III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

I am a Geospatial Data Scientist at Dendra Systems. Here I develop algorithms to process terabytes of satellite and airborne data – work that enables Dendra Systems to monitor and restore complex and biodiverse ecosystems. Prior to Dendra, I spent multiple years studying and working in the field of Geoinformatics in Ukraine, Sweden, and Germany, after which I earned my PhD degree in Geography at UNSW in Australia. Currently, my work is centered around the integration of Remote Sensing and Machine Learning for monitoring reforestation and ecological restoration projects.

2. What motivated you to compete in this challenge?

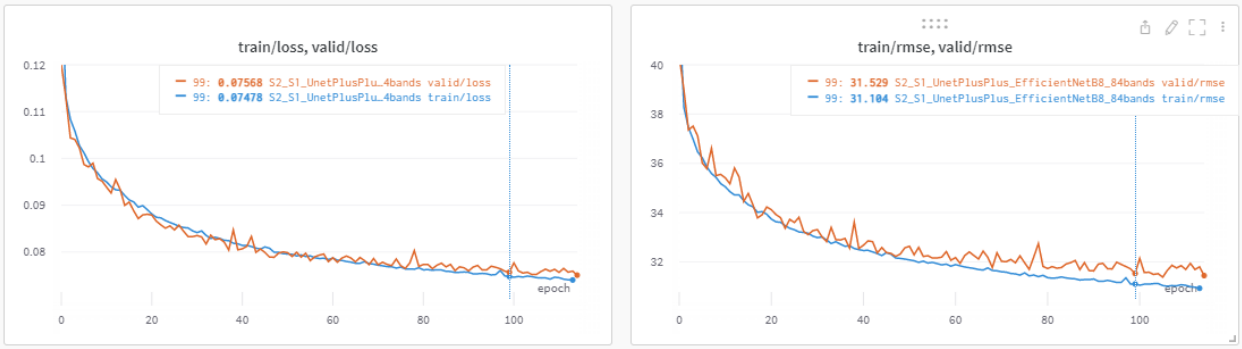
As the first DrivenData competition I have participated in, I was drawn to the BioMassters challenge because of the importance of its mission. Forests play a critical role in removing carbon dioxide from the atmosphere and storing it as biomass. The BioMassters challenge is one of the global community's efforts to protect and restore forests, and I wanted to be part of the solution. Hopefully, my work will bring us a step closer to knowing how much carbon is being sequestered by forests and monitoring change over time, and to the ultimate goal of ensuring accountability and building community confidence in the value of carbon sequestration projects. I also wanted to see how my machine learning skills stack up against the best in the field.

3. High level summary of your approach: what did you do and why?

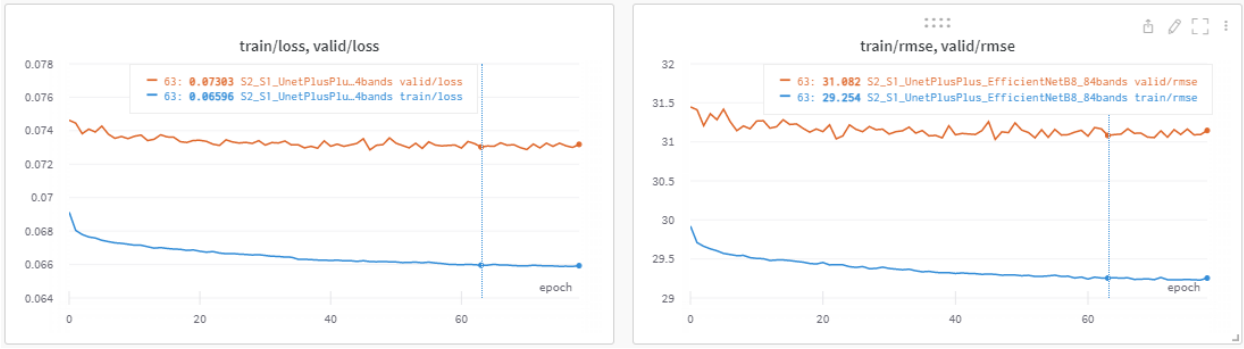
- *Sentinel-1 and Sentinel-2 imagery were preprocessed into 6 cloud-free median composites to reduce data dimensionality while preserving the maximum amount of information*
- *15 models were trained using a UNet++ architecture in combination with various encoders (e.g. se_resnext50_32x4d and efficientnet-b{7,8}) and median cloud-free composites. From the experiments, UNet++ showed better performance as compared to other decoders (e.g UNet, MANet etc.)*
- *The models were pretrained with multiple augmentations (HorizontalFlip, VerticalFlip, RandomRotate90, Transpose, ShiftScaleRotate), batch size of 32, AdamW optimizer with 0.001 initial learning rate, weight decay of 0.0001, and a ReduceLROnPlateau scheduler*
- *UNet++ models were optimized using a Huber loss to reduce the effect of outliers in the data for 200 epochs*
- *To improve the performance of each UNet++ model they were further fine-tuned after freezing pre-trained encoder weights for another 100 epochs*
- *For each model the average of 2 best predictions was used for further ensembling*
- *The ensemble of all 15 models using weighted average was used for the final submission*

4. Do you have any useful charts, graphs, or visualizations from the process?

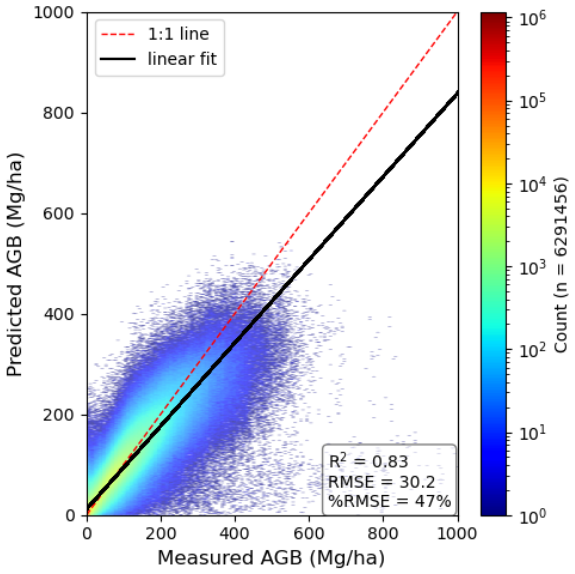
Here are typical learning curves of the UNet++ model with efficientnet-b7 encoder:



And here are the learning curves for fine-tuning the same model after freezing pre-trained encoder weights:



Here's the per-pixel dataset-wide performance of the ensemble model on the validation set:



5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

- *Median cloud-free image compositing (see: 1_preprocessing.ipynb), which allowed to reduce data dimensionality at the marginal loss of accuracy. For example, here is the performance of the UNet++ model with se_resnext50_32x4d encoder using different median compositing:*

Median compositing	2S	3S	4S	6S
Channels	28	42	56	84
RMSE (val)	29.3894	29.0204	28.7029	28.7986

```
def preprocess_s1(uID, indir, outdir, composite_type='4S')

def preprocess_s2(uID, indir, outdir, composite_type='4S')
```

- *Ensembling using the weighted average (see: 4_ensemble.ipynb), which allowed boosting the performance from RMSE of 28.5587 of the best individual model to RMSE of 27.8108 on the validation set:*

```
pred = np.stack((m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12,
m13, m14, m15), axis=0)
popt, pcov = curve_fit(f_15m, pred, gt)
```

- *Encoder weight freezing (see: utils.py and 2_train.ipynb) and further fine-tuning, which allowed improving the performance of each model on average by RMSE of ~0.1, which was enough to overtake the next best submission on the private leaderboard by RMSE of 0.0064.*

```
def freeze_encoder(model):
    for child in model.encoder.children():
        for param in child.parameters():
            param.requires_grad = False
    return

freeze_encoder(model)
```

6. Please provide the machine specs and time you used to run your model.
    - *CPU (model): AMD Ryzen 9 3950X*
    - *GPU (model or N/A): NVIDIA GeForce RTX 3090*
    - *Memory (GB): 128 GB*
    - *OS: Windows 11*
    - *Train duration: ~360 hours (15 models @ ~24 hours/model) on 1 GPU*
    - *Inference duration: ~3.5 hours (~15 min/model)*
7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?
    - *Make sure you train models in a mixed precision regime (precision=16) to fit batch sizes of 32 into the memory on RTX 3090*
    - *When training models using 6S median composite data, I ran into NaNs in the validation loss, which I "fixed" using y_hat = torch.nan_to_num(y_hat) (see L877 in utils.py)*
8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

    Yes, half-through model training I decreased the validation set from 10% (869 samples) to 1.1% (96 samples). So, the first 9 models were trained using 10% validation set, and the last 6 models were trained using 1.1% validation set.
9. How did you evaluate performance of the model other than the provided metric, if at all?

I only calculated Huber loss and RMSE on the validation set, which I decreased from 10% to 1.1% for some models (see above).

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    - *MSE loss on logarithmic and sqrt scaled agb data*
    - *Model ensembling using LightGBM*
    - *Model stacking using a CNN-based meta learner*
    - *Adding vegetation indices for Sentinel-2 and VV/VH ratio for Sentinel-1 images*
    - *TweedieLoss and RMSLELoss*

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    - *Incorporating time and location information for each pixel (i.e. latitude and longitude)*
    - *Incorporating elevation and land cover information*
    - *Continue experimenting with other loss functions*
    - *Cross-validation*
    - *Potentially better architectures (e.g. UNet3+ or Transformers)*
    - *Getting more GPUs for hyperparameter tuning and faster training/inference*