FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTING TECHNOLOGIES

AY 2023- 24 EVEN

18CSE352T NEURO-FUZZY AND GENETIC PROGRAMMING

Case Study Implementation Report

Enhanced Pedestrian Detection Using Optimized Deep Convolution Neural Network For Smart Building Surveillance

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

REG NO - RA2111003010551

NAME - SHIVANSH DENGLA

Objective:

The objective of this case study is to implement and evaluate an optimized VGG-16 (OVGG-16) architecture for pedestrian detection in images, leveraging deep learning techniques. The specific goals include dataset acquisition and preparation, model development using the OVGG-16 architecture, performance evaluation against baseline models, and analysis of the results to determine the effectiveness of the proposed approach.

Dataset Acquisition and Preparation:

The first objective is to acquire the INRIA dataset and prepare it for model training and evaluation. This involves dividing the dataset into training and validation sets, standardizing image sizes, and ensuring class balance between pedestrian and non-pedestrian images. The dataset serves as the foundation for training and testing the pedestrian detection models.

Model Development:

The second objective is to develop the OVGG-16 architecture tailored for pedestrian detection. This involves optimizing the VGG-16 architecture by fine-tuning hyperparameters and layers to enhance its performance in detecting pedestrians. The model is constructed using deep convolutional neural networks (CNNs) to leverage their feature extraction capabilities, thereby improving pedestrian detection accuracy.

Performance Evaluation:

The third objective is to evaluate the performance of the OVGG-16 model and compare it with baseline models, including the original VGG-16 architecture and a hybrid machine learning model (HMPD). Performance metrics such as accuracy, precision, recall, and F-measure are calculated to assess the model's effectiveness in detecting pedestrians. Confusion matrices and performance measures provide insights into the model's strengths and weaknesses.

Results and Analysis:

The final objective is to analyze the results of the performance evaluation and draw conclusions regarding the effectiveness of the OVGG-16 model. The analysis focuses on comparing the performance of the OVGG-16 model with baseline models and identifying areas where the optimized architecture excels. Insights gained from the analysis contribute to understanding the potential of deep learning techniques, particularly CNNs, in improving pedestrian detection systems for various applications.

Software Used:

The case study on implementing and evaluating the optimized VGG-16 architecture for pedestrian detection would involve the use of several software tools and frameworks for data preparation, model development, training, evaluation, and analysis. Here are some key software components typically used in such a project:

1. Python: Python serves as the primary programming language for implementing the deep learning model, as well as for data manipulation, visualization, and analysis.

2. Deep Learning Frameworks: Deep learning frameworks such as TensorFlow, PyTorch, or Keras are essential for building and training the convolutional neural network (CNN) models. These frameworks provide high-level APIs for constructing neural networks and offer efficient computation on GPUs for accelerated training.

3. Computer Vision Libraries: Libraries like OpenCV (Open Source Computer Vision Library) are utilized for image processing tasks such as loading, resizing, and augmenting images, as well as for performing operations like edge detection and image transformations.

4. Data Visualization Tools: Tools like Matplotlib and Seaborn are used for visualizing the dataset distributions, model training curves, and performance metrics such as precision-recall curves and confusion matrices.

5. IDEs (Integrated Development Environments): IDEs such as Jupyter Notebook, PyCharm, or Visual Studio Code provide interactive environments for writing and executing Python code, facilitating rapid prototyping and experimentation with deep learning models.

6. Version Control Systems: Version control systems like Git, along with platforms like GitHub or GitLab, are employed for managing project code, collaborating with team members, and tracking changes to the codebase over time.

7. Data Annotation Tools: Tools like LabelImg or VGG Image Annotator (VIA) may be used for annotating pedestrian bounding boxes in images, which is essential for creating ground truth datasets for model training and evaluation.

8. Performance Monitoring and Logging Tools: Tools for monitoring model training progress, such as TensorBoard (for TensorFlow) or PyTorch Lightning, help track metrics, visualize model architectures, and debug training issues.

Difficulty Faced:

During the implementation of the case study on optimizing the VGG-16 architecture for pedestrian detection, several challenges and difficulties may be encountered:

1. Model Optimization: One of the primary difficulties lies in optimizing the VGG-16 architecture for pedestrian detection. While VGG-16 is a powerful convolutional neural network (CNN) architecture, fine-tuning it for pedestrian detection requires careful adjustments to the model architecture, hyperparameters, and training strategies. Achieving the right balance between model complexity, training data, and computational resources can be challenging and may require extensive experimentation and tuning.

2. Data Collection and Annotation: Gathering and annotating a high-quality dataset of pedestrian images with accurate bounding box annotations can be a time-consuming and labor-intensive process. Ensuring sufficient diversity in the dataset to capture various pedestrian poses, occlusions, and environmental conditions adds to the complexity. Moreover, maintaining data consistency and quality control throughout the annotation process is crucial to ensure the effectiveness of the trained model.

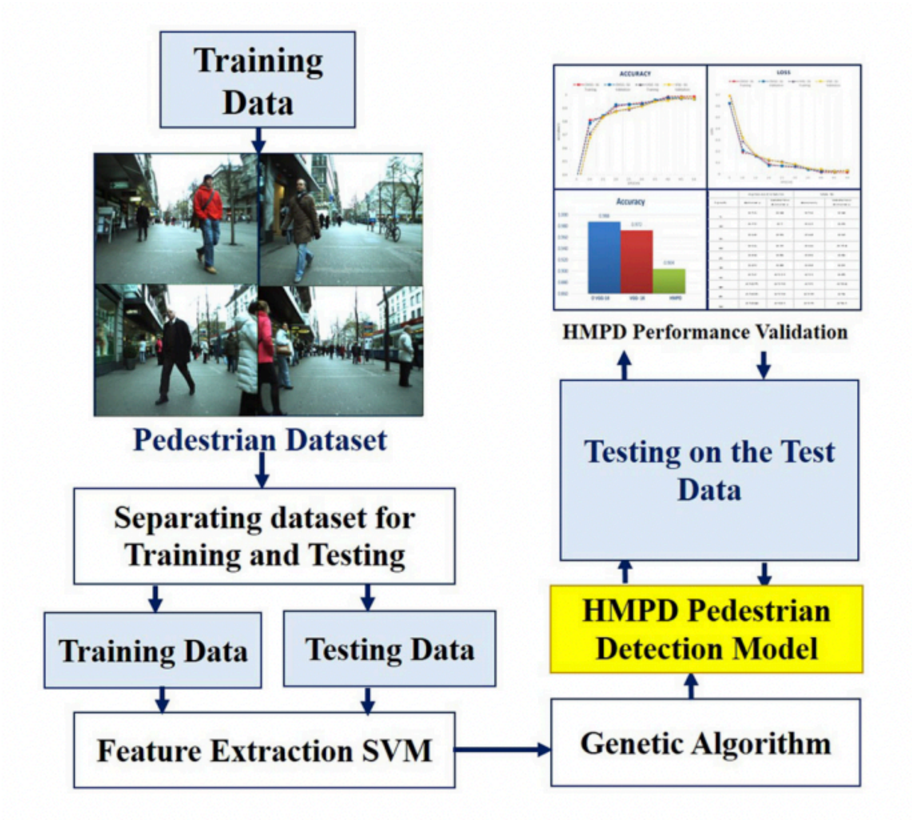
3. Computational Resources: Training deep learning models, especially complex architectures like VGG-16, requires significant computational resources, including powerful GPUs and large amounts of memory. Acquiring and managing these resources can be costly, particularly for researchers or organizations with limited budgets. Optimizing the training pipeline to make efficient use of available hardware resources becomes essential to minimize training time and costs.

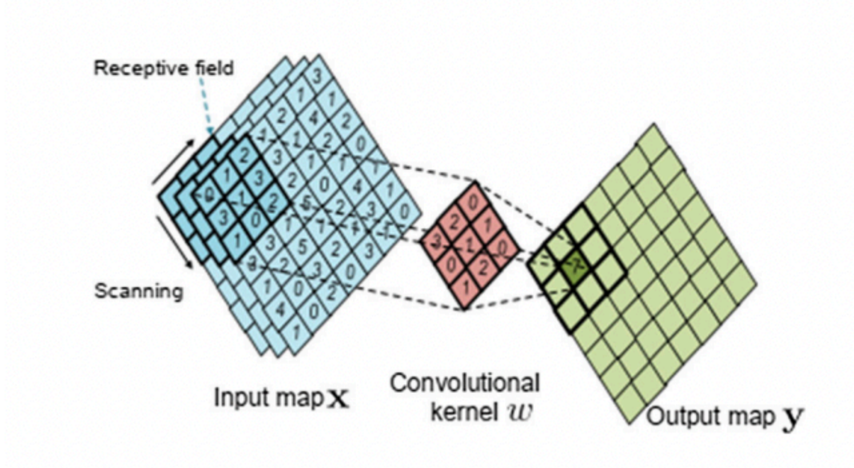
4. Hyperparameter Tuning: Tuning hyperparameters such as learning rate, batch size, and regularization techniques is crucial for achieving optimal model performance. However, identifying the right set of hyperparameters that balance model generalization and overfitting can be challenging. Conducting systematic hyperparameter search experiments and analyzing the impact of different configurations on model performance require careful planning and execution.

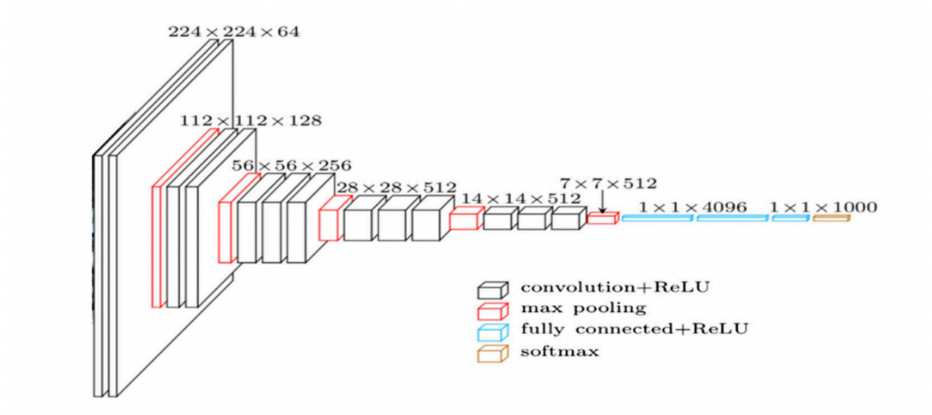
5. Evaluation and Interpretation: Evaluating the performance of the trained pedestrian detection model involves metrics such as precision, recall, and mean average precision (mAP). Interpreting these metrics and understanding the model's strengths and weaknesses in real-world scenarios requires domain expertise and qualitative analysis of model predictions. Additionally, identifying failure cases and areas for improvement based on evaluation results is essential for refining the model and iteratively improving its performance.

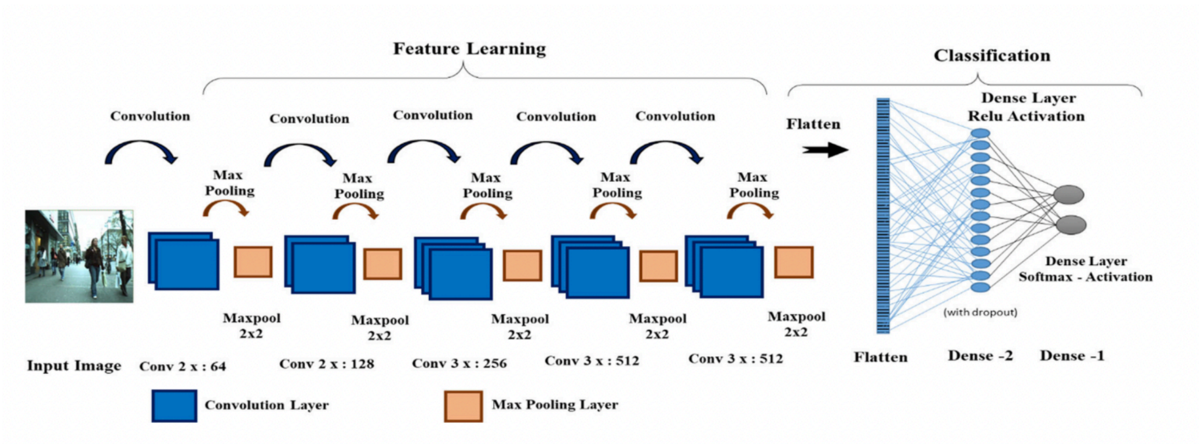
Addressing these challenges requires a combination of technical expertise, domain knowledge, and effective problem-solving strategies. By carefully navigating these difficulties and leveraging appropriate tools and methodologies, researchers can overcome obstacles and make meaningful progress in optimizing deep learning models for pedestrian detection applications.

Screen Shots:









Video Link (Make it Public):

https://drive.google.com/file/d/1qsDEUf7qi6GwY52ZfI1IPf_XtLrSGa69/view?usp=sharing

GitHub Link(Make it Public):

https://github.com/ShivanshDengla/Neurofuzzy-Journal-Paper