Enhanced Media Experience with AI-Powered Commercial Detection and Replacement

BeagleBoard.org
Mar 17, 2024

Table of contents

Chapter 1

Introduction

Leveraging the capabilities of BeagleBoard's powerful processing units, the project will focus on creating a real-time, efficient solution that enhances media consumption experiences by seamlessly integrating custom audio streams during commercial breaks.

1.1 Summary links

- **Contributor:** Aryan Nanda
- **Mentors:** Jason Kridner, Deepak Khatri
- **GSoC Repository:** TBD

1.2 Status

This project is currently just a proposal.

1.3 Proposal

- Created accounts across OpenBeagle, Discord and Beagle Forum
- The PR Request for Cross Compilation: #185
- Created a project proposal using the proposed template

1.4 About

- Find my Resume here
- **Forum:** u/aryan_nanda
- **OpenBeagle:** aryan_nanda
- **Github:** AryanNanda17
- **School:** Veermata Jijabai Technological Institute (VJTI)
- **Country:** India
- **Primary language:** English, Hindi
- **Typical work hours:** 9AM-5PM Indian Standard Time

- **Previous GSoC participation:** This would be my first time participating in GSOC

Chapter 2

About the Project

Project name: Enhanced Media Experience with AI-Powered Commercial Detection and Replacement

2.1 Description

I propose developing a **GStreamer Plugin** capable of processing video inputs based on their classification. The plugin will identify commercials and either replace them with alternative content or obscure them, while also substituting the audio with predefined streams. This enhancement aims to improve the media consumption experience by eliminating unnecessary interruptions. I intend to **explore various video classification models** to achieve accurate detection and utilize TensorFlow Lite to leverage the **native accelerators of BeagleBone AI-64** for high-performance, real-time inferencing with minimal latency. I believe real-time high-performance would be the most critical thing for this project and I intend on testing a few different ways to see which one works best.

2.2 Goals and Objectives

The goal of this project is to detect and replace commercials in video streams on BeagleBoard hardware using a GStreamer pipeline which includes a model that accurately detects commercials with minimal latency. Comparison of different model accuracy can be done by doing some manual analysis and trying different video classification models and to finally use the best performing option to be included in the GStreamer pipeline for inferencing of real-time videos. This would be the result presented at the end of the project timeline. For phase 1 evaluation, the goal is to build a training dataset, preprocess it and fine-tune and train a Video Classification model to identify commercials segments in a video accurately. For phase 2 evaluation, the goal is to use the the best model identified in phase 1 for commercial detection and build a GStreamer pipeline which would do video processing based on commercial segments classification and using native accelerators present in BeagleBone-ai-64 for high-performance.

In order to accomplish this project the following objectives needs to be met.

1. **Phase 1:-**
    - Develop a dataset of videos and corresponding labels indicating the presence of commercials in specific segments.
    - Preprocess the dataset to ensure it's suitable for input into deep learning models. Moreover divide the datset into train, validation and test set.
    - Fine-tune various deep learning models and train them on the prepared dataset to identify the most accurate one for commercial detection in videos.
    - Save all trained models to local disk and perform real-time inference using OpenCV to determine the model that yields the best results with high-performance.

2. **Phase 2:-**
    - Based on all the options tried in Phase 1, decide on the final model to be used in the GStreamer pipeline.
    - Compiling the model and generating artifacts so that we can use it in TFLite Runtime.
    - Building a GStreamer pipeline that would take real-time input of media and would identify the commercial segments in it.
    - If the commercial segment is identified the GStreamer pipeline would either replace them with alternative content or obscure them, while also substituting the audio with predefined streams.
    - Enhancing the Real-time performance using native hardware Accelerators present in BeagleBone-Ai-64.

Chapter 3

Methods

In this section, I will individually specify the training dataset, model, GStreamer Pipeline etc. methods that I plan on using in greater details.

3.1 Building training Dataset

To train the model effectively, we need a dataset with accurate labels. Since a suitable commercial video dataset isn't readily available, I'll create one. This dataset will consist of two classes: commercial and non-commercial. To build this dataset, I'll refer to the Youtube-8M dataset, which includes videos categorized as TV advertisements. I'll download these videos and organize them into two folders: commercial and non-commercial. However, since the Youtube-8M dataset provides encoded feature vectors instead of the actual videos, direct usage would result in significant latency. Therefore, I'll use it as a reference and download the videos labeled by it as advertisements to build our dataset. After the dataset is ready I will preprocess it to ensure it's suitable for input into deep learning models. Moreover I'll divide the datset into train, validation and test set.

3.2 Video Classification models

MoViNets is a good model for our task as it can operate on streaming videos for online inference. The main reason behind trying out MoViNets first is becaue it does quick and continuous analysis of incoming video streams. MoViNet utilizes NAS(Neural Architecture Search) to balance accuracy and efficiency, incorporates stream buffers for constant memory usage, and improves accuracy via temporal ensembles. The MoViNet architecture uses 3D convolutions that are "causal". Causal convolution ensures that the output at time t is computed using only inputs up to time t. This allows for efficient streaming. This make MoViNets a perfect choice for our case. So, I will fine tune the MoViNets model and will train it on the commercial detection dataset.

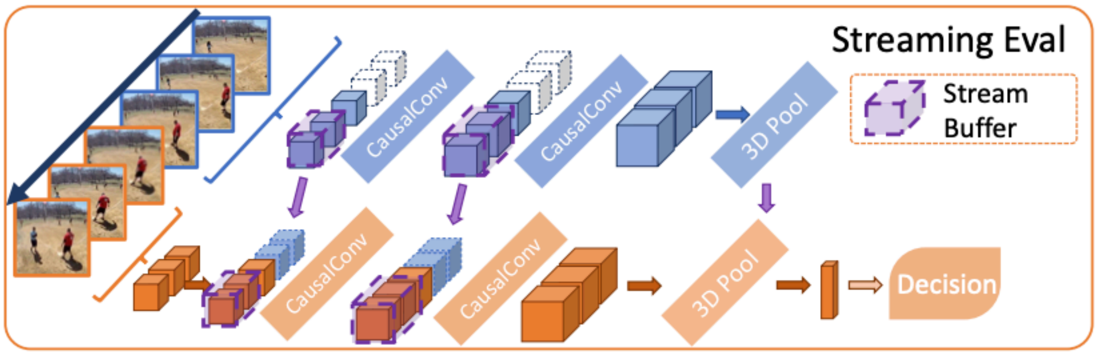

Figure 1: Stream buffer in MoViNets

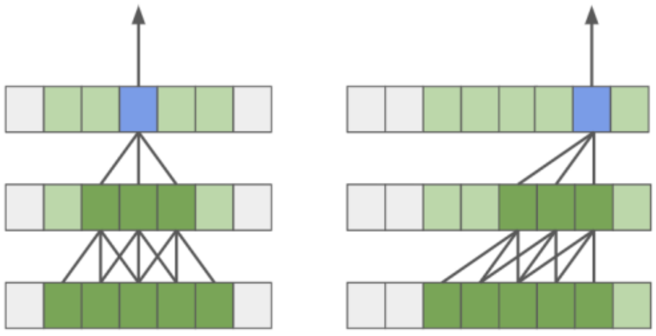

Figure 2: Standard Convolution Vs Causal Convolution

If MoViNet does not perform well than we can use other models like ResNet-50+LSTMs. Since a video is just a series of frames, a naive video classification method would be pass each frame from a video file through a CNN, classify each frame individually and independently of each other, choose the label with the largest corresponding probability, label the frame, and assign the most assigned image label to the video. To solve the problem of "prediction flickering", where the label for the video changes rapidly when scenes get labeled differently. I will use **rolling prediction averaging** to reduce "flickering" in results.

The Conv+LSTMs model will perform well as it considers both the spatial and temporal features of videos just like a Conv3D model. The only reason it is not my first choice is because MoViNets are considered to be better for real-time performance.

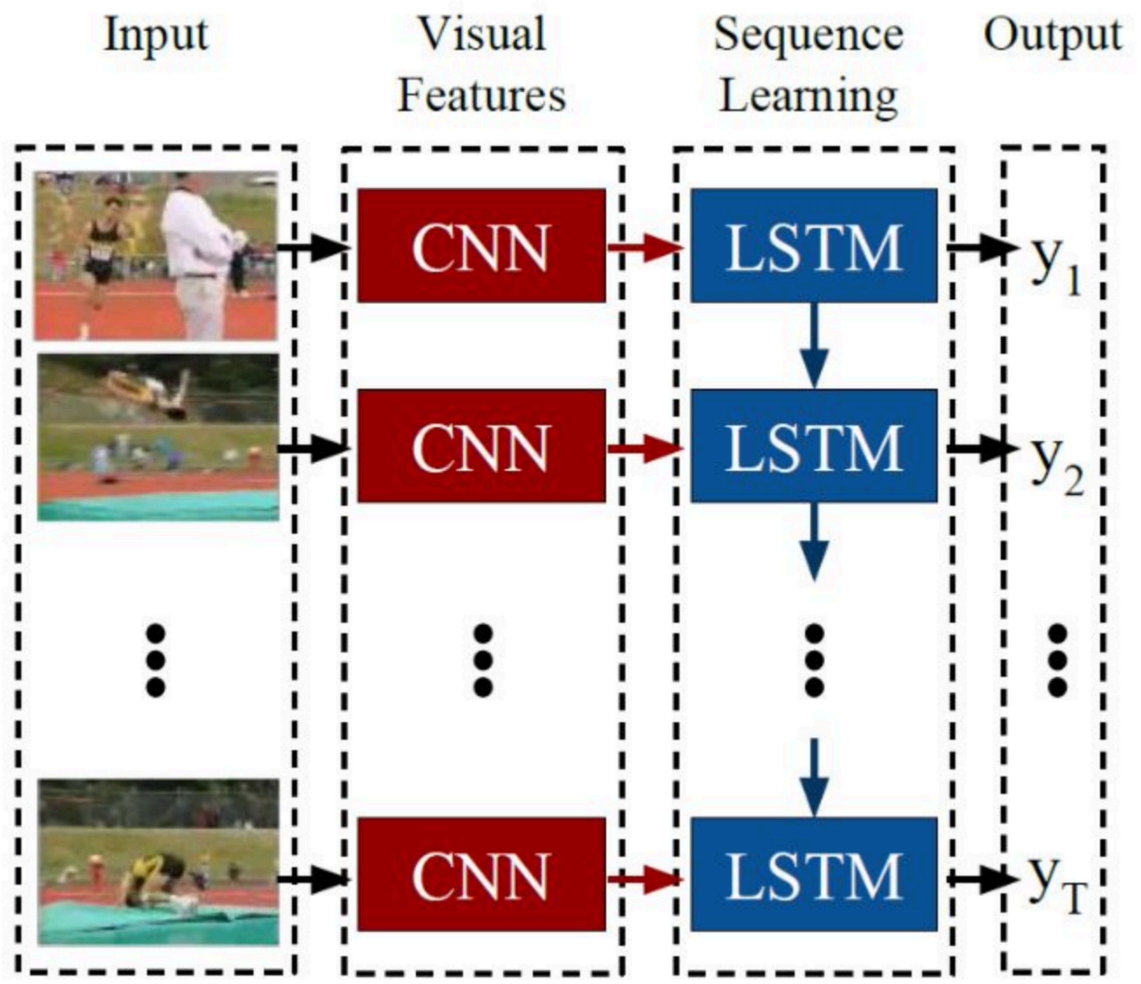

Figure 3: Conv+LSTMs

3.2.1 Optional method with Video Vision Transformers

This is a pure Transformer based model which extracts spatio-temporal tokens from the input video, which are then encoded by a series of transformer layers. I have kept this as an optional method because our problem is of binary classification(either Commercial or Non-Commercial), so using such a complex model for this small problem may not be as efficient as other models.

3.3 Choosing the Best Performing model

After training the models, I'll assess their performance using evaluation metrics and conduct real-time inference on a sample video containing both commercial and non-commercial segments. I'll select the model with the highest accuracy and integrate it into the GStreamer pipeline for further processing.

3.4 Model Execution on BeagleBone AI-64

The BeagleBone AI-64 Linux for Edge AI supports importing pre-trained custom models to run inference on target. Moreover, Edge AI BeagleBone AI-64 images have TensorFlow Lite already installed with acceleration enabled. The Debian-based SDK makes use of pre-compiled DNN (Deep Neural Network) models and performs inference using various OSRT (open source runtime) such as TFLite runtime, ONNX runtime etc.

In order to infer a DNN, SDK expects the DNN and associated artifacts in the below directory structure.

```
project_root
│
├── param.yaml
│
├── artifacts
│    ├── 264_tidl_io_1.bin
│    ├── 264_tidl_net.bin
│    ├── 264_tidl_net.bin.layer_info.txt
│    ├── 264_tidl_net.bin_netLog.txt
│    ├── 264_tidl_net.bin.svg
│    ├── allowedNode.txt
│    └── runtimes_visualization.svg
│
└── model
     └── ssd_mobilenet_v2_300_float.tflite
```

1. model: This directory contains the DNN being targeted to infer
2. artifacts: This directory contains the artifacts generated after the compilation of DNN for SDK.
3. param.yaml: A configuration file in yaml format to provide basic information about DNN, and associated pre and post processing parameters.

Therefore, after choosing the model to be used in GStreamer pipeline, I will generate the artifacts directory by following these instructions.

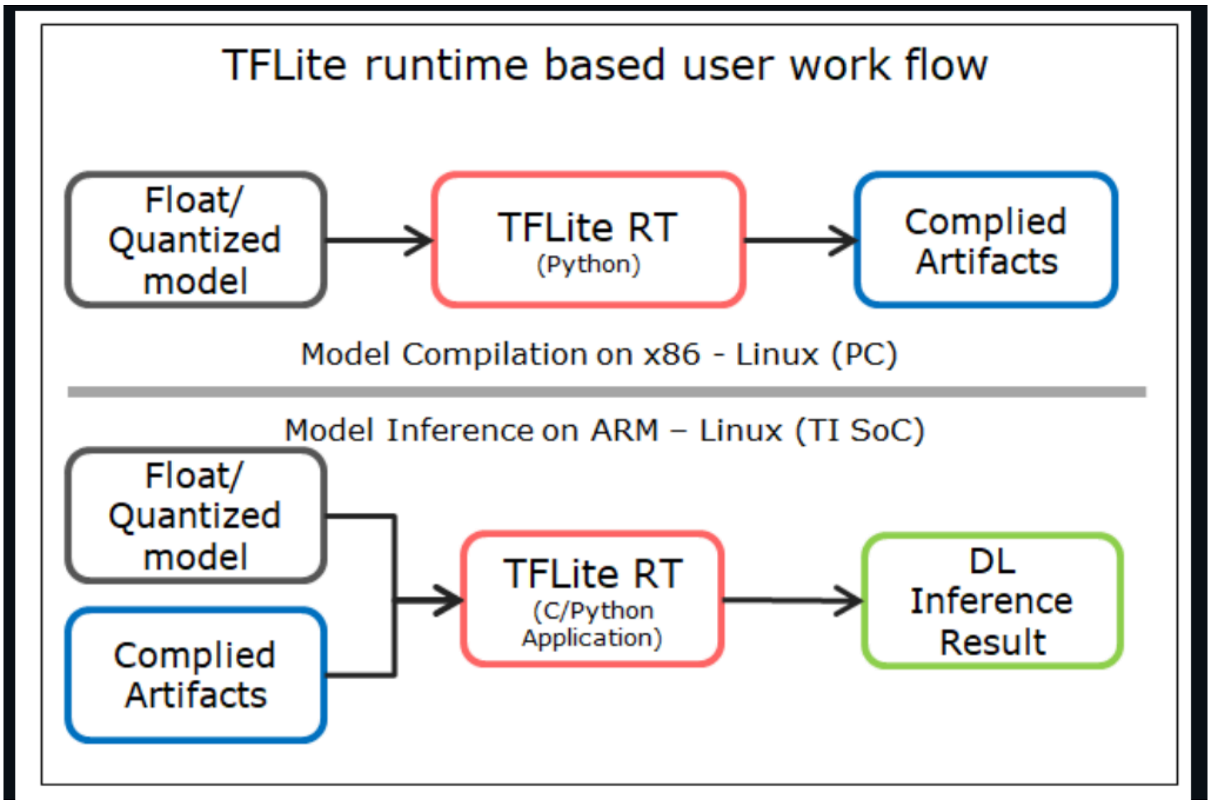
TFLite runtime based user work flow
Float/ Quantized model
TFLite RT (Python)
Complied Artifacts
Model Compilation on x86 - Linux (PC)
Model Inference on ARM – Linux (TI SoC)
Float/ Quantized model
Complied Artifacts
TFLite RT (C/Python Application)
DL Inference Result

Figure 4: TFLite Runtime

3.5 GStreamer Pipeline

The data flow in the GStreamer pipeline at a high level can be split into 3-parts:-

1. Input Pipeline - Grabs a frame from the input source.
2. Output Pipeline - Sends the output to the display.
3. Compute Pipeline - Performs pre-processing, inference and post-processing.

I will create a GStreamer Pipeline that will receive an input of a video and it will grab it frame by frame. The frame will be split into two paths. The "analytics" path resizes the input maintaining the aspect ratio and crops the input to match the resolution required to run the deep learning network. The "visualization" path is provided to the post-processing module which overlays the detected classes. If a commercial video is detected, we apply blurring to the video frames and replace the audio. If a non-commercial video is detected, proceed with the normal visualization process without blurring or replacing the audio. Post-processed output is given to HW mosaic plugin which positions and resizes the output window on an empty background before sending to display.

The following GStreamer input and output pipeline describes how I will build the GStreamer pipeline:-

- GStreamer input pipeline:

```
v4l2src device=/dev/video0 ! \
video/x-raw,format=NV12,width=1920,height=1080 ! \
tiovxmultiscaler name=split_01 ! \
split_01. ! queue ! video/x-raw, width=320, height=320 ! \
tiovxdlpreproc data-type=10 channel-order=1 mean-0=128.000000 mean-1=128.
↪000000 mean-2=128.000000 scale-0=0.007812 scale-1=0.007812 scale-2=0.
↪007812 tensor-format=rgb out-pool-size=4 ! \
application/x-tensor-tiovx ! appsink name=pre_0 max-buffers=2 drop=true \
split_01. ! queue ! video/x-raw, width=1280, height=720 ! \
```

(continues on next page)

(continued from previous page)

```
tiovxdlcolorconvert target=1 out-pool-size=4 ! \
video/x-raw, format=RGB ! appsink name=sen_0 max-buffers=2 drop=true
```

- GStreamer output pipeline:

```
appsrc format=GST_FORMAT_TIME is-live=true block=true do-timestamp=true␣
↪name=post_0 ! \
    tiovxdlcolorconvert ! video/x-raw,format=NV12,width=1280,height=720 ! \
    queue ! mosaic_0.sink_0

appsrc format=GST_FORMAT_TIME block=true num-buffers=1 name=background_0 ! \
    tiovxdlcolorconvert ! video/x-raw,format=NV12,width=1920,height=1080 ! \
    queue ! mosaic_0.background

tiovxmosaic name=mosaic_0 \
    sink_0::startx=320  sink_0::starty=180  sink_0::width=1280  sink_
↪0::height=720 \
    ! video/x-raw,format=NV12,width=1920,height=1080 ! \
    kmssink sync=false driver-name=tidss
```

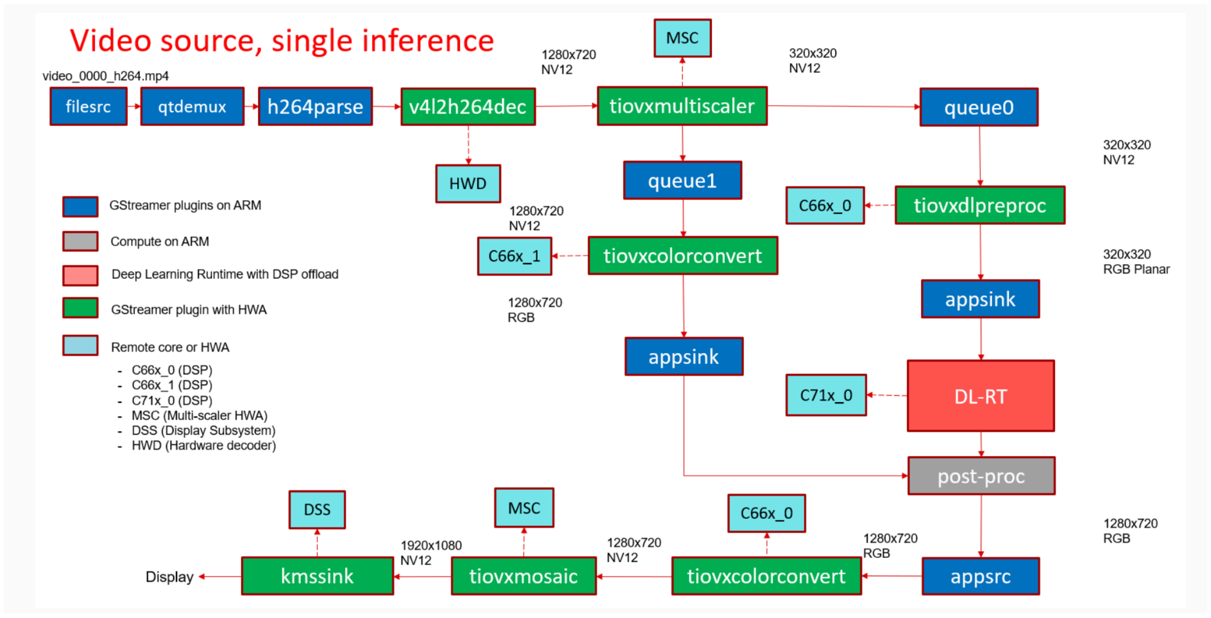

Figure 5: GStreamer Pipeline

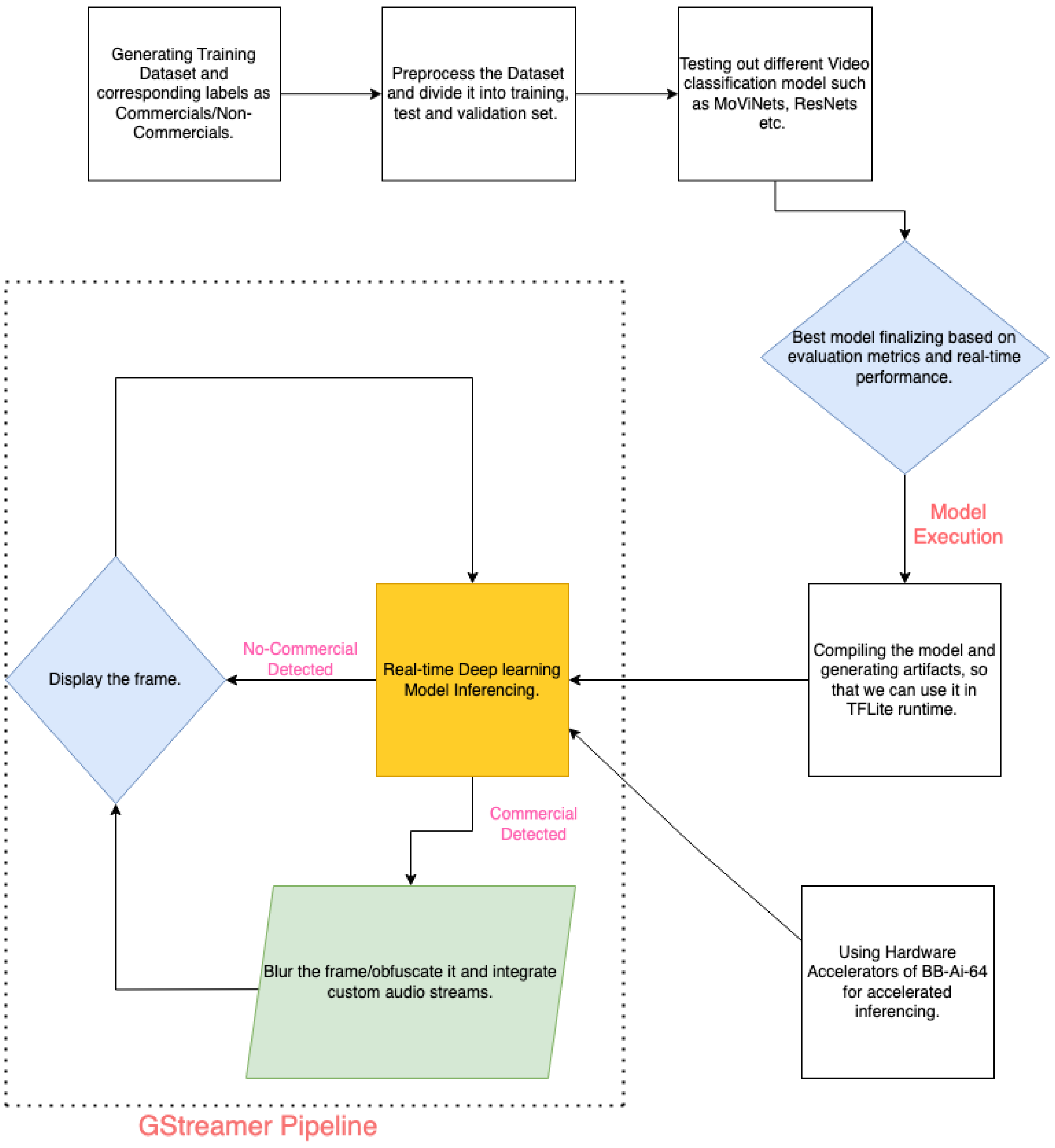
Generating Training Dataset and corresponding labels as Commercials/Non-Commercials.
Preprocess the Dataset and divide it into training, test and validation set.
Testing out different Video classification model such as MoViNets, ResNets etc.
Best model finalizing based on evaluation metrics and real-time performance.
Model Execution
Compiling the model and generating artifacts, so that we can use it in TFLite runtime.
Using Hardware Accelerators of BB-Ai-64 for accelerated inferencing.
Real-time Deep learning Model Inferencing.
No-Commercial Detected
Display the frame.
Commercial Detected
Blur the frame/obfuscate it and integrate custom audio streams.
GStreamer Pipeline

Figure 6: Project Workflow

3.6 Software

- Python
- C++
- TensorFlow
- TFlite
- GStreamer
- OpenCV
- Build Systems

3.7 Hardware

- Ability to capture and display video streams using Beagleboard ai-64

Chapter 4

Timeline

4.1 Timeline summary

Date	Activity
April 3 - May 1	Understanding GStreamer pipeline and TFLite runtime of BeagleBone-Ai-64.
May 1 - May 10	Discussing implementation ideas with mentors.
May 10 - May 31	Focus on college exams.
June 1 - June 3	Start coding and introductory video
June 3	Release introductory video and complete milestone #1
June 10	*Complete milestone #2*
June 17	*Complete milestone #3*
June 24	*Complete milestone #4*
July 1	*Complete milestone #5*
July 8	Submit midterm evaluations
July 15	*Complete milestone #6*
July 22	*Complete milestone #7*
July 29	*Complete milestone #8*
August 5	*Complete milestone #9*
August 12	*Complete milestone #10*
August 19	Submit final project video, submit final work to GSoC site and complete final mentor evaluation

4.2 Timeline detailed

4.3 Community Bonding Period (May 1st - May 10th)

- Discuss implementation idea with mentors.
- Discuss the Scope of the project.

4.4 Milestone #1, Introductory YouTube video (June 3rd)

- Making an Introductory Video.
- Starting working on dataset.

4.5 Milestone #2 (June 10th)

- Develop a dataset of videos and corresponding labels indicating the presence of commercials in specific segments.

- Preprocess the dataset to ensure it's suitable for input into deep learning models. Moreover divide the datset into train, validation and test set.

4.6 Milestone #3 (June 17th)

- Fine-tune various deep learning models and train them on the prepared dataset to identify the most accurate one for commercial detection in videos.

4.7 Milestone #4 (June 24th)

- Continuing working on model and fine-tuning it to give the best results.

4.8 Milestone #5 (July 1st)

- Save all trained models to local disk and perform real-time inference using OpenCV to determine the model that yields the best results with high-performance.

4.9 Submit midterm evaluations (July 8th)

- Completing all the phase 1 tasks if reamaining.

Important: July 12 - 18:00 UTC: Midterm evaluation deadline (standard coding period)

4.10 Milestone #6 (July 15th)

- Based on all the options tried in Phase 1, decide on the final model to be used in the GStreamer pipeline.
- Compiling the model and generating artifacts so that we can use it in TFLite Runtime.

4.11 Milestone #7 (July 22nd)

- Building a GStreamer pipeline that would take real-time input of media and would identify the commercial segments in it.

4.12 Milestone #8 (July 29th)

- Continuing working on the GStreamer pipeline.
- If the commercial segment is identified the GStreamer pipeline would either replace them with alternative content or obscure them, while also substituting the audio with predefined streams.

4.13 Milestone #9 (Aug 5th)

- Enhancing the Real-time performance using native hardware Accelerators present in BeagleBone-Ai-64.

4.14 Milestone #10 (Aug 12th)

- Continuing working on optimizing the real-time output for best performance.

4.15 Final YouTube video (Aug 19th)

- Submit final project video, submit final work to GSoC site and complete final mentor evaluation

4.16 Final Submission (Aug 24nd)

Important: **August 19 - 26 - 18:00 UTC:** Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

August 26 - September 2 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

4.17 Initial results (September 3)

Important: **September 3 - November 4:** GSoC contributors with extended timelines continue coding

November 4 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 11 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with extended deadline

Chapter 5

Experience and approch

This project requires prior experience with machine learning, multimedia processing and embedded systems.

- As a good starting point for this project, I build a Sports Video Classification model and did **Video Processing on it based on Video classification** using OpenCV(Video Processing based on Video classification is an important part of the project). Demo
- **We will be building Pure C++ GStreamer pipeline from input to output so experience with C++ Codebases and build systems is required.**
    - Relevant contribution - https://github.com/SRA-VJTI/Pixels_Seminar/pull/123 (OpenCV/C++)
- I have Previously Worked on the project GestureSense in which I did Image Processing based on Image classification using OpenCV/Python.
- I have past experience with esp-32 microcontroller and I have Previously Worked on a project Multi-Code-Esp in which I build a multi-code esp component.
- **Experience in Open-Source**

    Contributed at pymc repository. Added enhancements.

    - https://github.com/pymc-devs/pymc/pull/7132 (merged)
    - https://github.com/pymc-devs/pymc/pull/7125 (merged)

    Resolved one issue in OpenCV repository.

    - https://github.com/opencv/opencv/issues/22177 (merged)
- **Contributions in openbeagle.org/gsoc**
    - Added New idea.
    - Improved Documentation

5.1 Contingency

- **If I get stuck on my project and my mentor isn't around, I will use the following resources:-**
    - MoViNets
    - Video Visual Transformer
    - GStreamer Docs
    - BeagleBone-Ai-64
- Moreover, the BeagleBoard community is extremely helpful and active in resolving doubts, which makes it a great going for the project resources and clarification.

5.2 Benefit

This project will not only enhance the media consumption experience for users of BeagleBoard hardware but also serve as an educational resource on integrating AI and machine learning capabilities into embedded systems. It will provide valuable insights into:

- The practical challenges of deploying neural network models in resource-constrained environments.
- The development of custom GStreamer plugins for multimedia processing.
- Real-world applications of machine learning in enhancing digital media experiences.

5.3 Misc

- The PR Request for Cross Compilation: #185
- Relevant Coursework: Neural Networks and Deep Learning, Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization, Convolutional Neural Networks