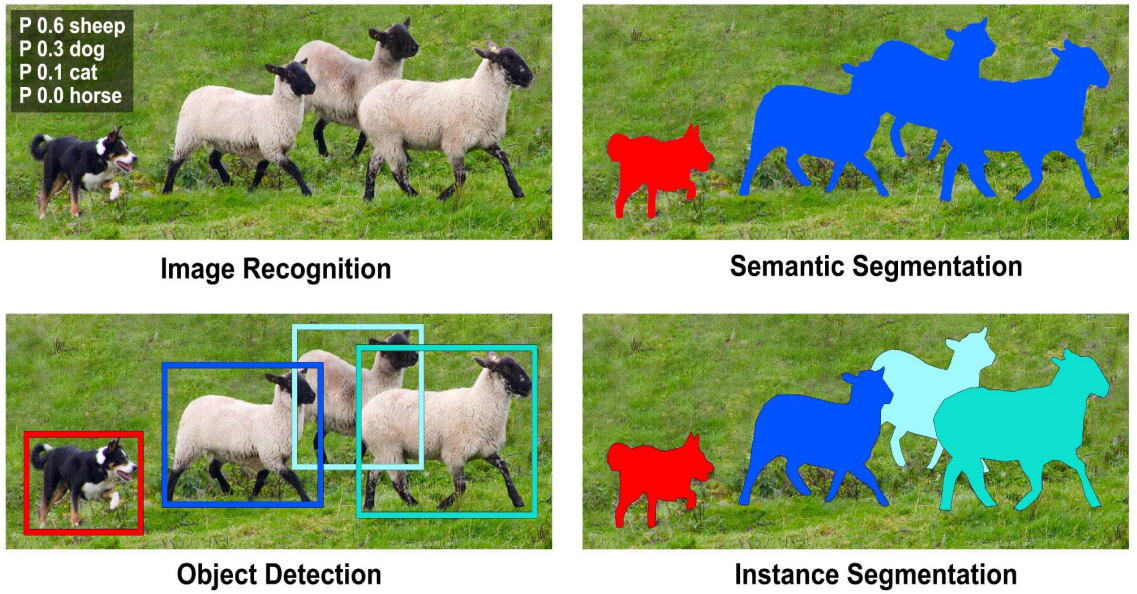
EFFICIENT FPGA IMPLEMENTATION
FOR VISUAL OBJECT SEGMENTATION

Harshit Sharma,Keerthi Priya Lankapalli, Sowmya Tippi Reddy

INTRODUCTION

- Computer Vision is used in various domains like agriculture, automation systems, autonomous vehicles, robots, security etc.

- Our task is focussed on FPGA simulation of models used for object segmentation by analysis of images from video frames taken from various input sources.



PROBLEM STATEMENT

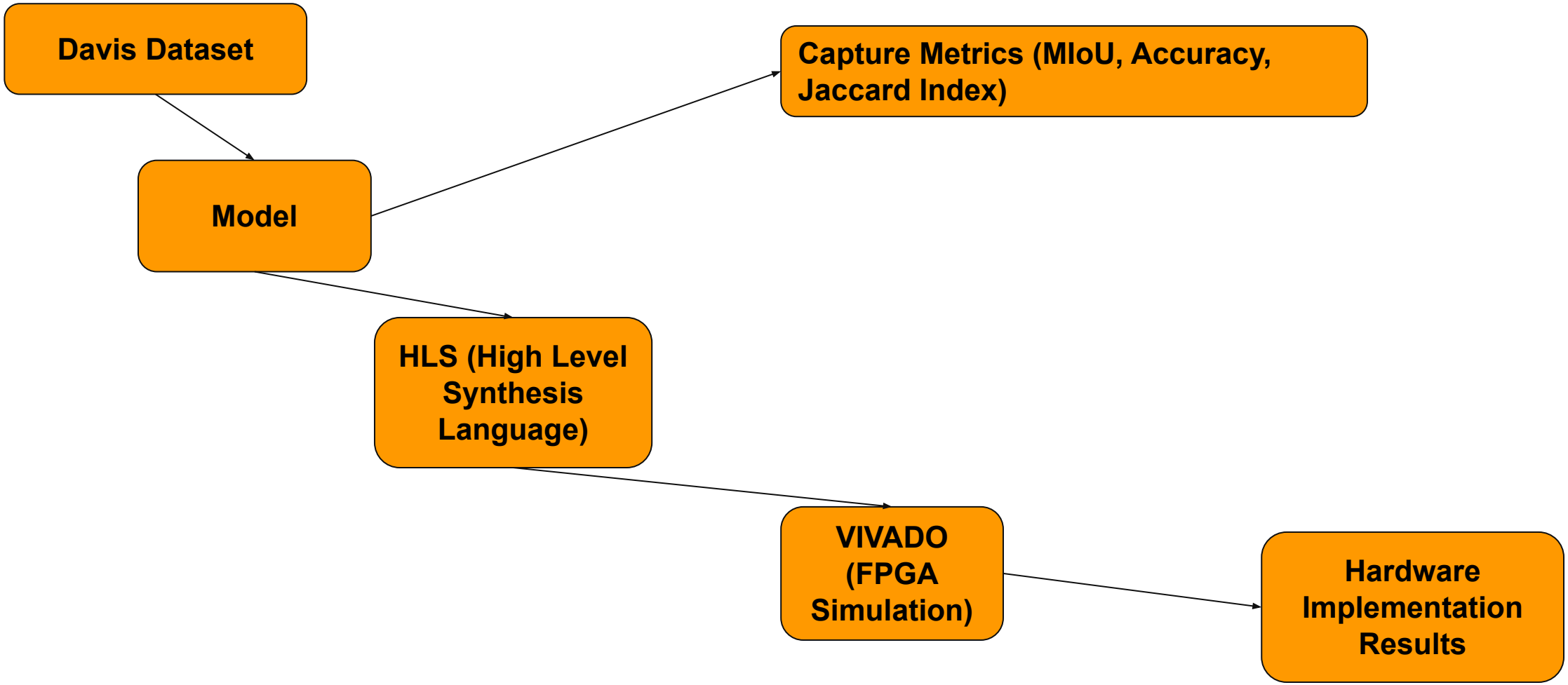
- Field-Programmable Gate Arrays (FPGAs) facilitate parallel processing and help to reduce latency in image processing tasks. eg. Semantic segmentation of videos

- FPGA implementation of the segmentation algorithms can provide faster results with a lower power consumption and high flexibility.

- In this project, visual object segmentation was performed on the DAVIS 2016 dataset, and the model was implemented on FPGA simulation using VIVADO.

- Analyze the performance of the model using metrics like accuracy, mean IoU, Jaccard Score and, constraint performance metrics like resource consumption (e.g. Latency, Digital Signal processors (DSPs), Look up tables (LUT), Flip-Flops (FF), BRAM (Block RAMs)).

PROPOSED SOLUTION



IMPLEMENTATION

- FPGA simulation on VIVADO does not work with models written as custom classes and the support for pytorch is limited to basic layers and networks only. So, we implemented the most implemented state of the art model UNET for the task of visual object segmentation.

- Since the model was lighter in comparison with the other state of the art models, we were able to train the model with our personal compute resources over selected vehicle specific classes from DAVIS 2016 class.

- Due to the nascent state of the research in FPGA implementation of ML models, not all the layers are available in the code infrastructure for conversion including Conv2DTranspose.

- So, we have replaced it with a combination of Upsampling 2D and Conv2D layers in the architecture which also proves to be more memory-efficient than using a Conv2D transpose layer. The Conv2D transpose layer requires more memory because it needs to compute a dense matrix multiplication, while UpSampling followed by a Conv2D layer only requires computing a sparse matrix multiplication.

- The Tensorflow U-Net model was modified as stated above and this model was used to perform the object segmentation task on DAVIS 2016 dataset. The model was trained on 1000 car images from the dataset.

IMPLEMENTATION CHALLENGES

- The unrolling limit in VIVADO is set to 4096 by default. Unrolling limit refers to the maximum number of iterations that can be unrolled in a loop.
- Increasing the unrolling limit would increase the size of the circuit and the memory required to synthesize.
- This would lead to longer synthesis times and higher resource utilization.
- Due to the default unrolling limit, the state-of-the-art models like U-Net and X-Mem couldn’t be simulated on FPGA in Vivado.

MODEL OPTIMIZATION

- The U-NET model with 450,000 parameters achieved the best results in terms of accuracy, jaccard index and mIoU score.
- Due to its, complexity and size it was not feasible for implementation in FPGA.
- So, we tried to optimize the model by reducing the number of parameters by controlling filter size and number of layers.
- The U-NET model parameters were reduced in steps from 450,000 to 7,000
- Due to this, the accuracy, jaccard index and mIoU scores were gradually reduced. Can we do better?

KNOWLEDGE DISTILLATION

- The transfer of knowledge from a large, complex model (teacher model) to a smaller model(student model).
- The goal is to improve the performance of the smaller model by leveraging the knowledge of the larger model.
- KD_450k is the teacher model and 21k_network is the student model (KD_450k-21k_network)
- Knowledge distillation also drastically reduced the power consumption from 1359 uJ to 306 uJ

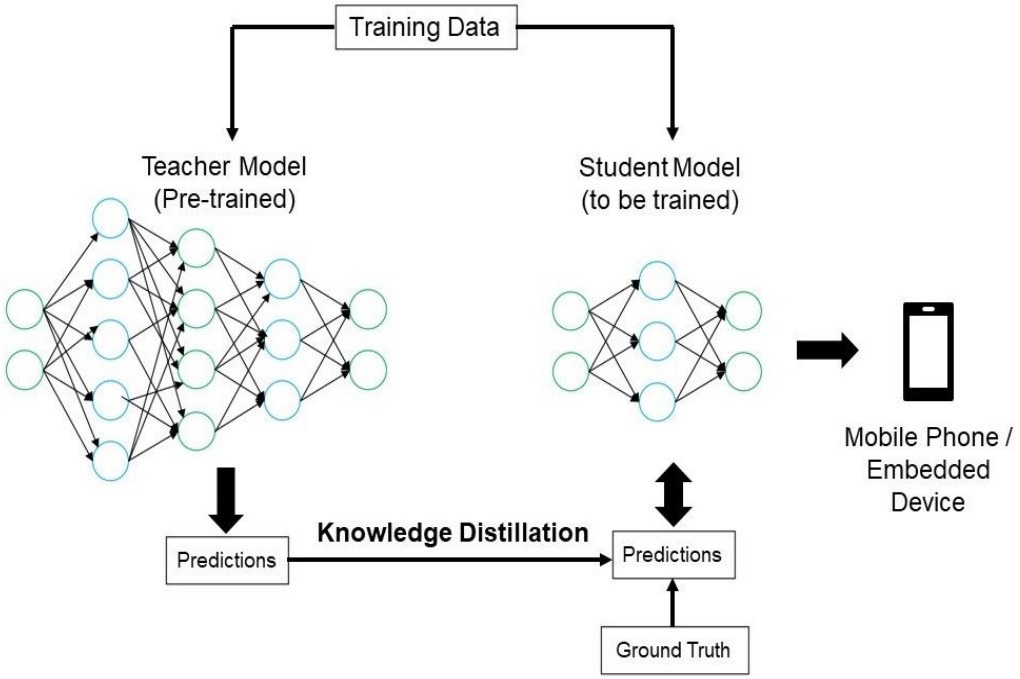


Fig 1: Knowledge Distillation

ADDITIONAL MODEL OPTIMIZATIONS

- **PRUNING** is a technique used to reduce the size of a deep neural network by removing unnecessary connections or neurons. It helps to reduce number of parameters and hence computation.

- **QUANTIZATION** is a technique used to reduce the precision of weights and activations in a deep neural network. This aims at reducing the number of bits used to represent the weights and activations.

- Quantization significantly reduces the memory usage and improves the computational efficiency. Our models are currently use a 16 bit quantization factor while converting to FPGA project/model.

MODEL EVALUATION

Latency = clock rate * cycles = 4.375 ns * 68801 ~= **3.01 ms**

	Software Performance Metrics			**Hardware Performance Metrics (Resource Optimization)**				
Models	Accuracy	MoU	Jaccard Index	BRAM	LUT	FF	DSPs	Total Energy
Unet-Lite-7K	0.959	0.872	0.870	3748	5159	210362	422097	62.28 uJ
KD-450k-21K	**0.985**	**0.959**	**0.959**	**5122**	**14998**	**491374**	**959521**	**305.92 uJ**
KD-450k-12K	0.983	0.955	0.955	4124	10025	347353	663348	71.07 uJ
KD-450k-7K	0.979	0.926	0.927	3748	5119	210505	423232	62.28 uJ

Table 1: Overall Software and Hardware metrics and results for various models

RESULTS

- The results of evaluating the model on the test set are described in Table 1.
- The results of some sample predictions are displayed in Fig 2.
- We observed a continuous decrease in loss and continual increase in the accuracy values during the training process.
- These results prove that this model outperformed the existing state-of-the-art models for object segmentation tasks.

Test Metrics	Unet-Lite-450k	KD-450k-21K
Accuracy	0.985	0.985
IoU Coefficient	0.966	0.959
Jaccard Index	0.966	0.959

Table 2: Evaluation Results

Fig 2: Sample predictions from images of DAVIS-2016 with 21k parameters with Knowledge Distillation

Fig 3: Sample predictions from images of DAVIS-2016 with 450k parameters model

CONCLUSION & FUTURE WORK

- We have successfully implemented the KD_450k_21k model on an FPGA simulation using Vivado.
- We have observed that Knowledge distillation seems to be an effective way of reducing model complexity and resource consumption for ML inference, which is a key factor for FPGA implementation.
- Hence, Knowledge distillation could be used to learn from more complex models like X-Mem.
- We plan to work on the additional optimization techniques like Auto Quantization and analyse their performance in future.

THANK YOU