Comparative analysis on ViT and Small task specific vision model using Sen1Floos11 Dataset

Reyad Hasan Rinkon

May 14, 2025

1 Introduction

The sen1floods dataset is globally diverse collection of satellite images that is created to support flood mapping research. It contains 446 manually labeled 512*512 image chip from 11 flood events, making it ideal for training on flood-maping task[3]The Sen1Floods11 dataset is a useful resource for building and testing deep learning models that detect floods using Sentinel-1 radar data. Since radar can be useful with cloud scenarios as it's great for getting quick, reliable information during floods.
Recently, geospatial AI has made a lot of progress thanks to various benchmark models and robust datasets [4][5] [6].On the other hand , models made and trained for specific tasks such as (CNNs) are still widely used because they work well with a smaller compute power and a small dataset.
This project focuses on fine-tuning the Prithvi-EO-2.0 foundational model (a pre-trained vision transformer developed by NASA and IBM)[7] and compares it with a small task-specific model trained from scratch using the sen1Floods11 dataset. By testing both models on the same flood mapping task, we want to see the trade-off between big pre-trained models and small task-specific models for real-world flood detection. Some key challenges are making the model adaptable with only a small amount of labeled data and testing how sensitive it is to different amounts of data.

2 Literature Review

The Sen1Floods11 dataset has become a standard for evaluating flood segmentation models.[8]. Research using this dataset shows that Sentinel-1 works better than Sentinel-2 for real-time flood mapping because it can see through clouds.[8]. However dealing with speckle noise is still a challenge.
Pretrained vision transformers (ViTs) and smaller models made for specific tasks have different tradeoffs between speed and accuracy.Pretrained ViTs like TinyViT perform really well (84.8% accuracy on ImageNet-1k with just 21 million parameters) by learning from bigger models through knowledge distillation,

which also makes them easier to use for other downstream tasks.[1] Prithvi-EO-2.0 is pretrained on 4.2M Harmonized Landsat-Sentinel samples, introduced temporal and location embedding. In Sen1Floods11 dataset it achieves 88.61 % accuracy mIoU through finetuning. The 600M paramter version of this model outperformed many task specific CNNs in cross domain generalization.

Convolutional neural networks (CNNs) have dominated flood segmentation tasks, particularly U-Net variants achieving median F1 scores of 0.91 for flood detection using Sentinel-1 SAR data.[9] While CNNs traditionally outperformed ViTs on limited data, recent advancements show smaller ViTs pretrained with multi-scale strategies can match or exceed larger models in accuracy and computational efficiency (2-4x fewer resources) [2]. Although unseen small dataset lead to poor generalization as transformer require large data to do good generalization.

3 Proposed Work

For the Prithvi-2.0 model I choose "prithvi eo v2 300 mae" variant as a backbone for the pretrained ViT model task. It takes 6 spectral band as a single input (Blue, Green, Red, Narrow IR, SWIR1, SWIR2).These bands were selected to align with the Harmonized Landsat-Sentinel (HLS) product used during pretraining.There are no version of PrithVi-2.0 pretrained on only SAR bands or inputs excluding these 6. I process these band from the Sen1Floods11 dataset. This model also takes temporal snapshots as a input. I used it by copying the single-date data 4× along the temporal axis. Duplication mimics temporal consistency as real time-series data is unavailable.

I added a CNN head after the Prithvi-2.0 model.The CNN head I attached consists of four sequential CNN blocks designed to process the model's ViT encoder outputs into a flood mask. Where each block includes a `Conv2D` layer, a ReLU activation function, and a Batch Normalization layer. The model outputs a tensor of shape `Batch` $\times$ `512` $\times$ `512`, with predicted class values ranging from 0 to 2. To align with the ground truth mask labels of -1, 0, and 1, the mask is adjusted during the training process.

I conducted experiments by training the model with different fraction of training data(100%, 50%, 25% and 10%).I used the training, testing and validation split that were already present in the dataset. There were total 256 data for training.And the others were equally split on testing and validation. For calculating the validation and testing mIoU the number of validation and training data were not changed to ensure identical condition.I evaluated the models various training strategy by calculating the mIoU (mean intersection over union). To find out the CNN heads importance in this training I also ran a similar experiment without the CNN head. This time I used the same model but pretrained on the same Sen1Floods11 dataset.

For the task specific small model. I used a CNN based model that follows the encoder decoder architecture (Figure 2). There are 4 encoder block and 4 decoder block. Each encoder block consists of a `Conv2D` layer, followed by Batch

Normalization, a ReLU activation function, and a MaxPooling layer. And each decoder block consists of a `Conv2D` layer, followed by Batch Normalization, a ReLU activation function, and an Upsampling layer. Very similar to encoder block just instead of Maxpooling I used a upsampling layer. Also ran the similar experiment with similar data fraction percentages and calculated mIoU. While calculating the mIoU I decided to ignore the No Data (-1 labeled in the masks).

The input size is similar to that of Prithvi-2.0 based model. I maintained the same input data dimensions (copying along the temporal axis) to ensure identical scenario.

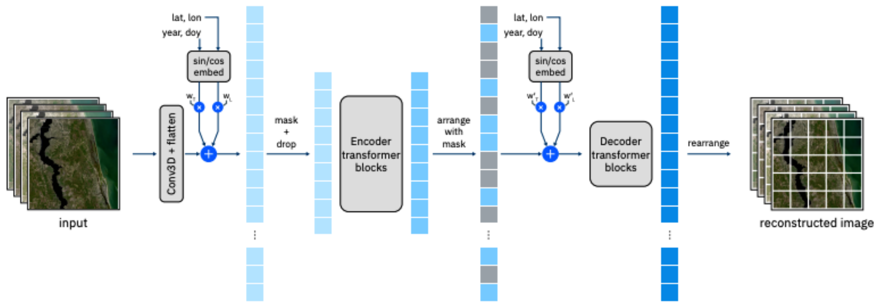

Figure 1: Prithvi Architecture [7]

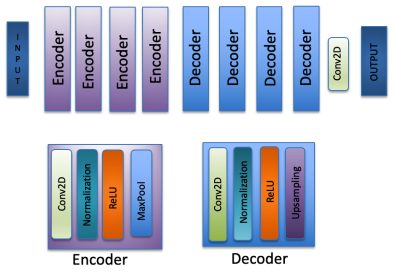

Figure 2: Block Diagram of CNN based model

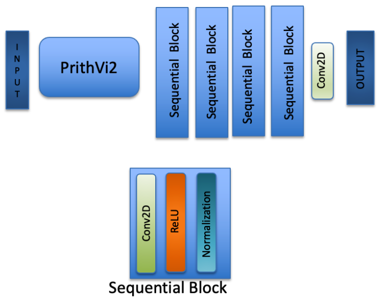

Figure 3: Block Diagram of Prithvi-2.0 based model

4 Results and Discussion

From the Figure 4 where it shows the sensitivity analysis. The effect of the training data fraction with respect to mIoU. We can see that for the 100% data fraction the model performed reasonably well. After 100 epochs it went above 0.9 that is on average more than 90% of the time the model returned similar result as the ground truth. With 50% of the training data it also showed remarkable results. That is with only 127 training data. It went above the 100% section in the first few epochs. But after running it for a long time it converged where it actually should be. But for 25% and 10% the mIoU curve fluctuated. That means model is not performing consistently in these situations. And for the experiment that is run without the CNN head with sen1Flood11 pretrained prithvi-2.0 model the graph becomes a straight like along x-axis. And with changing the data fraction it did not effect at all. That specify the importance of the CNN head.

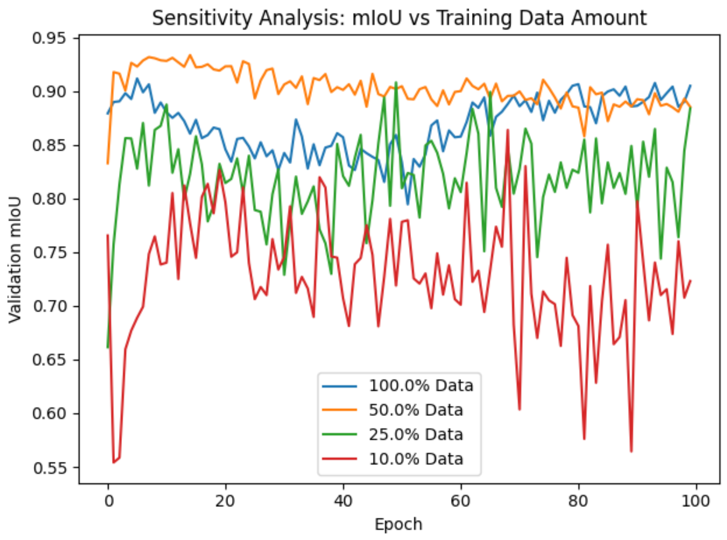

Figure 4: Prithvi2.0: % Data vs Validation mIoU

The figure 5 shows the similar analysis for CNN based model. It shows that it is a small model compared to the Prithvi-2.0 (around 100 times smaller in terms of trainable parameter) but it showed very good results than the counterpart.For 100% data it correctly gave accurate output for around 95% of the time. The 50% module also performed well and does not fluctuate that much compared to 25% and 10% section.

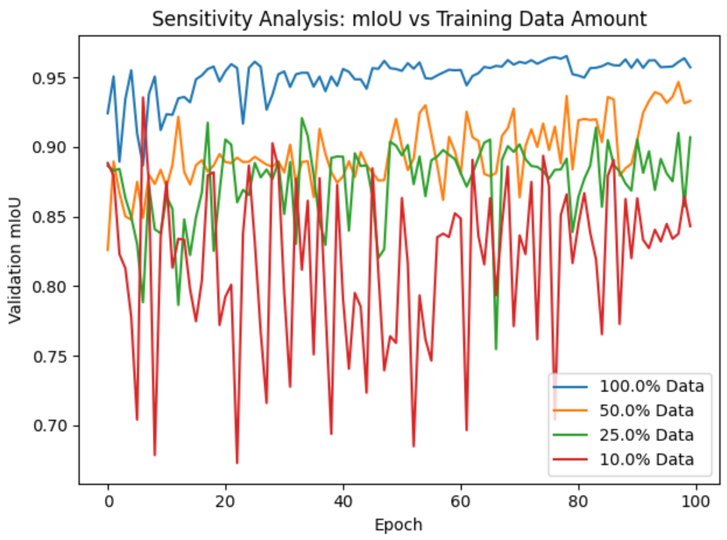

Figure 5: Task-Specific Model: % Data vs Validation mIoU

% Data/Test mIoU	**Prithvi2.0 with CNN head**	**CNN**	**Backbone Prithvi2.0**
100.0% Data	0.9044	0.9391	0.886
50.0% Data	0.868	0.916	0.886
25.0% Data	0.8571	0.9016	0.886
10.0% Data	0.6945	0.8179	0.886

Table 1: Comparison of Test mIoU with different training data proportions (ViT vs. Task specific model).

From Table 1 we can see that CNN based model outperformed in terms of test mIoU than Prithvi-2.0 with CNN head. The test here performed on the test dataset that is provided in the Sen1Floods11 dataset. And among backbone and backbone with CNN head the later learned the dataset with 100% data better than the pretrained model.

5 References

References

[1] Wu, Kan, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. "Tinyvit: Fast pretraining distillation for small vision transformers." In European conference on computer vision, pp. 68-85. Cham: Springer Nature Switzerland, 2022.

[2] Nauen, Tobias Christian, Sebastian Palacio, Federico Raue, and Andreas Dengel. "Which Transformer to Favor: A Comparative Analysis of Efficiency in Vision Transformers." In 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 6955-6966. IEEE, 2025.

[3] Bonafilia, Derrick, Beth Tellman, Tyler Anderson, and Erica Issenberg. "Sen1Floods11: A georeferenced dataset to train and test deep learning flood algorithms for sentinel-1." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp. 210-211. 2020.

[4] Blanchon, Victor. "UC Merced Land Use Dataset." Hugging Face, August 21, 2024. Available: https://huggingface.co/datasets/blanchon/UC$_{M}erced.$

[5] Earth Engine Data Catalog — Google for Developers. (n.d.). Google for Developers. https://developers.google.com/earth-engine/datasets

[6] SAT-4 and SAT-6 airborne datasets. (n.d.). https://csc.lsu.edu/ saikat/deepsat/

[7] Szwarcman, Daniela, Sujit Roy, Paolo Fraccaro, orsteinn Elí Gíslason, Benedikt Blumenstiel, Rinki Ghosal, Pedro Henrique de Oliveira et al. "Prithvi-eo-2.0: A versatile multi-temporal foundation model for earth observation applications." arXiv preprint arXiv:2412.02732 (2024).

[8] Konapala, Goutam, Sujay V. Kumar, and Shahryar Khalique Ahmad. "Exploring Sentinel-1 and Sentinel-2 diversity for flood inundation mapping using deep learning." ISPRS Journal of Photogrammetry and Remote Sensing 180 (2021): 163-173.

[9] Tavus, Beste, Recep Can, and Sultan Kocaman. "A CNN-based flood mapping approach using sentinel-1 data." ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 3 (2022): 549-556.