DL-GPU PROJECT 2:

Exploration of TPU Architectures for Optimized Transformer Performance in Image Detection of Drainage Crossings

Presented by Aikaterini Panteleaki, Amir Nazeri, and Denys Godwin

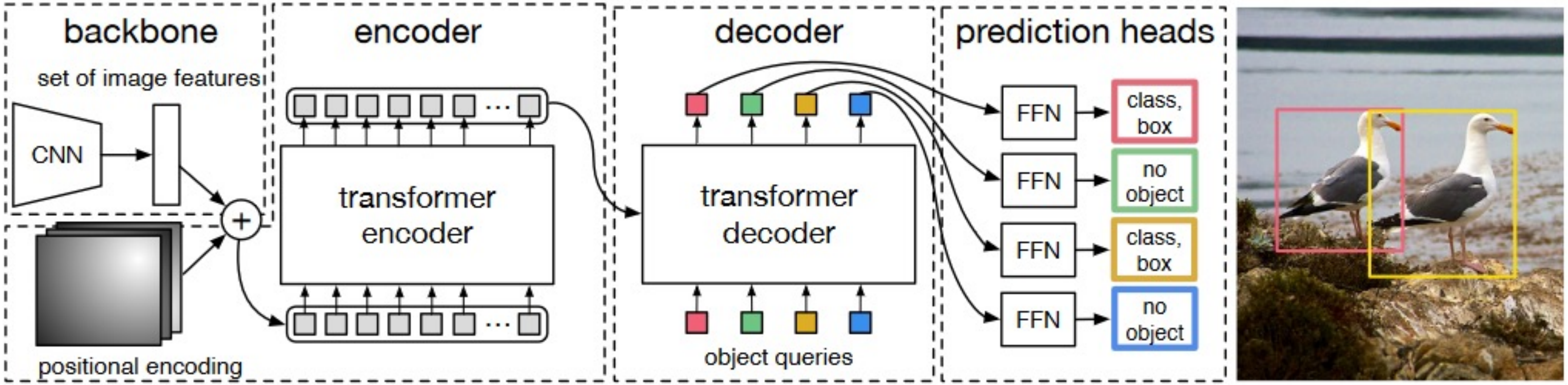
PROJECT DESCRIPTION

- As high-resolution digital elevation models (HRDEMs) continue to improve the mapping of hydrographic features, the challenge of accurately identifying under-road drainage structures like culverts remains significant.
- Utilizing the Transformer architecture, renowned for its ability to handle complex visual data through self-attention mechanisms, this project refines image analysis techniques for environmental science.
- By optimizing the performance of Transformer models through targeted TPU architecture exploration using the Scalesim tool , this project minimizes latency, compute time, and cost while maintaining high accuracy when using the model for inference.

THE MODEL

DETECTION -TRANSFORMER (DETR)

- Introduced by Facebook AI in 2020.



DETR FEATURES

- Key advantages of DETR over traditional object detection:
- End-to-end training pipeline: No need for hand-crafted component in Faster RCNN or YOLO
- Transformer architecture: transformer component leverages self-attention mechanism to capture complex and long-dependencies.
- Simplified architecture: No RPN or non-maximum suppression (NMS). Abstracted by Bipartite matching loss.
- Easier Scalability: Embedded transformer accelerates processing of large data.

COCO DATASET

- COCO2017 dataset, containing 118k training images and 5k validation images.
- 7 instances per image on average and up to 63 instances in a single training image.
- Includes annotated with 80 object categories and 5 captions describing the scene..


```json
"categories": [
    {
        "id": 1,
        "name": "car",
        "supercategory": "vehicle"
    },
    {
        "id": 2,
        "name": "truck",
        "supercategory": "vehicle"
    }
]
```

```json
"images": [
    {
        "id": 1,
        "width": 640,
        "height": 480,
        "file_name": "000000397133.jpg",
        "license": 1,
        "flickr_url": "https://www.flickr.com/photos/adrianrosebrock/397133",
        "coco_url": "http://images.cocodataset.org/val2017/000000397133.jpg",
        "date_captured": "2013-11-14 17:02:52"
    },
```

DETR RESULTS ON COCO

- Inference on validation set:

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.420
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.624
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.442
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.205
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.458
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.611
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.333
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.533
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.574
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.312
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.628
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.805
```

MODELS' COMPARISON

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

DATASET PREPARATION

THE DATASET

- The original dataset as shared is formatted to match Pascal VOC standard for object detection
- Each single-channel 800x800 .tif file has a corresponding .xml file containing the annotations for the image
- .tif files represent LiDAR-derived High-Resolution Digital Elevation Model (HRDEM) data
  - Elevation represented as units above sea level
  - Dataset stats:

Mean	Standard Devation	Max	Min	Mean Elevation Difference
3,960.79	10,892.09	47,075	3.48	114.2667

PRE-PROCESSING THE DATASET

- Images with invalid pixel values (either –1E38 or anomalously high values) are filtered in prepare_dataset.ipynb:

```
df.at[index, 'valid'] = False if (np.max(data) - np.min(data)) > 10000 else True
```

- Images are converted to relative elevation from absolute elevation with code from get_normalization.ipynb:

```
min_value = data.min()
new_data = data - min_value
```

- Then, the mean and standard deviation is extracted from the entire dataset with the code on the right to generate new stats:

Mean	Standard Devation	Max	Min	Mean Elevation Difference
55.82	185.58	3581	0	114.36

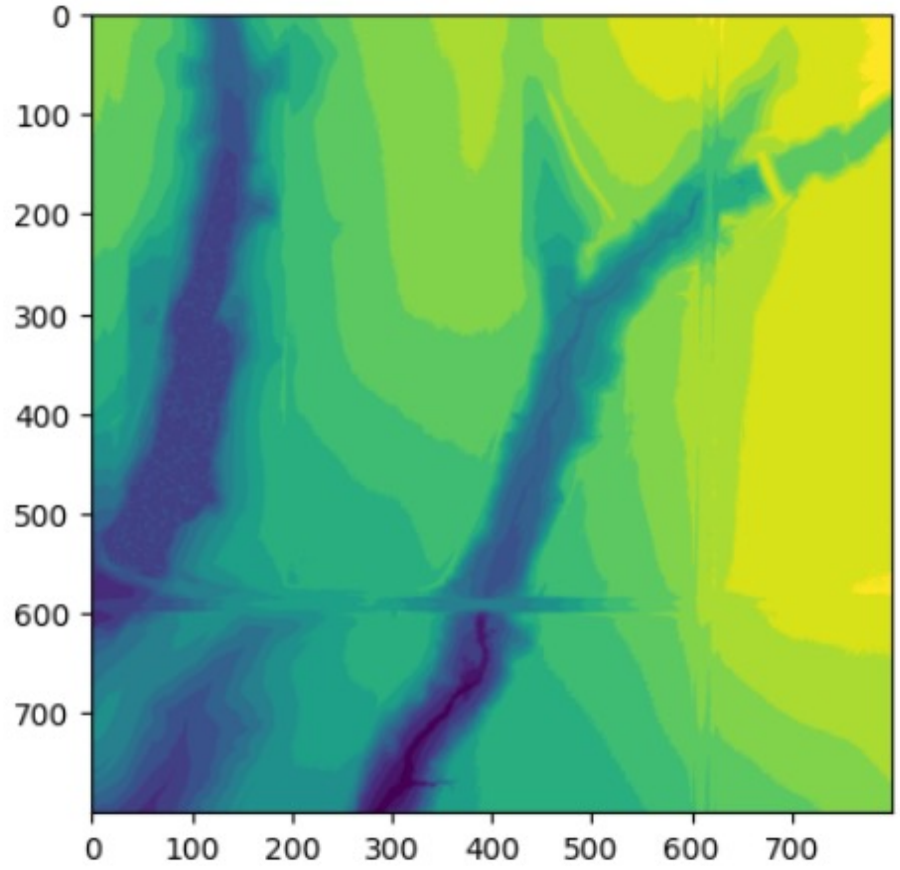
```
psum    = torch.tensor([0.0])
psum_sq = torch.tensor([0.0])
min = torch.tensor

index = 0
# loop through images
for inputs in tqdm(image_loader):
    psum += inputs.sum()
    psum_sq += (inputs ** 2).sum()
    index += 1

# pixel count
count = len(tif_files) * 800 * 800

# mean and STD
total_mean = psum / count
total_var  = (psum_sq / count) - (total_mean ** 2)
total_std  = torch.sqrt(total_var)
```

MODIFYING THE DATASET TO FIT THE MODEL

- DETR requires a COCO-formatted dataset
- Image chips were split randomly into train (70%), validate (20%), and test (10%) directories
- Image ids and properties were stored in a COCO-formatted json
- Annotation bounding boxes were extracted from the xml files and compiled into the COCO-formatted JSON, where each annotation is associated with an image id
  - This is implemented in prepare_dataset.ipynb

MODIFYING THE MODEL TO FIT THE DATASET

- The default COCO dataloader imported from pycocotools uses PIL to load imagery in RGB format
  - This leads to degradation of resolution (see example, right)
  - Solved by writing a custom dataloader which uses rasterio to load .tif files, repeat along the first axis for 3 channels, and convert to torch.Tensor format.



CHALLENGES IN ADAPTING DETR TO HRDEM DATA

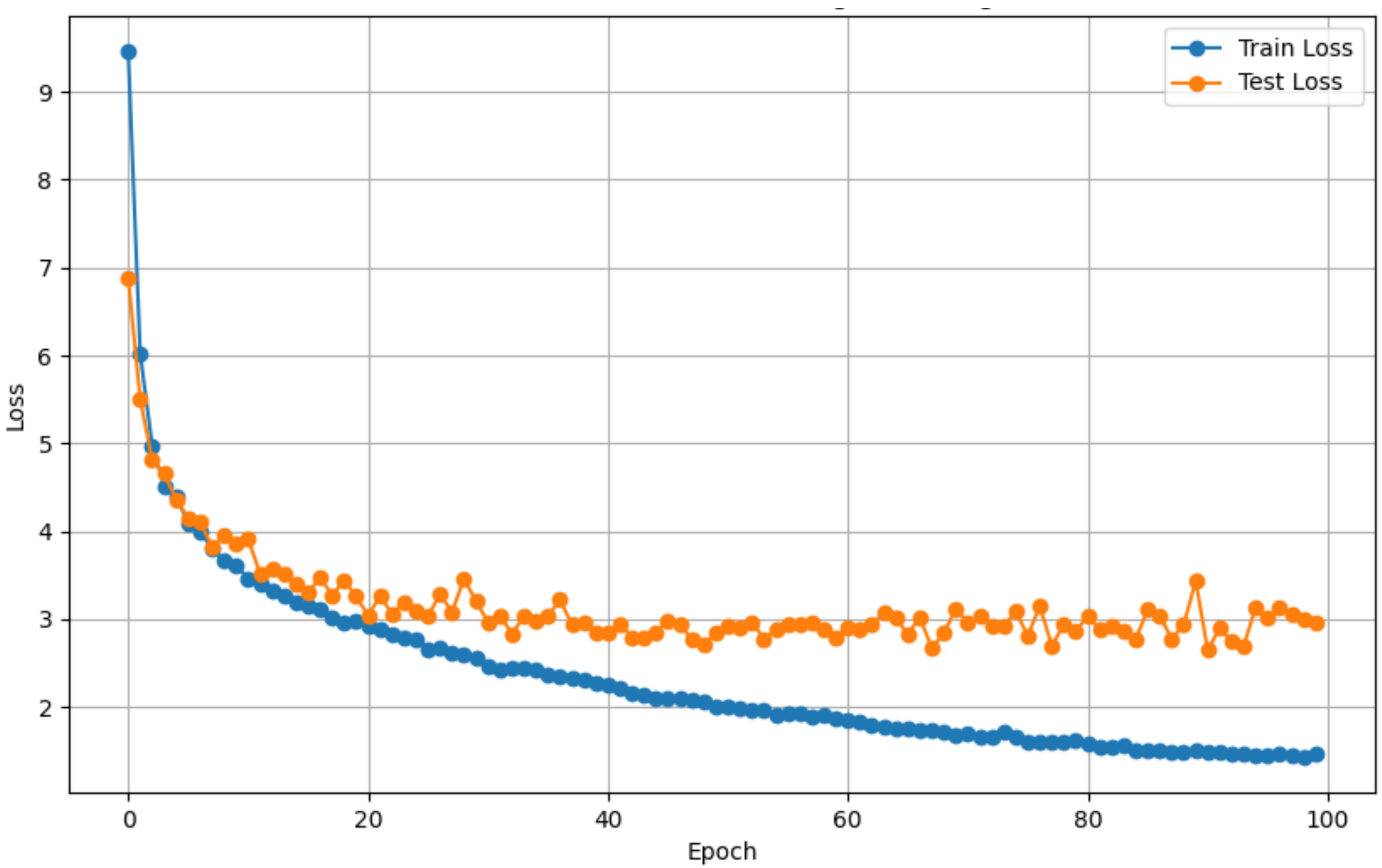
- The data augmentations and transforms expect input in a PIL Image.Image format, and therefore will need to be modified to expect torch.Tensor input
- COCO formatting requires precision in conversion
  - Issues arose around using str for int fields and vice versa
  - Issues arose around image ids not matching correctly with corresponding annotations

MODEL PERFORMANCE

TEST DESCRIPTION

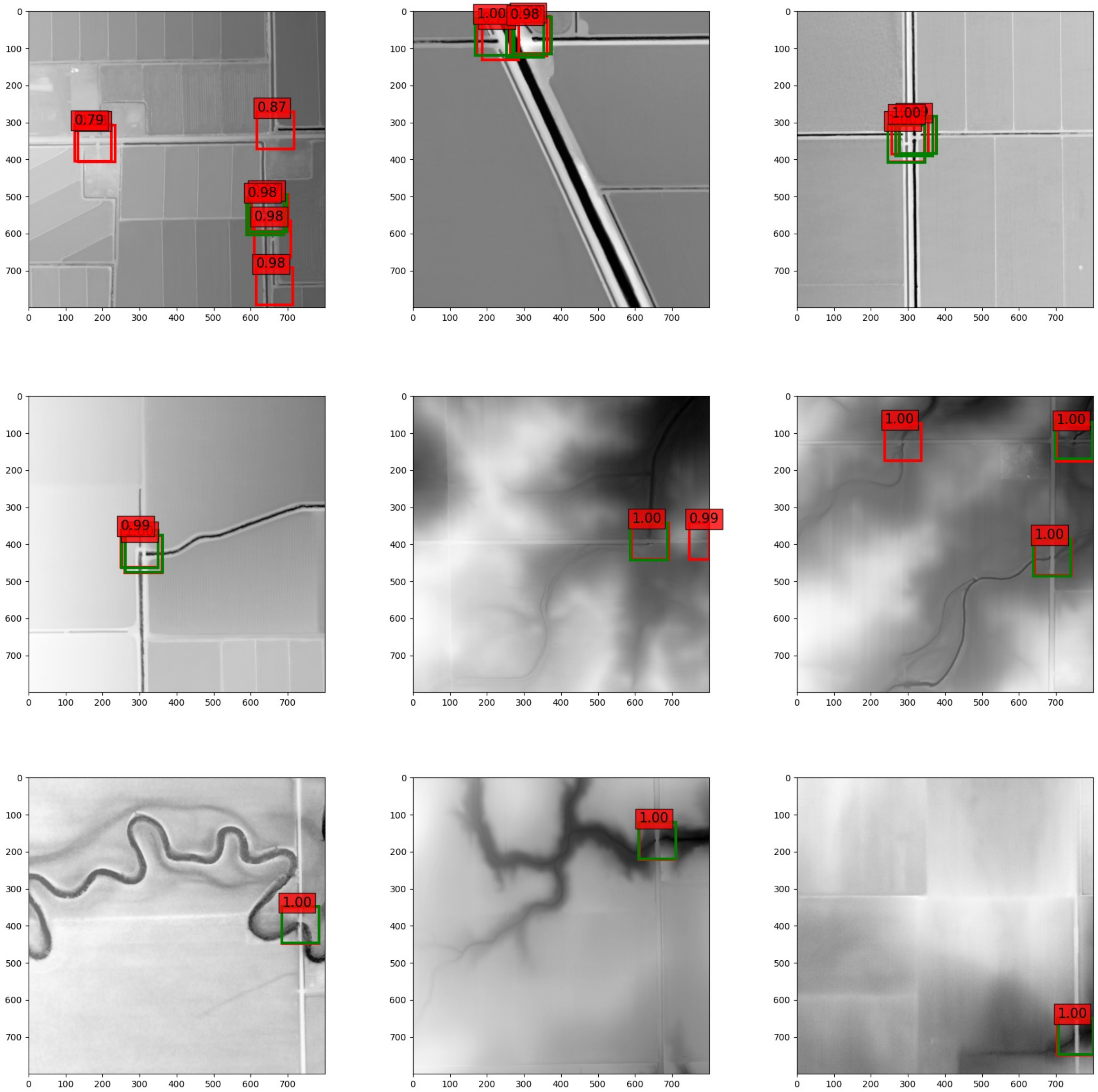
- Two preliminary model tests were run, each using a batch size of 8 and training for 100 Epochs
  - Each training took 13 hours and 20 minutes with a maximum GPU utilization of 7.6GB on one Quadro RTX 5000
- Model 1 is the stock 90-class model pretrained on the COCO dataset
  - "Drainage Culvert" is assigned to category 1, replacing "Person"
- Model 2 is a custom binary classifier model with randomly initialized weights

MODEL 1: PRETRAINED PERFORMANCE



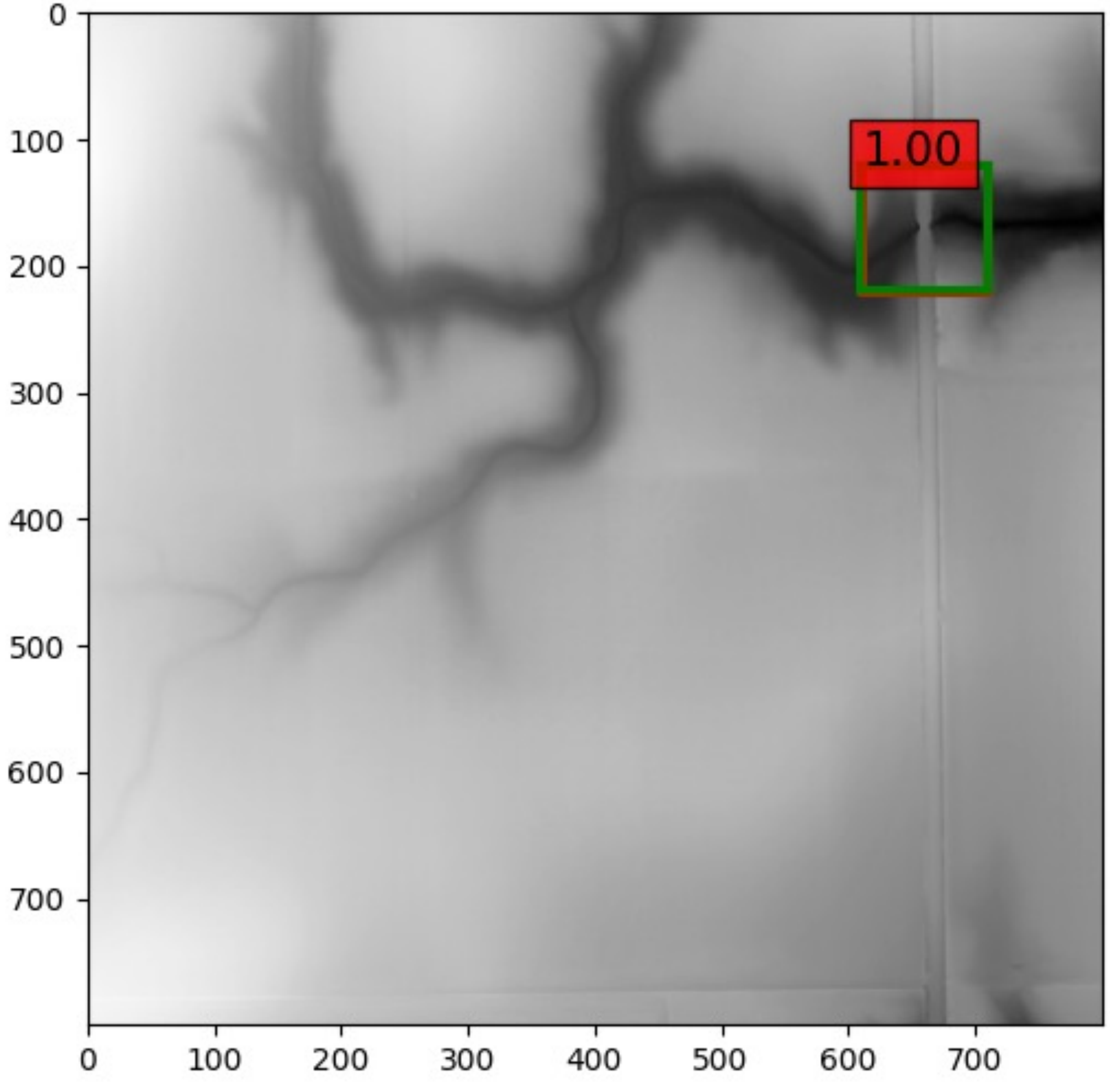
MODEL 1 VISUALIZATIONS: PRETRAINED

- All test set visualizations for Model 1 may be viewed at https://rb.gy/1gczzw
- To avoid cherry-picking, this is every 70th visualization in the test set (idx 0, 70, 140 etc.)
- Only predictions with over 70% confidence are visualized
- Of these visualizations:
  - 6 match the target perfectly
  - None contain a miss
  - 1 matches the target AND finds an unambiguous culvert that the human classifier missed
  - 2 match the target and find ambiguous candidates for a culvert



MODEL 1 VISUALIZATIONS: PRETRAINED

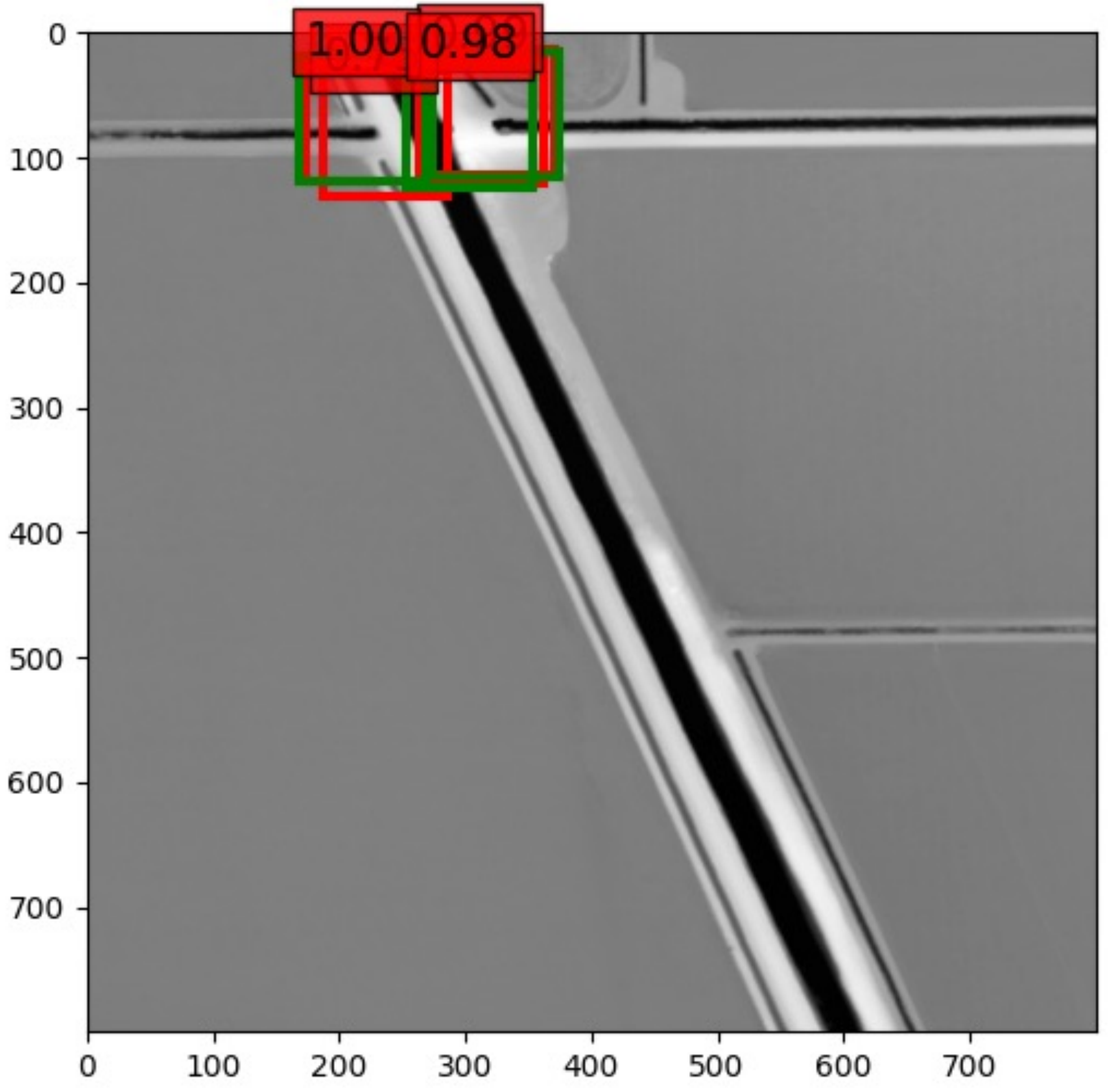
Case 1: Single Easy Object



MODEL 1 VISUALIZATIONS: PRETRAINED

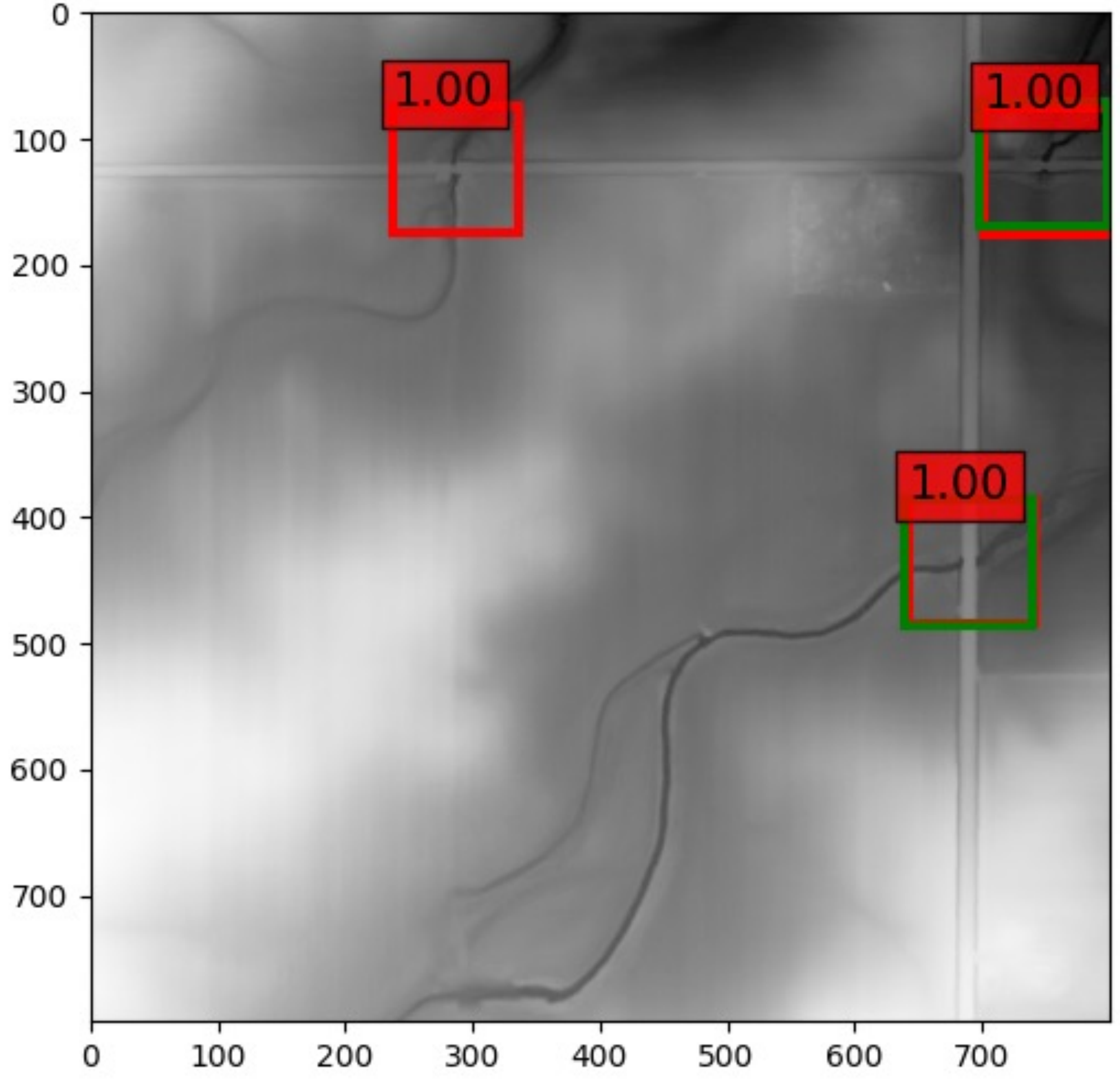
Case 2: Multiple Objects

- Multiple drainage culverts are detected for each target box – this is likely because this is a complex system of multiple culverts



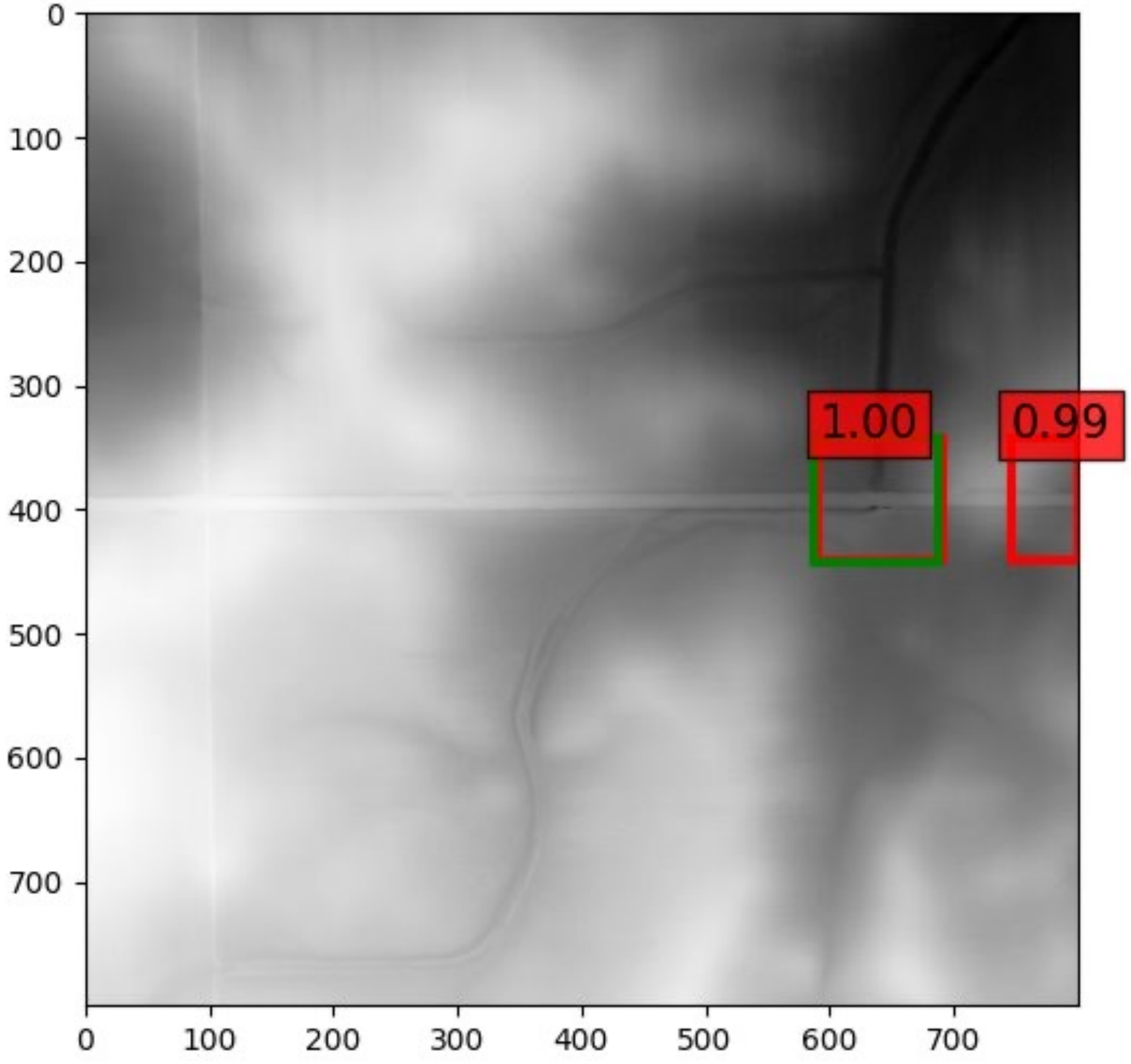
MODEL 1 VISUALIZATIONS: PRETRAINED

Case 3: Model finds an unambiguous culvert not labeled in the dataset



MODEL 1 VISUALIZATIONS: PRETRAINED

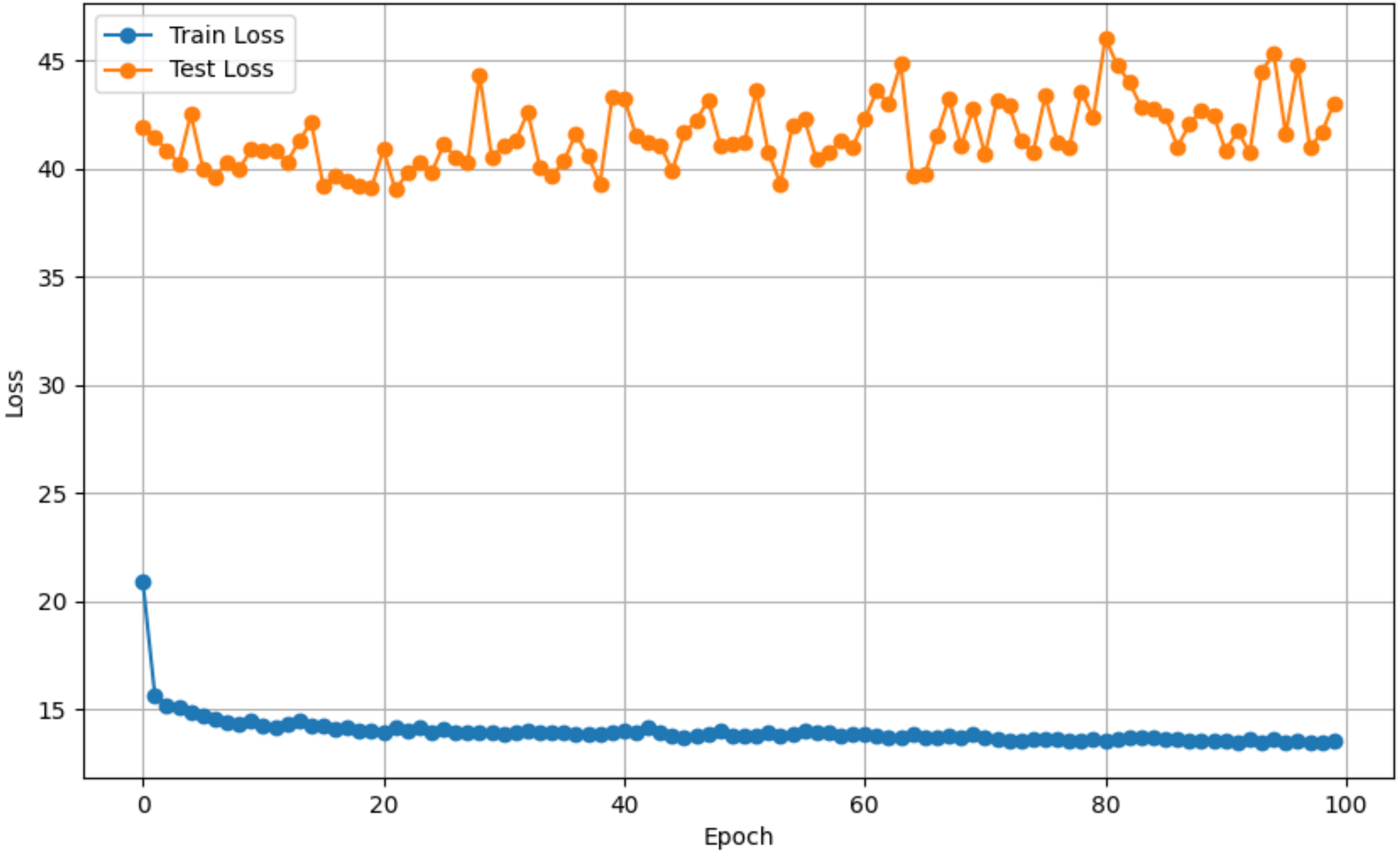
Case 4: Human-in-the-loop training may be beneficial to distinguish ambiguous examples – the false positive here is an edge case



MODEL 1 STATISTICS

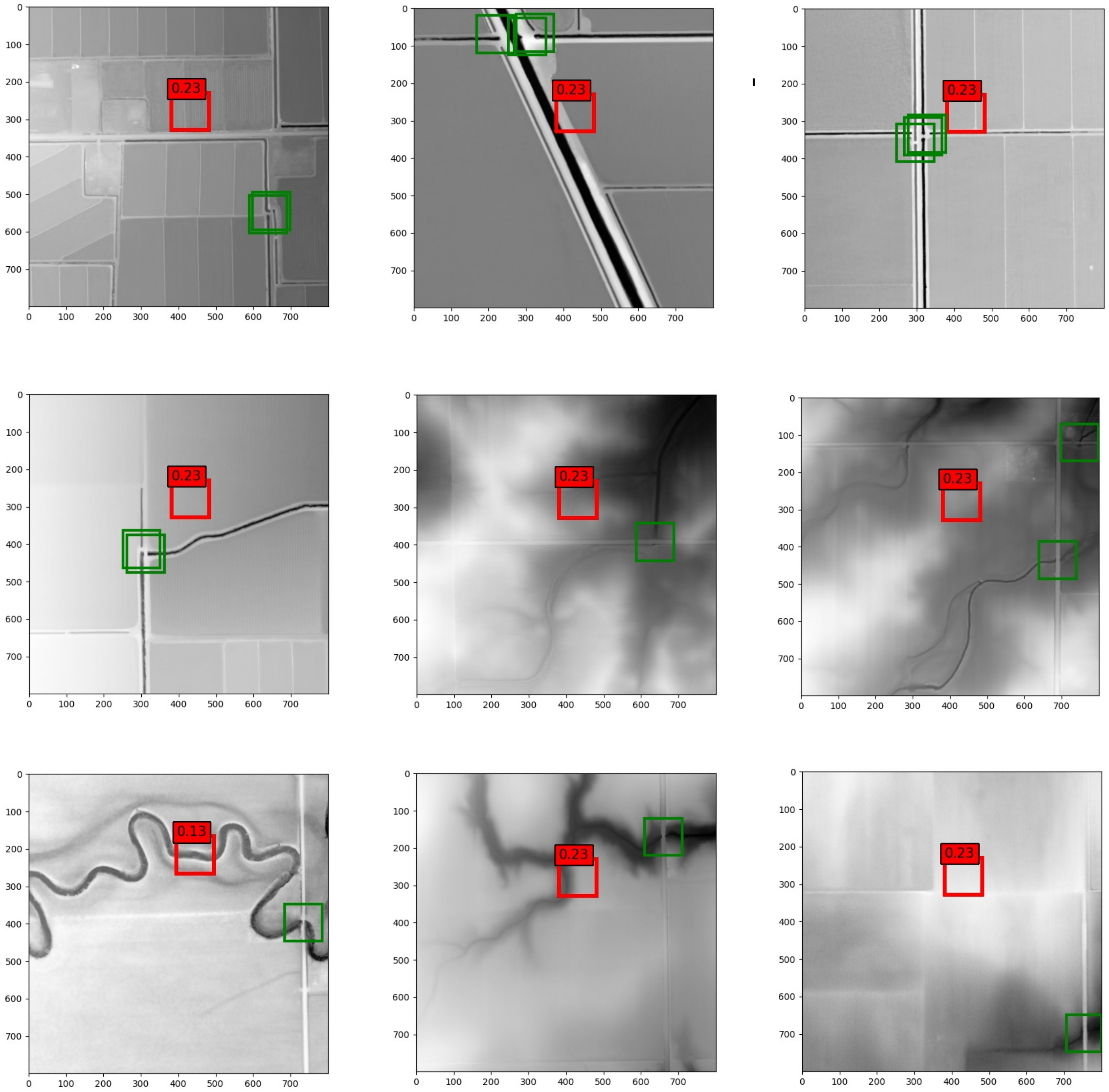
Metric	Test Set Performance
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.645
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.788
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.747
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.538
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.674
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.322
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.825
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.888
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.828
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.904

MODEL 2: FROM SCRATCH PERFORMANCE



MODEL 2 VISUALIZATIONS: FROM SCRATCH

- Training finds a bad local minimum – this training was repeated multiple times with similar results each time.
- Different hyperparameters and loading pre-trained CNN and encoder weights may yield better results for a more lightweight binary classifier.



MODEL 2 STATISTICS

Metric	Test Set Performance
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.000
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.000
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.000
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.000
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.002
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.002
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.002
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.000
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.002

SCALESIM

Systolic CNN AcceLErator Simulator

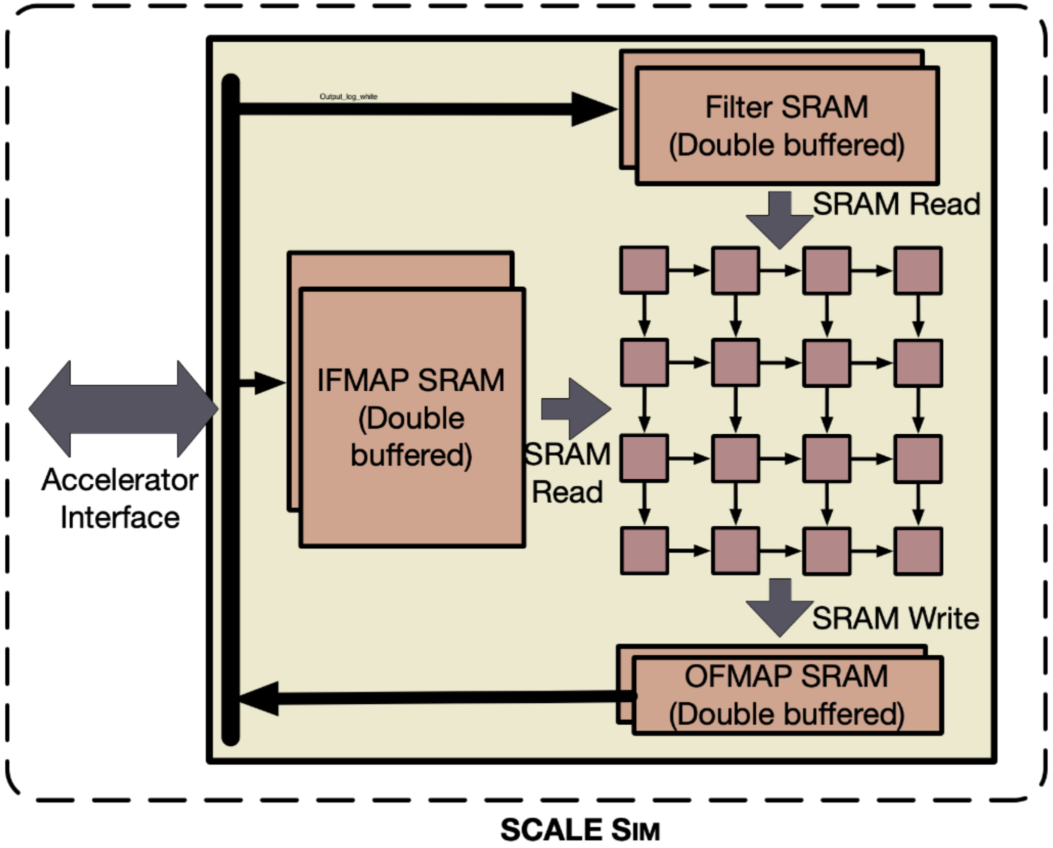
USING SCALESIM FOR ARCHITECTURE SEARCH

Configurable Parameters:

- Systolic Array Height and Width
- Memory size of Filter, IFMAP and OFMAP
- Bandwidth
- Number of Memory Banks
- Dataflow type (ws, os, is)

Performance Related Output:

- Latency
- Mapping Efficiency
- Compute Utilization
- Per Layer



DESIGN SPACE EXPLORATION

8 configurable parameters:

- 2 Array dimensions [260, 400] → 140^2 choices
- 3 SRAM sizes [1024, 35*1024] → 34816^3 choices
- 3 types of Dataflow [ws, os, is] → 3 choices
- Number of Memory Banks [1, 4] → 4 choices
- Bandwidth [1, 64] → 64 choices
- Design Space size ≈ 7e20 points

ScaleSim experiment time:

- 800 x 800 input image → 35 minutes per experiment → 35 * 7e20 = 174 days for a greedy approach

DESIGN SPACE EXPLORATION PRUNING

Solution:

- Limit range of available configuration values
- Use smaller input image size (260 x 260)
- Use random search or genetic algorithm for optimal design point

PyGAD:

- open-source Python Library for genetic algorithm optimization
- Custom fitness function, using one optimization objective, the total latency

PYGAD GENETIC ALGORITHM

Genetic Algorithm properties:

- 50 generations
- 4 parent chromosomes
- 10 solutions per population
- 8 genes per chromosome

Chromosome Structure:

Array Height	Array Width	IFMAP size	Filter mem size	OFMAP size	Dataflow	Bandwidth	Memory Banks

GENETIC ALGORITHM

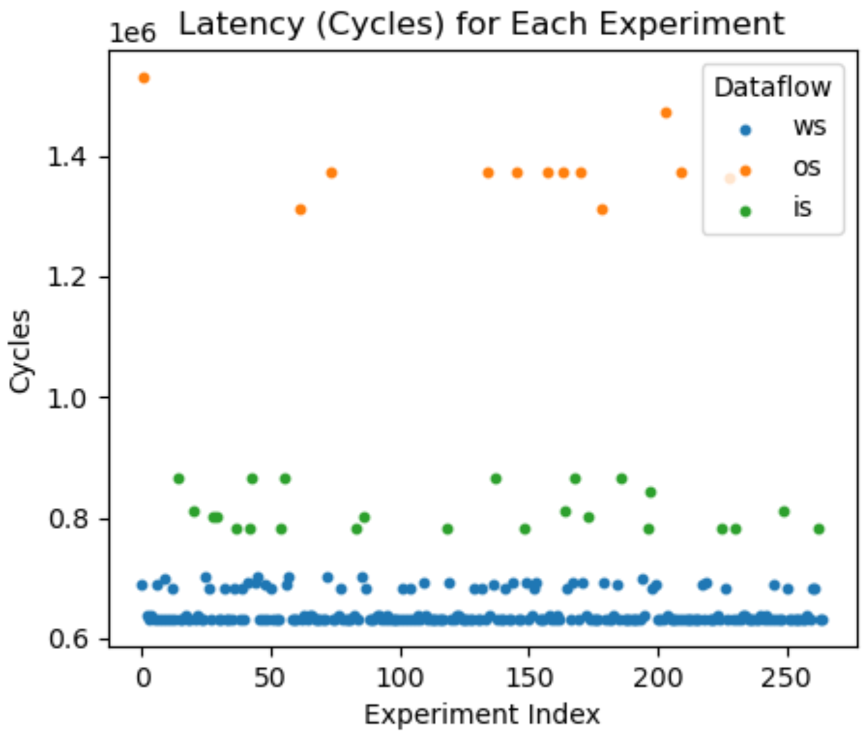
Chromosome Values

- Array Height and Width [260, 320]
- IFMAP, Filter SRAM [1 KB, 10 KB]
- OFMAP SRAM [512 bytes, 5 KB]
- Dataflow: ws, os, is
- Bandwidth bytes [1, 9]
- Memory Banks [1, 3]

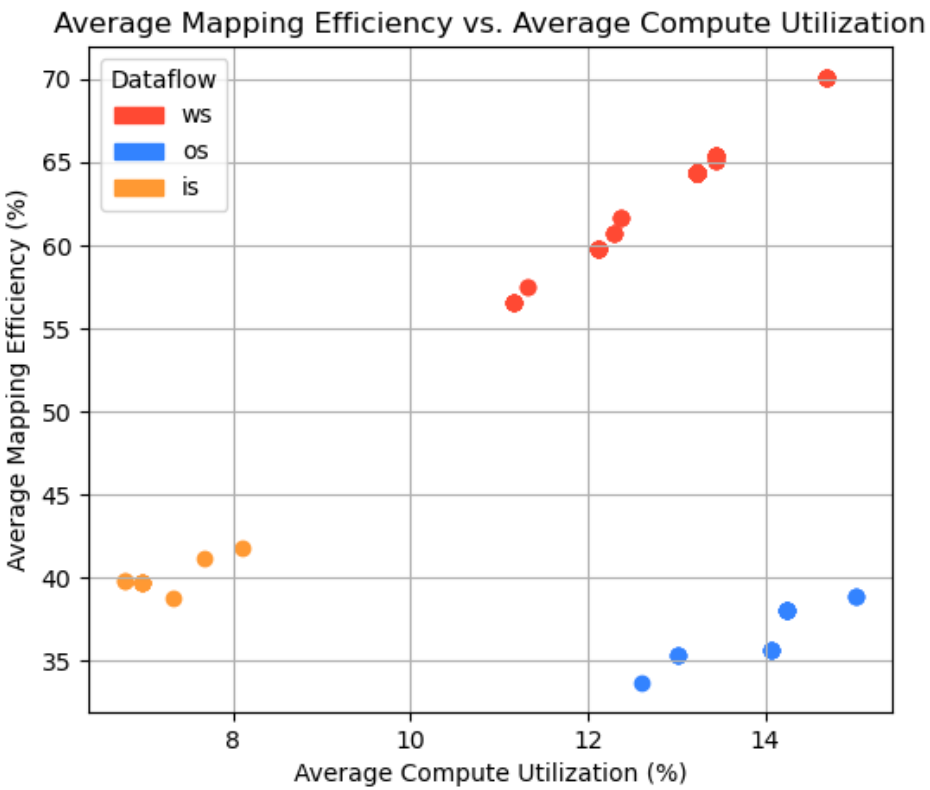
Converged after 2 days by executing 265 experiments

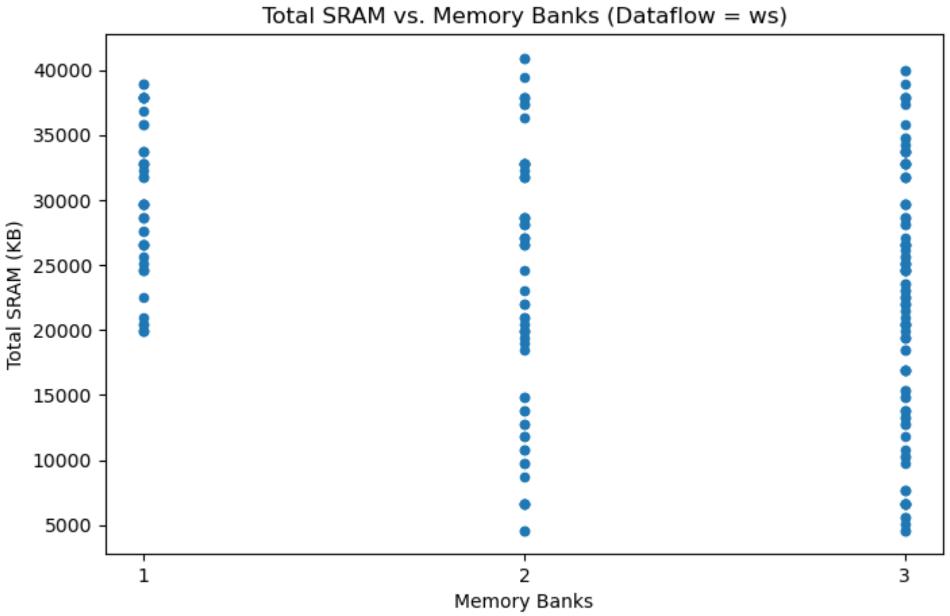
OPTIMAL SOLUTION FOR 260 X 260 INPUT

Parameter	Optimal Value
Systolic Array dimensions	300 x 260
IFMAP SRAM size	1 KB
Filter SRAM size	2 KB
OFMAP SRAM size	3584 bytes
Dataflow	Weight Stationary
Bandwidth	9 bytes
Memory Banks	2
Total Latency	631451 cycles



OPTIMAL TPU ARCHITECTURE





ISSUES AND FUTURE WORK

1. Overtraining may occur prematurely as no random transforms are applied in training
2. The model is worse at detecting boxes in the edge of the image
3. The performant model processed 800x800 image chips, while the most efficient TPU architecture is found using a model with a 260x260 input size

Possible solution to all 3: implement random cropping as a training transform, cropping to 260x260 inputs, preventing overtraining and constraining model search parameters.

- This would most likely benefit from using a smaller bounding box size, e.g. 50x50
- Then, during inference, use a 260x260 sliding window with overlap to predict drainage crossing locations
- This is all possible in the current codebase with a fixed transforms script and retraining