

Real Time Video Upscaling

Joshua Lau, James Garcia, Kunal Barde (B0)

Use Case/Application Area

Problem:

- People want to watch old home videos and movies
- Don't want to plan for upscaling videos ahead of time
- Don't want/know how to do it online or on a computer

Solution:

Plug-and-play, real-time, video super-resolution device - Enhancing 240p videos to 1080p.



Solution Approach

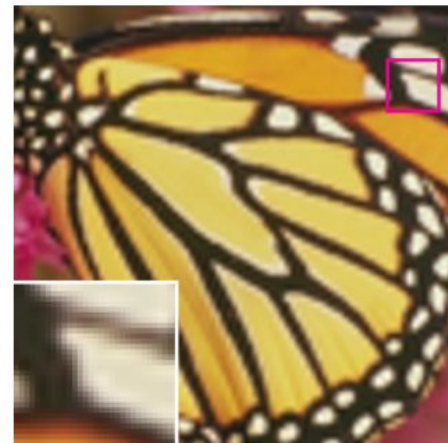
- Trained upscaling algorithm
 - Trained on a variety of videos (Dataset from CDVL)
 - Based off paper of similar scope (SRCNN)
- Final product must be quick enough to be real-time
 - No A/V desynchronization
 - No buffering/lag
- User-friendly
 - Plug-and-play
 - Output high quality video in 1080p
 - Portable enough to move hardware around without much hassle

Quantitative Requirements

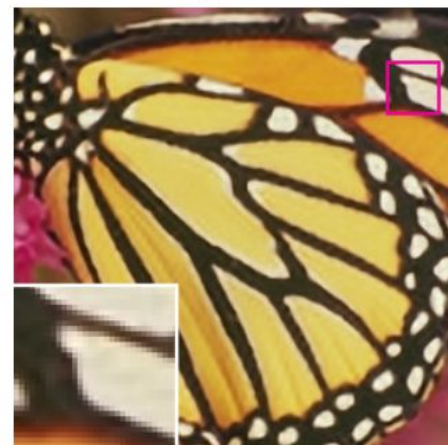
Requirement	Metric	Why?
Scaling Factor	4.5x	From 240p widescreen to 1080p full HD
Super Resolution Quality	SSIM>0.66	Beat average bicubic interpolation SSIM score to justify using CNN
Latency	<60ms	A/V synchronisation (via EBU R37) On-demand playability
Throughput	<33.4ms/frame (≥ 30 FPS)	Maintaining FPS without buffering or lag

Model Justification

- CNN vs. DSP (bicubic) vs. DSP (filters)
 - Bicubic is fast but throws away useful data
 - DSP filters require designer fine tuning
 - CNN only requires tuning of few hyperparameters + training
 - CNN have been shown with higher performance
- SSIM vs. VMAF
 - Further benchmarking on VMAF revealed excessive runtimes at-scale
 - SSIM is fast and standard across research papers



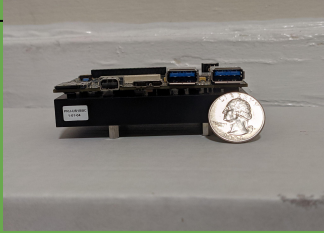
Bicubic / 24.04 dB



SRCNN / 27.95 dB

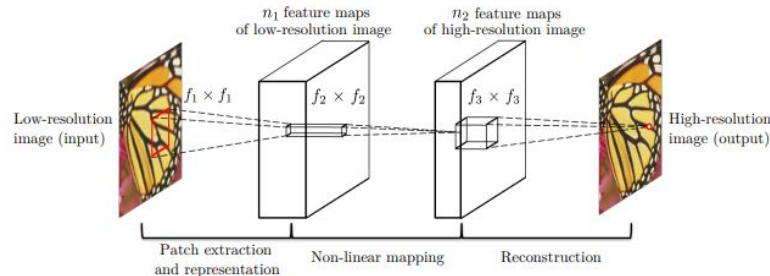
System Specification (Hardware Device)

Hardware Choice	Why / why not?
CPU	High Latency, Low Throughput
GPU	Insufficient Modes of Computation
ASIC	Too Static, Long Development Time
FPGA	Allows Acceleration
Ultra96	Native <u>1080p, 60FPS</u> DisplayPort output Dedicated <u>ARM core</u> to act as host for FPGA Onboard <u>USB and WiFi</u> capability for data I/O



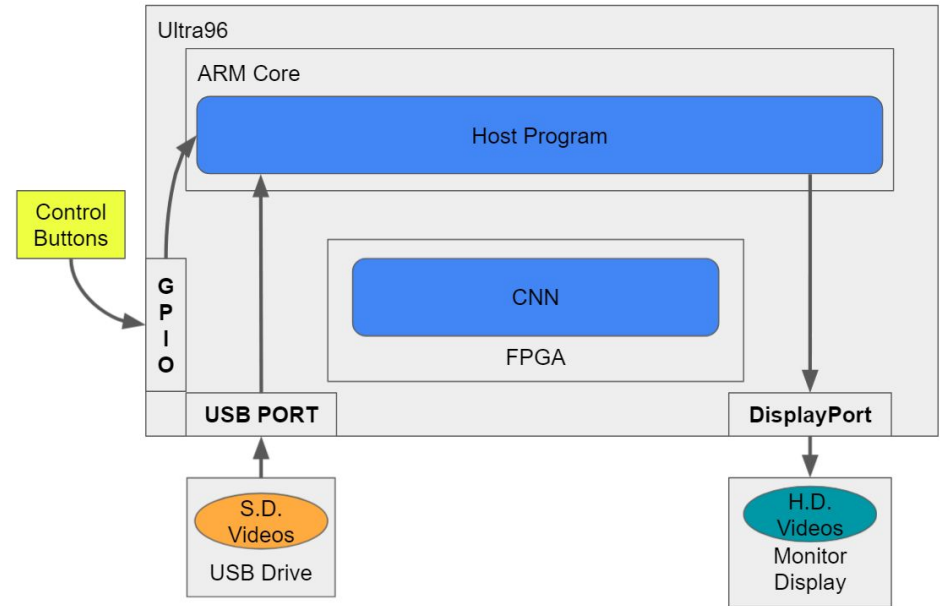
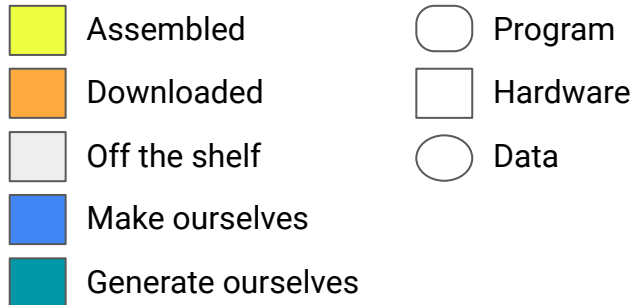
Implementation Plan (Software Model)

- Upscale image from lower to higher resolution using bicubic interpolation, which is part of the preprocessing of the frames of the video. (off the shelf algorithm)
- After the interpolation is applied to the image, we plan on feeding this through 3 convolution layers responsible for the super-resolution.
- First convolutional layer -> Patch extraction and representation
- Second convolutional layer -> Non-linear mapping
- Last layer -> Reconstructing into high resolution image
- Use MSE (Mean Squared Error) as loss function for training



Implementation Plan (Hardware Device)

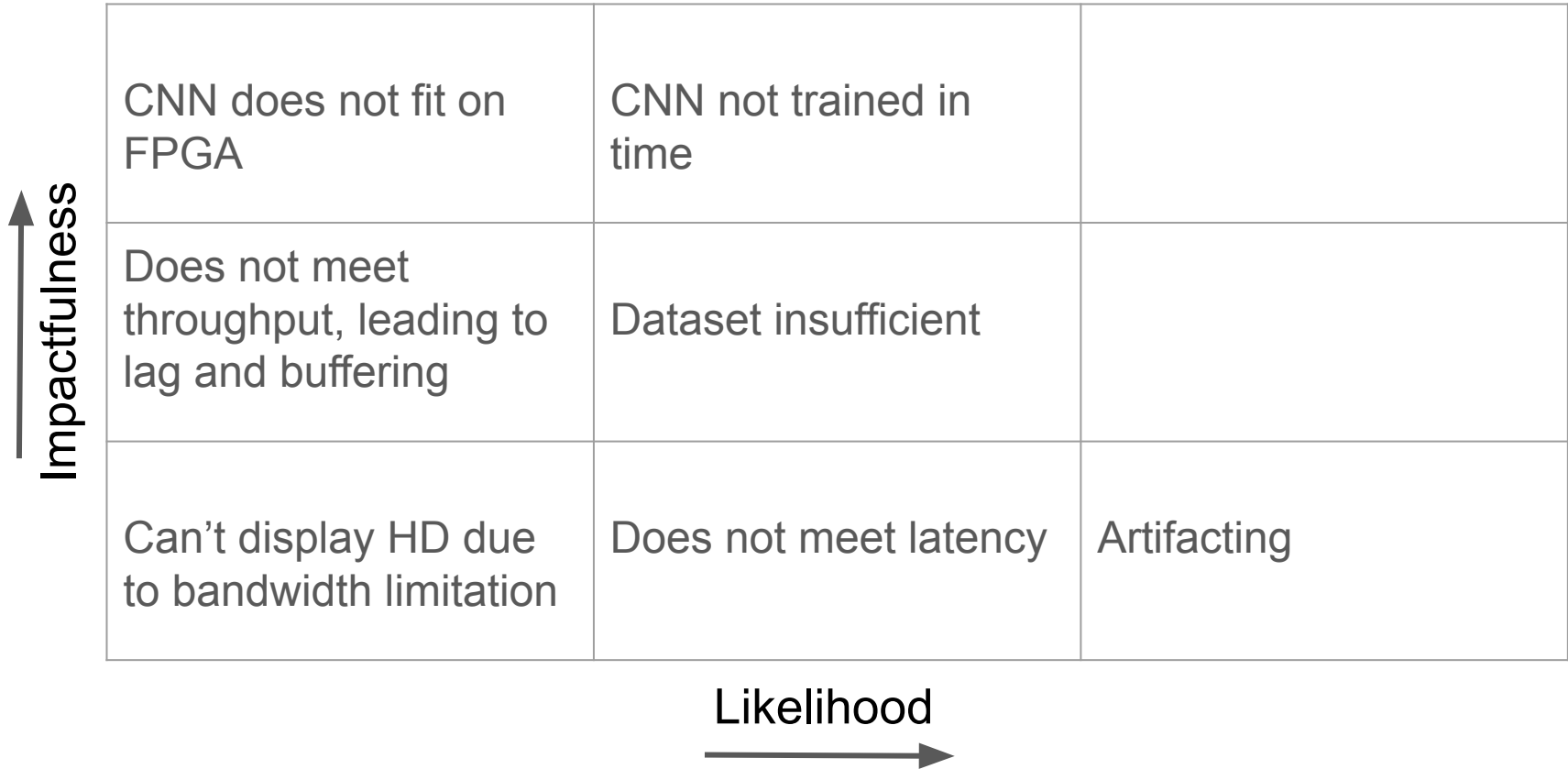
- Write CNN in Vivado HLS
- CNN will be accelerated by FPGA and HLS meta-language
- Weights hard-coded from pre-trained CNN



Metrics and Validation

Metric	Method	Value
Latency	Implement benchmark with ARM core for single image	<60ms
Throughput	Implement benchmark with ARM core for long runs	<33.3ms/frame >30FPS
Super Resolution Quality	Score output image with SSIM	SSIM>0.90

Risk Management



A risk management matrix with 'Impactfulness' on the vertical axis and 'Likelihood' on the horizontal axis. The matrix is a 3x3 grid. The top-left cell contains 'CNN does not fit on FPGA'. The top-middle cell contains 'CNN not trained in time'. The middle-left cell contains 'Does not meet throughput, leading to lag and buffering'. The middle-middle cell contains 'Dataset insufficient'. The bottom-left cell contains 'Can't display HD due to bandwidth limitation'. The bottom-middle cell contains 'Does not meet latency'. The bottom-right cell contains 'Artifacting'.

CNN does not fit on FPGA	CNN not trained in time	
Does not meet throughput, leading to lag and buffering	Dataset insufficient	
Can't display HD due to bandwidth limitation	Does not meet latency	Artifacting

Project Management

TASK TITLE	Project Proposal		Design Presentation				Interim Demo				Final Presentation
	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
	9/20	9/27	10/4	10/11	10/18	10/25	11/1	11/7	11/15	11/22	11/29
Hardware											
Acquire Ultra96	JSG										
Acquire Peripherals	JSG										
Research I/O	JSG	JSG	JSG + KB								
Implement I/O			KB + JSG								
Test I/O			JSG + KB								
Get Comms between ARM Core and FPGA		JSG									
Write Math Functions for CNN in Vivado HLS				JSG	JSG	JSG					
Validate HW				KB	KB	KB	KB				
Port SW model onto FPGA							JSG + KB				
Validating FPGA model against SV model								ALL			
Software											
Research DSP vs CNN models	ALL										
Acquire AWS Credits		KB									
Setup AWS		KB	KB + JL								
Acquire Dataset	JL										
Familiarize VMAF Documentation	JL	JL									
Research specific CNN models		KB + JL	KB + JL								
Benchmark VMAF		JSG + JL									
Research SSIM	KB										
Benchmark SSIM		KB + JL									
Benchmark CNN Models		JL + JSG	JL								
Develop Python Code for Training			JL + KB	JL	JL						
Train Model				JL	JL						
Test/Evaluate Model				JL	JL						
Log/Export Weights					JL						
Misc											
Slack/Slop									ALL	ALL	
Milestones											
Proposal Presentation	JSG										
Design Presentation			KB								
Design Review Report			ALL	ALL							
Interim Demo							ALL	ALL			
Final Presentation											JL

JSG
JL
KB

James
Joshua
Kunal

