III. Model documentation and write-up

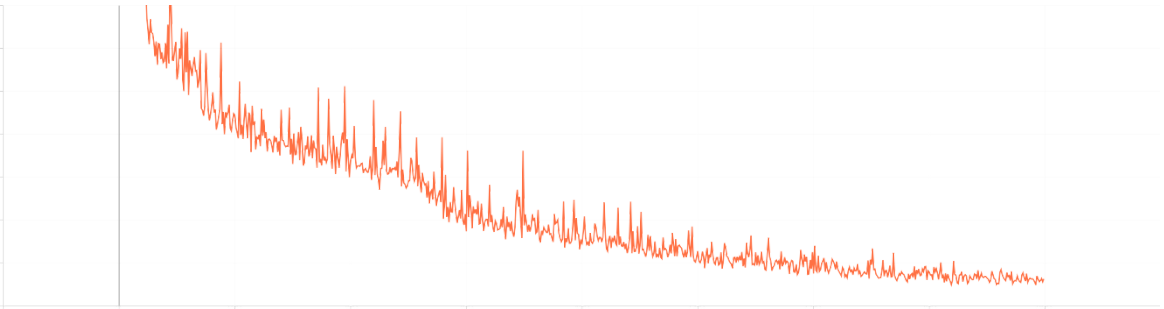
Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

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'm doing research at the University of Montréal in computer graphics. I got my master's in mathematics at the Novosibirsk State University, Russia. I am involved in an educational program where I teach machine and deep learning courses. I am a research supervisor of undergraduate students and bachelors' theses.

2. What motivated you to compete in this challenge?
   Machine learning is my passion and I often take part in competitions.

3. High level summary of your approach: what did you do and why?
   The solution is based on an UNet model with a shared encoder with aggregation via attention. The inputs to the encoder are 15-band images with a resolution of 256x256 from joint Sentinel-1 and Sentinel-2 satellite missions. The encoder is shared for all 12 months. The outputs are aggregated via self-attention. Finally, a decoder takes as inputs the aggregated features and predicts a single yearly agbm. Optimization was performed using `AdamW` optimizer and `CosineAnnelingLR` scheduler and `RMSE` loss. We don't compute loss for high agbm values (>400). We use vertical flips, rotations, and random month dropout as augmentations. Month dropout simply removes images.

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



5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   1) Attention model to aggregate features of ecnoder
   ```
   models.py:8:
   class AttentionPooling(nn.Module):
       def __init__(self, embedding_size, hid=None):
           super().__init__()
   ```

```
        hid = embedding_size if hid is None else hid

        self.attn = nn.Sequential(
            nn.Linear(embedding_size, hid),
            nn.LayerNorm(hid),
            nn.GELU(),
            nn.Linear(hid, 1)
        )

    def forward(self, x, bs, mask=None):
        # x: [B*T, D, H, W]
        # mask: [B, T]
        _, d, h, w = x.size()
        x = x.view(bs, -1, d, h, w)
        x = x.permute(0, 1, 3, 4, 2)  # [bs, t, h, w, d]

        # x:    [B, T, *, D]
        attn_logits = self.attn(x)  # [B, T, *, 1]
        if mask is not None:
            attn_logits[mask] = -torch.inf

        attn_weights = attn_logits.softmax(dim=1)  # [B, T, *, 1]

        x = attn_weights * x  # [B, T, *, D]
        x = x.sum(dim=1)   # [B, *, D]

        x = x.permute(0, 3, 1, 2)     # [bs, d, h, w]

        return x, attn_weights
```

2) Month dropout augmentation

```
dataset.py:89:
    if random.random() > 0.5:  # "word" dropout
        while True:
            mask2 = np.random.rand(*mask.shape) < 0.3
            mask3 = np.logical_or(mask, mask2)
            if not mask3.all():
                break

        mask = mask3
        imgs[mask2] = 0
```

3) RMSE loss which ignores very high agbm values

```
train.py:405:
class NoNaNRMSE(nn.Module):
    def __init__(self, threshold=400):
        super().__init__()

        self.threshold = threshold

    def forward(self, logits, target):
        not_nan = target < self.threshold

        logits = logits.squeeze(1)
        diff = logits - target
        diff[~not_nan] = 0
        diff2 = torch.square(diff)
        diff2m = (diff2 / not_nan.sum((-1, -2), keepdim=True)).sum((-1, -2))
        diff2msqrt = torch.sqrt(diff2m)

        rmse = diff2msqrt.mean(0)

        return rmse
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): AMD Milan 7413 @ 2.65 GHz 128M cache L3
   - GPU (model or N/A): Nvidia A100 40GB VRAM
   - Memory (GB): 64
   - OS: GNU/Linux
   - Train duration: 8 days
   - Inference duration: 5 min

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.)?
   The submission is generated using Nvidia A100 GPU 40Gb. I got slightly different results on Nvidia V100 32Gb.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   No

9. How did you evaluate performance of the model other than the provided metric, if at all?
   -

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

I tried 3D CNNs, segformer models, aggregation level on different levels (before encoder, after encoder, after decoder), different attention models, group norm.

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?
    Make predictions non-negative, i.e. put `relu` layer on top logits