cuFLAVR

A CUDA implementation of ML-based frame interpolation

Aditya Hota
ahota@seas.upenn.edu
GitHub: adityahota
CMPE (BSE'21) + ROBO (MSE'22)

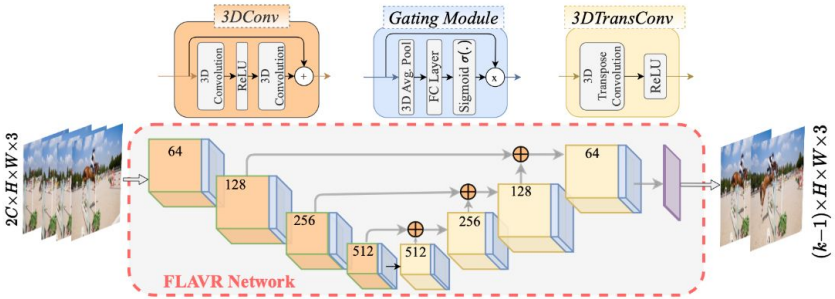
Richard Chen
laurelin@seas.upenn.edu
GitHub: LaurelinTheGold
CMPE (BSE'22) + ROBO (MSE'23)

Kaan Erdogmus
kaanberk@seas.upenn.edu
GitHub: kaan9
CIS (BSE'22) + WH (BSEcon'22)

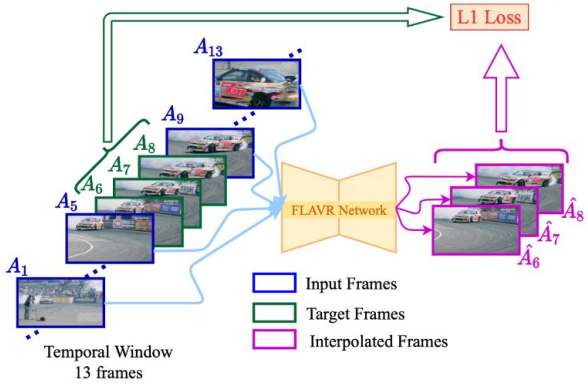
CIS 565 | Team 03 | Fall 2021

What is FLAVR?

- Flow-Agnostic Video Representations for Fast Frame Interpolations
  - Developed by Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran
- Adds frames to videos to increase frame rate
  - Our focus is on 2x frame interpolation

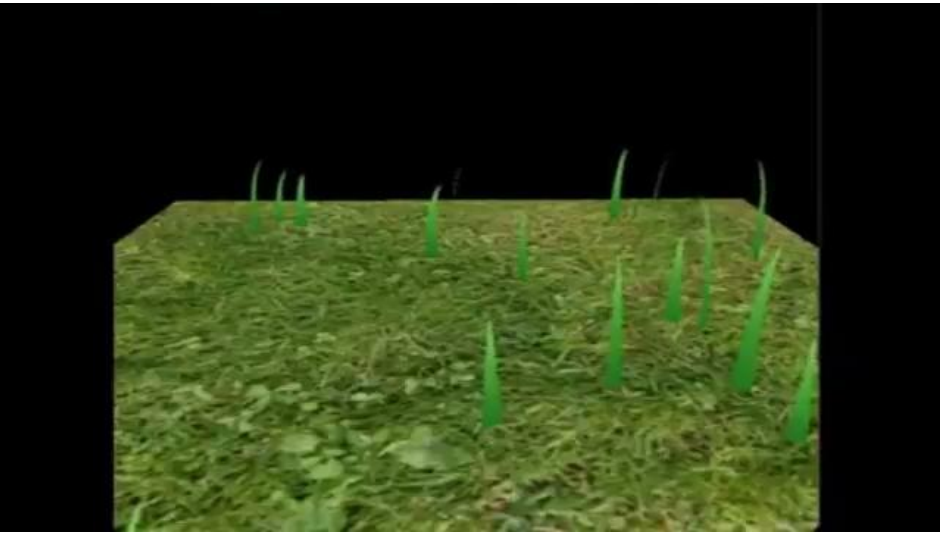


(a) Overview of the proposed architecture

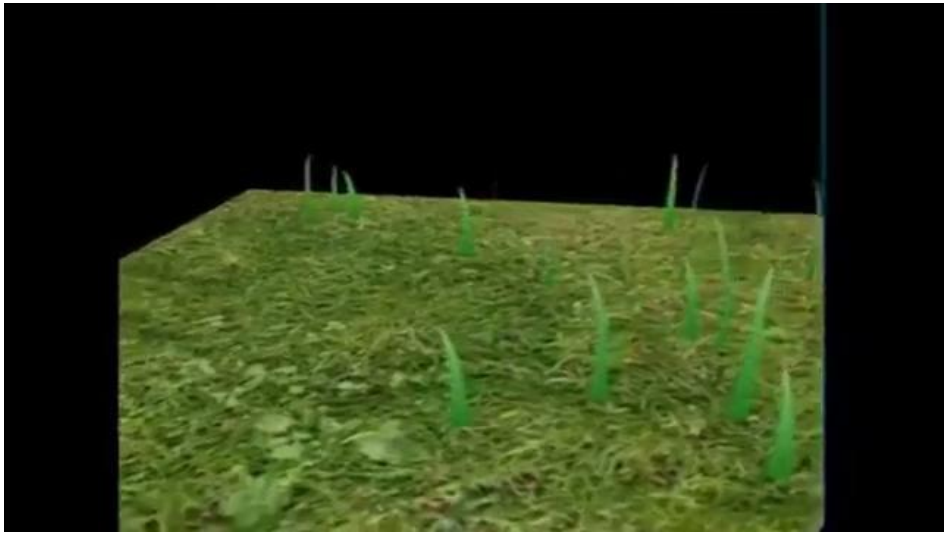


(b) Sampling procedure

Video Smoothing



Original (15 FPS)



2x Interpolation (30 FPS)

Video source: Aditya’s Vulkan Grass Renderer

Video Slow-Mo

Original

Slow-Mo (2x)

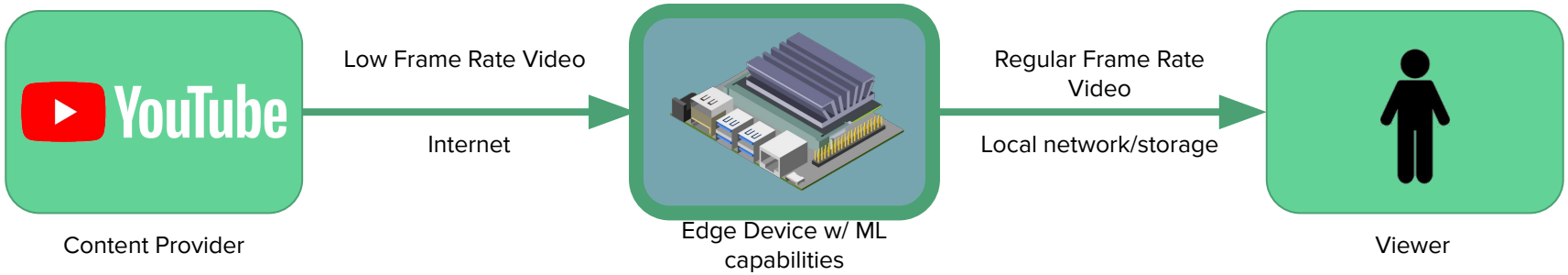
Video source: BBC Earth

Motivation

- High resolution content delivery is costly
  - High bandwidth due to large files
- Reducing frame rate can cut data transfer in half
  - Interpolate on edge devices to recover lost frames
- Artists can render movies while drawing fewer frames

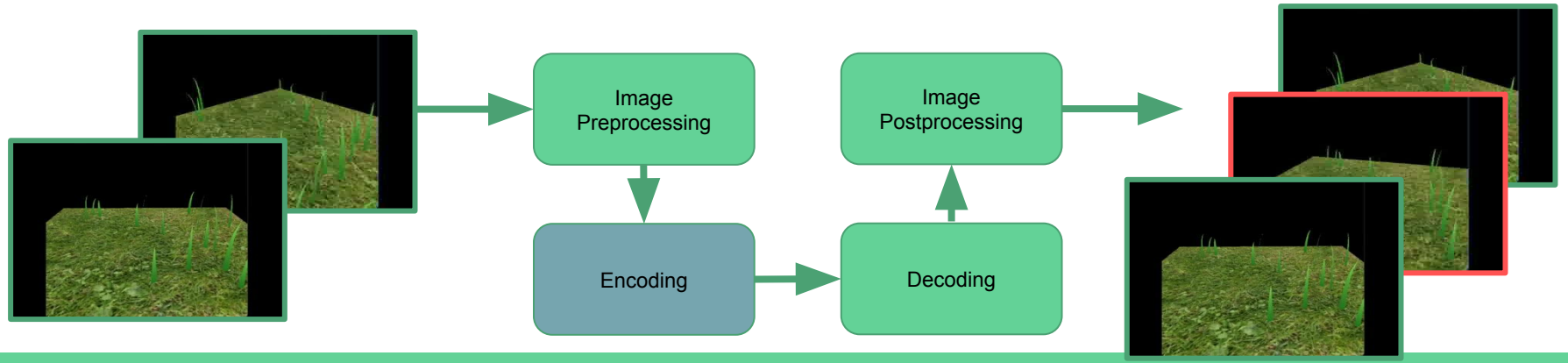
Motivation

- Want to implement PyTorch model more device agnostically
  - cuDNN would require fewer dependencies
  - Theoretically better performance
  - Can be implemented on embedded NVIDIA devices

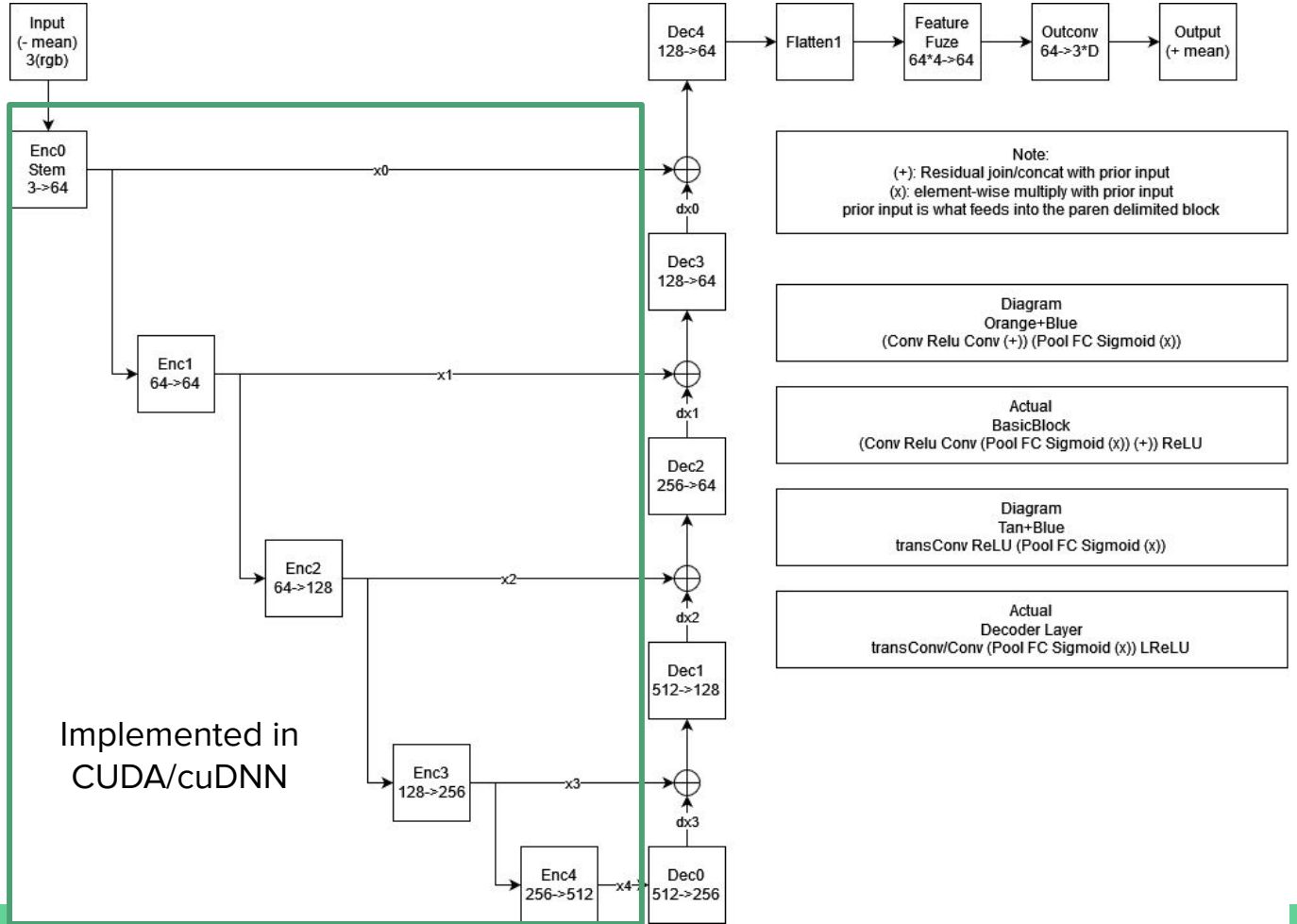


Our Work

- Implemented encoder of FLAVR model in cuDNN
- Developed custom kernels/modules for unsupported cuDNN operations
  - Leaky ReLU
  - Adaptive Average Pooling
- Created modular API for inference using cuDNN
  - Support for 2D+3D convolutions, activations, pooling, matrix math
- Created pipeline to interop cuDNN code with Torch model to leverage decoder

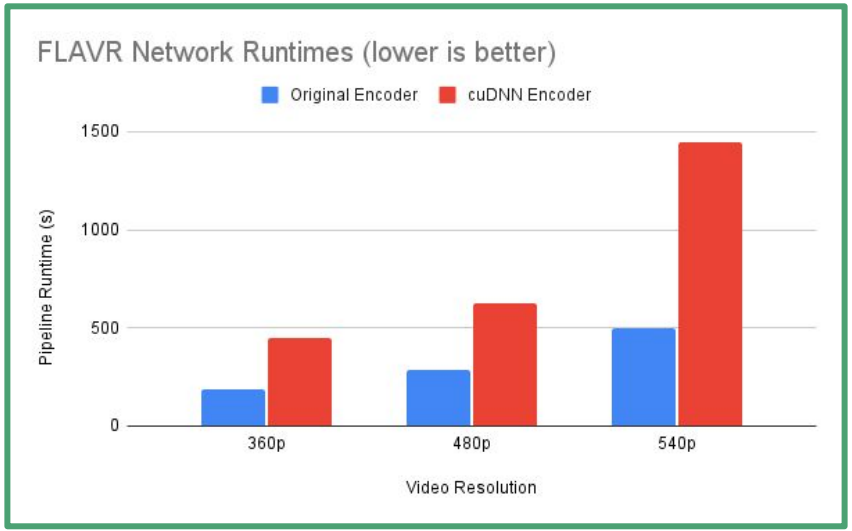


FLAVR Model


Input
(- mean)
3(rgb)
Enc0
Stem
3->64
Enc1
64->64
Enc2
64->128
Enc3
128->256
Enc4
256->512
Dec4
128->64
Flatten1
Feature
Fuze
64*4->64
Outconv
64->3*D
Output
(+ mean)
Dec3
128->64
Dec2
256->64
Dec1
512->128
Dec0
512->256
x0
x1
x2
x3
x4
dx0
dx1
dx2
dx3
Implemented in
CUDA/cuDNN
Note:
(+): Residual join/concat with prior input
(x): element-wise multiply with prior input
prior input is what feeds into the paren delimited block
Diagram
Orange+Blue
(Conv Relu Conv (+)) (Pool FC Sigmoid (x))
Actual
BasicBlock
(Conv Relu Conv (Pool FC Sigmoid (x)) (+)) ReLU
Diagram
Tan+Blue
transConv ReLU (Pool FC Sigmoid (x))
Actual
Decoder Layer
transConv/Conv (Pool FC Sigmoid (x)) LReLU

Results

- Videos produced are **almost indistinguishable** from original model
- API extracts away 120+ lines of cuDNN boilerplate
  - 3D Convolution can be constructed and run in < 10 lines
- Performance is slower, but optimizations can be made



Tested on GTX 1050Ti with Max-Q Design, 4 GB VMemory, Ubuntu 20.04

Model Performance

- The model produces very visually appealing outputs
  - In fact, differences are hard to spot
- Left: original (30 FPS); Right: upsampled (smoothened to 30 FPS)
  - Minor loss of detail on squirrel's leg and branch

Original

Upsampled

Future Areas of Exploration

- Implement U-Net style decoder in cuDNN
- Moving memory allocations to layer constructors
  - Reduces overhead of layer run function

- Use tensor cores for matrix math
- Use custom kernels to fuse operations

- Compare performance with TensorRT
- Train network with cuDNN

Top: original
Bottom: slow-mo (2x)
Image from PrettyAnimations

Acknowledgements

Special thanks to authors Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran upon whose code we based our pipeline

Papers:

Kalluri, T., Pathak, D., Chandraker, M., & Tran, D. (2020). Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv preprint arXiv:2012.08512*.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 6450-6459).

2D Convolutions Guide:

Peter Goldsborough: 2D Convolutions using cuDNN

GitHub

https://github.com/adityahota/CIS565-Final-Project-SlowMo

Thank you!
Q&A

Bonus: Purely Interpolated Video

Bonus: Flowers

Original

Slow-Mo (2x)

Video source: Gif It Me