrate the scores of the F1-based model with the basic CNN model and utilize data augmentation and transfer learning (fine-tuning the pre-trained models) techniques to avoid overfitting toward the minority classes and to generalize the model. This approach will significantly improve the performance of the minority classes and maintain the performance of the majority ones.

The proposed model is depicted in Figure 5.1, which includes real-time data augmentation module, CNN transfer learning module (will be discussed in chapter 6), and dynamic sampling module. Real-time data augmentation module is used to generate the transformed images for each training batch, transfer learning module is utilized to fine-tune the model, and dynamic sampling module is designed to automatically generate new samples based on the performance of the reference set.
5.2.1 Real-time Data Augmentation

One of the main challenges of deep learning is that large amounts of labeled data are required to achieve a reasonable detection performance. One solution to this limitation is data augmentation which artificially creates training data via multiple transformations. Augmentation can improve the generalization and prevent overfitting while reducing the need for large-scale datasets. This process can be done either offline before training the model or real-time in each iteration of learning. In offline augmentation, we need to re-create the dataset before starting the training process. However, in real-time augmentation, we only transform a small batch of images that are required for each training iteration. In this step, we generate batches of image data via real-time data augmentation. This approach directly augments the input data to the model in the data space. Following [55], several random transformations including rotation, shear, flip, brightness, and shift are applied to the training samples, so that the model never sees twice the same image.
5.2.2 Dynamic Sampling in CNNs

The training data used for transfer learning is critical to calibrate the parameters in the CNN model. In this dissertation, the dynamic sampling mechanism is proposed to tackle the imbalanced data problem. Inspired by how humans practice similar questions to avoid the same error happening again, we utilize the performance metric on the reference dataset to adjust the class distribution of the training samples. Here, the F1-score metric is used, and the score is calculated based on the one-against-all assumption for each class.

Algorithm 2 Model Training Framework with F1-Based Dynamic Sampling

**Input:** Training Images \( X_{\text{train}} \), Reference Images \( X_{\text{ref}} \), Initial Model \( M_S \), and Class List \( C \).

**Output:** Dynamic-Sampling-Based Model \( M^1 \).

1. \( M^1_0 \leftarrow M_S \), \( i \leftarrow 1 \), \( N^* \leftarrow \frac{|X_{\text{train}}|}{|C|} \);
2. for all class \( c_j \in C \) do
3. \( N_{i,j} \leftarrow N^* \);
4. end for
5. while \( \neg \)IsFullyTrained\((M^1_{i-1})\) do
6. \( X_i \leftarrow \) ImageSampling\((X_{\text{train}}, N_i)\);
7. \( M^1_i \leftarrow \) Train\((M^1_{i-1}, X_i)\);
8. \( F1_i \leftarrow \) UpdateF1\((M^1_i, X_{\text{ref}})\);
9. for all class \( c_j \in C \) do
10. \( N_{i+1,j} \leftarrow \) UpdateSampleSize\((F1_i, c_j)\);
11. end for
12. \( i \leftarrow i + 1 \)
13. \( M^1 \leftarrow M^1_{i-1} \)

In the proposed method, the target model \( M^1 \) is initialized by \( M_S \), trained by the set of images \( X_{\text{train}} \), and the dynamic sampling is performed based on the set of images \( X_{\text{ref}} \). All the classes are given in the list \( C = \{c_j\} \). Algorithm 2 shows the model training framework with dynamic sampling, where \( |X_{\text{train}}| \) is the size of the training dataset and \( |C| \) is the number of classes. In iteration \( i \), the training images \( X_i \) are sampled from \( X_{\text{train}} \) in the target domain and augmented as mentioned in
The number of the images of each class is determined by its F1-scores in the previous iteration $F_{i-1}$. After the model is trained by the generated sample set, the updated model is used to predict the concept of each image in the reference dataset $X_{\text{ref}}$, where the images are completely different from those in the testing dataset (obtained from either a different camera or different time). The F1-scores of class $c_j$ in iteration $i$, $f_{i,j}$, are thus calculated:

$$f_{i,j} = \frac{2 \cdot \text{Rec}_{i,j} \cdot \text{Pre}_{i,j}}{\text{Rec}_{i,j} + \text{Pre}_{i,j}}$$

(5.1)

where $\text{Pre}_{i,j}$ and $\text{Rec}_{i,j}$ are the precision and recall metrics of the class $c_j$ in iteration $i$. Note that $F_i = \{f_{i,j}\}$ is the vector of the F1-scores of all the classes in iteration $i$. If a class has a higher F1-score, it can be better distinguished from the other classes in $C$. Hence, it becomes more important to improve the performance of the classes with lower F1-scores and thus more samples (images) from these classes will be selected in the next iteration; while the total number of images trained in each iteration remains the same. Eq. (5.2) defines the number of images of $c_j$ of the next iteration. The number of images in any class $c_j$ is initialized to $N^*$ which is the average number of samples in all classes, i.e., $f_{0,j} = N^*$.

$$\text{UpdateSampleSize}(F_i, c_j) = \frac{1 - f_{i,j}}{\sum_{c_k \in C} (1 - f_{i,k})} \times N^*$$

(5.2)

### 5.2.3 Experimental Analysis

**Dataset Description and Preprocessing.** For this experiment, we use an image dataset automatically collected from network cameras as described in [162]. The data from network cameras have a wide range of characteristics. They may have different formats (such as Motion JPEG and MP4), different spatial resolutions (i.e., numbers of pixels), and temporal resolutions (i.e., frame rates). Some cameras allow viewers to select resolutions and frame rates but most cameras allow no options. The
data from some cameras may be noisy due to many reasons, for example, lens covered by dust, sand, and spider webs, or the views are blocked by trees. These restrictions impose additional challenges to data analysis. The system has no control over the data quality because the authors do not own these cameras. The pre-processing phase focuses on organizing and refining the dataset by creating annotations and reducing noise. In the current annotation workflow, multi-labels are not taken into account. Therefore, each image only represents one scene and focuses on its most significant concept.

The initial dataset contains thousands of images retrieved from the cameras, which are then split into smaller directories for annotation. The team uses a Java annotation software to label the images and save the results to a comma-separated file. The file includes the image names (camera ids with timestamp) and the corresponding labels. The final cleaned dataset contains over 10,000 images captured from network cameras. Those images include 19 semantic concepts (scenes) such as highway, intersection, yard, and mountain. The dataset is carefully divided into 70% training, 10% reference, and 20% testing so that each set includes different samples from all classes. All the images are resized to 299*299 pixels. In this dataset, the Positive to Negative (P/N) ratio for each concept ranges from 1.088 (concept “highway”) to as low as 0.002 (concept “airport”), which leads to an imbalanced data classification problem.

**Result Evaluation.** The F1-score (Avg. F1) is adopted as the main evaluation metric since it is the most valuable comparison metric for imbalanced data and is the trade-offs between precision and recall. Moreover, the Weighted Average F1-score (WAvg. F1) and top-1 accuracy (Acc.) metrics are used to show that the proposed model can improve not only the prediction of individual minority classes but also the overall performance results. Here, WAvg. F1 is the average of the F1-scores
of each class times its ratio of positive to all samples. The results of the proposed network are compared with the following models: (1) “Basic CNN”: a model based on VGGNet [163] running from scratch on our dataset (deeper models such as Inception will not converge well on this dataset); (2) “Deep CNN features+SVM”: using a deep CNN model as a fixed feature extractor and a linear support vector machine as a classifier; (3) “TL+No Aug.”: a fine-tuned CNN model without data augmentation; (4) “TL+Basic Aug.”: a fine-tuned model with real-time data augmentation; and finally (5) “TL+Balanced Aug.”: a fine-tuned CNN model plus a modified data augmentation in which each training batch includes a balanced number of classes. This model utilizes both oversampling and undersampling techniques. In all transfer learning models, Inception-v3 is used as the base CNN model. Stochastic Gradient Descent (SGD) [164] is used as the optimization with learning rate 0.0001 and momentum 0.9. The “ImageDataGenerator” layer in Keras [165] is used for augmentation. Specifically, the augmentation parameters used in this work are: shear_range=0.2, horizontal_flip=True, rotation_range=10, width_shift_range=0.2, and height_shift_range=0.2. Moreover, the threshold of model fusion, Tr, is selected as 0.3 based on the model performance on the reference data.

Table 5.3 illustrates the detailed performance results on this dataset. As can be inferred from the table, training a CNN from scratch performs the worst for all three evaluation metrics. This is due to the need for large-scale datasets to accurately update the random weights in CNNs. Transfer learning can significantly improve the results compared to basic CNN as shown in the third row of the table. However, it still cannot handle imbalanced data precisely. Similarly, the “TL+No Aug.” model performs poorly on the dataset regarding the Avg. F1-score. However, the model “TL+Basic Aug.” increases all the metrics compared to the no augmentation model. The “TL+Balanced Aug.” model (a hybrid oversampling and undersampling
model) can obviously improve the Avg. F1-score; however, its Acc. and WAvg. F1-scores are less than the ones in the original augmentation model. In other words, conventional imbalanced data techniques boost the performance of the minority classes by sacrificing the majority ones. Finally, the last row of the table shows how the proposed method improves the performance results for all three evaluation metrics. That is, it improves the prediction performance of the minority classes and also maintains the average accuracy.

Table 5.3: Performance evaluation of the proposed dynamic sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
<th>Avg. F1</th>
<th>WAvg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic CNN</td>
<td>0.649</td>
<td>0.254</td>
<td>0.630</td>
</tr>
<tr>
<td>Deep CNN Features+SVM</td>
<td>0.746</td>
<td>0.528</td>
<td>0.747</td>
</tr>
<tr>
<td>TL+No Aug.</td>
<td>0.765</td>
<td>0.432</td>
<td>0.755</td>
</tr>
<tr>
<td>TL+Basic Aug.</td>
<td>0.792</td>
<td>0.502</td>
<td>0.779</td>
</tr>
<tr>
<td>TL+Balanced Aug.</td>
<td>0.759</td>
<td>0.553</td>
<td>0.766</td>
</tr>
<tr>
<td>Proposed Dynamic Sampling Model</td>
<td>0.802</td>
<td>0.599</td>
<td>0.794</td>
</tr>
</tbody>
</table>

Figure 5.2: Comparison of F1-scores for each concept in the network camera dataset

The visualized performance results are demonstrated in Figure 5.2 which shows each concept (class) along with its P/N ratio in the parentheses. As can be seen from
this figure, the distribution of the data is highly skewed. Several concepts have very low P/N ratios (e.g., airport, bridge, and playground), few concepts have higher P/N ratios (e.g., mountain view, intersection, and water), and the "highway" concept has a very high P/N ratio. "Basic CNN" has the lowest F1-score in all classes and cannot detect any instances in classes with very low P/N ratios. The "TL+No Aug." model improves the results compared to "Basic CNN", but it still cannot detect the minority classes. "Deep CNN features+SVM" performs better than all other models in concept "Park+Building", while it performs poorly in almost all other classes. "TL+Balance Aug." can detect some instances in the minority classes, though its performance is much lower than the "TL+Basic Aug." model in other classes (e.g., highway and intersection). Despite the good performance of the "TL+Basic Aug." model in some concepts (e.g., yard and street), it cannot detect any instances in the low P/N ratio classes. Finally, the proposed model can significantly improve the detection performance of the minority classes, while maintaining or even improving the performance of all other concepts. This shows the effectiveness of the proposed model to classify imbalanced and heterogeneous data from the real-world datasets.

5.3 Conclusion

This chapter proposes two sampling methods to handle imbalanced data classification. First, it introduces the new spatio-temporal oversampling technique which generates synthetic videos to handle imbalanced data. Then, a dynamic sampling model is proposed which is based on CNNs together with real-time data augmentation to enhance the performance results for both minority and majority classes. The experimental results show the effectiveness of the proposed models on two imbalanced datasets and a real-time visual dataset captured by the network cameras.
The necessity of automatic semantic analysis in multimedia data is apparent in many real-world applications [12]. Specifically, video event detection is an important and challenging task in multimedia management systems. Over the last decade, researchers have been looking for automatic techniques to detect the most interesting events and concepts from multimedia data [117, 166, 167].

Till now, numerous deep learning architectures have been proposed for a variety of applications. However, it is almost impossible for a single model to work well for all scenarios and datasets. It is also difficult to handle imbalanced and big multimedia data because of overfitting, information loss, and additional bias [168].

This chapter presents several techniques for multimedia deep static representation learning to support multimedia semantic event detection using advanced deep learning.

6.1 Ensemble Deep Learning

In this dissertation, we propose an ensemble deep learning model, which not only overcomes the imbalanced data issue in multimedia big data, but also decreases the information loss and over-fitting problems caused by single models. Inspired by the great success of deep learning, it is used for deep feature analysis with the application to video event detection. Thereafter, an ensemble approach is developed based on the performance of each weak learner (Support Vector Machine (SVM) classifier) on each deep feature set to improve the semantic event detection in imbalanced datasets.
Figure 6.1: The proposed ensemble deep learning model

The Ensemble Deep Learning (EDL) model consists of a mixture of feature extractors using deep learning techniques which are integrated with the proposed ensemble algorithm. The whole model is divided into three main modules, namely (1) preprocessing, (2) deep feature extraction, and (3) classification (as shown in Figure 6.1). The classification module also includes the training, validation, and testing steps.
6.1.1 Preprocessing

The first step in every data analysis is how to preprocess the unstructured data. This step is domain specific and each type of data (e.g., audio, image, video, and text) may require its own preprocessing routines. For video processing in this study, we utilize an automatic and unsupervised shot boundary detection approach [169] based on the object tracking and image segmentation techniques. Using this approach, shot boundaries of each raw video are detected and the first frame of each shot is chosen as the keyframe. The first frame of each shot can be considered as the most distinctive one as it is the boundary of two successive shots. After the preprocessing, the selected keyframes are used for event detection in videos.

6.1.2 Deep Feature Extraction

Before 2010, research studies in computer vision mostly focused on improving the handcrafted features and generating more discriminative attributes from the data [170]. Some common and powerful handcrafted features include HOG [171], CEDD [172], and SIFT [173] for visual data and MFCCs [174] for aural data. However, this progress started to slow down between 2010 and 2012 with the advent of new deep learning techniques such as CNNs. In recent years, deep learning is growing very fast and has exceedingly raised the performance results. In this study, we also decide to take advantage of this emerging algorithm and apply it as a feature extractor to our data. For this purpose, several rich and deep feature extraction models are integrated in a proper manner. The deep feature extraction module is based on the CNN algorithm and utilizes the pre-trained models using transfer learning.
Convolutional Neural Networks

CNNs [103] are an advanced version of MultiLayer Perceptron (MLP) networks. However, in CNNs, most neurons are locally connected instead of fully connected, which highly increases the training speed and reduces over-fitting by eliminating a vast amount of parameters in the network.

Unlike MLP, the inputs of each layer in CNNs are arranged in three dimensions: width, height, and depth. For example, for a 256×256×3 image input, the width and height equal 256 and 3 is the depth of this input which refers to the channel number (e.g., RGB). Each neuron in CNNs is connected to a small region of its previous layer. In overall, there are three main layers to build a convolutional network architecture: (1) Convolutional layer, (2) Pooling layer, and (3) Fully connected layer [175]. A CNN includes a stack of convolutional layers followed by a pooling layer and is usually ended with a fully connected layer as shown in the deep feature extraction module in Figure 6.1.

In the convolutional layer, the neurons are connected to local regions in the input, each generating a dot product between a small region in the input volume and their corresponding weights. As a result, a number of feature maps are generated, by convolving (sliding) filters over all spatial locations in the input data. In other words, the feature maps are obtained by the convolution of the input data with a linear filter (and bias term addition) followed by a nonlinear activation function as illustrated in Equation (6.1), where $x_{ij}^k$ refers to the $k^{th}$ feature map at a given layer, $i$ and $j$ are the input dimensions, and $x_{ij}^{k-1}$ is the input data from the previous layer. Filters of the $k^{th}$ layer are determined by $W_{ij}^k$ (weights) and $b_j^k$ (bias).

$$x_{ij}^k = f((W_{ij}^k * x_{ij}^{k-1}) + b_j^k).$$ (6.1)
Finally, the activation function or nonlinearity is shown with $f$. One of the mostly used activation functions for deep learning is Rectified Linear Unit (ReLu) ($f(x) = \text{max}(0, x)$) which increases the nonlinearity and shows better performance compared to the conventional ones (e.g., sigmoid, tanh, etc.).

After each convolutional layer, there exists a pooling layer which applies a nonlinear downsampling operation along the width and height (spatial dimensions) of the image input given in Equation (6.2), where $\beta_{ij}^k$ is a multiplicative bias and $\text{down}(.)$ is a subsampling function (e.g., max, average, etc.). Therefore, using the pooling layer, the size of each activation map is reduced, which makes the representation more manageable. It also handles over-fitting and provides additional robustness to the network.

$$x_{ij}^k = f(\beta_{ij}^k \text{down}(x_{ij}^{k-1}) + b_j^k).$$

Finally, the fully connected layer is used as the last layer of CNNs to compute more high-level reasoning or the class scores. Similar to traditional neural networks, all neurons or activation maps from the previous convolutional-subsampling layer are fully connected to a single neuron in this layer.

**Feature Extraction using Transfer Learning**

In this work, several advanced and successful deep learning architectures are utilized for visual feature extraction. For this purpose, instead of training an entire CNN from scratch, we take the pre-trained reference models and treat the convolutional networks as feature extractors for new datasets. Theses reference models are pre-trained on very large-scale datasets. Specifically, we select those models which have more impacts on the image processing field in recent years. The ImageNet dataset [155] which contains millions of images with 1000 concept categories is used
Table 6.1: Pre-trained reference models for feature extraction

<table>
<thead>
<tr>
<th>Method</th>
<th>Challenges</th>
<th># layers</th>
<th># categories</th>
<th>dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>ILSVRC 2012</td>
<td>8</td>
<td>1000</td>
<td>ImageNet</td>
</tr>
<tr>
<td>R-CNN</td>
<td>ILSVRC 2013</td>
<td>7</td>
<td>200</td>
<td>ImageNet</td>
</tr>
<tr>
<td></td>
<td>VOC 2012</td>
<td></td>
<td></td>
<td>PASCAL VOC</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>ILSVRC 2014</td>
<td>22</td>
<td>1000</td>
<td>ImageNet</td>
</tr>
<tr>
<td>ResNet</td>
<td>ILSVRC 2015</td>
<td>152</td>
<td>1000</td>
<td>ImageNet</td>
</tr>
<tr>
<td></td>
<td>COCO 2015</td>
<td></td>
<td></td>
<td>COCO</td>
</tr>
</tbody>
</table>

to train all such models. Therefore, we run these pre-trained models on our datasets and generate the features (also known as CNN codes) for all images. These feature sets are further used for the classification. Table 6.1 presents a summary of the CNN models used in this work for feature extraction. As can be seen from the table, a variety of models with different numbers of layers and architectures are used.

### 6.1.3 Classification

After extracting features from the aforementioned reference models using an unsupervised transfer learning, we employ a new ensemble technique to alleviate the over-fitting problem and to improve the performance. First, the extracted deep features are analyzed to find the importance of each feature set extracted from each deep learning model in a supervised manner. In addition, an enhanced ensemble method is proposed to optimally integrate the trained models. This method effectively adjusts the weight coefficients for the classification module (please refer to Figure 1). First, we train $k$ classification models (weak classifiers in the ensemble), each trained on a feature set. Thereafter, the weight coefficient of each classifier is adjusted based on its classification performance on the validation dataset. The classification step includes two parts: deep ensemble learning and testing.
Deep Ensemble Learning

The training procedure of the proposed deep ensemble learning is illustrated in Algorithm 5. In the first step, we divide the dataset into three parts: training $T$, validation $V$, and testing $T'$. Suppose the training set is defined as $T = \{(t_1, c_1), (t_2, c_2), \ldots, (t_N, c_N)\}$, where $t_i$ is the $i^{th}$ training instance, $c_i$ is the instance class (e.g., for a binary classification task $c_i \in \{0, 1\}$), and $N$ is the size of the training set. Moreover, we store all the feature sets extracted from all deep learning models in $Fr$. This is another input of the training algorithm.

The proposed ensemble learning can be seen as a bootstrap aggregation (bagging) which involves all the weak learners (classifiers) in the voting. However, in this algorithm, a weighted voting is generated rather than assigning an equal weight to each learner. In addition, the weights are assigned based on a metric for imbalanced data. Therefore, the results are improved toward the minority class, while the performance of the majority class is maintained as high as possible. The weak learners or models are defined as $M = \{M_j, j = 1, 2, \ldots, k\}$, each trained using a linear SVM as shown in Lines 1-3 of Algorithm 5 where $k$ is the number of total weak learners. SVM is used as the main classifier as it has shown promising results when it integrates with deep learning [176]. After weak learners $M_j (j = 1, 2, \ldots, k)$ are trained using the training instances, each model is evaluated using the validation set $V$ as shown in Lines 4-7 of Algorithm 5. For the evaluation and adjusting the weight coefficients, the F1 measure is used.

Thereafter, the weight of each trained model $M_j$ is calculated using the ratio of the corresponding F1 score ($F1_j$) to the sum of the scores for all models as shown in Equation (6.3).

$$W_j = \frac{F1_j}{\sum_{j=1}^{k} F1_j}. \quad (6.3)$$
The weight coefficient assignees higher values (probability) to the models that are more confident about their prediction. Finally, the algorithm returns each model and its corresponding weight $W_j$ to be further used in the testing module.

**Algorithm 3** Training of Ensemble Deep Learning

**Input:** Training instances $T\{ (t_i, c_i), \ i = 1, 2, \cdots, N \}$, Validation instances $V\{ (v_i, c_i), \ i = 1, 2, \cdots, N_2 \}$, Feature set $Fr = \{ F_j, \ j = 1, 2, \cdots, k \}$.

**Output:** Weight matrix $W_j$, Trained models $M_j$.

1: for all $F_j \in Fr (j = 1, \cdots, k)$ do
2: \hs $M_j \leftarrow$ SVM($T, F_j$);
3: end for
4: for all $F_j \in Fr (j = 1, \cdots, k)$ do
5: \hs $F_{1j} \leftarrow$ Validate($V, F_j$);
6: \hs $W_j = \frac{F_{1j}}{\sum_{j=1}^{k} F_{1j}}$;
7: end for
8: return $W_j, M_j$

**Testing**

Algorithm 4 illustrates the testing procedure. The first input of this algorithm includes the testing set $T'\{ (t'_i), \ i = 1, 2, \cdots, N_3 \}$, where $t'_i$ is the $i^{th}$ testing instance and $N_3$ is the total number of testing instances. In addition to the instances, for each training model $M_j$, its feature set $F_j$ and weight matrix $W_j$ are given as the input of this algorithm to predict the labels of the testing instances. In order to achieve this, we calculate a weighted sum of the $k$ models (or weighted voting). As can be seen in Line 2-4 of Algorithm 4 the labels $L_j (j = 1, \cdots, k)$ generated by the $j^{th}$ weak learner is calculated for each testing instance. Afterwards, the final label $PL_i$ is calculated as shown in Line 5 of Algorithm 4. Thus, if the generated weighted sum is greater than half, the label is predicted as positive. Accordingly, the weak learners with higher validation performance have higher impacts on the testing prediction in the proposed ensemble algorithm.
Algorithm 4 Testing of Ensemble Deep Learning

**Input:** Testing instances $T'\{(t'_i), \ i = 1, 2, \cdots, N_3\}$, Feature set $F_r = \{F_j, \ j = 1, 2, \cdots, k\}$, trained models $M_j$s and the weight matrices $W_j$s.

**Output:** Predicted labels $PL_i$.

1: for all $t'_i \in T'(i = 1, 2, \cdots, N_3)$ do  
2: \hspace{1em} for all $F_j \in F_r(j = 1, \cdots, k)$ do  
3: \hspace{2em} $L_j \leftarrow M_j(t'_i, F_j)$;  
4: \hspace{1em} end for  
5: $PL_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{k} L_j \ast W_j \geq \frac{1}{2}, \\ 0 & \text{otherwise} \end{cases}$  
6: end for  
7: return $PL_i$

6.1.4 Experimental Analysis

The proposed Ensemble Deep Learning (EDL) model can be generally applied to a variety of real-world problems such as image, audio, and text classification. In this work, we specifically evaluated our model on two video datasets in order to detect semantic events. The first dataset includes videos containing natural disasters, while the second one is a public large-scale video dataset called TRECVID.

Since both datasets are highly imbalanced, usual metrics such as accuracy and mean-square error may not be effective and reliable. The reason is that the conventional classifiers, which are mostly biased to the majority class, may show very high accuracy on this class while we are more interested in the minority class. Therefore, the proposed model is evaluated using the common measurement metrics for imbalanced data. Specifically, the confusion matrix parameters including True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), as well as Precision, Recall, and F1 measure are employed to evaluate the EDL performance.

The deep learning model utilized in the following experiments is called Caffe [177]. It includes the advanced deep learning techniques and the state-of-art architectures. The main advantage of Caffe is its rich and updated pre-trained reference mod-
els, which can be easily used for fine-tuning and transfer learning. Among all the reference models, we utilize the most successful ones in the literature including R-CNN, CaffeNet, GoogleNet, AlexNet, and ResNet to leverage in the proposed ensemble deep learning model. To do so, we extract a variety of feature sets from video keyframes using the aforementioned Caffe reference models as explained in Section 6.1.2. All features are extracted from the last fully-connected layer (Inner-Product type) of each model. For instance, layer “fc-rcnn” of R-CNN, layer “fc8” of CaffeNet and AlexNet, “loss3/classifier” of GoogleNet, and “fc1000” of ResNet are used. All models are originally trained on the ImageNet dataset, a very large-scale image database including 1000 classes. The last layer of each selected model generates a 1000-dimension feature vector except the R-CNN, which generates 200 features in its fully connected layer.

**Evaluation of EDL on Disaster Dataset**

The disaster dataset is collected from the YouTube videos including seven natural disasters such as flood, damage, fire, mud-rock, tornado, and lightning. In overall, this dataset includes about 80 videos. After applying the video shot boundary detection and keyframe selection techniques, 6884 shots are extracted from this dataset [178]. The average positive/negative (P/N) ratio of the disaster dataset is 0.051. This ratio shows the imbalanced distribution of the data. Figure 6.2 shows some example keyframes from the disaster dataset.

For the disaster dataset, the proposed EDL model is compared with two sets of algorithms: the handcrafted features (or engineering features) and the deep learning features. For the first group, several low-level and mid-level features such as HOG, CEDD, color histogram, texture, and wavelet are extracted. The overall feature set for each keyframe includes 707 visual attributes. While, in the second group, the
Figure 6.2: Sample keyframes for each concept in the disaster dataset.
features are generated by applying the deep learning reference models directly on each keyframe. After feature extraction, we apply several well-known classifiers including Decision Tree (DT), Multiple Correspondence Analysis (MCA) [179], SVM, and a boosting algorithm on the handcrafted features. The last one is an ensemble algorithm which can be considered as a credible benchmark to be compared with our ensemble model. The SVM classifier is also utilized for the second group (deep features) as it has been proven to be a successful classifier when it is integrated with deep learning techniques. To have a fair comparison, all the classifiers are tuned to reach to their highest results on this dataset and they are evaluated through a 3-fold cross validation.

Table 6.2 shows the average precision, recall, and F1 score for both handcrafted and deep feature groups integrated with different classification algorithms. The last row also shows the performance of the proposed EDL algorithm. As can be conclude from the table, the proposed model improves the performance results in comparison with all the techniques in both groups. In other words, it not only outperforms all the conventional classifiers integrated with the engineering features, but also beats the recent well-known deep neural networks such as GoogleNet and AlexNet. By looking deeper on the results, one can infer that SVM and ensemble (boosting) techniques acquire the highest performance in terms of F1-score in the handcrafted features group. Specifically, SVM has the highest precision compared to all other algorithms including our proposed EDL. However, its low recall value decreases its overall F1 score. It is worth mentioning that a higher recall value, or in other words lower false negative, is more preferable in an imbalanced data where the correct detection of minority class is vital (e.g., in a cancer detection application). Therefore, integrating the deep features with the SVM classifier in a reasonable manner can increase the recall value and F1 measure significantly as
Table 6.2: Average performance of various feature sets and classifiers on the disaster dataset

<table>
<thead>
<tr>
<th>Features</th>
<th>Classifier</th>
<th>precision</th>
<th>recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>handcrafted</td>
<td>DT</td>
<td>0.816</td>
<td>0.823</td>
<td>0.819</td>
</tr>
<tr>
<td>handcrafted</td>
<td>MCA</td>
<td>0.894</td>
<td>0.720</td>
<td>0.782</td>
</tr>
<tr>
<td>handcrafted</td>
<td>Boosting</td>
<td>0.910</td>
<td>0.841</td>
<td>0.867</td>
</tr>
<tr>
<td>handcrafted</td>
<td>SVM</td>
<td>0.957</td>
<td>0.802</td>
<td>0.868</td>
</tr>
<tr>
<td>R-CNN</td>
<td>SVM</td>
<td>0.930</td>
<td>0.722</td>
<td>0.794</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>SVM</td>
<td>0.918</td>
<td>0.840</td>
<td>0.875</td>
</tr>
<tr>
<td>CaffeNet</td>
<td>SVM</td>
<td>0.919</td>
<td>0.840</td>
<td>0.876</td>
</tr>
<tr>
<td>AlexNet</td>
<td>SVM</td>
<td>0.924</td>
<td>0.859</td>
<td>0.888</td>
</tr>
<tr>
<td>deep features</td>
<td>EDL</td>
<td>0.949</td>
<td>0.883</td>
<td>0.913</td>
</tr>
</tbody>
</table>

shown in the second group of the results (deep learning features) in Table 6.2. In this group, AlexNet reaches to the highest F1 score compared to other deep learning techniques. This interesting fact shows that very deep and complex architectures (e.g., GoogleNet) cannot be always useful for different types of datasets. Sometimes a lighter version of deep neural networks not only is more efficient than the complex ones, but also can be generalized for different similar tasks. For example, in this experiment, although the nature of ImageNet is very different with our disaster dataset, but it can be seen that most of the pre-trained models (e.g., AlexNet) on ImageNet can classify disaster events in videos with a reasonable performance. Finally, the proposed model utilizes the power of deep learning features integrated with a new ensemble technique to improve the overall performance in terms of recall and F1 score. The overall F1 score is calculated as 0.913 which is 4.5% higher than the best classifier in the first group and 2.5% higher than the best result in the second group.

Figure 6.3 visualizes the performance results of each deep learning algorithm in which the y-axis refers to the F1 score and the x-axis shows different disaster events. It can be seen from this figure that the proposed EDL model improves the F1 score
Figure 6.3: Performance evaluation for different concepts on the disaster dataset for all disaster events compared to other techniques. In this figure, in most cases, R-CNN has the lowest performance which can be due to two main reasons. First, its architecture is mainly designed for region-based object detection and semantic segmentation, so that its architecture does not properly match a frame-based video event detection task. In addition, it generates 200 features which include less information than other selected deep learning architectures which generate 1000 features in their last layer. CaffeNet and GoogleNet have achieved very close average performance despite of their different architectures. More specifically, CaffeNet reaches a higher performance for tornado and damage concepts, while GoogleNet beats CaffeNet in fire and snow. AlexNet outperforms all other deep learning techniques in terms of F1 score for almost all concepts except lighting and snow. Finally, our proposed technique could successfully improve the results on all semantic events and outperforms the state-of-the-art deep learning algorithms.
Evaluation of EDL on TRECVID Dataset

More experiments are conducted to further demonstrate the effectiveness of the proposed EDL model. For this purpose, the TRECVID 2011 IACC.1.B dataset including the Internet Archive videos under the Creative Commons licenses is selected as the evaluation benchmark. In the TRECVID Semantic Indexing (SIN) task, similar to the disaster dataset, the goal is to detect the semantic concepts contained in the video shots. The automatic assignment of semantic labels or tags is a fundamental step for further video browsing, search, and filtering, to name a few. In overall, the IACC.1.B dataset includes hundred thousands of training and testing video keyframes and 346 concepts. In this work, 20,000 keyframes (the first 10,000 instances from the training and the first 10,000 ones from the testing data) are selected to evaluate our EDL model. In this dataset, a concept refers to a high-level semantic content or object such as person, vehicle, and sky. Figure 6.4 demonstrates several sample keyframes in this dataset. The main challenge of this dataset is its highly imbalanced or skewed distribution. Table 6.3 presents the statistics of the selected concepts in the training and testing sets. These concepts are selected due to their popularity and also the variety of the P/N ratios they have. Therefore, we can evaluate the behavior and functionality of the proposed model in various situations. For example, the concepts “Person”, “Outdoor”, and “Road” include more positive instances (P/N ratio is above 15%) in the training set compared to other concepts. The average P/N ratios for the training set and testing set are 0.087 and 0.039, respectively.

In this experiment, the results from our model are compared with the ones from Tokyo Institute of Technology (TiTech) which was selected as the best performance in the TRECVID 2011 semantic indexing task. The TiTech group extracted several low-level features such as SIFT and MFCCs from each video shot.
Figure 6.4: Sample keyframes for each concept in TRECVID dataset
Table 6.3: Statistic summary of selected concepts in TRECVID

<table>
<thead>
<tr>
<th>Concept</th>
<th>Training P/N Ratio</th>
<th>Testing P/N Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>0.3714</td>
<td>0.1511</td>
</tr>
<tr>
<td>Outdoor</td>
<td>0.1801</td>
<td>0.1490</td>
</tr>
<tr>
<td>Indoor</td>
<td>0.0589</td>
<td>0.0218</td>
</tr>
<tr>
<td>Face</td>
<td>0.0993</td>
<td>0.0035</td>
</tr>
<tr>
<td>Male</td>
<td>0.0418</td>
<td>0.0179</td>
</tr>
<tr>
<td>Vehicle</td>
<td>0.0253</td>
<td>0.0560</td>
</tr>
<tr>
<td>Sky</td>
<td>0.0275</td>
<td>0.0119</td>
</tr>
<tr>
<td>Vegetation</td>
<td>0.0675</td>
<td>0.0233</td>
</tr>
<tr>
<td>Text</td>
<td>0.0804</td>
<td>0.0188</td>
</tr>
<tr>
<td>Entertainment</td>
<td>0.0413</td>
<td>0.0006</td>
</tr>
<tr>
<td>Road</td>
<td>0.165</td>
<td>0.0165</td>
</tr>
</tbody>
</table>

Therefore, they utilized the handcrafted features including both visual and audio features. Thereafter, an advanced tree-structured Gaussian Mixture Model (GMM) is proposed to model the distribution of low-level features using the maximum a posteriori (MAP) adaptation. In addition, similar to the disaster dataset, the EDL model is compared with several deep features integrated with the SVM classifier. For this experiment, we replaced the R-CNN with the ResNet due to the low performance of R-CNN in the previous experiment. In addition, the CaffeNet is removed as it has a very similar architecture to the AlexNet. Accordingly, this time we only have three weak learners in our EDL model.

Table 6.4 shows the precision and recall values of all deep learning algorithms as well as the ones from the proposed EDL for each concept in this dataset (since the precision and recall values of TiTech model are not available, they are not listed in this table). As mentioned earlier, the recall metric is more important than the precision in an imbalanced dataset. Thus, the proposed method can achieve the highest recall value (0.436) while maintaining the precision as high as possible as shown in Table 6.4. Based on this table, the proposed EDL model beats all other methods in terms of the recall value in most concepts. Although there are a few concepts
Table 6.4: Performance evaluation for different concepts on the TRECVID dataset

<table>
<thead>
<tr>
<th>concept</th>
<th>AlexNet pre</th>
<th>AlexNet rec</th>
<th>GoogleNet pre</th>
<th>GoogleNet rec</th>
<th>ResNet pre</th>
<th>ResNet rec</th>
<th>EDL pre</th>
<th>EDL rec</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>0.321</td>
<td>0.424</td>
<td>0.334</td>
<td>0.487</td>
<td>0.337</td>
<td>0.5</td>
<td>0.332</td>
<td>0.538</td>
</tr>
<tr>
<td>outdoor</td>
<td>0.488</td>
<td>0.641</td>
<td>0.502</td>
<td>0.672</td>
<td>0.504</td>
<td>0.681</td>
<td>0.483</td>
<td>0.730</td>
</tr>
<tr>
<td>indoor</td>
<td>0.500</td>
<td>0.019</td>
<td>0.333</td>
<td>0.005</td>
<td>0.175</td>
<td>0.052</td>
<td>0.2</td>
<td>0.066</td>
</tr>
<tr>
<td>face</td>
<td>0.075</td>
<td>0.514</td>
<td>0.085</td>
<td>0.571</td>
<td>0.085</td>
<td>0.714</td>
<td>0.092</td>
<td>0.571</td>
</tr>
<tr>
<td>male</td>
<td>0.529</td>
<td>0.051</td>
<td>0.531</td>
<td>0.097</td>
<td>0.392</td>
<td>0.114</td>
<td>0.393</td>
<td>0.125</td>
</tr>
<tr>
<td>vehicle</td>
<td>0.481</td>
<td>0.094</td>
<td>0.494</td>
<td>0.145</td>
<td>0.393</td>
<td>0.158</td>
<td>0.407</td>
<td>0.177</td>
</tr>
<tr>
<td>sky</td>
<td>0.294</td>
<td>0.339</td>
<td>0.220</td>
<td>0.373</td>
<td>0.201</td>
<td>0.398</td>
<td>0.208</td>
<td>0.441</td>
</tr>
<tr>
<td>vegetation</td>
<td>0.269</td>
<td>0.553</td>
<td>0.303</td>
<td>0.662</td>
<td>0.292</td>
<td>0.754</td>
<td>0.328</td>
<td>0.675</td>
</tr>
<tr>
<td>text</td>
<td>0.177</td>
<td>0.595</td>
<td>0.178</td>
<td>0.665</td>
<td>0.178</td>
<td>0.616</td>
<td>0.200</td>
<td>0.643</td>
</tr>
<tr>
<td>entertainment</td>
<td>0.083</td>
<td>0.667</td>
<td>0.071</td>
<td>0.167</td>
<td>0.022</td>
<td>0.167</td>
<td>0.083</td>
<td>0.667</td>
</tr>
<tr>
<td>road</td>
<td>0.167</td>
<td>0.025</td>
<td>0.397</td>
<td>0.167</td>
<td>0.213</td>
<td>0.16</td>
<td>0.397</td>
<td>0.167</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.308</strong></td>
<td><strong>0.357</strong></td>
<td><strong>0.313</strong></td>
<td><strong>0.365</strong></td>
<td><strong>0.254</strong></td>
<td><strong>0.392</strong></td>
<td><strong>0.284</strong></td>
<td><strong>0.436</strong></td>
</tr>
</tbody>
</table>

such as “text” and “vegetation” having higher recall values in other methods than the EDL, their low precision decreases the overall F1 score significantly. Thus, our proposed method tries to keep the balance between these two metrics and provides higher F1 scores in such concepts. This phenomenon can be also seen in Figure 6.5, which visualizes the F1 scores for all the benchmark algorithms including the TiTech and all other deep learning models. As can be inferred from the figure, the proposed EDL outperforms all other methods in all concepts except sky. In this concept, although the proposed EDL detects the most positive instances, AlexNet achieves the highest F1 value because of its low false positive or its high precision. Another important fact can be concluded from Figure 6.5 is the low F1 scores achieved by the TiTech (the best results in the SIN task in TRECVID 2011) compared to deep learning techniques. In other words, similar to the results acquired using the disaster dataset, deep learning features (even those extracted from shallow and simple architectures) contain more information to discriminate the objects than the handcrafted features in many applications.
Figure 6.5: Performance evaluation on the TRECVID dataset

Figure 6.6: A comparison of True Positive value on the TRECVID dataset
Finally, Figure 6.6 depicts the number of positive instances predicted correctly (True Positive) by each deep learning technique. It can be seen that the proposed EDL detects much more positive instances in many concepts such as “person”, “outdoor”, and “indoor”. It also maintains TP as high as possible for highly imbalanced concepts. For instance, the “Entertainment” concept has the lowest TP in the testing set (please refer to Table 6.3), which means that about only six instances are positive among 10,000 instances in this concept. The EDL and AlexNet can detect four out of six positive instances, while the GoogleNet and ResNet can only detect one positive instance and other five ones are classified as negative.

In summary, based on the experiments on two different datasets with skewed distributions, the proposed EDL achieves promising performance compared to other well-known techniques in this area.

6.2 Efficient Deep Residual-Inception Network

The recent advancement in image recognition was obtained with deeper and wider networks [1, 2]. One example is the residual architecture [1] which reaches to 152 layers, almost 8 times deeper than GoogLeNet [2] and VGG nets [105]. As the network grows in depth (or sometimes in width [2]), the features can be enriched and more high-level features can be extracted compared to those from early layers.

Now the question is whether deeper networks always generate better performance. In other words, does stacking more layers lead to better learning? In addition, are extremely deep networks computationally reasonable for and applicable to different applications? Based on several experiments reported in [1, 184], network improvements and better learning are not as easy as stacking more layers and a careful design is indispensable. The first problem driven by the depth increase
is called “vanishing” gradients \[106, 184\], in which the network cannot be trained with regards to the gradient based algorithms like back propagations and the convergence is prevented from the early layers. There have been several solutions in the literature to address this problem by using rectified linear activation instead of common activation functions (e.g., sigmoid or tanh) \[185\] and normalization layers \[186\]. Another issue is when the training accuracy is saturated and suddenly starts to degrade, which is not due to over-fitting. One solution to this problem is addressed in [1] using Residual Learning.

To address the aforementioned problems, in this work, we propose a new deep learning architecture based on the traditional Convolutional Neural Networks (CNNs) integrated into two levels of Residual-Inception combination. This architecture is successfully tested on two different multimedia datasets. Specifically, it is applied on a video event detection task containing natural disaster events. The overall Deep Residual-Inception network improves the losses compared to the most recent deep learning architectures and also significantly decreases the computational costs.

The proposed network contains three main modules, including traditional convolutional neural networks, residual connections for training deep architectures, along with inception modules for retaining computational efficiency.

### 6.2.1 Residual Module

The residuals are essential for very deep network to avoid the degradation problem. Suppose \(H(x)\) is an underlying mapping to be fit by few neighbor layers, where \(x\) is the first input. Based on the report in [187], several nonlinear layers are capable of asymptotically approximating complicated functions. Therefore, instead of approximating \(H(x)\) using the neighbor layers, a residual function \(F(x) := H(x) - x\) will
be approximated by these layers. Hence, a residual block (as shown in Figure 6.7) is defined as follows.

\[ y^k = F(x^k, W^k) + x^k, \]  

(6.4)

where \( x^k \) and \( y^k \) are the input and output vectors and \( F \) represents the residual mapping and the connection \( (F + x) \) is performed by an element-wise addition.

In our view, utilizing residual networks helps the network to learn both weights and depths at the same time. In addition, we ensure the new layer \((N + 1)\) is learning something new by providing the output of the previous layer \((N)\) without any modification to the output of the current layer \((N + 1)\). This technique handles both vanishing gradient and degradation problems in very deep networks.

### 6.2.2 Inception Module

The inception module significantly improves the computational efficiency while scaling up the network. This module heavily utilizes NIN in its internal architecture for two reasons: (1) to reduce the input dimension and eliminate the computational bottlenecks and (2) to increase not only the network depth, but also its width to improve the overall performance. In other words, since a bigger size means a larger
number of parameters, which causes overfitting in deep networks, leveraging sparsity even inside the convolutions leads to better results. Therefore, the filter-level sparsity blocks are introduced in the inception module. The filter sizes are $1 \times 1$, $3 \times 3$, and $5 \times 5$. All layers along with their output filter banks are combined and concatenated into one output vector. In addition, pooling is added in each inception since it is essential for convolutional networks. To further compress the network and reduce the dimension, $1 \times 1$ convolutional layers are added before each expensive convolutions. One sample of this module is used in our network as shown in Figure 6.8.

### 6.2.3 Network Architecture

To handle the issue of overfitting, vanishing gradient, and network saturation problems, we study the combination of residual and inception modules. As the first layers generate low-level abstraction while the higher layers provide more high-level features from the data, the proposed network starts with a light version of each module and the ratio of convolutions and Residual-Inception blocks are gradually increased. The proposed network utilizes a few numbers of Residual-Inception stacks rather than very deep stacks of each single module. It starts with the traditional CNNs.
along with a lighter version of residual in conjunction with an inception module. Then, an increased dimension of Residual-Inception is added on top of the previous layer. At the end of the last residual block, an average pooling, a dropout, a fully connected layer, and softmax are added to generate the final classification results. This block is also added to the end of the last inception block groups. In this case, we can evaluate which module generates smaller losses in each training step. Figure 6.9 depicts a schematic view of the proposed architecture. Table 6.5 also shows the detailed architecture. In residual blocks, downsampling is performed directly by the convolutional layers using the stride of 2. ReLu is used as an activation function in all convolutions including those inside residual and inception modules. However, it is removed after each element-wise addition operation [188]. We use dropout [189] after average pooling to avoid overfitting. The total number of layers is 33 including 3 CNNs, 21 residual layers, 8 inception layers, and 1 fully connected layer. This network is designed efficiently, which can be run even on devices with limited resources.
Figure 6.9: The proposed deep Residual-Inception network
6.2.4 Experimental Analysis

Datasets. Two datasets are selected to evaluate the proposed network. First, the disaster dataset proposed in the previous section. Second, we conducted more experiments on CIFAR-10 [3], a large public dataset consisting of 60,000 $32 \times 32$ color images in 10 classes (shown in Figure 6.10). It is divided into 50k training and 10k testing images. The main focus is to show the functionality of the proposed network on a large dataset compared to well-known deep learning algorithms in different training iterations and times. However, we do not intend to push the state-of-the-art results which also utilized other techniques such as augmentation, ensemble, randomized input order, and sampling methodologies [2]. Therefore, a simple architecture of the proposed network, as well as the ones of the comparison benchmarks are used in these experiments.

Figure 6.10: CIFAR-10 image dataset [3]

Experiment Setups and Results. For the disaster dataset, the network input is
224 × 224 images and channel-wise (pixel) mean is used instead of mean image [105].

The learning rate is set to 0.0001 to train the network slowly and avoid overfitting.

The input of the network for the CIFAR-10 is 32 × 32 images with subtracting the mean pixel. We start with a base learning rate of 0.01 and divide it by 10 every 20k iterations. For both datasets, SGD with a momentum of 0.9 and weight decay of 0.0001 is selected to train the model.

Caffe [177] is used as the deep learning framework. Our proposed network is compared with two successful deep learning networks: GoogLeNet (Inception) [2] with 22 layers and Deep Residual with 50 layers (proposed by Microsoft [1]). For this experiment, we used the CPU-based implementation on 6 servers with 64 processors.

Figure 6.11 depicts the patterns of the learning in the proposed deep Residual-Inception network compared to two selected benchmarks (Inception and Residual network). Specifically, in Figure 6.11(a), although the proposed network starts with higher losses, it starts to converge after less than 10,000 iterations. The inception network has very low losses at first but it does not show any improvement from early stages, which can be due to over-fitting of this wide and deep network. The residual network shows a similar behavior as our network, but it still has higher losses than the proposed Residual-Inception network in all iterations, due to its very deep architecture. Similar patterns have been shown in Figure 6.11(b) on CIFAR-10 which includes more data and classes than the disaster dataset. In this figure, the inception network has higher training losses in all iterations; while the Residual-Inception network and the deep residual network have lower losses, respectively. Therefore, based on this experiment, one can conclude that a compact combination of these two benchmarks can converge earlier and produce lower losses.

Finally, since interesting video event detection is the main purpose of this work, we have utilized the proposed Deep Residual-Inception network to analyze its be-
behavior on each disaster class. For this purpose, a binary classification is conducted based on the 3-fold cross validation. As this dataset is highly imbalanced, the precision, recall, and F1 values are used as the evaluation metrics instead of accuracy. Table 6.6 shows the detailed results for each disaster concept. As can be seen from the table, the proposed deep network achieves very promising F1 scores in almost all classes. For example, lightening has the highest F1 score compared to the other classes, which can be due to its discriminative features (e.g., lights) in the corresponding images. Damage and snow classes have the lowest recall and F1 results respectively, because of their complex image texture and color. All in all, the average F1 score of binary classification on the disaster dataset is 0.914, which is higher than the previous work on this dataset.

6.3 Conclusion

In this chapter, we target multimedia data representation learning with the goal of handling large and imbalanced datasets. We study the advantages of utilizing
Table 6.6: Performance evaluation for different concepts on the disaster dataset

<table>
<thead>
<tr>
<th>Event</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flood</td>
<td>0.920</td>
<td>0.943</td>
<td>0.932</td>
</tr>
<tr>
<td>Damage</td>
<td>0.879</td>
<td>0.785</td>
<td>0.829</td>
</tr>
<tr>
<td>Fire</td>
<td>0.965</td>
<td>0.940</td>
<td>0.952</td>
</tr>
<tr>
<td>Mud-rock</td>
<td>0.971</td>
<td>0.923</td>
<td>0.947</td>
</tr>
<tr>
<td>Tornado</td>
<td>0.940</td>
<td>0.897</td>
<td>0.918</td>
</tr>
<tr>
<td>Lightening</td>
<td>0.979</td>
<td>0.968</td>
<td>0.973</td>
</tr>
<tr>
<td>Snow</td>
<td>0.914</td>
<td>0.798</td>
<td>0.849</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.938</td>
<td>0.893</td>
<td><strong>0.914</strong></td>
</tr>
</tbody>
</table>

deep learning and transfer learning techniques for feature analysis. First, multiple feature sets are extracted from the well-known deep learning algorithms. Thereafter, a novel ensemble deep classifier is developed to fuse different deep feature sets, as well as the results from each weak learner. This model alleviates the issue of imbalanced data, a very prevalent and unavoidable problem in real-world applications. The proposed model is extensively evaluated using two large-scale video datasets, namely a natural disaster dataset and the popular TRECVID dataset. The experimental analysis has been conducted to compare the performance of the proposed EDL model with the ones in other state-of-the-art machine learning algorithms. Specifically, its performance is compared with both handcrafted and deep features groups, integrated with several well-known classifiers. Based on the experimental results, the proposed model outperforms both groups of algorithms in two datasets with different concepts, which demonstrate its advantage effectiveness for video event detection.

In addition, this chapter presents a new deep learning technique called Deep Residual-Inception network, which not only improves the performance but also significantly speeds up the training and convergence processes. In summary, based on the experiments on two different multimedia datasets, the proposed Deep Residual-
inception network has shown its superiority and effectiveness while maintaining low computational costs.
As deep learning has been widespread in a wide range of applications, its training speed and convergence have become crucial. This chapter presents a new algorithm for hyper-parameter adaption in deep neural networks to enhance the training process. Specifically, an automatic drop-based learning rate scheduling is proposed to improve the SGD algorithm in deep learning. This work alleviates the task of selecting an appropriate learning rate by analyzing the statistical trends of the training process in an online manner. It automatically decides when to drop the learning rate based on the losses in the previous training iterations. The trivial computational costs of the trend analysis is ignorable compared to the gradient descent computation.

7.1 Trend-based Learning Rate Annealing

The proposed Trend-based Learning Rate Annealing (T-LRA) algorithm is applied to the CNNs on a classification task to evaluate its effectiveness. To our best knowledge, this is the first work that schedules the learning rate using statistical trend analysis. The advantages of this approach are: 1) automatic setting of a learning rate based on the previous training trends; 2) negligible computation over gradient descent; and 3) applicable to the deep neural networks and large datasets.

In this work, SGD is utilized for the training of deep learning networks and two well-known trend analysis techniques (i.e., the Mann-Kendall and Cox-Stuart tests) are leveraged for automatic learning rate adaptation as described in the following sections.
7.1.1 **Stochastic Gradient Descent**

Supervised learning is the most popular machine learning technique for either deep or shallow networks. In a supervised learning, an objective function measuring the error or distance between the actual and desired outputs is computed. The learning process includes adapting its internal parameters in order to minimize this error. Such adjustable parameters are also called “weights”. Deep neural networks may contain millions of weights which need to be updated during the training process. In order to update the weights properly, in each iteration, a gradient vector is computed which measures the error when the weight is increased by a very small factor. Then, using the opposite direction of the gradient vector, the weights are updated. If the gradient vector is negative, the direction of the steepest descent takes the objective function to the average low output error or its minimal point [47].

SGD is one of the most common procedures used to minimize the objective function, especially for neural networks and deep learning. In overall, it consists of computing the outputs, errors, and the average gradient for a few input examples. Accordingly, the weights are updated based on such information. After repeating this process for many small training sets, it stops when the error or loss stops decreasing. This process is surprisingly fast compared to its batch version which employs all training examples in each iteration [131].

A neural network consists of an input layer $X = \{x_1, ..., x_i, ..., x_N\}$ including $N$ input examples, the hidden layers containing $K$ neurons $H = \{h_1, ..., h_k, ..., h_K\}$, and an output layer including $M$ outputs $Y = \{y_1, ..., y_j, ..., y_M\}$. The neurons are connected to each other with $w_{ik}$ or $w_{kj}'$ which indicates the weights between the $i^{th}$ input and the $k^{th}$ hidden neuron or the $k^{th}$ hidden neuron and the $j^{th}$ output neuron. To simplify, the whole weights can be considered as the entries of a general weight matrix $W$. A simple version of this network is shown in Figure 7.1. An output of
a neuron $y_j$ is calculated using Equation (7.1) and the optimization function $L$ is defined in Equation (7.2) [190].

$$y_j = f\left(\sum w'_{kj} \ast h_k\right); \quad (7.1)$$

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} E(x_i) + \lambda r(W); \quad (7.2)$$

where $f$ is the activation function (e.g., sigmoid, tanh, or ReLU) which produces the non-linearity in a neural network. $E$ is the loss on data instance $x_i$ (e.g., $E = 1/2 \sum_{j=1}^{M} (y_j - t_j)^2$, where $t_j$ is the actual output and $y_j$ is the predicted one). The regularization term is $r(W)$ with the weight $\lambda$. Since the input size $|D|$ may be very large in practice, a stochastic approximation of the optimization function is used ($N \ll |D|$) as shown in Equation (7.2). The loss function $E$ is computed in the forward pass of the backpropagation neural network, while its gradient $\Delta E_w$ is calculated in the backward pass. In particular, SGD updates the weights using a linear combination of the previous weight update $V_t$ and the negative gradient $\Delta L(W_t)$ (given in Equation (7.4)), where $W_t$ and $W_{t+1}$ are the previous and updated weight matrices, respectively.

$$V_{t+1} = \mu V_t - \alpha \Delta L(W_t); \quad (7.3)$$

$$W_{t+1} = W_t + (V_{t+1}). \quad (7.4)$$

There are two important hyperparameters in the weight update formula that need to be assigned carefully: (1) the learning rate $\alpha$ or the negative gradient weight, and (2) the momentum $\mu$ or the weight of the previous update. Regarding the learning rate which heavily affects the network performance in practice, it has been shown that the drop-based techniques are slightly preferable and efficient for deep neural networks [1] [127]. Therefore, an automatic and fast learning rate schedule is proposed in this work, so that there is no need to manually select the number of training iterations in which the learning rate should be decreased.
7.1.2 Non-Parametric Trend Analysis

In a non-parametric trend test, no assumption of normality is required and a null hypothesis, $H_0$, is that the data population is identically distributed and comes from an independent population [191]. Since the errors in a deep neural network may not be normally distributed, two well-known non-parametric trend tests are utilized in this work, which are described in the following sections. In a hypothesis test, the $p$-value determines the statistical significance and plays a key role in interpreting the data statistics. To find a specific difference in an experiment, it is assumed that the null hypothesis $H_0$ is true. If the $p$-value is small (less than a significant value $\phi$), then the null hypothesis is rejected which shows a significant change in the data observations; while a large $p$-value (greater than $\phi$) indicates the acceptance of the $H_0$.

Mann-Kendall Trend Test

The Mann-Kendall is a popular non-parametric trend test commonly used to detect monotonic trends in series, especially for climate or environmental data [192] [193]. Suppose a set of observations are denoted as $\{x_1, x_2, ..., x_N\}$ ordered in time. The
Mann-Kendall statistic is given as Equation (7.5) [191, 194]:

\[
S = \sum_{k=1}^{N-1} \sum_{j=k+1}^{N} \text{sgn}(x_j - x_k)
\]  

(7.5)

where

\[
\text{sgn}(x_j - x_k) = \begin{cases} 
1, & \text{if } x_j - x_k > 0 \\
0, & \text{if } x_j - x_k = 0 \\
-1, & \text{if } x_j - x_k < 0 
\end{cases}
\]  

(7.6)

The mean of the Mann-Kendall statistic is \( E[S] = 0 \) and its variance \( \sigma^2 \) is calculated as:

\[
\sigma^2 = \frac{N(N-1)(2N+5) - \sum_{j=1}^{q} t_j(t_j - 1)(2t_j + 5)}{18}
\]

(7.7)

where the number of data points is denoted as \( N \), the number of the tied groups in the data set is \( q \), and the number of data points in the \( j^{th} \) tied group is \( t_j \).

Finally, the test statistic \( Z \) is calculated using \( S \) and \( \sigma^2 \) (given in Equation (7.8)):

\[
Z = \begin{cases} 
\frac{S - 1}{\sigma}, & \text{if } S > 0 \\
0, & \text{if } S = 0 \\
\frac{S + 1}{\sigma}, & \text{if } S < 0 
\end{cases}
\]  

(7.8)

Similar to other two-sided tests, the \( p \)-value in the Mann-Kendall represents the probability of the error regarding the null hypothesis \( H_0 \). This probability shows whether there is no trend or a significant change in the time series. If \( Z \) is negative (positive) and the probability is greater than the level of significance, there is a decreasing (increasing) trend in the data series, otherwise there is no significant trend.

**Cox-Stuart Trend Test**

Another trend analysis utilized in this work is a sign test proposed by Cox and Stuart [195]. This is a simple test for a monotonic (increasing or decreasing) trend
analysis. Using the notions of the previous section, the data observations are paired as:

\[ \{x_1, x_{c+1}\}, \{x_2, x_{c+2}\}, \ldots, \{x_{N-c}, x_N\}. \tag{7.9} \]

where \( c = \frac{N}{2} \) if \( N \) is even and \( c = \frac{N+1}{2} \) if \( N \) is odd. Then, a sign test is applied as follows. First, the differences between each pair is taken. In an increasing trend, it is expected most of the differences to be positive. On the contrary, a predominance of negative differences demonstrates a decreasing trend. Specifically, the Cox-Stuart test for \( N > 30 \) is calculated as \([191]\):

\[ Z = \frac{|Sg - \frac{N}{6}|}{\sqrt{\frac{N}{12}}} \tag{7.10} \]

where the maximum number of the signs is denoted as \( Sg \). Again, \( p\)-value represents the probability of the error regrading the selected significance level.

### 7.1.3 Online Learning Rate Schedule

In a training process, the loss function could be subjected to gradual changes or decay. In the drop-based techniques, it is desirable to drop the learning rate after several iterations. To automatically determine the number of iterations, an online learning rate schedule is proposed using the non-parametric trend analysis techniques. Algorithm 5 presents the overall training algorithm. The input of this algorithm contains the training input \( X \) and initial weight matrix \( W \) which will be iteratively updated using the SGD algorithm as shown in line 5. There are three hyperparameters: the learning rate \( \alpha \), the decay factor \( \theta \), and a new hyperparameter \( \varphi \) which will be initialized in this algorithm. Decay is the factor to be used for learning rate annealing. Similar to other learning parameters, the initialization values (\( \alpha_0 \) and \( \theta_0 \)) may be altered for different datasets. The \( \varphi \) is defined as the level of significance for the trend analysis.
Algorithm 5 Training deep neural network algorithm

1: procedure DPLearning(X,W) \Comment{Training the network}
2: \hspace{1em} $\alpha = \alpha_0$, $\theta = \theta_0$, $\varphi = \varphi_0$;
3: \hspace{1em} for all iterations $i \in (1, \cdots, I')$ do
4: \hspace{2em} for $j = 1$ to $I/I'$ do
5: \hspace{3em} $W = SGD(X,W,\alpha)$; \Comment{Update the weight}
6: \hspace{3em} $L[j] = ExtractLoss(W)$;
7: \hspace{1em} LRScheduler($L$, $\alpha$, $\theta$, $\varphi$);

After initialization, the training is started and the network weight is updated in each iteration. Suppose the network is going to be trained for $I$ iteration (e.g., 100,000). Thus, the whole learning process is divided into $I'$ steps (e.g., 20 steps with each 5,000 iterations). In each iteration, SGD is employed on the data using the current weight matrix $W$ and the learning rate value. Then the corresponding loss (error) is extracted (line 6 in Algorithm 5) as explained in Section 7.1.1. This process will be executed for $\frac{I}{I'}$ iterations.

After that, a scheduler is employed to update the hyperparameters ($\alpha$, $\theta$, and $\varphi$) as shown in Algorithm 6. This algorithm illustrates the whole procedure of the proposed online learning rate annealing. First, a time-series object $TS$ of the losses is created as shown in line 3 in Algorithm 6. This object is used as the input of the trend analysis function (e.g., Mann-Kendall or Cox-Stuart) in order to detect the trends in the losses curves. The learning rate is updated if there is no significant change in the losses or if the trend is positive, which means the losses are increasing. The significant level is defined by $\varphi$ and is updated whenever the learning rate is decreased. If the loss plateaus, the learning rate is divided by $\theta$. In addition, the decay factor $\theta$ is divided by 2 to reduce the learning rate decay over time. Similarly, the significance level $\varphi$ is multiplied by 2 as the loss changes are reduced exponentially over time. In other words, the losses are reduced very fast in the early iterations so the $p$-value should be very small (e.g., less than 0.05),
while in the last iterations where the loss curve is going to be flatten, the \( p\)-value threshold should be increased (e.g., 0.1).

Algorithm 6 The proposed online learning rate annealing algorithm

```
1: procedure LRScheduler\((L, \alpha, \theta, \varphi)\) \> Schedule the learning rate
2: \hspace{1em} SampleSet = L;
3: \hspace{1em} TS = TimeSeries(SampleSet);
4: \hspace{1em} Tr = MannKendall(TS); \> Tr=CoxStuart(TS)
5: \hspace{1em} if sign > 0 \textbf{ or } Tr.p-value > \varphi \textbf{ then}
6: \hspace{2em} \alpha = \frac{\alpha}{2};
7: \hspace{2em} \theta = \frac{\theta}{2};
8: \hspace{2em} \varphi = \varphi \ast 2;
```

### 7.2 Experimental Analysis

In this work, the main focus is to show the functionality and effectiveness of the proposed T-LRA on the stat-of-the-art algorithms and large public datasets. For this purpose, CNN is used as the learning model and the experiments are conducted on a challenging multimedia task, namely concept and image classification. SGD algorithm is highly used in CNNs to optimize the network and improve the losses. Therefore, the proposed T-LRA can automatically schedule the learning rate in CNNs.

The proposed T-LRA is a general learning rate scheduler that can be used for different classifiers and applications. Specifically, in this work, it is applied on a successful CNN architecture called Network In Network (NIN) \([4]\). This architecture is selected as it has shown very promising performance on image recognition and classification, while it has a very simple and straightforward architecture. The difference between NIN and the conventional CNNs is the utilization of advanced micro neural networks to abstract the data within the receptive field. In overall, The Deep NIN consists of the following layers (also shown in Figure 7.2). First, a
stack of mlpconv layers is used which replaces the Generalized Linear Model (GLM) in traditional CNNs with MultiLayer Perceptron (MLP) to convolve over the input. Then, oversampling layers are used after each mlpconv and followed by a dropout layer. The dropout layer can somehow prevent from overfitting in fully connected layers [55, 189]. Finally, the global average pooling and cost layers are added in the last layer. It is worth mentioning that a simple architecture of the NIN network is used to only focus on the functionality of the proposed learning rate scheduler. Therefore, other techniques such as augmentation, ensemble, and sampling have not been conducted in these experiments.

As mentioned earlier, the T-LRA algorithm can be run in parallel with the NIN algorithm. Every 5K iterations of the NIN training, T-LRA is called and calculates the trend in the current training curve. If a significant trend is observed in the curve, the learning rate remains constant, otherwise it is updated as explained in Algorithm 6. The 5K-iteration criterion is selected because enough data (losses in this work) is needed for a statistical test. In addition, based on the experiments, it is an adequate and reasonable range to find significant changes in the training curve.

The proposed T-LRA is evaluated on CIFAR-10 [3]. As mentioned earlier, this dataset includes 60,000 images (50,000 for training and 10,000 for testing) with labels and is used for image classification and object recognition. It is also composed of 10 classes as shown in Figure 6.10. The main challenge in this dataset is that it...
includes tiny color images with $32 \times 32$ resolution, taken from the dataset of 80 million tiny images [196]. Therefore, it may be thought it does not include enough information to detect objects effectively. However, this public dataset is used for many multimedia and computer vision competitions every year and even a small improvement in accuracy (e.g., 1%) can distinguish the proposed model.

Several experiments have been conducted to demonstrate the effectiveness of the proposed T-LRA. Both Cox-Stuart and Mann-Kendall trend analysis techniques are used to schedule the learning rate. The results are also compared to the benchmark technique with the original SGD. Although there are other learning rate scheduling techniques (e.g., drop-based or time-based schedules), they are not used as the comparison benchmarks because this is a subjective problem and there is no specific rule when the learning rate should be dropped. Different studies use different epochs to reduce or increase the rate which is completely based on trial and error. This is another reason why an automatic scheduler is needed to handle this issue in a general manner. The evaluation criteria include common metrics such as accuracy and losses (as explained in Section 7.1.1 Equation (7.2)).

Caffe is used to train the NIN network on the CIFAR-10 dataset. The NIN network is trained on CPU mode using 6 servers with 64 processor.

The network input for this dataset is $32 \times 32$ with mean subtraction. The solver parameters include the base learning rate $\alpha_0$ of 0.01 and SGD with momentum 0.9. In total, the network is trained for more than 100,000 iterations. The initial decay factor $\theta$ is selected as 10, which means the learning rate is divided by 10 in the first steps, while this factor is reduced gradually (i.e., divided by 2 each time when $\alpha$ changes) to lessen the effects of the learning rate changes in the final stages. Another important parameter is the significance level $\phi$ which controls the range of p-value in the trend analysis algorithms. It is initialized to 0.05 and increased gradually due
to the significant loss changes in early iterations and small loss changes in the final iterations.

Figure 7.3 shows the behavior of the training and testing in three methods. The first model is the original SGD without the learning rate scheduling algorithm. The second and third models are the proposed learning rate annealing based on the Cox-Stuart and Mann-Kendall trend analysis techniques, respectively. The first plot (Figure 7.3 (a)) shows the training losses of these three models for more than 70,000 iterations. As can be inferred from this plot, both Mann-Kendall and Cox-Stuart have lower training losses compared to the original SGD algorithm. Specifically, Cox-Stuart losses are decreased around 30K iterations compared to both original and Mann-Kendall approaches, while Mann-Kendall training losses are suddenly decreased around 40K iterations and stays as low as possible compared to Cox-Stuart and the original one. On the other hand, the test losses plot (Figure 7.3 (b)), shows the supremacy of the Cox-Stuart than Mann-Kendall as it has lower losses, especially after 30K iterations. In this plot, though, the average losses for Mann-Kendall are much smaller than the original algorithm. Finally, the last plot (Figure 7.3 (c)) shows the comparison between the test accuracy of each algorithm in different iterations. As can be seen from this plot, all these three algorithms have the same accuracy in the first steps. At 30K iterations, the Cox-Stuart p-value is no longer smaller than the selected significance level (e.g., 0.05) which causes a drop in the learning rate value. This drop improves the accuracy from 0.844 to 0.860 as shown in the second row of Table 7.1. Around 40K iterations, a similar case happens for Mann-kendall where the accuracy increases significantly from 0.834 to 0.863 as depicted in both Figure 7.3 (c) and the third row of Table 7.1. It can be seen that the original SGD does not have this improvement even after many iterations because it may be trapped in a local minimum. The final results during the 70K iterations
Figure 7.3: Performance comparison of original SGD and T-LRA using Cox-Stuart and Mann-Kendall
Table 7.1: Accuracy comparison on CIFAR-10 for different iterations and two different values for the significance factor \((\varphi_0)\)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>30K</th>
<th>40K</th>
<th>50K</th>
<th>60K</th>
<th>70K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.844</td>
<td>0.834</td>
<td>0.845</td>
<td>0.846</td>
<td>0.845</td>
</tr>
<tr>
<td>Cox-Stuart ((\varphi_0 = 0.05))</td>
<td>0.844</td>
<td>0.860</td>
<td>0.861</td>
<td>0.862</td>
<td>0.862</td>
</tr>
<tr>
<td>Mann-Kendall ((\varphi_0 = 0.05))</td>
<td>0.844</td>
<td>0.834</td>
<td>0.863</td>
<td>0.867</td>
<td>0.867</td>
</tr>
<tr>
<td>Mann-Kendall ((\varphi_0 = 0.1))</td>
<td>0.844</td>
<td>0.834</td>
<td>0.845</td>
<td>0.867</td>
<td>0.870</td>
</tr>
</tbody>
</table>

are shown in Table 7.1. The accuracy of the proposed learning rate annealing could reach to 0.862 and 0.867 using the Cox-Stuart and the Mann-Kendall trend analysis techniques, respectively.

One improvement is to increase the initial value of the significance factor \(\varphi_0\) from 0.05 to 0.1 since the loss trend usually changes very slowly compared to the other trend analysis applications (e.g., environmental changes). However, this improvement causes more computational cost because it needs more iterations to reach its maximum performance. To show the effects of the significance factor, another experiment is conducted using the NIN network and the Mann-Kendall trend analysis (because it shows a higher accuracy than the Cox-Stuart one in the previous results). For this purpose, \(\varphi_0\) is initialized to 0.1 and multiplied by 2 every time the learning rate is decreased. The results are shown in the last row of Table 7.1.

As can be seen from this table, the proposed algorithm has similar results as the original SGD until iteration 50K. Then, it shows a significant increase around 60K and finally reaches to 0.87 at 70K, which is higher than all the other methods in this iteration. The learning rate annealing is executed every 5K iterations during the training of the deep network. Finally, the maximum accuracy of 0.879 is obtained, which is 3.3\% higher than the one in the SGD algorithm.
The aforementioned results show the effectiveness of the proposed learning rate annealing compared to the algorithms that do not leverage any scheduling method. In addition, it automates the process of learning rate modifications, which means it can reach to the highest accuracy without manually changing the hyper-parameters. Moreover, due to the very light processing of the proposed algorithm, its computational time can be completely disregarded compared to the heavy SGD costs. Therefore, it can be easily integrated with online algorithms and applications. Regarding the efficiency of the proposed method, as explained in Table 7.1, the proposed method can achieve a higher performance in fewer iterations. For example in 50k iterations, the accuracy of 0.863 is obtained using the Mann-Kendall trend analysis with $\varphi_0 = 0.05$; while the original SGD achieves the accuracy of 0.845 and can only increase it by 0.1% after 10k iterations. Moreover, for the proposed method, the accuracy is increased by 0.3% after 10k iterations, which shows the effectiveness and efficiency of the proposed method.

7.3 Conclusion

This chapter presents a novel learning rate annealing (scheduling) using two light and efficient trend analysis approaches (namely, Cox-Stuart and Mann-Kendall). This automatic and online drop-based technique reduces the learning rate value gradually to avoid trapping in a local minimum in a training loss curve, where there may exist a global minimum. Specifically, it is applied on a popular deep learning architecture called Network In Network and evaluated using a public large-scale image dataset called CIFAR-10. The proposed algorithm improves the results of the original stochastic gradient descent, used in many learning algorithms such as backpropagation. In overall, the classification accuracy on the testing data is
increased by 3.3% compared to the SGD algorithm. In addition, the proposed algorithm reaches the highest accuracy in a smaller number of iterations and reduces the computational costs of training.
CHAPTER 8

DEEP SPATIO-TEMPORAL LEARNING

This chapter presents new solutions for analyzing spatio-temporal multimedia data. Specifically, a new spatio-temporal model for large-scale and imbalanced video classification using deep learning is proposed [141]. This work introduces a new spatio-temporal model for video analysis. First, spatial information is extracted from the video sequences using the pre-trained CNNs. Thereafter, these sequences are fed to the proposed two-layer residual bidirectional LSTM, and finally the video classes are predicted in the final fully connected layer. The experimental results demonstrate the ability of the proposed model with respect to the prediction performance and efficiency.

The main challenges in video classification are threefold: (1) There are large variations between the frames throughout the whole video (for example, the existence of various objects and scenes in one video such as tree, building, human, and water in a disaster event), (2) There are a large number of frames needed to be processed for each video, (3) The video data is multimodal and spatio-temporal in nature. Due to all these challenges, video content analysis and classification is a complex and big data problem requiring accurate and efficient learning models.

With the advent of deep learning, new methodologies have been proposed to address the problem of large-scale video classification [109]. Specifically, Convolutional Neural Networks (CNNs) [55] and Recurrent Neural Networks (RNNs) [197] are employed for modeling static and temporal information. Despite the great success of deep neural networks in visual data classification, there remain challenges and rooms for improvement. To address these challenges, this chapter presents a new deep learning model that extracts static and temporal information from videos and reduces the overall training process using transfer learning.
8.1 Spatio-Temporal Representation Learning

8.1.1 Transfer Learning with Deep CNNs

As described in chapter 6, several deep learning architectures like Inception-v3 [198] and ResNet [1] have been widely used as pre-trained CNNs for image processing applications. These networks were originally trained on a very large-scale dataset “ImageNet” for several weeks and are widely used to extract generic appearance features in different applications. We have shown the effectiveness of deep features compared to the traditional handcrafted features [143]. In addition, utilizing pre-trained models can significantly expedite the whole training process on the new dataset. Depending on the target dataset and its similarity to the source dataset, the pre-trained CNNs can be truncated in various layers.

8.1.2 LSTM

LSTM networks have internal memory cells which are able to learn the long-term dependencies of sequential frames. In addition, they overcome exploding gradients in the temporal domain (vanishing problem) by providing temporal shortcut paths. Due to the simple input concatenation and activation applied in RNNs, it can remember information for a short time. Different from RNNs, LSTMs have a more complex structure assisting them to remember information for a longer period of time. As shown in Figure 8.1 (a), when a new information arrives, the input gate $i_t$, forget gate $f_t$, output gate $o_t$, and memory cell $c_t$ in the LSTM cell handle the information overwriting by comparing it with the inner memory. LSTM gates are designed to control the forgetting, updating, and remembering processes and enable
gradients to smoothly flow through time. As a result, only the information that is needed are selectively passed.

Let $\sigma$ be the sigmoid non-linearity which squashes the inputs to a range between $[0, 1]$, and $\tanh(x)$ be the hyperbolic tangent non-linearity which squashes its input $x$ to a range between $[-1, 1]$. The LSTM parameter updates at time step $t$ given inputs $x_t, h_t, c_t$ are defined as follows \[136\]:

\[
\begin{align*}
    i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i); \\
    f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f); \\
    c_t &= f_t.c_{t-1} + i_t.tanh(W_c[h_{t-1}, x_t] + b_c); \\
    o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o); \\
    h_t &= o_t.tanh(c_t).
\end{align*}
\]

where $W_k$ and $b_k$ refer to the weight and bias of $k = \{i, f, c, o\}$, respectively. In order to gradually learn the connections of input $i_t$, forget $f_t$, and output $o_t$ gates, they are component-wise multiplied by the input, hidden output, and memory cell.

### 8.1.3 Bidirectional LSTM

The original LSTMs have one direction and predict the output based only on previous information. Hence, some information may be lost in a one-directional network. Similar to human trajectories, Bidirectional LSTMs (BiLSTMs) are continuous and consider both former and subsequent information. As a result, it can capture bidirectional global temporal information in video sequences. Figure 8.1(b) illustrates a BiLSTM in which the input set is defined as $x = \{x_0, x_1, ..., x_t, x_{t+1}\}$ and the output set as $y = \{y_0, y_1, ..., y_t, y_{t+1}\}$ and the hidden layer as $h = \{h_0, h_1, ..., h_t, h_{t+1}\}$. In the hidden layers, there are forward sequences $\rightarrow$ and backward sequences $\leftarrow$. The
Figure 8.1: The architectures of (a) the LSTM cell and (b) unfold Bidirectional LSTM.

The parameters of BiLSTM at time $t$ can be defined as follows [199]:

$$h^+ = g(U_{h^+}x_t + W_{h^+} + b_{h^+}); \quad (8.6)$$

$$h^- = g(U_{h^-}x_t + W_{h^-} + b_{h^-}); \quad (8.7)$$

$$y_t = g(V_{h^+}h^+ + V_{h^-}h^- + b_y). \quad (8.8)$$

where $g$ is an activation function such as ReLu ($g(a) = \text{Max}(0, a)$), $U$ refers to the weight matrix from the input to the hidden layers, $W$ is the weight from the hidden to the hidden layers, $V$ denotes the weight from the hidden to the output layers, and $b_s$ denotes the bias of $s = \{h^-, h^+, y\}$.

### 8.1.4 CNN-Residual Bidirectional LSTM

The proposed deep learning model includes spatial, temporal, and prediction components. The video input flows in the spatial dimension (vertical direction) and temporal dimension (horizontal direction) and the corresponding classes are detected in the last prediction layer. In the spatial component, deep CNN features are extracted for every frame from every video using transfer learning and converted into the sequences of extracted features. As described in chapter [6] several deep
learning architectures like Inception-v3 \cite{inception} and ResNet \cite{resnet} have been widely used as pre-trained CNNs for image processing applications. These networks were originally trained on a very large-scale dataset “ImageNet” for several weeks and are widely used to extract generic appearance features in different applications. We have shown the effectiveness of deep features compared to the traditional handcrafted features \cite{handcraft}. In addition, utilizing pre-trained models can significantly expedite the whole training process on the new dataset. Depending on the target dataset and its similarity to the source dataset, the pre-trained CNNs can be truncated in various layers.

In the temporal component, the CNN feature sequences are fed into the proposed residual bidirectional LSTM as the time series to preserve the continuous temporal information. Residual connections can overcome the gradient transmission by forwarding the information from the upper layers directly through the network using an “addition” operator \cite{residual}. This simple connection can significantly improve the training process since the lower information can transmit to the upper layer directly through a highway. The residual connection provides not only the temporal shortcut paths but also an additional spatial shortcut path for efficient training of deep LSTM networks. Therefore, it gives a flexibility to the LSTM cells to deal with the vanishing or exploding gradients. Different from original LSTM, residual LSTM adds a shortcut path to the output layer $h_t$ instead of accumulating a highway path on an internal memory cell $c_t$. The shortcut can be the output of any lower layers, though the exact previous output of Bidirectional LSTM is used in this research.
Then the network parameters are updated as follows:

\[ h_0 = \sigma(W_0 x + b_0); \quad (8.9) \]

\[ h_l = \sigma(W_l h_{l-1} + b_l) + h_{l-1}; \quad (8.10) \]

\[ y = \sigma(W_y h_{L-1} + b_y) + h_{L-1}. \quad (8.11) \]

where \( l = \{1, 2, ..., L - 1\} \) and \( L \) is the total number of residual layers. In this work, we use two residual layers (i.e., \( L = 2 \)).

The proposed model can access and discover more information in advance due to its backward passes and also can avoid overfitting and vanishing gradients due to its residual connection. In this study, a two-layer residual bidirectional LSTM is designed \((L = 2)\), followed by a batch normalization which is connected to the last element from its previous layer. In the final temporal layer, only the last element of the output is selected and batch normalization is applied because it normalizes the input across a mini-batch and generates simpler feature representations in the hidden layers. Therefore, it overcomes gradient vanishing and prevents outliers at the test time. In addition, L2 regularization is utilized to generalize the model and to reduce overfitting to the training data. More specifically, each parameter of the objective function is penalized by its squared magnitude as follows:

\[
E(W) = \frac{1}{2} \sum_{n=0}^{N-1} (t_n - y(x_n, W))^2 + \frac{\lambda}{2} \| W \|^2 \quad (8.12)
\]

where \( E(W) \) is the objective function, \( t_n \) is the actual class value of the \( n^{th} \) instance in the training batch, \( N \) is the total number of instances, and \( y \) is the output based on input \( x_n \) and weight \( W \). The last term is the L2 regularization term including a penalty weight of \( \lambda \).

Dropout is also directly added to each bidirectional LSTM layer. Dropout is a regularization technique which randomly ignores some neurons during the training,
and so their contribution to the activation is temporarily deactivated. As a result, we can significantly prevent overfitting. Finally, the prediction component includes two fully connected layers and a dropout in between, which generates the final classes.

8.1.5 Experimental Analysis

In this work, the proposed model is applied to a large-scale video dataset to evaluate its performance. Specifically, the public UCF101 action recognition dataset [49] is used that contains 13,320 videos with 101 action categories. UCF101 is one of the most challenging datasets due to its diversity in terms of actions, views, background, camera motion, etc. It is also selected because of its popularity in the literature and many available benchmarks that can be used to evaluate our model. The first train/test split of this dataset suggested by the reference website is used in this experiment.

In the preprocessing step, we first extract all the frames form each video. Thereafter, we extract the features of every video frame through the last pooling layer of Inception-v3, resulting in a feature set with 2048 dimensions. These extracted features are later grouped into sequences. For the sake of simplicity and similar to the experiments in [161], alpha is selected as 40. In other words, we turn each video into a 40-frame sequence. For temporal analysis, a two-layer Residual Bidirectional LSTM with 1024-wide followed by a 1024 fully connected layer and 50% dropout is used. This relatively shallow network outperforms other deep stacked Residual Bidirectional LSTM models. We use Adam stochastic optimization with an aggressively small learning rate 0.000001 and L2 regularization with $\lambda = 0.0003$.

As mentioned earlier, we do not extract any extra engineering features such as motion and dense trajectories to reduce the complexity and enhance the efficiency.
of the model. Thus, the proposed model is compared with the existing work without the optical flow and iDT features. Since UCF101 has relatively a great balance of training data in all classes, accuracy and top-5 accuracy are used as the evaluation metrics.

Table 8.1 summarizes the experimental evaluation with the comparison against the state-of-the-art models on UCF-101. The comparison models include: (1) a simple CNN-MLP model [161] in which the features are extracted from each frame with Inception-v3 and then the sequences are passed to a Multi-layer Perceptron (MLP), (2) a 10-dimension 3D convolutional network [111], (3) a single frame CNN using the pre-trained CaffeNet and applying a linear Support Vector Machine (SVM) for classification [111], (4) a slow fusion CNN model that slowly integrates temporal information with the spatial one through the network [109], (5) a deep spatio-temporal model called Long-term Recurrent Convolutional Networks (LRCN) based on CNN and LSTM [115], and (6) a single frame CNN and LSTM [52]. All these methods are based on RGB frames only and no optical flow or iDT features are used. As can be inferred from this table, our proposed model (CNN-ResBiLSTM) improves the classification performance (accuracy) compared to all other models. It is worth mentioning that, although the existing work did not usually mention the speed of training their models, it is obvious that the models utilizing transfer learning for feature analysis perform much faster than those trained from scratch on the raw data.

To further demonstrate the effectiveness of the proposed residual bidirectional LSTM, we conduct several experiments on the UCF101 dataset as shown in Table 8.2. These results show the importance of bidirectional and residual connections combined with LSTM in spatio-temporal video analysis. The comparison models include: (1) a frame-based CNN and softmax for generating final classes. This model
Table 8.1: Classification results on UCF101. The proposed CNN-ResBiLSTM compared with baselines and state-of-the-art models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Distributed CNN+RNN (end to end) [161]</td>
<td>20.0</td>
</tr>
<tr>
<td>CNN-MLP [161]</td>
<td>70.0</td>
</tr>
<tr>
<td>C3D [111]</td>
<td>52.8</td>
</tr>
<tr>
<td>Frame-based CNN + linear SVM [111]</td>
<td>68.8</td>
</tr>
<tr>
<td>Slow Fusion CNN [109]</td>
<td>65.4</td>
</tr>
<tr>
<td>LRCN [115]</td>
<td>71.1</td>
</tr>
<tr>
<td>Frame-based CNN+LSTM [52]</td>
<td>73.3</td>
</tr>
<tr>
<td>CNN-ResBiLSTM</td>
<td><strong>77.6</strong></td>
</tr>
</tbody>
</table>

only considers static features in single frames and ignores the temporal information between the frame sequences. We fine-tune Inception-v3 by freezing the top layers of the network and updating the weights in only the final layers. This simple model surprisingly generates a promising performance compared to the complex models such as C3D. (2) a model based on the CNN features and a simple LSTM. Although this model utilizes the temporal information using LSTM cells, it cannot improve the results of the frame-based model. (3) adding residual connections to the previous model. This model cannot increase the performance notably. (4) a model with bidirectional connections. This model can enhance the accuracy by 2%. (5) fusing the CNN Bidirectional LSTM and a simple CNN MLP to further boost the performance. However, as can be seen from the table, the fusion cannot help in this case. Finally, our proposed model (CNN-ResBiLSTM) can increase both accuracy and Top-5 accuracy compared to the best comparison result (CNN-BiLSTM) by almost 3.4% and 2.7%, respectively, which is promising for this dataset.

Figure 8.2 (a-b) visualizes the loss and accuracy comparison of each model during the training process. It can be inferred from these plots that the proposed method (ResBiLSTM) can converge faster than the other benchmarks and generate lower
Table 8.2: Performance evaluation results on UCF101 using different techniques based on our experimental setup.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Acc (%)</th>
<th>Top-5 Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame-based CNN (IncV3) + Softmax</td>
<td>73.4</td>
<td>91.6</td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>72.2</td>
<td>92.2</td>
</tr>
<tr>
<td>CNN-ResLSTM</td>
<td>72.5</td>
<td>91.6</td>
</tr>
<tr>
<td>CNN-BiLSTM</td>
<td>74.2</td>
<td>91.4</td>
</tr>
<tr>
<td>CNN BiLSTM+ MLP Fusion</td>
<td>73.6</td>
<td>91.4</td>
</tr>
<tr>
<td>CNN-ResBiLSTM</td>
<td>77.6</td>
<td>94.1</td>
</tr>
</tbody>
</table>

losses and higher accuracies in almost all the iterations. The LSTM model has the slowest convergence while BiLSTM and ResLSTM can lessen this problem of LSTM. Finally, the proposed model can learn forward and backward connections in each video sequence and leverage the temporal shortcut paths to expedite the training convergence and reach to the higher performance faster.

8.2 An Integrated Spatio-Temporal Sampling and Deep Learning Model

In chapter 5, a spatio-temporal video sampling model is proposed for imbalanced data classification. We combine the idea of our sampling model with the proposed spatio-temporal deep learning model to consider both spatial and temporal information in multimedia data while handling imbalanced data problem. Although initial work has been done in these two directions, more efforts need to be dedicated to integrate the existing solutions. Therefore, we propose to integrate these two models as shown in Figure 8.3. The model will include spatio-temporal synthetic oversampling, spatial, temporal, and prediction components. First, the video oversampling
will be employed to overcome the imbalanced data problem, and then both static and temporal features will be extracted from multimedia data using pre-trained CNNs and ResBiLSTM. Specifically, the proposed oversampling algorithm will be extended as shown in Algorithm 7.

In addition to the new oversampled videos $\hat{V} = \{\hat{v}_{i,j,fr}|i = 1, \cdots, N; j = 1, \cdots, M; fr = 1, \cdots, freq_j\}$, generated by the previous algorithm in chapter 5, this time the sequences of spatial features $\text{Sequences} = \{\text{Seq}_{i,j,fr}|i = 1, \cdots, N; j = 1, \cdots, M; fr = 1, \cdots, freq_j\}$ are generated where $\text{Seq}_{i,j,fr}$ is the feature sequence related to the oversampled video $\hat{v}_{i,j,fr}$. The static features are generated for the
Figure 8.3: The proposed integrated spatio-temporal deep learning model for imbalanced video classification
corresponding image using the pre-trained models as explained in chapter 6. These frames generated for each video are stitched together as a sequence $Seq_{i,j,fr}$ to be easily used in the next layers for video temporal analysis. Using this technique, the deep features are extracted once for each frame and may be used several times through the training process. Therefore, there is no need to continuously pass the original images through the CNN every time the same frame is read. The spatio-temporal synthetic oversampling algorithm returns these sequences to be used as the input of the temporal deep learning model.

### 8.2.1 Experimental Analysis

In order to demonstrate the effectiveness of the proposed integrated model, we conduct some preliminary experiments similar to the one shown in chapter 5 for disaster video dataset and imbalanced UCF101. Tables 8.3 and 8.4 summarize the experimental results. The last two rows show the results of the proposed integrated model without and with class weighting, respectively. It can be seen from the tables, the proposed CNN-ResBiLSTM together with the proposed video oversampling further improves the results and reaches to 70% accuracy for disaster dataset and 72% for UCF101 dataset. Specifically for the disaster dataset, compared to the original CNN-LSTM, the proposed techniques can enhance the accuracy and F1 measure by more than 11% and 0.17, respectively. Similarly, the accuracy and F1 measure are improved about 4% for UCF101.

### 8.3 Conclusion

This chapter presents a new spatio-temporal model for large-scale video classification using deep learning. The model introduces a new spatio-temporal deep learning
Algorithm 7 The proposed spatio-temporal synthetic oversampling algorithm for an imbalanced video dataset

**Input:** Original training video dataset $V = \{v_{i,j}|i = 1,\ldots,N; j = 1,\ldots,M\}$, Class list $CL = \{cl_j|j = 1,\ldots,M\}$, Maximum number of video samples $\delta$, and sequence size $\alpha$.

**Output:** Oversampled video dataset $\hat{V} = \{\hat{v}_{i,j,fr}|i = 1,\ldots,N; j = 1,\ldots,M; fr = 1,\ldots,freq\}$ and Sequences of spatial features $\text{Sequences} = \{Seq_{i,j,fr}|i = 1,\ldots,N; j = 1,\ldots,M; fr = 1,\ldots,freq\}$.

1: $\hat{V} \leftarrow \{\}; \text{Sequences} \leftarrow \{\}$
2: for all class $cl_j \in CL$ do
3: \hspace{1em} $freq_j \leftarrow \left\lceil \frac{\delta}{nv_j} \right\rceil$
4: \hspace{2em} for all video $v_{i,j} \in V$ do
5: \hspace{3em} $frm_{i,j} \leftarrow \text{GETFRAMES}(v_{i,j})$
6: \hspace{2em} \hspace{1em} for $fr = 1,\ldots,freq_j$ do
7: \hspace{3em} \hspace{2em} \hspace{1em} if $\text{SizeOf}(frames_{i,j}) > \alpha$ then
8: \hspace{3em} \hspace{3em} \hspace{2em} $frm_{i,j} \leftarrow \text{RANDDOWN}(frm_{i,j})$
9: \hspace{3em} \hspace{2em} \hspace{2em} else
10: \hspace{3em} \hspace{3em} \hspace{2em} $frm_{i,j} \leftarrow \text{UPSAMPLE}(frm_{i,j})$
11: \hspace{3em} \hspace{2em} \hspace{2em} end if
12: \hspace{3em} \hspace{2em} $Seq_{i,j,fr} \leftarrow \{\}$
13: \hspace{3em} \hspace{2em} $\hat{v}_{i,j,fr} \leftarrow \{\}$
14: \hspace{3em} \hspace{2em} for all $Img \in frm_{i,j}$ do
15: \hspace{3em} \hspace{3em} $\hat{Img} \leftarrow \text{RANDAUG}(Img)$
16: \hspace{3em} \hspace{3em} $\hat{v}_{i,j,fr} \leftarrow \hat{v}_{i,j,fr} + \hat{Img}$
17: \hspace{3em} \hspace{3em} $\text{features} \leftarrow \text{SPATIALFEATURES}(\hat{Img})$
18: \hspace{3em} \hspace{3em} $Seq_{i,j,fr} \leftarrow Seq_{i,j,fr} + \text{features}$
19: \hspace{3em} end for
20: \hspace{2em} $\hat{V} \leftarrow \hat{V} + \hat{v}_{i,j,fr}$
21: \hspace{2em} $\text{Sequences} \leftarrow \text{Sequences} + Seq_{i,j,fr}$
22: end for
23: end for
24: return $\hat{V}, \text{Sequences}$
Table 8.3: Performance evaluation results on Disaster dataset.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Acc</th>
<th>F1</th>
<th>Weighted F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No video oversampling</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.589</td>
<td>0.339</td>
<td>0.526</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.663</td>
<td>0.428</td>
<td>0.654</td>
</tr>
<tr>
<td><strong>With video oversampling</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.671</td>
<td>0.456</td>
<td>0.662</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.678</td>
<td>0.477</td>
<td>0.688</td>
</tr>
<tr>
<td><strong>Proposed model</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-ResBiLSTM</td>
<td>0.681</td>
<td>0.493</td>
<td>0.678</td>
</tr>
<tr>
<td>CNN-ResBiLSTM+ class weighting</td>
<td>0.700</td>
<td>0.513</td>
<td>0.706</td>
</tr>
</tbody>
</table>

Table 8.4: Performance evaluation results on imbalanced UCF101.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Acc</th>
<th>F1</th>
<th>Weighted F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No video oversampling</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.685</td>
<td>0.655</td>
<td>0.670</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.680</td>
<td>0.660</td>
<td>0.670</td>
</tr>
<tr>
<td><strong>With spatio-temporal video oversampling</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>0.706</td>
<td>0.684</td>
<td>0.696</td>
</tr>
<tr>
<td>CNN-LSTM+ class weighting</td>
<td>0.690</td>
<td>0.669</td>
<td>0.679</td>
</tr>
<tr>
<td><strong>Proposed model</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN-ResBiLSTM</td>
<td>0.723</td>
<td>0.702</td>
<td>0.717</td>
</tr>
<tr>
<td>CNN-ResBiLSTM+ class weighting</td>
<td>0.705</td>
<td>0.686</td>
<td>0.696</td>
</tr>
</tbody>
</table>
method using CNN and LSTM models. First, the spatial information is extracted from the video sequences using the pre-trained CNNs. Thereafter, these sequences are fed to the proposed two-layer residual bidirectional LSTM, and finally the video classes are predicted in the final fully connected layer. In addition, this model is integrated with the proposed dynamic sampling to enhance the classification performance especially for minor or complicated classes. The experimental results demonstrate the ability of the proposed model with respect to the prediction performance and efficiency.
In the previous chapters, we mainly focus on visual (image/video) data. However, other data modalities may contain valuable information which can be utilized in the framework. Despite the great success of deep learning in the processing of single data modalities, there are still a few research studies focusing on multimodal deep learning frameworks [149, 200]. This problem is mainly due to the limited available datasets that contain multiple data modalities including text, audio, video, etc.

In addition, many real-world data samples can be represented with multiple labels. For example, an image may contain multiple objects or a video may contain various events. In such cases, the data samples cannot be easily categorized by a single class. Therefore, Multi-Label Classification (MLC) is a necessity to solve these problems. In MLC, different from single-label classification, each instance is assigned to multiple labels simultaneously. Due to the high dimensionality of the data, the enormous number of label combinations, and the complex correlation between the labels, MLC is more challenging than a single-label classification problem. Besides, for a multimedia dataset containing multiple data types, it is essential to discover the correlation between both labels and data modalities.

Considering all these challenges, in this chapter, we present a new model for multi-label multimodal data classification using advanced deep neural networks. In addition, we consider the imbalanced data problem to further enhance the detection performance for both minority and majority classes. This model is specifically evaluated on a multimodal dataset designed for natural disaster information retrieval and management. However, it can be easily extended for other multimodal multi-label datasets. The contributions of this work include: (1) deep feature extraction using spatio-temporal deep learning models for each modality (text, audio, and im-
age); (2) a new fusion technique which considers the relation between both labels and data modalities while considering the imbalanced data problem; (3) a modified disaster-based video dataset which is designed for multi-label multimodal video classification.

The proposed multi-label multimodal deep learning model is shown in Figure 9.1. The input of our model includes disaster videos which contain visual and audio clips as well as text descriptions. For each data modality, we extract static features using the state-of-the-art pre-trained deep learning models. In the next step, temporal features are extracted using the advanced Recurrent Neural Networks (RNNs). Then, in the fusion module, we concatenate the features from each modality and apply a Random Forest feature selection to remove the irrelevant features. Finally, the selected features are used as the input of the multi-label multimodal weighted Support Vector Machine (SVM) to generate the final classification results.

9.1 Static Feature Extraction Module

Static feature sets include visual, audio, and text features as explained below.

9.1.1 Visual Feature Extraction

In a video classification, visual data play an important role in detecting various concepts. In this research, we used a pre-trained deep learning model to extract the visual features from video clips. First, each video clip is subsampled to $\alpha$ frames. To do so, video clips with more than $\alpha$ frames are essentially down-sampled, while those with less than $\alpha$ frames are upsampled. For example, both videos with $\alpha + 1$ and $\alpha + 500$ frames will be reduced to $\alpha$. Similar to our previous work [14], downsampling and up-sampling are done in a random manner. First, a number $\text{skip/rep}$
Figure 9.1: The proposed multi-label multimodal deep learning model.
is generated \((\text{skip/rep} = \frac{\text{frame-length}}{\alpha})\). For down-sampling, a rescaled list of video frames is returned by generating a random number for each \(\text{skip}\) range and selecting the corresponding frame for that range. On the other hand, for up-sampling, when the number of frames in a video is less than \(\alpha\), each frame is repeated \(\text{rep}\) times to generate the rescaled \(\alpha\) frame list. This step is necessary to have a fixed-size input for the temporal model.

In the next step, spatial features are automatically extracted using a popular pre-trained model called Inception-v3 [198]. This CNN-based model is an advanced and efficient version of the original Inception architecture proposed by Google in 2014. In particular, it factorizes convolutions into smaller convolutions (e.g., traditional \(7 \times 7\) convolutions are factorized into \(3 \times 3\) convolutions) and also adds batch normalization to the fully connected layer of the auxiliary classifier. Inception-v3 is originally trained on ImageNet [55]. The features generated by transfer learning based on the pre-trained models can be used for smaller but similar datasets. Transfer learning can significantly speed-up the training process in deep learning compared to when the model is completely trained from scratch using the new dataset. In addition, it alleviates the necessity of having a very large-scale labeled dataset for training the deep learning models. Depending on the similarity between the source dataset (e.g., ImageNet) and target dataset (e.g., disaster dataset), different layers of the pre-trained model can be used for feature extraction. In this work, Inception-v3 is truncated in the last average pooling layer to generate the features for each frame. Thereafter, each video clip is turned into an \(\alpha\)-frames sequence of deep features. The spatial feature extraction is only done once in an offline manner, and its results can be used many times in the training process. Thus, it is not necessary to pass the original raw images through the network every time they are read. We used the last average pooling layer of Inception-V3 for feature extraction.
9.1.2 Audio Feature Extraction

Audio features are extracted through SoundNet [201], a pre-trained model of natural sound representations. It utilizes a student-teacher training procedure and learns the acoustic representations from the unlabeled videos by leveraging the discriminative cross-modalities knowledge between the visual and audio data. The audio features are learned from the image features extracted from ImageNet [55] and Places [202]. It consists of several convolutional layers and pooling layers. In our proposed framework, the eight-layer model is used as the feature extractor, and the features are extracted from the \textit{conv7} layer, which is the convolutional layer before the last layer, that predicts the probabilities of classes in the ImageNet and Places datasets. The output matrix of the audio features is determined by the size of $TIME \times DIM$, where $TIME$ is the length of the sequences affected by the length of the audio clip, and the feature dimension is fixed as $DIM$ for a certain sample rate. In our model, 1024 audio features are generated for each short time series.

9.1.3 Textual Feature Extraction

In comparison to visual and audio data, text data is capable of providing rich information which precisely describes various situations. By adding the knowledge learned by the textual model, the multimodal model could capture important semantic information [203]. The text data is extracted from the video description from all the videos. Preprocessing is performed to clean and format the textual data, which includes stop words and punctuation removal and tokenization. Then, the textual data is transformed into the vector space by using a pre-trained word embedding model called GloVe [204]. GloVe first learns a word co-occurrence counts matrix and generates the vector space representation based on the co-occurrence of
each pair of words with a soft constraint:

\[
\gamma_i^T \gamma_j + b_i + b_j = \log(X_{ij}) \quad (9.1)
\]

where \(X_{ij}\) is the word pair \(i\) and \(j\), \(\gamma_i\) and \(\gamma_j\) are the word vectors for words \(i\) and \(j\), \(b_i\) and \(b_j\) are the biases term for words \(i\) and \(j\). Then, the co-occurrence matrix is reduced to generate the final word vector. The objective of the cost function \(J\) is to penalize rare word pairs which carry less information:

\[
f(X_{ij}) = \begin{cases} 
(X_{ij}/X_{\max})^\alpha & \text{if } X_{ij} < X_{\max} \\
1 & \text{otherwise}
\end{cases} \quad (9.2)
\]

\[
J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(\gamma_i^T \gamma_j + b_i + b_j - \log(X_{ij}))^2 \quad (9.3)
\]

where \(V\) is the total number of words, \(f(x)\) is the weighting function, \(X_{\max}\) is the cutoff threshold, and \(\alpha\) is the tunable parameter.

### 9.2 Temporal Feature Extraction Module

Video data includes a series of frames and there is valuable temporal information between the frames’ sequences. This temporal information can be seen among the visual and audio frames as well as the video textual data. After we extract the static features, we extract the temporal features from each modality using deep RNNs. Specifically, we extend our previously proposed model called Residual-Bidirectional Long short-term memory (ResBiLSTM) [141] to extract the temporal features from not only the visual data but also audio and text data.
9.3 Fusion Module

The output from the ResBiLSTM network consists of segments of temporal features that contain the relevant information for various concepts. By incorporating the early outputs from the temporal networks, the semantic correlation from different modalities could be preserved and utilized. The overall fusion model is illustrated in Algorithm 8. The unimodal vector representation $\vec{v}_i$, $\vec{a}_i$, $\vec{t}_i$ from visual, audio, and text models are concatenated to form a single vector representation $\vec{f}_i$. Then, all the vectors $\vec{f}_i$ are grouped and formed the new dataset $F$ based on the original ordering of the instances. The new vector has 384 dimensions that may cause various problems such as overfitting and slower training time. Therefore, dimensional reduction and feature selection techniques are applied. Random Forest (RF) is a tree-based ensemble learning algorithm that constructs multiple decision trees through the training phase and produces the final prediction score based on the majority vote of each classifier [205]. We use $F$ as the input of RF classifier and calculate the mean decrease of Gini Impurity (GI) of each feature. The GI is defined as:

$$GI = \sum_{j=1}^{|C|} P(j) \times (1 - P(j))$$

where $|C|$ is the size of the concepts, $P(j)$ is the probability of an input be classified as class $j$. While training, the total decrease of Gini impurity for each feature is computed on the decision tree level. Then, the impurity decrease from each feature is averaged on the whole forest. Based on the mean decrease of Gini impurity, feature ranking $R$ is generated. In real-world data, the distribution of the number of instances for different concepts may be heavily skewed. This imbalanced class problem could negatively impact the performance of the classifier since most
of the machine learning models assume the classes' distribution are uniform. Thus, the cost function of SVM is modified to penalize the misclassification of instances that belong to the minority classes. The new cost function is defined as:

\[ J = \frac{1}{\sum_{j}^{\lvert C \rvert} \delta_j} \sum_{i}^{N} \delta_j \cdot \max(0, 1 - y_i(w_i^T x_i + b_i)) \]  

(9.5)

where \( \delta_j \) is the inverse frequency of the number of instances containing class \( c_j \), \( \lvert C \rvert \) is the size of the concepts, \( N \) is the total number of instances, \( y_i \) is the label of \( i^{th} \) instance, \( x_i \) is the input instance, \( w_i \) and \( b_i \) are the learned weight and bias terms.

The original multi-label ground truth \( L \) is transformed into the single-label form \( \hat{L} \) using the label powerset algorithm. The weighted SVM is trained with the new ground truth label setup using the recursive feature elimination approach. This approach recursively drops the lowest ranked feature \( r_k \) in all the instances from input \( F \) based on the feature ranking \( R \). During each iteration, the prediction result will be recorded and compared with the previous score. If the latest score is not improved then the previous best result (\( S \)) will be returned.

### 9.4 Experimental Analysis

**Dataset.** The data used in this work is based on the dataset collected and used in our previous work [148]. The original dataset contains 1,540 video and audio clips that are extracted from 419 Youtube videos related to 2017 hurricane Harvey and Irma. We extend the original dataset by 1) adding text (extracted from the video descriptions) as a new modality, and 2) transforming the original single label problem into a multi-label problem. The statistics information of the disaster dataset is shown in Table 9.1.
**Algorithm 8** The proposed fusion algorithm

**Input:** Audio feature $A$, Video feature $V$, text feature $T$ and ground truth label $L$

**Output:** Final prediction score $S$

1: $F \leftarrow \{\}$
2: for $\vec{a}_i \in A$, $\vec{v}_i \in V$, $\vec{t}_i \in T$ do
3: $\vec{f}_i \leftarrow \text{concatenate}(\vec{a}_i, \vec{v}_i, \vec{t}_i)$
4: $F \leftarrow F \cup \vec{f}_i$
5: end for
6: $R \leftarrow \text{RandomForest}(F)$
7: $IF \leftarrow \{\}$
8: $\hat{L} \leftarrow \text{LabelPowerSet}(L)$
9: for $c_j \in C, j = 1, 2, ..., |C|$ do
10: $\delta_j \leftarrow \frac{1}{|F \in c_j|}$
11: $IF \leftarrow IF \cup \delta_j$
12: end for
13: for $r_k \in R, k = |R - 1|, |R - 2|, ..., 1$ do
14: $F \leftarrow F - r_k$
15: $s_k \leftarrow \text{WeightedSVM}(IF, F, \hat{L})$
16: if $s_k < s_{k-1}$ then
17: $S \leftarrow s_k$
18: return $S$
19: end if
20: $S \leftarrow s_k$
21: end for
22: return $S$

<table>
<thead>
<tr>
<th>No.</th>
<th>Concepts</th>
<th># of Instances</th>
<th>P/N Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Demo</td>
<td>150</td>
<td>0.047</td>
</tr>
<tr>
<td>2</td>
<td>Emergency Response</td>
<td>338</td>
<td>0.105</td>
</tr>
<tr>
<td>3</td>
<td>Flood/Storm</td>
<td>971</td>
<td>0.301</td>
</tr>
<tr>
<td>4</td>
<td>Human Relief</td>
<td>273</td>
<td>0.085</td>
</tr>
<tr>
<td>5</td>
<td>Damage</td>
<td>371</td>
<td>0.115</td>
</tr>
<tr>
<td>6</td>
<td>Victim</td>
<td>311</td>
<td>0.096</td>
</tr>
<tr>
<td>7</td>
<td>Speak/Briefing/Interview</td>
<td>811</td>
<td>0.251</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>3,225</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Setup. Different metrics are required to evaluate the performance of MLC compared to those used in the single label classification. In the literature, several metrics have been adopted \cite{206}. The evaluation metrics applied for our proposed model include Hamming Loss, micro-averaged F-measure and mean average precision.

The Hamming Loss (HL) represents the proportion of the misclassified labels to the total number of labels.

\[
HL = \frac{1}{|N|} \sum_{i=1}^{N} \frac{Y_i \oplus \Theta_i}{|C|}
\]  

(9.6)

where \(N\) is the total number of samples, \(|C|\) is the total number of concepts, \(Y_i\) is the ground truth label, \(\Theta_i\) is the prediction results, and \(\oplus\) is the binary logical “exclusive or” operator. Micro averaged F-measure (MicroF1) calculates the micro-averaged F1-score of all classes by counting the global True Positives (TP), False Negatives (FN) and False Positives (FP) across all classes. Mean Average Precision (MAP) calculates the average of the Average Precision (AP) over all the instances.

The dataset is randomly split into 60% training, 20% validation and 20% testing. In addition, we keep the distribution of classes almost similar between training, validation, and testing datasets. All model parameters are tuned using the validation dataset. The total numbers of static features for visual, audio, and text are 2048, 1024, and 1000, respectively. The temporal feature extraction model is composed of two bidirectional residual LSTM layers with 10% dropout, one dense layer using the ReLu activation function with 50% dropout and the final dense layer using Sigmoid activation function. The binary cross entropy is used as the cost function for the network training. For the weighted SVM classifier in the fusion model, a linear kernel is applied, a 0.9 penalty parameter for the error term is used and the shrinking heuristic is enabled.
**Experimental Results.** To demonstrate the effectiveness of the proposed multi-label multimodal deep learning model, it is compared with several baselines as follows. Single visual, audio, and textual models including static features from Inception-V3, SoundNet, and Glove, respectively, each combined with a dense layer for classification. The second group of baselines includes the combinations of two different modalities (e.g., visual+audio, visual+text, text+audio). We also compared the proposed model with two different fusion techniques including early fusion and late fusion. In early fusion, the static features are concatenated and then we apply LSTM to generate the temporal features followed by dense layers to generate the final scores. On the other hand, the late fusion concatenates the temporal features from each modality and apply the dense layers for classification.

Table 9.2 shows the detailed performance results of the baselines and the proposed model. It can be seen from the table that the single text models perform better than the visual and audio models. Specifically, text model achieves 0.78 and 0.69 micro F1 and MAP, respectively. The visual model also achieves a reasonable performance which is significantly higher than the audio model. These results illustrate the importance of textual and visual data in event detection and disaster information management applications. In the next step, every two various modalities are combined to generate the classification results. Surprisingly, the audio+text model achieves the highest performance (micro F1 of 0.86) among all these three combinations. This is mainly due to the fact that audio and text can complete each other better than visual+audio or visual+text. For example, audio can easily detect concepts “speak” and “flood”, but it cannot perform well for “damage” or “human relief” concepts, while text performs well in such concepts.

Finally, we used all the three modalities to further improve the results. It can be inferred from the table that simply concatenating the static features using early
Table 9.2: Performance evaluation results on the disaster dataset

<table>
<thead>
<tr>
<th>Approach</th>
<th>Features</th>
<th>Micro F1</th>
<th>HL</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single modal</td>
<td>visual</td>
<td>0.6767</td>
<td>0.1586</td>
<td>0.6015</td>
</tr>
<tr>
<td>Single modal</td>
<td>audio</td>
<td>0.5022</td>
<td>0.2565</td>
<td>0.4197</td>
</tr>
<tr>
<td>Single modal</td>
<td>text</td>
<td>0.7789</td>
<td>0.1187</td>
<td>0.6945</td>
</tr>
<tr>
<td>Two modalities</td>
<td>visual+audio</td>
<td>0.6667</td>
<td>0.1652</td>
<td>0.5928</td>
</tr>
<tr>
<td>Two modalities</td>
<td>visual+text</td>
<td>0.823</td>
<td>0.0969</td>
<td>0.7472</td>
</tr>
<tr>
<td>Two modalities</td>
<td>text+audio</td>
<td>0.8586</td>
<td>0.078</td>
<td>0.7882</td>
</tr>
<tr>
<td>Three modalities</td>
<td>visual+audio +text</td>
<td>0.812</td>
<td>0.102</td>
<td>0.7351</td>
</tr>
<tr>
<td>Three modalities</td>
<td>visual+audio +text</td>
<td>0.9022</td>
<td>0.0575</td>
<td>0.8409</td>
</tr>
<tr>
<td>Proposed model</td>
<td>visual+audio +text</td>
<td>0.9414</td>
<td>0.0348</td>
<td>0.8993</td>
</tr>
</tbody>
</table>

Fusion cannot improve the classification performance compared to the two modalities models. This is mainly due to the different nature of the feature sets that cannot be easily combined in the early levels. However, if the features are fused in the final levels (after applying the temporal module), we can achieve a significant improvement in the final performance (e.g., 0.90 micro F1). Finally, we further improve the performance by applying our proposed fusion technique which includes late fusion followed by RF feature selection and a weighted SVM for imbalanced data classification. As a result, we could beat all the benchmarks. Specifically, the micro F1, HL, and MAP reach 0.94, 0.03, and 0.90, respectively. In other words, the F1 score is improved by 4% and MAP is improved by almost 6% compared to the best result (late fusion).

We further demonstrate the effectiveness of the proposed model in Figure 9.2, in which our model is compared with the other two fusion techniques (early and late fusions). This figure visualizes the micro F1 results for each concept in the disaster dataset. It can be observed from the figure that the proposed model beats early and late fusions in all the concepts. For a few concepts such as “speak” and “flood”,
the late fusion’s performance is very close to the ones from our method. However, in other concepts such as “demo” and “emergency response”, there is a big gap between our performance and other fusion techniques. As shown in Table I, these concepts have lower P/N ratios compared to “speak” and “flood”. Therefore, the proposed model can successfully enhance the performance of the minority classes without scarifying the majority ones.

In summary, the proposed multi-label multimodal imbalanced data classification model achieves an outstanding performance for a very challenging and complex dataset.

9.5 Conclusion

This chapter presents a new multi-label multimodal model based on deep neural networks for imbalanced data classification. The proposed model includes static feature extraction for each modality using transfer learning, temporal feature analysis using ResBiLSTM, and a new fusion module which considers the correlation between both data modalities and labels. The proposed model also handles the
imbanced data problem by automatically assigning a weight to each class during the classification. This model is evaluated using the disaster video dataset.
10.1 Conclusions

In this dissertation, a comprehensive framework is proposed for spatio-temporal multimedia big data analytics based on deep neural networks. It includes several main components as follows: (1) synthetic data generation, (2) automatic sampling for imbalanced data, (3) deep representation learning, (4) automatic hyper-parameter learning, (5) deep spatio-temporal learning, and (6) multimodal deep learning. These components are integrated as a coherent entity to provide new solutions for existing challenges in multimedia big data. Each component is summarized as follows:

- Novel synthetic data generation models are proposed based on simulators and adversarial networks to overcome the difficulties of collecting large-scale real-world data. Specifically, they are designed to generate data for rare events, that are difficult to be obtained from the real world. Domain and scenario randomization are leveraged to bridge the reality gap between simulation and the real world while Cycle-GAN is used to generate data for rare cases from normal data automatically. These techniques are combined with deep learning to solve two important applications, namely autonomous driving in simulation and flood detection.

- A new sampling model is proposed to overcome data imbalanced problem in multimedia big data. This model is a combination of two new sampling models namely spatio-temporal synthetic oversampling and dynamic sampling approaches. The first model is used before the training process starts and
tries to balance the dataset by generating new synthetic video data using random frame selection and augmentation. The second model, however, is integrated with CNNs and updates the data samples of each class based on its performance in each training iteration.

- A new deep learning model is proposed to extract discriminative features for multimedia semantic event detection. This model consists of an ensemble of several pre-trained deep models for feature extraction and a new combination called Residual-Inception layers. The proposed deep feature extraction uses transfer learning techniques and improves the detection performance compared to the conventional hand-crafted features. Moreover, the proposed Residual-Inception model helps further improve the classification results and significantly enhance the training speed compared to very deep and complicated learning models.

- A new algorithm is proposed to automatically adjust the hyper-parameters in DNNs. This algorithm is called T-LRA, which is a general learning rate scheduler for SGD algorithm based on statistical trend analysis methods. It automatically modifies the learning rate based on the previous training trends and can achieve higher prediction performance in fewer training iterations.

- A novel spatio-temporal representation learning is presented to not only extract the static features from multimedia data but also leverages the temporal information in time-series data, such as video and audio. The proposed model shows how spatial and temporal information can be integrated effectively to improve the final classification results. This component utilizes deep learning techniques such as CNNs for spatial data analysis and LSTM for temporal feature analysis in multimedia data.
After extracting static and temporal features from multimedia data, how to integrate different data modalities in an effective and efficient manner becomes important. For this purpose, a new fusion model is proposed to handle multi-modal data classification. Specifically, a multi-label multimodal deep learning model is proposed as a fusion technique to consider the correlation between classes and different data modalities. The proposed fusion model is used specifically to detect semantic concepts from videos containing natural disasters and can be used for disaster management systems.

10.2 Future Work

It is shown in the previous chapters the effectiveness and efficiency of the proposed framework (depicted in Figure 3.1) for multimedia big data analytics. However, there are still several challenges that need to be considered in the future work as explained below.

10.2.1 Automatic Hyper-Parameter Learning

In this dissertation, the proposed hyper-parameter learning technique mainly focused on learning rate scheduling. It uses a statistical drop-based mechanism in which the learning rate parameter will be reduced gradually based on the training losses. It is already shown that this technique can adjust the step size until it reaches the minimum loss. However, if the model gets stuck in the local minimum, it is still challenging to reach the global minimum. Some optimizers have tried to address this issue by introducing adaptive momentum parameter \[130\]. However, these techniques are not always successful to get out of the local minimum. This is also the limitation of the proposed T-LRA algorithm. To address this issue, the future work
should focus on designing effective SGD Restart (SGDR) techniques \cite{207, 208}. In other words, instead of applying various types of learning rate decay, we should also reset the learning rate after several iterations. This process, also known as “cyclic learning rates”, is shown in Figure 10.1.

The important question is how to decide the number of epochs to restart the learning rate. Loshchilov and Hutter \cite{207} utilized the cosine function as the learning rate restart function. However, instead of setting each cycle to the same period of time, the length of the restart period should be expanded as we get closer to the global minimum. In future work, the T-LRA algorithm will be extended to automatically restart the learning rate based on the trend of the training loss.

In addition to the learning rate, there exists other hyper-parameters and parameters that need to be automated. For example, the whole structure of DNNs including the number of layers, kernel size, number of filters as well as global hyper-parameters (e.g., momentum) can be determined using revolutionary and genetic algorithms \cite{209, 210, 211}. Yet, evolving DNNs is computationally expensive, as
a complete training process is needed for each newly generated model. Therefore, more advanced techniques are required to only select the best generated models that will not only save the training time but also enhance the model performance within fewer iterations.

### 10.2.2 Integrated Synthetic Data Generation

Two different synthetic data generation models are proposed in this dissertation. The first model is based on Cycle-GAN that performs an image to image style transfer. Specifically, in our flood event detection application, it transfers abnormal styles such as blurry, night, and rainy to normal flood images and augments it to the regular flood images to enhance the DNN model. While in the second model, simulators (e.g., Unity Game engine) are used to generate image samples. Then, various domain and scenario randomizations are applied to bridge the reality gap.

In future work, we propose to integrate these two techniques to further reduce the gap between synthetic and real-world data. In other words, simulators can be used to automatically generate large-scale samples of images for different conditions and then adversarial models can be applied to these data to transfer the style of the real-world images to the simulation data. Using this method, we will not need any annotated real-world data. Instead, we can quickly generate labeled data with simulators and then collect a series of real-world images (without any labels). Cycle-GAN can transfer the annotated simulated data to the real-world data.

The preliminary results of our integrated simulation and Cycle-GAN model are shown in Figures 10.2 and 10.3. The first set of results show how the images can be transferred from a photo-realistic simulation world to the real world (Figure 10.2). From this figure, it can be seen that transferring from simulation to the real world
is very challenging (i.e., center images are very noisy and some objects are missing). This issue is mainly due to the complexity of the real-world images and the huge difference between these two domains. Therefore, we did another experiment to transfer the images from a primitive simulation (Domain 1 in Chapter 4) to a more realistic world (Domain 2 in Chapter 4) as shown in Figure 10.3. The results from the second experiment are more promising as the two domains are more similar (both are from simulators) and have less complexity. Thus, in the future work, one can transfer the style of realistic simulation to a primitive inexpensive simulation and later use these data to train a DNN model that is able to work on the real world.

10.2.3 Other Future Work

Other Modalities

The proposed multimodal deep learning model utilizes different data modalities in order to enhance the classification performance. The current model utilizes visual,
Unsupervised Learning

This dissertation utilizes several unsupervised learning techniques such as Cycle-GAN for image-to-image translation, transfer learning for static feature extraction, and data augmentation for generating additional synthetically modified data. Nevertheless, the main focus of this study is video classification and supervised learning. The main challenge in supervised learning is the annotation process which is very tedious and time-consuming. For instance, it took us several months and lots of human efforts to collect and annotate our disaster dataset. To tackle this challenge, many researchers have focused on unsupervised learning techniques such as unsu-
pervised generative adversarial networks and autoencoders [212] [213]. In addition, low-shot learning (learning from a few examples) can be considered as an alternative when there are a few supervised data [214]. These techniques can be leveraged in our framework in the future.

Online Learning

In general, the network topologies and architectures in deep learning are time static (i.e., they are pre-defined before the learning starts) and are also time invariant [47]. This restriction on time complexity poses a serious challenge when the data is streamed online. Online learning previously came into mainstream research [215], but only modest advancement has been observed in online deep learning. Conventionally, DNNs are built upon the Stochastic Gradient Descent (SGD) approach in which the training samples are used individually to update the model parameters with a known label. The need is that rather than the sequential processing of each sample, the updates should be applied as batch processing. One approach was presented in [216] where the samples in each batch are treated as Independent and Identically Distributed (IID). The batch processing approach proportionally balances the computing resources and execution time.

Another challenge that stacks up on the issue of online learning is high-velocity data with time varying distributions. This challenge represents the retail and banking data pipelines that hold tremendous business values. The current premise is that the data is largely close in time to safely assume piece-wise stationarity, and thus having a similar distribution. This assumption characterizes data with a certain degree of correlation and develops the models accordingly, as discussed in [217]. Unfortunately, these non-stationary data streams are not IID and are often longi-
tudinal data streams. Moreover, online learning is often memory delimited, harder to parallelize and requires a linear learning rate on each input sample.

In the future, developing methods that are capable of online learning from non-IID data would be a big leap forward for multimedia big data deep learning.

**Big Data and Distributed Systems**

This dissertation mainly concentrates on improving algorithms and machine learning techniques for efficient multimedia big data analytics. In the future, advanced big data analytics techniques such as distributed systems, cloud computing, and edge computing will be used in this framework to further reduce the computational costs and also speed up the training process. For example, data parallelism and model parallelism techniques can be used to train ML models in a distributed system [218].

Both data-parallel and model-parallel strategies have their own limitations. On one hand, if data parallelism has too many training modules, it has to decrease the learning rate to make the training procedure smooth. On the other hand, if model parallelism has too many segmentations, the output from the nodes will increase sharply and reduce the efficiency accordingly [219]. Generally speaking, the larger the dataset is, the more beneficial it is to have data parallelism. The larger the deep learning model is, the more suitable it is to have model parallelism. Besides, compared to data parallelism, it is hard to hide the communication needed for synchronization in model parallelism because only partial information is included in each node for the whole batch, though some advanced frameworks like TensorFlow[220] support asynchronous kernels to save the communication cost. Thus, it is necessary to wait till the synchronization step finishes before moving forward to the next layer since the activities are unable to be processed with only partial information. The two kinds of strategies can be also fused to a hybrid model as discussed in
Both these techniques can be also integrated with genetic algorithms to distribute the evaluation of the fitness of population members among different GPUs and machines.


[71] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. Mc-
Grew, A. Ray, J. Schneider, P. Welinder, et al., “Domain randomization and
generative models for robotic grasping,” in IEEE/RSJ International Confer-

[72] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan,
L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al., “Using simulation and
domain adaptation to improve efficiency of deep robotic grasping,” in IEEE

[73] F. Sadeghi and S. Levine, “Cad2RL: Real single-image flight without a single

[74] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and
T. Brox, “A large dataset to train convolutional networks for disparity, optical
flow, and scene flow estimation,” in IEEE Conference on Computer Vision and

RGB-D: 5m photorealistic images of synthetic indoor trajectories with ground

[76] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van
Der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with
convolutional networks,” in IEEE International Conference on Computer Vi-

image features and synthetic images for deep learning,” in European Confer-

randomization for transferring deep neural networks from simulation to the
real world,” in IEEE/RSJ International Conference on Intelligent Robots and

pattern learning framework for content-based image retrieval,” IEEE Trans-
November 2006.


VITA

SAMIRA POUYANFAR

2005-2009
B.A., Computer Engineering
University of Isfahan
Isfahan, Iran

2010-2012
M.S., Artificial Intelligence
Sharif University of Technology
Tehran, Iran

2014-2019
Ph.D. Candidate, Computer Science
Florida International University
Miami, Florida

PUBLICATIONS


