CNN-Based Collision Detection for Autonomous Navigation: Enhanced with Optical Flow

Author: Letao Shi

Date: April 2025

Summary

This paper proposes a method to optimize the training of lightweight CNN models by combining optical flow algorithm. By introducing the optical flow algorithm, the problem of time discontinuity in the original training is solved, and the accuracy of the model is significantly enhanced.

Experimental results demonstrate that the integration of the Farnebäck optical flow algorithm increases prediction accuracy from 81.88% to 87.6%, while also enhancing performance under challenging conditions such as low illumination and partial occlusion.

This study verifies the feasibility of using dual-frame input to enhance the model's temporal perception on resource-constrained devices, and provides an effective improvement path for collision prediction tasks in embedded vision systems.

1. Background and Motivation

I initially worked on a Raspberry Pi-based car target tracking project. The hardware part used Raspberry Pi + L298N driver module. The car collected images in real time through the camera, and used the KCF (Kernelized Correlation Filter) tracking algorithm in OpenCV to automatically track the selected target. In the test environment with stable lighting and simple background, the system ran smoothly, but once it entered an environment with complex lighting and dynamic background, KCF was prone to tracking loss, especially when the target accelerated or was occluded.

In order to improve this problem, I began to look for a more robust image recognition solution on the Internet. In the process, I read Professor David Banjerdpongchai of Chulalongkorn University's paper "Selection of Lightweight CNN Models with Limited Computing Resources for Drone Collision Prediction". The paper proposed a strategy for lightweight CNN models to achieve collision prediction on Raspberry Pi devices, which is very suitable for my current project scenario. Inspired, I decided to reproduce the MobileNet model structure used in the paper and explore its effect on my self-built dataset.

After reproducing the original model, I found that its judgment in some video clips was still delayed or misjudged, with an accuracy of about 81.88%. So I increased the number of training rounds to 20, and the accuracy increased to 83.98%. After continuing to increase the number of training rounds, the model overfitted and the accuracy fluctuated greatly. I realized that its input was only a single frame of image and lacked temporal continuity. To this end, I further introduced Farnebäck optical flow information and created a 6-channel dual-frame input, in order to enhance the model's understanding of the object's motion trend and add an additional input channel of movement. After adjusting the structure and training strategy, the final model accuracy increased to 87.6%.

2. Model Architecture and Technical Plan

1) Baseline Model (MobileNet):
- Input: RGB image (224x224x3)
- Structure: CNN + Pooling + Fully Connected + Sigmoid (Binary)
- Framework: TensorFlow + Keras
- Accuracy: 81.88%

2) Improved Model with Optical Flow:
- Compute Farnebäck optical flow for adjacent frames
- Concatenate frame1 and frame2 into a 6-channel input (RGB+RGB)
- Use a data generator to feed dual frames
- Change input shape to (224, 224, 6)

Example model code:

```
base_model = models.Sequential([
    layers.InputLayer(input_shape=(224, 224, 6)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

3. Dataset and Training

- Used 93 videos from “uav-collision-avoidance”
- Used frames and labels (collision=0/1 from Excel files)
- 83 videos used for training, 10 for validation
- Verified paths before training to ensure data integrity

4. Experimental Results

- Final accuracy improved to 87.6%
- Performance remained stable under low-light and partial occlusion scenarios

5. Reflection and Significance

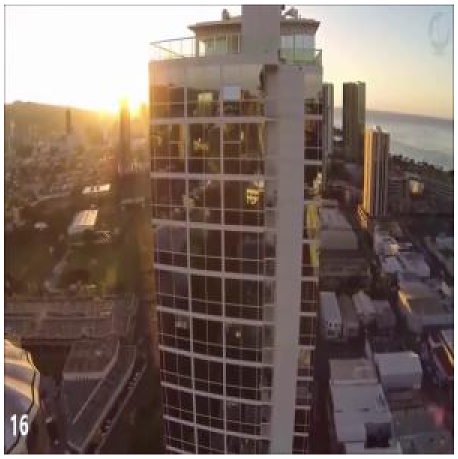
- This was my first project integrating deep learning with temporal image analysis
- Successfully reproduced the model training part of the paper
- Strengthen the ability in data preparation, model tuning, and performance analysis
- Learn a lot for further study in computer vision and intelligent systems
- Lays a solid foundation for my future research in autonomous systems, where understanding temporal dynamics is crucial.

Appendix

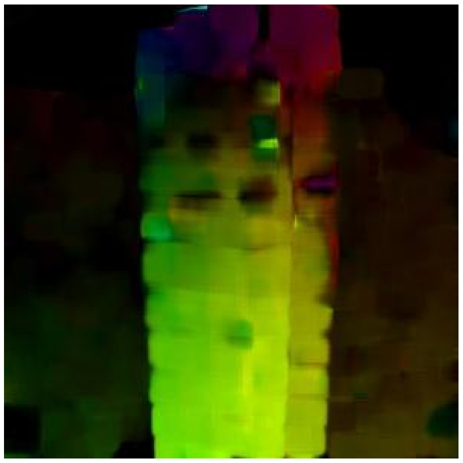
- Framework: TensorFlow 2.x + OpenCV
- Tools: PyCharm, Jupyter Notebook
- Language: Python 3.12.7
- Code: see file 'Combine.py', 'Train.py', 'Transform.py'. or view: terry002531/CNN_Collision_Report_OpticalFlow -
Training data: https://github.com/dario-pedro/uav-collision-avoidance/tree/master/colanet-web

"Original data set image, optical flow diagram and their combined diagram

1. Original diagram:



2. Optical flow diagram:



3. Combined diagram:

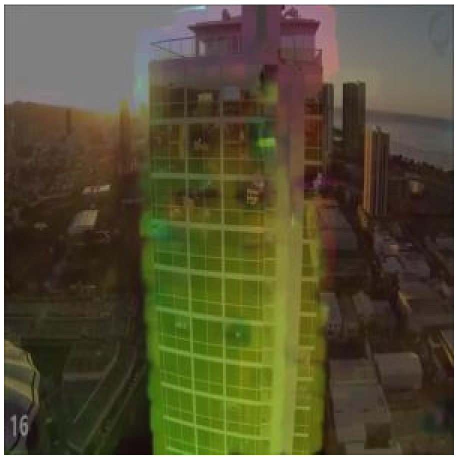
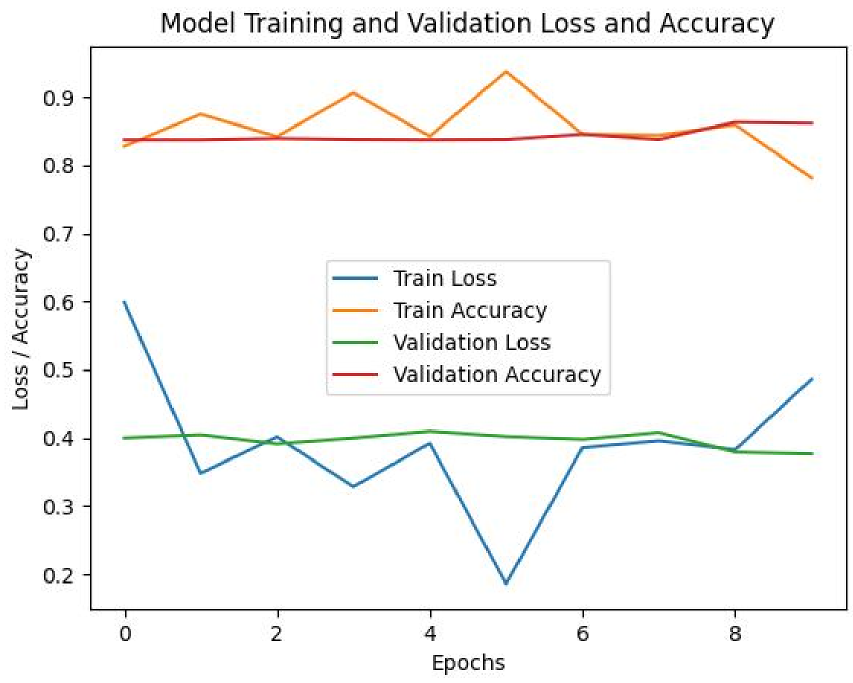
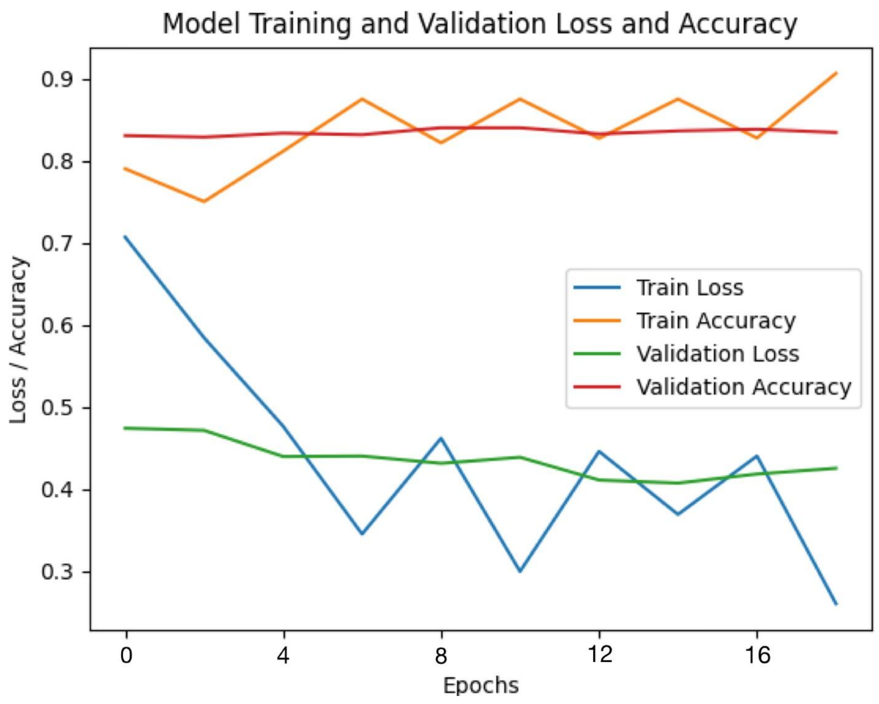


Image of training results

1. Results of 10 rounds of training on the original model：



2. The result of increasing the number of training rounds to 20：



3. The results of 10 rounds of training with optical flow algorithm are obtained：

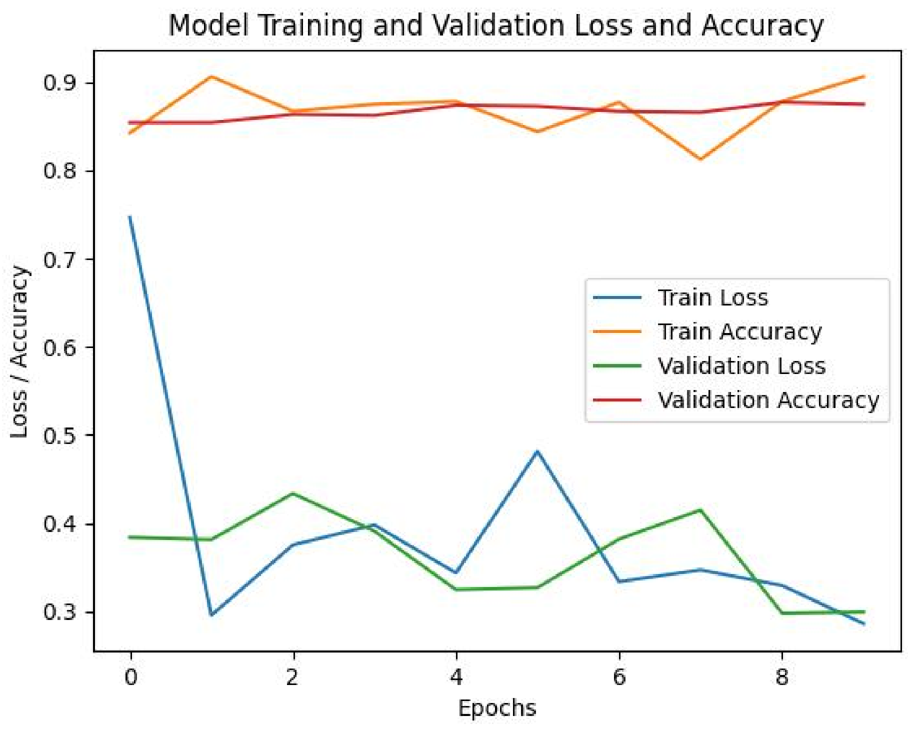


Table 1: Accuracy Comparison of Different Models

Model Version	Epochs	Accuracy
Baseline MobileNet	10	81.88%
Baseline MobileNet	20	83.98%
With Optical Flow (6-ch)	10	87.60%