Exploration of TPU Architectures for the Optimized Transformer in Drainage Crossing Detection

Amirhossein Nazeri*
Dept. of Automotive Engineering
Clemson University
Clemson, SC, USA
anazeri@g.clemson.edu

Denys W. Godwin*
Graduate School of Geography
Clark University
Worcester, MA, USA
dgodwin@clarku.edu

Aikaterini Maria Panteleaki*
Sch. of Electr., Comp., and Biomed. Engr.
Southern Illinois University
Carbondale, IL, USA
aikaterinimaria.panteleaki@siu.edu

Iraklis Anagnostopoulos
Sch. of Electr., Comp., and Biomed. Engr.
Southern Illinois University
Carbondale, IL, USA
iraklis.anagno@siu.edu

Michael I. Edidem, Ruopu Li
Sch. of Earth Systems and Sustain.
Southern Illinois University
Carbondale, IL, USA
michael.edidem@siu.edu, ruopu.li@siu.edu

Tong Shu†
Dept. of Computer Science and Engr.
University of North Texas
Denton, TX, USA
tong.shu@unt.edu

***Abstract*—Understanding hydrologic connectivity within landscapes is crucial for managing environmental challenges. Despite advancements in high-resolution Digital Elevation Models (DEMs) derived from Light Detection and Ranging (LiDAR) technology, accurately delineating hydrologic connectivity remains challenging due to disruptions caused by virtual flow barriers, such as roads and bridges. This study addresses this issue by enhancing the detection performance and reducing the latency of Transformer models for image detection of drainage crossings. We retrained a Detection Transformer (DETR) with a specialized recipe to improve culvert detection performance. Owing to the high susceptibility of LiDAR-based DEMs to measurement noise and varying data modalities, we conducted extensive data preprocessing to ensure DETR compatibility with the culvert dataset. Ablation studies on input size indicate that the model performs optimally with 800×800 pixel inputs, demonstrating its adaptability to new data modalities. Additionally, we employed Tensor Processing Units (TPUs) to decrease the model's latency. We developed a novel strategy to optimize TPU architecture, utilizing genetic algorithms to expedite the discovery of optimal TPU configurations for detection deployment. Our model surpasses the performance of previous models on the same task. This work not only addresses the computational complexities of deploying advanced object detection in environmental contexts but also significantly contributes to the precise and efficient monitoring of hydrologic connectivity.**

***Index Terms*—Object detection, Detection Transformer, DETR, TPU, Digital Elevation Model, LiDAR, hydrologic connectivit, culvert**

I. Introduction

Understanding hydrologic connectivity within landscapes is crucial for addressing various environmental management challenges, such as tracking nutrient transport in diffuse pollution runoff. This spatial characterization often relies on hydro-topographic delineation facilitated by Geographic Information Systems (GIS) and digital terrain models (DEMs). In recent years, high-resolution DEMs, primarily derived from Light Detection and Ranging (LiDAR) technology, have shown exceptional capabilities in depicting topographic details with high spatial precision, surpassing conventional DEMs [1]. Despite the advancements in LiDAR DEMs, accurately delineating hydrologic connectivity remains a significant challenge. Studies have indicated that drainage flowlines derived from LiDAR DEMs are often disrupted by virtual flow barriers, such as roads and bridges, which act as 'digital dam' [2], [3]. This issue is particularly pronounced in agricultural regions where rural road networks fragment the gentle terrain. Research has demonstrated that incorporating drainage structures like drainage crossings and bridges can enhance the accuracy of delineating drainage flows across landscapes [4], [5]. However, datasets of these drainage structures are frequently unavailable or inconsistent in format and quality.

Thus, there is a pressing need to develop an efficient approach to accurately and comprehensively map drainage structures, thereby improving the delineation of hydrologic connectivity using high-resolution LiDAR DEMs. Therefore, Due to the low quality and complexity of existing drainage structure datasets, there is a significant need to explore state-of-the-art object detection technologies to accurately identify and locate drainage structures. This approach will help to compensate for the inadequacies and poor quality of current datasets.

Object detection is an essential function in computer vision, with uses in various fields from autonomous driving and robotics, to medical and agriculture. It involves identifying and pinpointing objects within an image. Over time, various technologies and methods have been created to improve the accuracy and efficiency of object detection systems. Traditionally, these systems have heavily depended on Convolutional Neural Networks (CNNs). However, recent advancements have introduced transformer-based approaches, which provide sev-

This research work is done in the Smart High-performance and Ubiquitous Systems (SHU'S) lab at the University of North Texas.

* These authors contributed equally to this work.

† Corresponding author: Tong Shu, Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, USA (ORCID: 0000-0001-8617-1772)

eral benefits over the traditional methods. CNNs have been fundamental to object detection for many years due to their ability to capture spatial hierarchies in images through convolutional layers [6]–[8]. Notable CNN-based object detectors include Faster R-CNN and Single Shot MultiBox Detector (SSD). SSD, proposed by Liu et al. in 2016 [9], aims to improve the speed of object detection without significantly compromising accuracy. SSD uses a single-stage network that directly predicts object classes and bounding box offsets for a fixed set of default boxes of different scales and aspect ratios. This approach significantly reduces computational complexity, making it suitable for real-time applications. However, SSD can struggle with detecting smaller objects compared to two-stage detectors like Faster R-CNN. Introduced by Ren et al. in 2015, Faster R-CNN [10] transformed object detection by combining a Region Proposal Network (RPN) with the Fast R-CNN detector [11]. The RPN generates high-quality region proposals, which the Fast R-CNN then classifies and refines. This two-stage process enables accurate localization and classification, albeit with a higher computational cost.

Faster R-CNN had shown to be the de-facto technology among CNN-based object detectors. Despite its pioneering advancements in object detection, Faster R-CNN has notable drawbacks, particularly when used in various detection scenarios. One major issue is its computational inefficiency. The two-stage process, which involves generating region proposals followed by region classification and refinement, significantly increases processing time, making it less suitable for real-time applications. This complexity also requires substantial hardware resources, limiting its use on devices with limited computational power. Moreover, while Faster R-CNN is highly accurate in detecting larger objects, it can struggle with smaller objects and densely packed scenes, where the region proposal network might miss fine-grained details. This is problematic in applications like autonomous driving or surveillance, where detecting small or closely spaced objects is crucial [12], [13]. Additionally, the model's performance may degrade in scenarios that require rapid adaptation to dynamic environments due to the static nature of its training process.

Transformers, initially crafted for tasks in natural language processing, have made significant strides in computer vision recently. The advent of Vision Transformers (ViT) [14] and their subsequent application to object detection have marked considerable advancements in both efficacy and performance. In 2020, Carion et al. [15] introduced a transformative model for object detection known as DETR, which incorporates the use of transformers. This innovative system forecasts the locations and categories of objects by directly decoding image features, conceptualizing the task as one of predicting a collection of items. The DETR model streamlines the object detection process by forgoing conventional mechanisms such as Region Proposal Networks (RPNs) and Non-Maximum Suppression (NMS), thereby diminishing the likelihood of mistakes and streamlining operations. However, most recent research revealed that DETR faces some limitations when it comes to detecting small objects [16]. This is because the transformer in DETR primarily relies on semantic information from the highest level of abstraction in the deep network, which may not include the detailed information such as edges, texture, or color gradients that are crucial for locating small objects. This issue is even more critical when utilizing transformers for aerial applications where most instances are small objects within the images. On the other hand, DETR may require more training epochs to converge and can be slower at detecting small objects [17].

The advent of transformer-based models in Deep Neural Network (DNN) applications [18]–[20] has revolutionized domains such as computer vision. These models, however, require accelerators specifically designed to meet their computational demands. Tensor Processing Units (TPUs) [21] have emerged as a promising choice for such hardware accelerators, provided that their attributes are carefully selected for specific tasks. To enhance the performance of Transformer models for image detection of drainage crossings, we developed a novel strategy to optimize TPU architecture. Recognizing the computational challenges in deploying cutting-edge object detection technologies and auto-tuning the TPU architecture in a huge configuration space [22], [23], we leveraged genetic algorithms to streamline the discovery of optimal TPU configurations for the detection deployment. Building upon the state-of-the-art cost model, Scale-Sim [24], this method aimed to drastically reduce the total training cycles, thereby diminishing latency and boosting efficiency. Our work not only contributes to the precise and efficient monitoring of hydrologic connectivity, but also addresses the computational complexities of deploying advanced object detection in environmental contexts.

Contribution:

- Demonstrate ability of Pretrained DETR to adapt to LiDAR data for drainage crossing data: Training from the pre-trained model weights shows that the model is capable of predicitng bounding boxes which are not defined by object boundaries and instead depend on an intersection point of geographic features.
- Enhance dataset preprocessing for compatibility with DETR object detection model: Customized the COCO dataloader to process high-precision geographic data.
- Development of Optimized TPU Architecture: Created a framework for efficiently configuring TPUs specifically tailored for Transformer-based object detection models, such as DETR.

II. Related work

A. Object Detection

Advanced object detection networks are widely used for remote sensing applications essential for predicting, mapping, and mitigating natural disasters (e.g. flooding, fires, etc.), socioeconomic service delivery, or general urban and rural planning and management, as presented in [25]. An extensive study of methods for object detection using deep learning is presented in [26], where Amjoud et al. discuss a wide

range of vision networks, for object detection task on popular datasets. Recent research [27] highlights the benefits of transformer-aided detectors for aerial image object detection. Wang et al. investigated the object detection performance of RCNN variants with different CNN-based and transformer-based backbones on popular aerial datasets such as Airbus Aircraft Detection [28], and DOTA [29]. However, this work just partially benefited from transformers, as transformers were applied only in the backbone of the model rather than being used in training an end-to-end transformer-based object detector.

The academic interest towards image detection of drainage crossings specifically is also active. Custom CNN-based classifiers have been employed in [30] to classify drainage crossings in topographic data. A CNN-based model with custom architecture is developed in [31] to detect drainage crossing locations in 4-band digital orthophotos from USGS National Agriculture Imagery Program (NAIP). Although the proposed model performs well in determining whether a drainage crossing exists, it fails to accurately locate the drainage crossing. Thus it is not easily scalable to more complex tasks. A recent study has applied DETR and DINO transformer-based object detectors to detect drainage crossings from Light Detection and Ranging (LiDAR) digital terrain models (DEMs) datasets. The study compared the object detection performance of various CNN-based and transformer-based detectors, and reported that transformer-based detectors outperform traditional CNN-based detectors [32]. However, transformer detectors did not perform well in locating drainage crossings, as many drainage crossings were inaccurately positioned. We assume that this issue can be extensively improved by adequate model retraining and sufficient data preprocessing as the drainage crossing DEM-LiDAR dataset is fundamentally different than the MS-COCO dataset that the transformer detector is trained on.

Despite these extensive studies, object detection with transformers in overhead imagery remains challenging, due to substantial image volumes, inconsistent image resolution, small-sized objects, and highly complex backgrounds. These factors can significantly degrade the performance of object detection, necessitating a more specialized approach, which our research aims to address.

B. TPU architecture optimization

There has been a strong academic focus on exploring methods for optimizing TPU accelerators, emphasizing on enhancing the performance of Machine Learning (ML) tasks, such as inference and training [33], [34], [35]. In [36], Elbtity et al. proposed a TPU architecture with a runtime-reconfigurable dataflow strategy, significantly improving the performance of Convolutional Neural Network (CNN) workloads. Another paper [37] introduces a system that optimizes execution time of DNN models by deploying TPUs at the edge. Additionally, a recent survey [38] examines a more general hardware accelerator model, along with various strategies for full-stack optimization to speed up the execution of transformer models.

While these studies provide valuable insights about TPU optimization for general ML tasks, our work specifically focuses on extracting an optimal TPU configuration designed for transformer-based object detection models, such as DETR, as they have shown to be slower than their competitors. This approach aims to address the challenges posed by overhead imagery object detection, targeting drainage crossings recognition, while leveraging the efficiency of TPU accelerators.

III. Methodology

A. Detection Transformer

DETR integrates three principal components together: a CNN-based backbone, a transformer encoder-decoder architecture, and a detection head. Figure 1 demonstrates the architecture of DETR model. The CNN backbone is responsible for extracting feature maps from input images, which are then fed into the transformer encoder. The encoder processes these feature maps through self-attention mechanisms, capturing contextual relationships and spatial dependencies. Subsequently, the transformer decoder takes the encoded representations and generates object queries, predicting the bounding boxes and class labels for each detected object. The detection head, positioned at the final stage, refines these predictions into final object detections.

We modified the pre-trained DETR model by loading pre-trained weights for all layers except the last fully connected layer in the MLP classifier. Instead of using the 91-class classifier, we replaced the final layers with a binary classifier initialized with random weights. During training, all layers were updated through backpropagation. Initial experiments showed that freezing the backbone and using pre-trained weights failed to reduce loss, likely due to differences between the model's original training data and our single-channel geographic elevation data. As a result, we opted to retrain the entire model using the pre-trained weights as a starting point. Initial experiments in randomized weights also showed no reduction in loss during training.

B. Tensor Processing Unit Optimization

1) Performance estimation: The TPU is a specialized ASIC designed to accelerate machine learning workloads with its architecture centered around a systolic array of multiply-accumulate (MAC) units, optimized for matrix operations. It features various memory hierarchies for storing input/output data and neural network weights. Key optimization factors include the dimensions of the MAC array, which influence latency and throughput, and the SRAM memory sizes, affecting off-chip memory access and data reuse. Additionally, memory banks and bandwidth impact parallelism and data transfer efficiency. By tuning these parameters, the goal is to minimize latency in the DETR model inference process. To evaluate the performance of different TPU designs for the entire DETR model, we used the Scale-Sim cost model, a detailed cycle-accurate tool that simulates the execution of workloads on systolic array-based accelerators. The tool's modeling approach is compatible with industry-standard systolic array

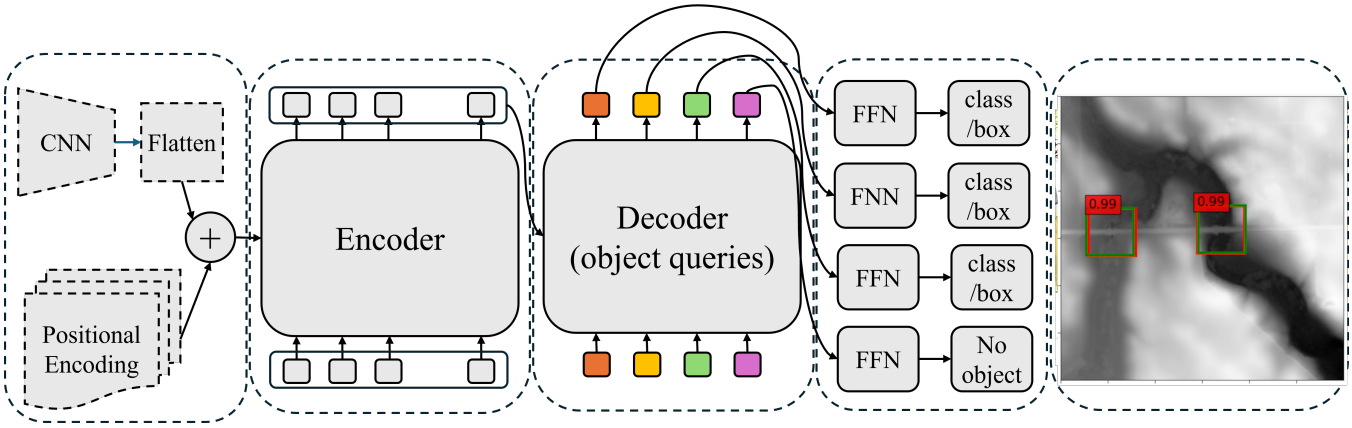


Fig. 1: The schematic of DETR architecture adopted for drainage crossing detection [15]

architectures, like commercial TPU implementations, ensuring realistic performance estimates for our target accelerator architecture. Scale-Sim's accuracy is a result of its comprehensive modeling of key architectural components, including memory hierarchies and data movement patterns, making it particularly suitable for evaluating complex neural architectures like DETR. For each TPU configuration, we simulate the convolutional backbone, the fully connected layers and the matrix multiplication operations within the DETR model. To achieve this, we convert the Transormers architecture layers to custom convolutional representations, so that they can be evaluated through Scale-Sim. Our optimization metric is the total cycle count across all involved layers, mainly because this quantity corresponds to the inference delay of identifying the drainage crossings, which is also the most important characteristic in object detection models. Moreover, the Scale-Sim cost model estimates the clock cycles considering both computations and memory accesses [39], so this metric balances effectively the trade-off between computations and memory for the entire model.

2) Genetic Algorithm: The design space of our optimization problem for the DETR model is very expansive, as these eight parameters can construct a space of 10^{20} design points. Moreover, the TPU configuration parameters have complex relationships, so a greedy algorithm or a simple heuristic method cannot be applied. To solve our problem efficiently, we leverage the genetic algorithms, which are optimization techniques ideal for vast and complex design spaces, inspired by principles of biological evolution. Their main function is to create a population of chromosomes and iteratively evolve it towards better solutions over the generations, by applying selection, crossover, and mutation operations. Our genetic algorithm was implemented using the PyGAD library [40], where we create each chromosome by encoding the aforementioned TPU parameters to eight genes. A population size of 100 was selected, to maintain diversity across the 8-dimensional search space, while 20 parent chromosomes mate with each other in every generation. Concerning the mutation type, we employed adaptive mutation to maintain diversity in the population, prevent any premature convergence, and encourage the exploration of new solutions [41], as opposed to constant mutation methods. Our fitness function integrates with Scale-Sim, to evaluate each candidate solution and extract the estimated inference total cycle count. Then, the genetic algorithm creates the next generation's population by applying genetic algorithm operations, focusing on minimizing the cycle count. This problem formulation ensures that the genetic algorithm evolves towards TPU configurations that offer minimal execution time of the entire DETR model.

IV. Experiments and Results

A. Dataset

The dataset consists of 6,012 LiDAR-derived DEM georeferenced rasters in TIF format, each with a 800m×800m footprint and a cell size of 1m×1m. Elevation data is stored as 32-bit floating point values, indicating meters above sea level, and comes from the USGS 3DEP program.

The rasters cover four watersheds in the Continental United States: Sacramento-Stone Corral in California, Vermilion River in Illinois, Maple River in North Dakota, and West Fork Big Blue in Nebraska. Drainage crossings within these watersheds were labeled as centroids, and corresponding rasters containing these centroids were extracted. Bounding boxes of 100m×100m were defined around these centroids, and the data was converted to the COCO format for use with the DETR model.

After filtering out anomalous rasters, 6,007 rasters with 13,141 drainage crossing bounding boxes were used. The Maple River Watershed data was reserved for transfer learning. A custom dataloader was implemented to handle the 32-bit DEM data, preserving precision using rasterio and applying z-normalization. Random flips and cropping were used for training augmentations, while center cropping was used for validation.

B. Experimental Setup

Experiments on model performance tested the impact of input size on the model's ability to identify desired bounding

boxes around drainage crossings using DEM data.

Loss is calculated on both boxes and classifications to update model parameters during training. For bounding boxes, Generalized Intersection over Union (GIoU) and L1 are used as the loss metrics. Cross-entropy is used as the classification loss metric. In all experiments, AdamW was used as an optimizer with a learning rate decay of 0.0001 and a stepped learning rate scheduler. This reduces learning rate limiter λ every 200 epochs by multiplying it and the default gamma value of $\gamma = 0.1$.

Experiment 1 uses inputs of size 800×800 with no cropping to re-train the pre-trained DETR model. Experiment 2 uses inputs cropped randomly to 600×600, Experiment 3 uses inputs cropped randomly to 400×400, and Experiment 4 uses inputs cropped randomly to 256×256. All other hyperparameters and the model size were held constant across all experiments.

All models were trained using the same image and label pairs. During validation and testing, images and labels are cropped to the correct sizes using center cropping to ensure comparability across epochs. However, given that image and labels are cropped, the validation and testing sets across experiments are not identical.

All experiments were run for 500 epochs. Training times and VRAM usage are shown in Table. I.

TABLE I: Precision and Recall on Initial and Transfer Datasets

Input	Total Time	Time/Epoch	Max VRAM (Gb)
800×800	67:04:47	08:03	5.863
600×600	49:59:40	06:00	3.479
400×400	40:48:23	04:34	2.051
256×256	34:06:57	04:06	1.409

The model checkpoint from the epoch with the lowest validation loss is chosen for inference on the test set and the transfer learning set. The results of these tests are explored in the Results and Discussion section of this paper.

Concerning the TPU architecture optimization part, we evaluated our algorithm by considering input images of dimensions 400×400. To ensure that the genetic algorithm produces only valid and realistic configurations, we specify the value range of each gene, as can be seen in Table II, which is close to properties of existing TPU accelerators [42]. The systolic array dimensions can be from 128 to 4×128 and the SRAM memory sizes can range from 1 KB to 10 KB each. The dataflow type can be either input, weight or output stationary, indicating which data group remains fixed when mapped on the MACs, throughout the computation. Memory bandwidth is also constrained from 500 to 1200 bytes per cycle, to reflect the typical real-case memory bandwidth of around 900 GB/sec with 700 MHz clock. Finally, the number of memory banks is limited to a range of 1 to 4, to balance parallelism levels with computational complexity.

C. Results and Discussion

Experiments show that maximal overall performance is achieved with 800×800 inputs, and that input size has an impact on performance metrics. Table III compares performance of object detection in terms of different input sizes on initial and transfer dataset. While table III shows that the model is capable of predicting correctly with 256×256 inputs, these inputs are too small for usable predictions overall. Subsequent experiments show decreasing precision and mixed effects on recall with smaller inputs. Mixed effects on recall may be due to the necessary cropping resulting in fewer target annotations per chip for the smaller input datasets. Best precision on predictions with an IoU of greater than 0.50 is with 600×600 inputs, showing that while localization may suffer with smaller inputs, the model is still capable of predicting drainage crossing locations accurately. Finally, the model performs exceptionally well, achieving high precision and recall across various IoU thresholds and maximum detection counts when input size is 800×800. However, on the transfer dataset, precision drops significantly, as well as recall when considering 10 and 100 maximum detections, while recall with one detection increases. The increase in single-detection recall may be due to the lower number of drainage crossings per image chip in the transfer dataset, making it easier to detect one drainage crossing given one detection. However, there may be unfamiliar styles of drainage crossings in the transfer dataset, leading to lower performance in finding all within a single image chip. Figure 4 shows that this model fails to predict in many cases.

TABLE II: Genetic Algorithm Gene space

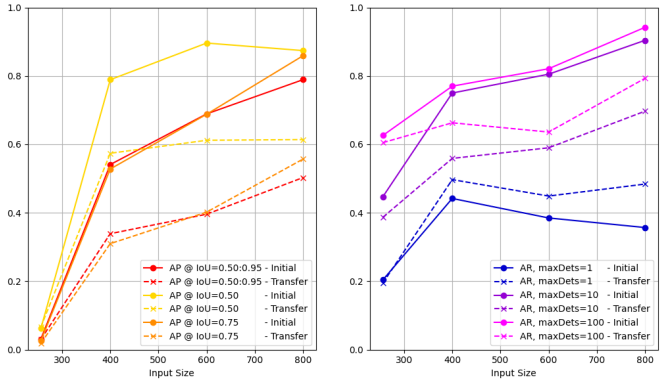
Gene	Value Range
Systolic Array Width	$[128, 4 \times 128]$
Systolic Array Height	$[128, 4 \times 128]$
Input Feature Memory	$[1, 10]$ KB
Weight Memory	$[1, 10]$ KB
Output Feature Memory	$[1, 10]$ KB
Bandwidth	$[500, 1200]$ bytes/cycle
# Memory Banks	$[1, 4]$
Mapping	Input/Output/Weight Stationary

Figure 2 compares model performance across input sizes for both initial and transfer datasets. Larger input sizes lead to better precision with stricter IoU thresholds, but have a more muted effect on IoU = 0.50. The decision on what input size is preferable would therefore depend on how important local precision is for model outputs in a given downstream application. The effect of input size on model recall is more muted, and in the case of recall given one detection, reversed past 400×400 inputs.

Cropping in Experiments 2 and 3 may contribute to better model performance than otherwise expected due to the ability to use random cropping during training, reducing over-training. Figure 3 demonstrates that validation loss for 800×800 inputs converges before Epoch 100 of training, and begins increasing after the learning rate reduction at Epoch 200. However, the validation loss of the model trained

TABLE III: Precision and Recall on Initial and Transfer Datasets. Average Precision = AP, Average Recall = AR.

	256×256		400×400		600×600		800×800	
Statistic	**Initial**	**Transfer**	**Initial**	**Transfer**	**Initial**	**Transfer**	**Initial**	**Transfer**
AP, IoU=0.50:0.95	0.031	0.030	0.541	0.339	0.689	0.396	**0.789**	0.503
AP, IoU=0.50	0.062	0.066	0.789	0.573	**0.896**	0.612	0.874	0.614
AP, IoU=0.75	0.026	0.019	0.528	0.310	0.688	0.402	**0.859**	0.557
AR, maxDets=1	0.204	0.195	**0.442**	0.497	0.305	0.449	0.357	0.484
AR, maxDets=10	0.447	0.307	0.750	0.559	0.805	0.590	**0.904**	0.697
AR, maxDets=100	0.626	0.605	0.770	0.663	0.821	0.636	**0.942**	0.793



(a) Precision (b) Recall

Fig. 2: Precision and Recall of each model on both the initial and transfer learning test sets.

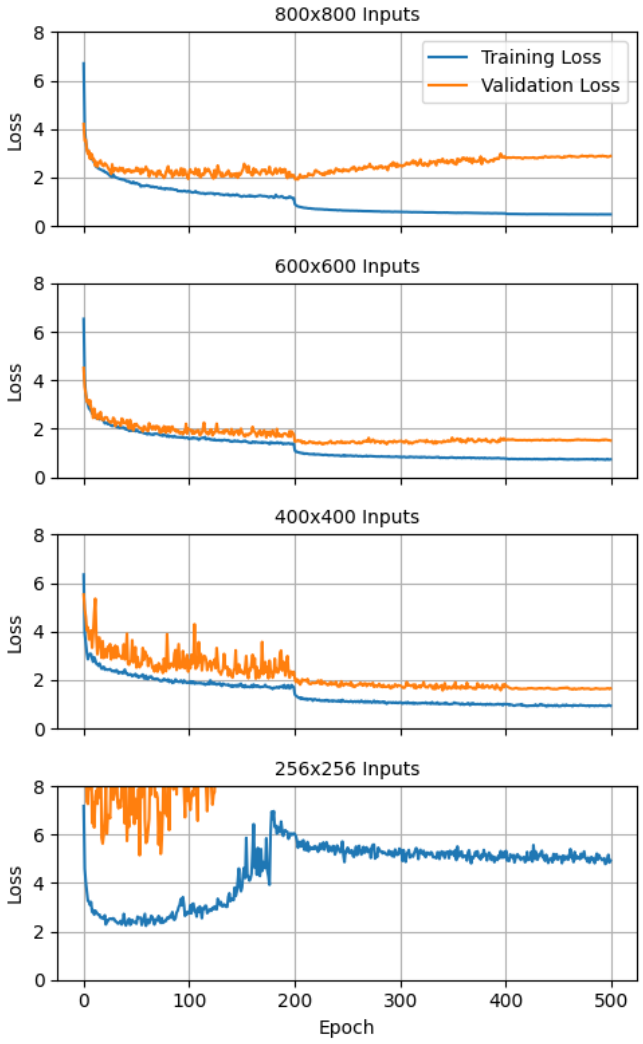


Fig. 3: Training loss and validation loss of each model over 500 epochs.

on 400×400 inputs converges after Epoch 400. These graphs also show that the model trained on 256×256 inputs loses stability and eventually reaches a bad local minimum at around Epoch 100. The implementation of random cropping for 800×800 inputs by drawing from a dataset with e.g. 2000×2000 chips may increase the performance of the model further.

Visualizations of the model outputs in Figure 4 show that on the initial dataset, models from Experiments 1, 2, and 3 are capable of identifying drainage crossings in complex systems with a high degree of accuracy. The outputs of the model trained on 800×800 inputs shows near-perfect performance on this particular example, despite the drainage system involving many sequential crossings underneath roadways in multiple locations across the DEM. Likewise, the visualizations from the transfer dataset show that the models are capable of high performance on data from an unseen watershed.

Overall, these results show the promise of adapting a pretrained hybrid transformer object detection model (DETR) to novel data modalities. The model is performant on single-channel, high-precision data with objects that have no edges or defined size. Input size has a strong effect on performance, showing that spatial context is important for this application. The trade-off between model efficiency and desired performance must be considered in downstream implementation. Additionally, fine-tuning a given model on a sample of a new watershed is recommended, given the drop in performance on the transfer dataset in this case.

D. Optimal TPU and Evaluation

Through the genetic algorithm optimization process, we derived a set of optimal TPU configurations, tailored specifically for the DETR model of 400×400 input image. While our algorithm converges in an optimal configuration, it is

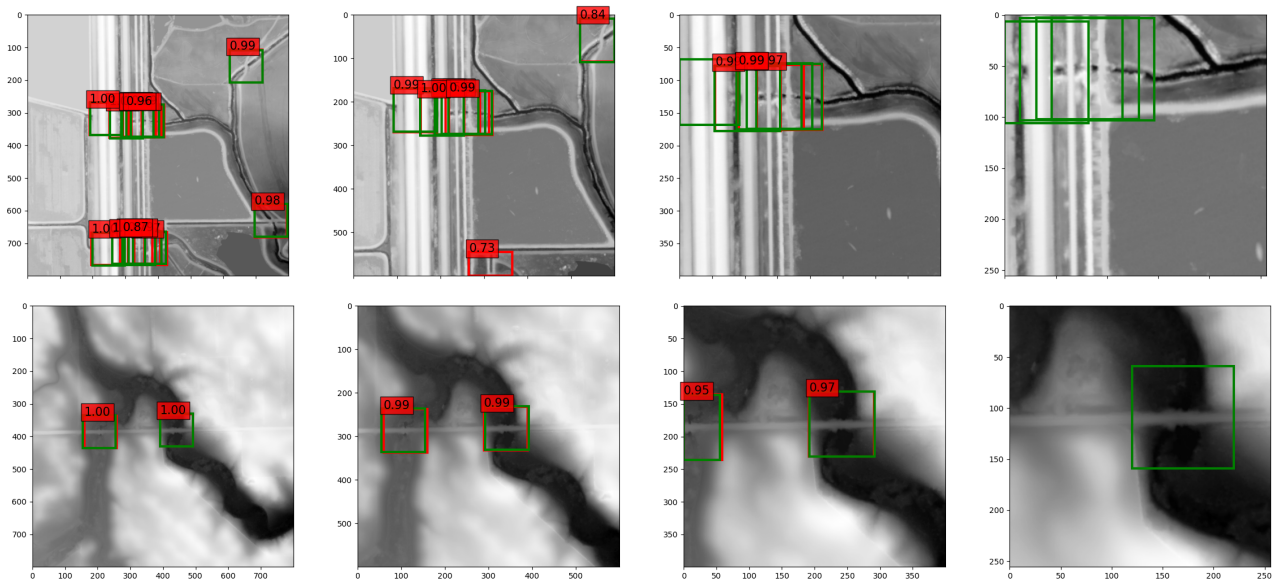


Fig. 4: Model outputs from initial test set (top) and transfer learning test set (bottom). From left to right: original image size of 800×800 is cropped to 600×600, 400×400, and 256×256. Ground truth bounding boxes are shown in green and model predictions with confidence over 0.7 are shown in red.

TABLE IV: TPU Configurations and Inference Cycles for 400×400 input image

Attribute	**Lowest Latency Solution**	**Average Latency Solution**	**Google's TPU v3**
# Systolic Arrays	1	1	2
Systolic Array Width	512	128	128
Systolic Array Height	256	256	128
Input Feature Memory	4 MB	2 MB	8 MB
Weight Memory	3 MB	3 MB	16 MB
Output Feature Memory	6 MB	1 MB	8 MB
Bandwidth	700 $B/cycle$	1100 $B/cycle$	900 $B/cycle$
# Memory Banks	2	2	2
Mapping	Weight Stationary	Weight Stationary	Weight Stationary
Cycles (Normalized)	0.6×	**0.9**×	1×
MAC Units (Normalized)	4×	**1**×	1×

crucial to note that there is no single straightforward recipe for our problem. The relationship between various architectural parameters and the estimated performance is quite complex and often non-intuitive.

Among the solutions of our genetic algorithm, the attributes of the lowest latency design are presented in Table IV. Our TPU configuration features a 512×256 systolic array of MAC units with 13 MB total size of SRAM memory. Moreover, the best option for memory banks is 2, while the most efficient mapping strategy is weight stationary dataflow. Finally, the memory bandwidth was optimal at 700 $bytes/cycle$. Another design point derived from the pool of solutions and with average latency, includes a smaller MAC Array of 128×256, SRAM memory of 6 MB, a high bandwidth capacity of 1100 $bytes/cycle$ and again weight stationary mapping. The latency of that average case compared to the fastest aforementioned solution is around 1.2× slower but with considerably less resources.

To evaluate the performance of our solutions, we established a baseline for comparison by configuring our cycle-accurate Scale-Sim tool to represent a real-world case of TPUs, similar to Google TPU v3 [42]. The baseline case incorporates two 128 × 128 systolic MAC arrays, 32 MB of on-chip SRAM cache memory and a memory bandwidth of 900 GB/s, that corresponds to 900 $bytes/cycle$ in a chip with clock speed of 1 GHz. As the mapping type, we choose weight stationary, which is very usual in existing designs, while the memory will comprise of 2 banks. All TPU configuration attributes are concentrated in Table IV, along with the total cycle count of DETR inference, as estimated by Scale-Sim. Our results show

that the latency oriented solution is much faster than the other two, as it employees the inference procedure $1.67\times$ faster than the Google TPU v3 model and $1.5\times$ faster than our average solution. However, this improvement in performance comes with a great increase in logic resource demand, as the fastest solution requires $4\times$ more MAC units from both the other two configurations. This leads us in choosing the average solution, which achieves a satisfactory balance between the trade-off latency and resource demand.

V. Conclusion

In this paper, we adapt a pre-trained object detection model (DETR) to a drainage crossing detection task using high-resolution LiDAR-derived DEM data. We test the responsiveness of the model to different input sizes and its performance on a transfer learning dataset from a watershed that was not represented in the training set. We then utilize a genetic algorithm to locate an optimal TPU architecture for this model.

We show that the pre-trained DETR model can be effectively adapted to a new data modality, surpassing the performance of previous models on the drainage crossing detection task as demonstrated in Jalalipour et al [32]. We show that there is a trade-off between input size and performance for this application. Additionally, model performance is lower on the transfer learning dataset, showing that fine-tuning should be performed when applying the model to watersheds with distinct physical geography.

Acknowledgment

This research is sponsored by the National Science Foundation under Grant No. OAC-2306184 with the University of North Texas and Grant No. BCS-1951741 with Southern Illinois University.

References

[1] J. N. Callow, K. P. Van Niel, and G. S. Boggs, "How does modifying a dem to reflect known hydrology affect subsequent terrain analysis?" *Journal of hydrology*, vol. 332, no. 1-2, pp. 30–39, 2007.

[2] R. Li, Z. Tang, X. Li, and J. Winter, "Drainage structure datasets and effects on lidar-derived surface flow modeling," *ISPRS International Journal of Geo-Information*, vol. 2, no. 4, pp. 1136–1152, 2013.

[3] G. Sofia, G. D. Fontana, and P. Tarolli, "High-resolution topography and anthropogenic feature extraction: Testing geomorphometric parameters in floodplains," *Hydrological Processes*, vol. 28, no. 4, pp. 2046–2061, 2014.

[4] J. B. Lindsay and K. Dhun, "Modelling surface drainage patterns in altered landscapes using lidar," *International Journal of Geographical Information Science*, vol. 29, no. 3, pp. 397–411, 2015.

[5] S. Bhadra, R. Li, D. Wu, G. Wang, and B. Rekabdar, "Assessing the impacts of anthropogenic drainage structures on hydrologic connectivity using high-resolution digital elevation models," *Transactions in GIS*, vol. 25, no. 5, pp. 2596–2611, 2021.

[6] Y. Zhang, D. Pandey, D. Wu, T. Kundu, R. Li, and T. Shu, "Accuracy-constrained efficiency optimization and GPU profiling of CNN inference for detecting drainage crossing locations," in *Proc. of Workshops of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W)*, Denver, CO, USA, Nov 2023, pp. 1780–1788.

[7] Y. Li, J. Baik, M. M. Rahman, I. Anagnostopoulos, R. Li, and T. Shu, "Pareto optimization of CNN models via hardware-aware neural architecture search for drainage crossing classification on resource-limited devices," in *Proc. of Workshops of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W)*, Denver, CO, USA, Nov 2023, pp. 1767–1775.

[8] T. Kundu and T. Shu, "HIOS: Hierarchical inter-operator scheduler for real-time inference of DAG-structured deep learning models on multiple GPUs," in *Proc. of the 25th IEEE International Conference on Cluster Computing (Cluster)*, Santa Fe, NM, USA, Nov 2023, pp. 95–106.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[11] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[12] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, pp. 261–318, 2020.

[13] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International journal of computer vision*, vol. 38, pp. 15–33, 2000.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[16] J. Huang and H. Wang, "Small object detection by detr via information augmentation and adaptive feature fusion," *arXiv preprint arXiv:2401.08017*, 2024.

[17] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[18] D. Pandey, J. Ghebremichael, Z. Qi, and T. Shu, "A comparative survey: Reusing small pre-trained models for efficient large model training," in *Proc. of Workshops of IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W)*, Atlanta, GA, USA, Nov 2024.

[19] D. Pandey and T. Shu, "AM-DGCNN: Leveraging graph attention networks and edge attributes for link classification in knowledge graphs," in *Proc. of Workshops of IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W)*, Atlanta, GA, USA, Nov 2024.

[20] J. M. Wozniak, P. Davis, T. Shu, J. Ozik, N. Collier, I. Foster, T. Brettin, and R. Stevens, "Scaling deep learning for cancer with advanced workflow storage integration," in *Proc. of the 4th Workshop on Machine Learning in HPC Environments (MLHPC) in conjunction with ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Dallas, TX, USA, Nov 2018, pp. 114–123.

[21] N. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles *et al.*, "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[22] T. Shu, Y. Guo, J. Wozniak, X. Ding, I. Foster, and T. Kurc, "Bootstrapping in-situ workflow auto-tuning via combining performance models of component applications," in *Proc. of ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*, St. Louis, MO, USA, Nov 2021, pp. 1–15.

[23] ——, "POSTER: In-situ workflow auto-tuning through combining component models," in *Proc. of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, Seoul, South Korea, Feb-Mar 2021, pp. 467–468.

[24] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of DNN accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 58–68.

[25] A. A. Adegun, J. V. Fonou Dombeu, S. Viriri, and J. Odindi, "State-of-the-art deep learning methods for objects detection in remote sensing satellite images," *Sensors*, vol. 23, no. 13, p. 5849, 2023.

[26] A. B. Amjoud and M. Amrouch, "Object detection using deep learning, CNNs and vision transformers: A review," *IEEE Access*, vol. 11, pp. 35 479–35 516, 2023.

[27] L. Wang and A. Tien, "Aerial image object detection with vision transformer detector (vitdet)," in *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, 2023, pp. 6450–6453.

[28] Airbus Geo, "Airbus aircraft detection," 2023, accessed: 2024-07-30. [Online]. Available: https://www.kaggle.com/datasets/airbusgeo/airbus-aircraftssample-dataset

[29] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, Z. Wang, Y. Gong, and S. Belongie, "Dota: A large-scale dataset for object detection in aerial images," *arXiv preprint*, 2017.

[30] D. Wu, R. Li, B. Rekabdar, C. Talbert, M. Edidem, and G. Wang, "Classification of drainage crossings on high-resolution digital elevation models: A deep learning approach," *GIScience & Remote Sensing*, vol. 60, no. 1, p. 2230706, 2023.

[31] Y. Zhang, D. Pandey, D. Wu, T. Kundu, R. Li, and T. Shu, "Accuracy-constrained efficiency optimization and GPU profiling of CNN inference for detecting drainage crossing locations," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1780–1788.

[32] S. Jalalipour, S. Ayyalasomayjula, H. Damrah, J. Lin, B. Rekabdar, and R. Li, "Deep learning-based spatial detection of drainage structures using advanced object detection methods," in *2023 Fifth International Conference on Transdisciplinary AI (TransAI)*. IEEE, 2023, pp. 1–10.

[33] D. Sanmartín Carrión, V. Prohaska, and O. Diez, "Exploration of TPUs for AI applications," in *International Conference on Advances in Computing Research*. Springer, 2024, pp. 559–559.

[34] A. Shahid and M. Mushtaq, "A survey comparing specialized hardware and evolution in TPUs for neural networks," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 2020, pp. 1–6.

[35] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, "An evaluation of edge TPU accelerators for convolutional neural networks," in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, 2022, pp. 79–91.

[36] M. Elbtity, P. Chandarana, and R. Zand, "Flex-tpu: A flexible TPU with runtime reconfigurable dataflow architecture," *arXiv preprint arXiv:2407.08700*, 2024.

[37] K.-C. Hsu and H.-W. Tseng, "Accelerating applications using edge tensor processing units," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–14.

[38] S. Kim, C. Hooper, T. Wattanawong, M. Kang, R. Yan, H. Genc, G. Dinh, Q. Huang, K. Keutzer, M. W. Mahoney *et al.*, "Full stack optimization of transformer inference: a survey," *arXiv preprint arXiv:2302.14017*, 2023.

[39] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "Scale-sim: Systolic CNN accelerator simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[40] PyGAD, "Pygad documentation," https://pygad.readthedocs.io/en/latest/, accessed: 2024-07-29.

[41] S. Blum, R. Puisa, J. Riedel, and M. Wintermantel, "Adaptive mutation strategies for evolutionary algorithms," in *The Annual Conference: EVEN at Weimarer Optimierungsund Stochastiktage*, vol. 2, 2001.

[42] T. N. Platform, "Deep dive on google's exascale TPUv4 AI systems," 2022, accessed: 2024-08-01. [Online]. Available: https://www.nextplatform.com/2022/10/11/deep-dive-on-googles-exascale-tpuv4-ai-systems/

ARTIFACT DESCRIPTION/EVALUATION APPENDIX

A. Summary of the Experiments Reported

1) Abstract: We provide this artifact appendix to enhance the reproducability of our results.

2) Artifacts: The GitHub link of source code: https://github.com/SHUs-Lab/BTSD24AN

B. Experimental Setup

1) Relevant Hardware Details:

```
Machine:
  Type: Kvm
  System: Supermicro
  product: SYS-4029GP-TRT2
  v: 123456789
  Mobo: Supermicro
  model: X11DPG-OT-CPU
  v: 1.01
  UEFI: American Megatrends v: 3.3
  date: 02/21/2020
CPU:
  Topology: 2x 24-Core
  model: Intel Xeon Gold 5220R
  bits: 64
  type: MT MCP SMP L2
  cache: 71.5 MiB
  Speed: 1000 MHz
  min/max: 1000/4000 MHz
Graphics:
  Device-1: ASPEED Graphics Family
    driver: ast
    v: kernel
  Device-2: NVIDIA TU104GL [Quadro RTX 5000]
    driver: nvidia
    v: 535.104.05
```

2) Operating Systems and Versions:

```
System:
  Kernel: 5.4.0-182-generic x86 64
  Distro: Ubuntu 20.04.6 LTS (Focal Fossa)
```

3) Applications and Versions:

```
Client: Docker Engine - Community
 Version: 24.0.7
 API version: 1.43
 Go version: go1.20.10
 Git commit: afdd53b
 Built: Thu Oct 26 09:08:01 2023
 OS/Arch: linux/amd64
 Context: default
Server: Docker Engine - Community
 Engine:
  Version: 24.0.7
  API version: 1.43 (minimum version 1.12)
  Go version: go1.20.10
  Git commit: 311b9ff
  Built: Thu Oct 26 09:08:01 2023
  OS/Arch: linux/amd64
  Experimental: false
 containerd:
  Version: 1.6.25
  GitCommit:
    d8f198a4ed8892c764191ef7b3b06d8a2eeb5c7f
 runc:
  Version: 1.1.10
  GitCommit: v1.1.10-0-g18a0cb0
 docker-init:
  Version: 0.19.0
  GitCommit: de40ad0
```

4) Libraries and Versions:

```
Docker image:
  pytorch/pytorch:2.3.0-cuda12.1-cudnn8-runtime

Installed libraries:
  cython==3.0.10
  numpy==1.23.3
  scipy==1.13.1
  onnx==1.16.1
  onnxruntime==1.18.0
  pandas==2.2.2
  rasterio==1.3.10
```

More complete information on the Docker image source, libraries, and versions are in the Dockerfile and requirements.txt within the GitHub repository.

5) Input Datasets: The preprocessed, training-ready dataset is available at: https://figshare.com/articles/dataset/Processed_Data_for_Exploration_of_TPU_Architectures_for_the_OptimizedTransformer_in_Drainage_Crossing_Detection_/27711249?file=50460957

C. Evaluation Experiments

1) Experiments on Transformer Design: The pre-trained DETR-ResNet50 model is cloned from Facebook Git repository: facebookresearch/detr.git
COCO-Evaluation Metrics are borrowed for the model's object detection performance.

2) Experiments on TPU Architecture Auto-tuning: DETR model with ResNet50 backbone model was loaded as pre-trained from HuggingFace library: facebook/detr-resnet-50

```
Installed libraries:
    pygad==3.0.0
    numpy
    scalesim
    pandas
    torch
    torchvision
    transformers
```