

# Real-Time Hyperspectral Signal Processing and Classification

2025 Final Presentation

Nash Rickert

Electrical and Computer Engineering Department, REU  
Montana State University

July 30, 2025

## Overview

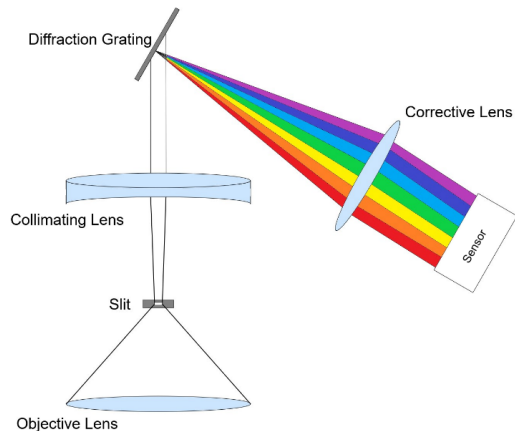
- Our goal is to implement **real-time hyperspectral classification**. This means that our model would be integrated with the data-collection process so that there is no need to handle large amounts of data after collection.
  - Currently it is often the case that large amounts of hyperspectral data will be collected in the field, then need to be processed separately before any results can be acted upon.
- This naturally means that the main goal of our project is to minimize the latency of the classification process so it matches the pace of data collection.

## Motivation

Real-time classification would provide immediate insight to field workers, allowing them to make important decisions quickly.

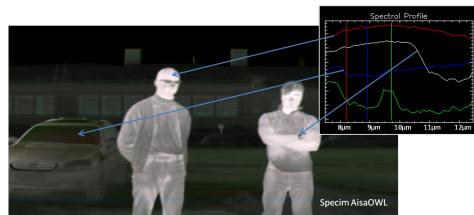
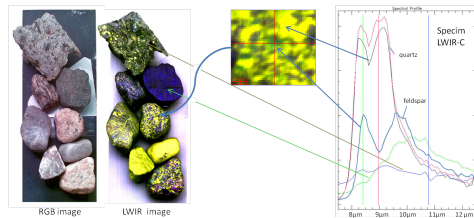
# Background: Hyperspectral Imaging (HSI)

- HSI uses diffraction to sample a continuous range of spectral bands.
- Because of its high dimensionality, hyperspectral data tends to be large and difficult to process.
  - A typical RGB image has 3 channels. Our sensor uses 1608. 535x larger.



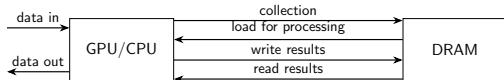
# Background: HSI for Classification

- Hyperspectral imaging data can be used to classify objects in images. Their high wavelength spectrum makes them especially useful for environmental monitoring, agriculture, and more.
- For our project, we're interested in identifying regions of forests with an abundance of ground fuel which puts them at risk of forest-fires.

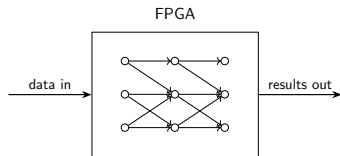


# Background: FPGAs

- An **FPGA** (Field Programmable Gate Array) is a programmable circuit.
- They are built out of logic elements, DSP slices (for performing simple addition and multiplication), and local memory (consisting of static RAM).
- They consist of a 'fabric' of resources where everything is interconnected (hence programmable).



**Figure:** Latency overhead of using a GPU for computation



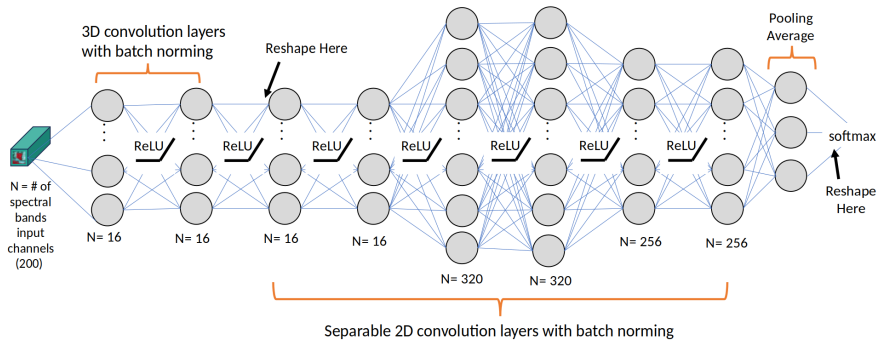
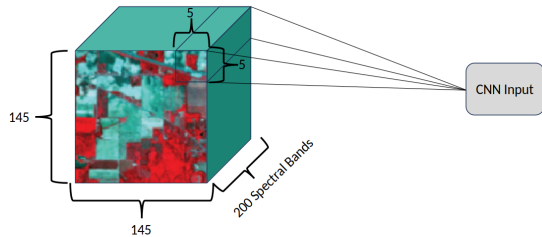
**Figure:** Lack of latency overhead of an FPGA inference stream

## Research Question

How much can different neural network model architectures for classifying hyperspectral data improve latency in an FPGA implementation without sacrificing accuracy?

## Baseline

- Previous work at MSU successfully reduced band dimension on a HSI dataset and performed high accuracy classification on it.
- 99.7% accuracy using a large convolutional neural network.
- Performs roughly 2 billion elementary add/multiply operations for each pixel classified.



## CNN Drawbacks

- A CNN allows us to gain spatial information about our image by taking into account pixels around each pixel
  - For real time classification, this requires storing data so our kernel can pass over it, interrupting the flow of data through our FPGA
- The CNN implementations are also quite large and might not be viable for our hardware needs.

## Kolmogorov Arnold Networks (KAN)

- Directly learns the nonlinear activation functions of a network.
- In an FPGA, these functions could be encoded directly as **lookup tables**, allowing us to approximate results much more quickly.
- If pixel-classification is viable, we can avoid storing/transferring data from memory for a CNN kernel.



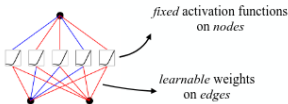
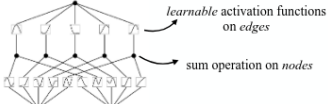
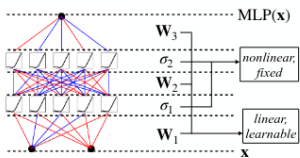
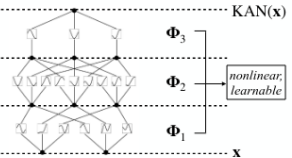
Model	<b>Multi-Layer Perceptron (MLP)</b>	<b>Kolmogorov-Arnold Network (KAN)</b>
Theorem	<b>Universal Approximation Theorem</b>	<b>Kolmogorov-Arnold Representation Theorem</b>
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(e)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	<p>(a)</p>  <p><i>fixed activation functions on nodes</i></p> <p><i>learnable weights on edges</i></p>	<p>(b)</p>  <p><i>learnable activation functions on edges</i></p> <p><i>sum operation on nodes</i></p>
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	<p>(c)</p>  <p><math>\mathbf{W}_3</math></p> <p><math>\sigma_2</math></p> <p><math>\mathbf{W}_2</math></p> <p><math>\sigma_1</math></p> <p><math>\mathbf{W}_1</math></p> <p><math>\mathbf{x}</math></p> <p><i>nonlinear, fixed</i></p> <p><i>linear, learnable</i></p>	<p>(d)</p>  <p><math>\Phi_3</math></p> <p><math>\Phi_2</math></p> <p><math>\Phi_1</math></p> <p><math>\mathbf{x}</math></p> <p><i>nonlinear, learnable</i></p>

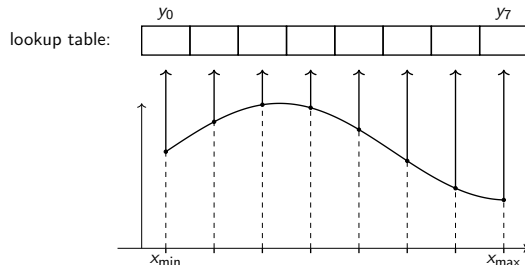
Figure 0.1: Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs)

- To properly benchmark and implement, we need to shift the models from the high level and opaque python implementations to something lower level.
- To do this, I transferred both the baseline CNN implementation and the KAN to C (for the latter, I created a custom C library that can be called directly from python).
  - This allows us to do preliminary benchmarking and provides an interface to swap out the computational parts of the classification directly to the FPGAs.

# More on Lookup Tables

- Each lookup table consists of both the actual stored values as well as the meta-data necessary do our lookups.

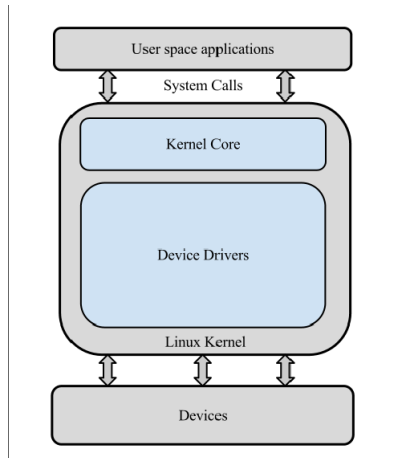
```
24 struct lkup_tbl {  
25 // The values of the lookup table  
26     float tbl[TBL_SIZE];  
27 // x val associated with table[0]  
28     float xmin;  
29 // x val associated with table[TBL_SIZE - 1]  
30     float xmax;  
31 // dist between x values. Roughly (xmax - xmin) / TBL_SIZE  
32     float xdists;  
33 // the reciprocal of xdists for division  
34     float inv_xdists;  
35 };  
36
```



- Thus to do a lookup on our table, we use `xmin`, `xmax`, `xdists`, and `inv_xdists` to do linear interpolation between the lookup table entries nearest to our input.
- These lookup tables take up a significant amount of memory – much more than is available locally in our FPGA (foreshadowing).

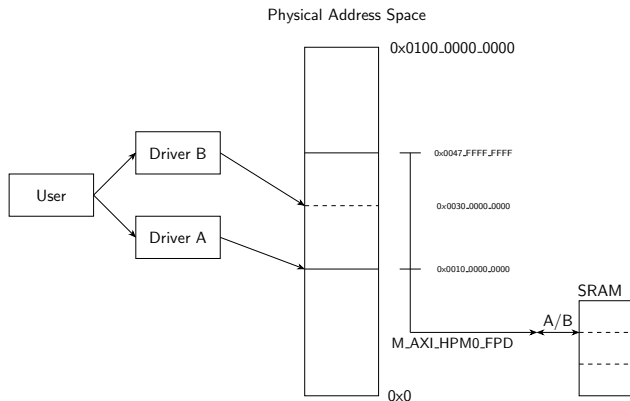
# Device Driver Introduction and Motivation

- A **device driver** is a piece of software that provides an interface between an operating system and the hardware devices it runs on.
- We want to write a driver that can transfer large amounts of data from system memory (DRAM) to local fabric memory (SRAM) on the FPGA efficiently.



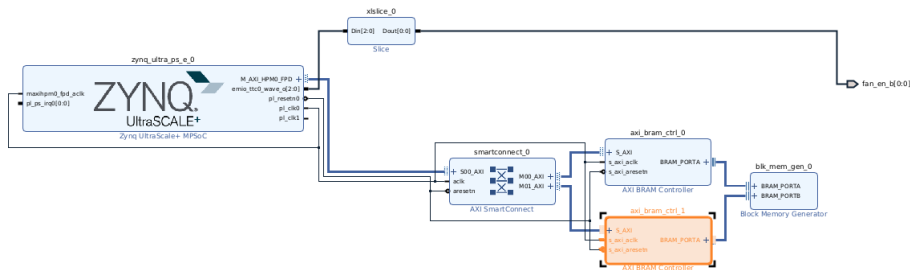
# Device Driver Design

- Significant portions of our board's address space are reserved for data buses and peripheral devices.
  - Thus we can use our driver to write to a portion of the address space that corresponds to a data bus accessible to our FPGA.
- A **device tree** in Linux describes the components of the hardware that Linux is running on.
- We create two entries in our device tree corresponding to addresses accessible to the FPGA through an AXI bus. The bus connects to separate ports in the static RAM of our FPGA.



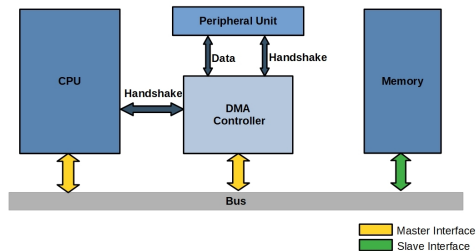
# Device Driver Implementation

- My driver combines elements of both a platform driver and a block driver (specifically a block ram-disk driver).
- Fabric design:



# Future Work

- KAN accuracy tradeoffs with different lookup table sizes and data type precisions.
- Model pruning.
- Benchmarking C implementations on the FPGA.
- Device driver latency overhead and pipelining.
- Direct Memory Access (DMA) Implementation.



## Project Summary

- This project began as an investigation of model architectures that would be suitable for hyperspectral classification on an FPGA.
- As it progressed, it became clear that a significant bottleneck for a real implementation would be loading data onto an FPGA, so I created a device driver that could accomplish that.

### Summary

Our goal is to achieve real-time HSI classification using a custom FPGA implementation. The real-time nature of this implementation would be highly useful for field work and real applications, but requires overcoming engineering challenges related to our hardware.



# Acknowledgements

- I would like to acknowledge my PI Ross Snider for the direction and assistance he has provided on the project.
- I would also like to thank Nat Sweeney, Zackery Backman, and Dirk Kaiser for the work that they have done and continue to do on the project.
- And a special thank you to my peers in the REU program that have made this summer fun and memorable for me, and for the support and encouragement they have provided throughout the summer.
  - And especially **Isabel Garcia** for reviewing my final report.
- This material is based upon work supported by the National Science Foundation under Grant No. 2349091.

# References I

- [1] CONTRIBUTORS, W. C.  
Hyperspectralcube.  
<https://commons.wikimedia.org/wiki/File:HyperspectralCube.jpg>, 2007.
- [2] CONTRIBUTORS, W. C.  
Hsi lwir stones.  
[https://commons.wikimedia.org/wiki/File:HSI\\_LWIR\\_stones.png](https://commons.wikimedia.org/wiki/File:HSI_LWIR_stones.png), 2011.
- [3] CONTRIBUTORS, W. C.  
Specim aisaowl outdoor.  
[https://en.wikipedia.org/wiki/File:Specim\\_aisaowl\\_outdoor.png](https://en.wikipedia.org/wiki/File:Specim_aisaowl_outdoor.png), 2011.
- [4] GANESH, P.  
Knowledge distillation: simplified, 2019.  
Available at: <https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764/>.
- [5] HUANG, P.-Y.  
Simpledarkblue beamer theme, 2021.  
GitHub repository.
- [6] KAISER, D.  
Eele490 undergraduate research, 2025.  
Available at: <https://github.com/rksnider/SmartHyperspectralCamera/blob/main/Distillation/Paper>.
- [7] LIU, Z., WANG, Y., VAIDYA, S., RUEHLE, F., HALVERSON, J., SOLJAČIĆ, M., HOU, T. Y., AND TEGMARK, M.  
Kan: Kolmogorov-arnold networks, 2025.
- [8] LOBANOV, V., FIRSOV, N., MYASNIKOV, E., KHABIBULLIN, R., AND NIKONOROV, A.  
Hyperkan: Kolmogorov-arnold networks make hyperspectral image classifiers smarter, 2024.

# References II

- [9] MORALES, G., SHEPPARD, J., LOGAN, R., AND SHAW, J.  
Hyperspectral band selection for multispectral image classification with convolutional networks.  
In *2021 International Joint Conference on Neural Networks (IJCNN)* (July 2021), IEEE, p. 1–8.
- [10] MORALES, G., SHEPPARD, J. W., LOGAN, R. D., AND SHAW, J. A.  
Hyperspectral dimensionality reduction based on inter-band redundancy analysis and greedy spectral selection.  
*Remote Sensing* 13, 18 (2021).
- [11] STOYANOV, Y.  
Direct memory access (dma) in embedded systems.  
<https://open4tech.com/direct-memory-access-dma-in-embedded-systems/>.
- [12] SWEENEY, N.  
Development of a smart unmanned system hyperspectral imager (sushi) for ground fuel characterization.
- [13] WANG, Y., YU, X., GAO, Y., SHA, J., WANG, J., GAO, L., ZHANG, Y., AND RONG, X.  
Spectralkan: Kolmogorov-arnold network for hyperspectral images change detection, 2024.
- [14] ZAKHAROV, I., MUTILIN, V., NOVIKOV, E., AND KHOROSHILOV, A.  
Generating environment model for linux device drivers.

# The End