One Month Progress Report

INTRODUCTION

This project has now passed the one-month mark, and it seems fitting for a more extensive progress report. To get everyone up to speed, I will summarize the project's problem statement, outline, and deliverables below.

Project Summary

As part of their work, the SOC Team requires the use of high-resolution satellite imagery. They ask from the imagery providers a large area, on the scale of an entire county, which quickly becomes expensive both in terms of money and compute. Moreover, since they are focusing on detecting construction changes, there are areas in a county that do not undergo construction. This means that some of the imagery that they are asking for is not needed. Thus, there is a clear opportunity to save costs by figuring out a way to limit the high-resolution imagery collected to just those areas that are very likely to experience construction.

Project Outline

The following is a rough outline of the project steps and their completion percentage. Note that the outline is subject to change as the project continues. Unexpected blockers or advancements could change or open new pathways for the project as it nears the finish line.

Step	Item	Sprint and Status	Notes
1	Choose locations with active construction.	**Sprint 134** **Done**	There is a lot of construction occurring north of Dallas, TX. I am focusing on geohashes `9vgm[d|6|0|1]`, among others nearby.
2	Collect imagery and other data for two locations. The dataset will include the following as features: time, true color, NDVI, elevation, slope, aspect, (optional: proximity to roads). *Key technology: Google Earth Engine.*	**Sprint 134** **Done**	Originally, this was done locally but I am shifting the storage location and logic to point to an S3 bucket at the following address: s3://rgc-zarr-store. Google Earth Engine offers low-resolution imagery for free, making it the perfect tool for this project.
3	Build training/validation/testing dataset or "datacube." *Key libraries: xarray, rasterio*	**Sprint 135** **Done**	I created additional channels in the dataset based on time, with some derived from NDVI, so that the model could capture time and landscape seasonality as a factor.
4	Model this project's architecture, training, and testing pipelines off of other projects. *Key libraries: PyTorch, GitHub (for researching other projects)*	**Sprint 135** **Done**	I based the model on two previous projects to help scaffold the time-aware model: • https://github.com/spaceml-org/RaVAEn • https://github.com/wwdAlger/DSFANet
5	Test model on each location. Location A will be used to train the model, and Location B will be used to test the model. Then the reverse: Location B will be used to train the model and Location A will be used to test the model.	**Sprint 135** **Done***	The MVP has been completed, namely a PyTorch "best model"… *…though work still needs to be done to improve the heatmap and prediction outputs.
6	Provide inference products that could take the form of heatmaps	**Sprint 136** **25%**	The model pipeline can output a rudimentary heatmap and change mask image, but more work needs to be done to improve them.
7	Get more data, improve and retrain the model, output predictions… rinse and repeat	**Sprint ≥136** **10%**	The initial heatmap and change masks are not satisfactory and certainly are not production-ready. I'd like to improve the model before declaring it finished.

One Month Progress Report

Project Deliverables

As stated in the project outline, I aim to produce a working model that can predict with reasonable accuracy where construction is likely to take place given an input area. I will also have the model and surrounding infrastructure output the following:

- Imagery of the input area
- A heatmap of likely change based on a threshold value
- A mask based on the threshold value and heatmap
- (Stretch goal) A Streamlit dashboard where a user can interact with the model
- Other outputs that the team requests

SPRINT 134 RECAP

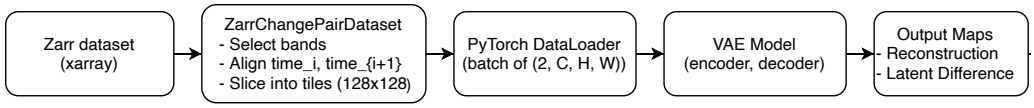
The beginning of this sprint coincided with the beginning of this project. As such, I focused on experimenting and testing the Google Earth Engine APIs. I also explored using other data sources, like NDVI and the road network[1], to enrich the model.

As the sprint went on, I improved the code's automation abilities by creating functions that could generate a full `xarray` dataset given a provided geohash.

SPRINT 135 RECAP

This sprint marked the completion of the initial model construction, training, validation, and testing phase. As the outline describes, I used two previous projects' methods when creating this project's model architecture, which I describe in more detail here.

The flowchart below shows the basic modeling strategy, where the model ingests a Zarr-based dataset, the model prepares the dataset for training, validation, and testing, and then loads it to the model



I opted for an unsupervised learning approach, that is, without labels pointing to where construction takes place, as it allowed me to work more quickly and not figure out ways to procure and register labels to the training and testing datasets. In a future iteration of the model, I may want to incorporate labels to the training set if we realize we need it to improve the model. One of the sources for this project, RaVAEn, is based on utilizing Variational Auto Encoders, or VAEs. These are designed to be used in unsupervised representation learning and are perfect for our use case. VAEs learn to encode input data into a compressed representation. They are useful for anomaly and change detection. Another key structure borrowed from RaVAEn is training the model to detect change based on two time points.

I incorporated DSFANet's use of time as a factor in the training pipeline by including channels derived from the timestamp, namely Δt (change in time since initial image) and day-of-year. I also added seasonality to the NDVI data by creating a normalized NDVI per season, so that changes in NDVI throughout the year are accounted for (i.e., winter vegetation is less thick than summer vegetation). DSFANet also focused on "slow feature loss", the idea that latent representations remain stable over time, unless there's a real change.

After some initial runs, I improved the stability of the metrics by normalizing the input channels so that outliers don't blow up the scores, and re-ran the model training.

SPRINT 136 RECAP THUS FAR

This week involved me improving helper tools surrounding the model. I wanted to create two tools: a CLI-based dataset maker that saves and updates datasets to the S3 bucket, and a CLI-based inference tool that takes a trained model and creates a report containing the t0 and t1 images, a heatmap, and a change mask.

The `build_dataset.py` script can dynamically update or create a dataset of a 5-precision geohash and save it in the S3 bucket. More work will have to be done to the ingestion logic so that the model can point to an S3 bucket instead of files on my local machine. The thinking here is that I will be able to train the model on more data, much more than I could store on my local machine. Each dataset can be quite large, anywhere between 5 and 40 GB when fully loaded, so the cloud-storage solution will be helpful. This function is largely complete. I would imagine it needs tweaks if downstream we realize we need the dataset to include certain things that aren't there right now.

The `inference.py` script is a work in progress. It still is pointing to a local file rather than the files on the S3 bucket. It can output the two time slice's images, a heatmap, a change mask, and some metrics, though the first time slice is currently showing up as black. This is a bug and will be dealt with.

In the second week of this sprint, I plan to update the model to accept S3-based datasets rather than local ones.

CONCLUSION AND NEXT STEPS

I am very pleased with the pace of progress with this project. We already have an MVP (we are able to train a model) and are working on improving the outputs of the `inference.py` script's predictions. I also think having a better-trained model will help the `inference.py` script give us more meaningful results.

With a month to go, I can confidently say that we are on track to finish this project on time.

I am curious about changing the focus of the model outputs from "I'll give you TWO time points in the past and tell me what HAS changed" to "I'll give you A SINGLE time point and tell me what WILL change". This is an artifact from relying on RaVAEn/DSFANet's model architectures. I think this new approach will be more helpful for the SOC Team, as when it comes time for them to procure imagery, it would be good for them to already have a sense of where the likely hotspots of construction will be. This ties directly to the problem statement for this project.

One stretch goal for this project would be to create a Streamlit dashboard where a user can interact with the trained model, making it user-friendly for those of differing technical abilities. Another stretch goal would be to dramatically increase the training areas to include those outside of the fast-growing cities in Texas. Other State's landscapes and construction behaviors will look differently (i.e., from space) and a better model would be able to better gracefully accept the diversity of urban landscapes in the US when making predictions.

One Month Progress Report

[1] Incorporating the road network I thought was a unique use of free data sources: I used Open StreetMap's tools to download a roads layer, clipped to the specific target geohash. You can navigate to their North American data here: https://download.geofabrik.de/north-america/us/. Open Street Map's roads data is updated daily, but merged in the following pattern:

- The past week is saved into individual layers.
- The months of the current year are saved, and
- Past years are merged into single files

The data is very large; Texas' road network was over 600 MB for each file. In order to convert the road network, which is a vector layer, into a raster layer (and thus added to the `xarray` dataset), I utilized a Kernel Density Estimation to create a continuous surface of scalar values, where higher values indicate a higher density of roads. The thinking is that construction requires the use of the road network, and if there aren't a lot of roads nearby, the likelihood of construction is low. I think the promise of this is high, though the significant drawback of using this layer is that the road network's change cadence is temporally a lot slower than satellite imagery. Changes in the landscape, picked up by a satellite, can happen on a day to second timescale, whereas the road network changes slowly. Perhaps a future model, if it could incorporate a much wider area, like a large portion of a state, would be able to make sense of the road network as a meaningful feature for the model.