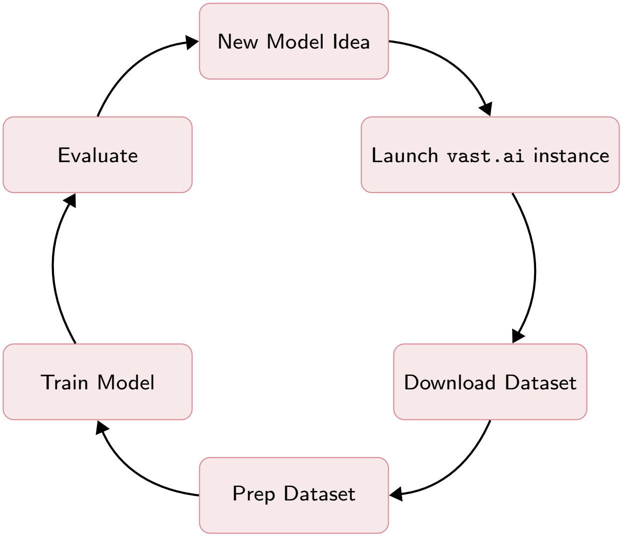
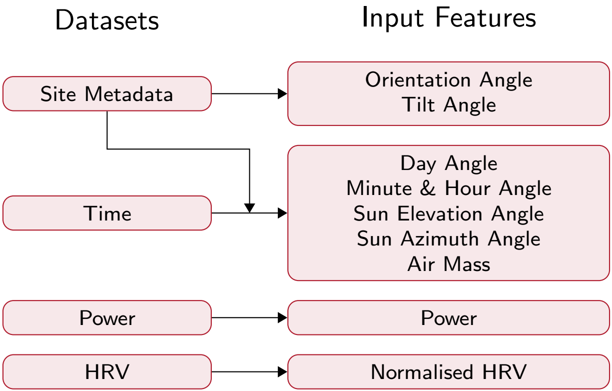
Observations

- This is an engineering challenge: clever optimisations $>$ research ideas
- Validation is volatile: ± 0.003 MAE on leaderboard for similar local validation score

Our approach



The Dataset

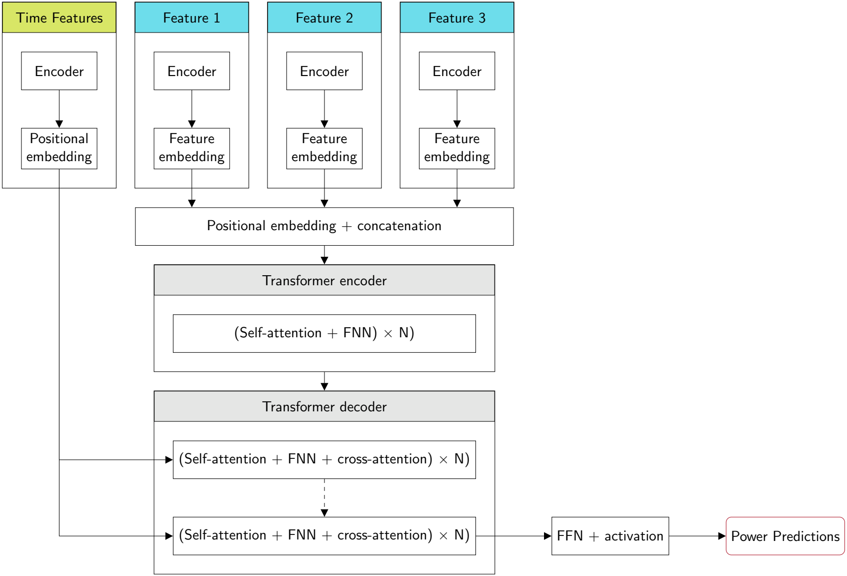


- All angles encoded as $(\sin\theta, \cos\theta)$ pair
- Prepped dataset saved as `torch.bfloat16`

The Architecture

1. Final architecture: CNN, FFNN $\rightarrow$ Transformer
2. Useful tricks: fancy transformer things, custom activation, stacking for image data
3. Other experiments: PerceiverIO, heightmap, weather forecasts
4. Misc. implementation details: code structure & hyper-parameters

Model Architecture



Tricks

- *Sub-LN* in the feedforward layer as per the *Magneto* architecture
- Grouped Query Attention in the transformer, *SwiGLU* activations
- Custom activation function after the final layer:

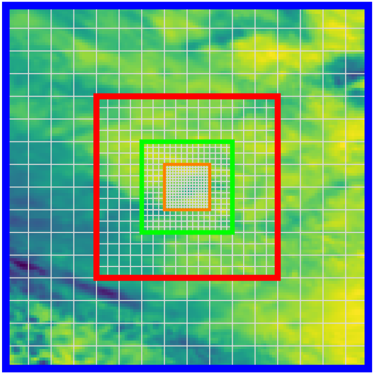
$$\mathbf{w} = \mathbf{p} \cdot \mathrm{softmax}(\mathbf{S})$$

$$\mathbf{n} = \frac{\mathbf{w}}{1 - \mathbf{w}}$$

$$\mathbf{y} = \frac{\mathbf{n}}{\mathbf{n} + e^{-\mathbf{x}}}$$

  Where: $\mathbf{p}$ is the vector of previously observed PV outputs, $\mathbf{S}$ a learned matrix (12×48) and $\mathbf{x}$ the output of the model before the activation. This is equivalent to a sigmoid centered around a weighed average of the inputs.

Stacked images



- We want to preserve low level details close to the target, and high level details further away
- Take N crops of the image, downsample them to the same resolution
- Stack the results on top of each other
- Multiply CNN channels by N and use N groups for each convolution layer, use fewer CNN layers

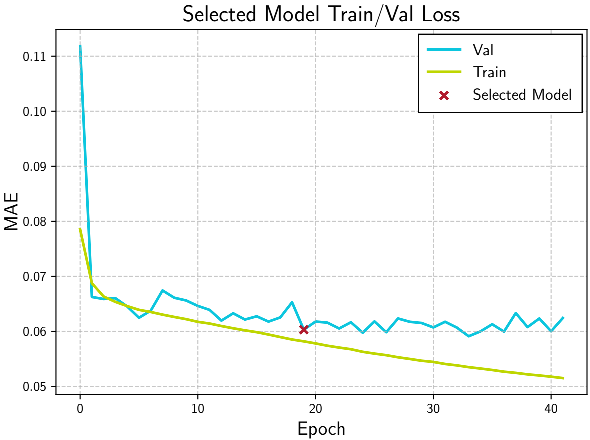
Other experiments

- **PerceiverIO**: Architecture easier to expand, slightly worse results
- **Elevation over chunk**: Over-fitting on site characteristics in a given year
- **Weather dataset**: Lack of time, straightforward addition

Misc. details

- CNN: `ResNet`-style, 3 layers per block, 5 blocks, a few optimisations, `Swish` activations
- Transformer: 16 heads, 8 groups, hidden dim: 640, forward expansion: 3, 5 layer for encoder and decoder, `SwiGLU` activations
- Optimiser: `RAdam`, LR: $1e-5$, WD: $5e-6$.
- PyTorch Lightning for training, W&B for tracking

Training



- Model size $\sim$ 46 M

Model Comparison

1. **Transformer**: 0.08437 4h, 0.06202 1h
   - Best performing model
   - Complex: self-attention requires custom model for each modality or it becomes computationally expensive
2. **PerceiverIO**: 0.08633 4h, 0.06252 1h
   - Worse than above Transformer
   - Easier to work with
   - Can use same tricks to improve
3. **EurNeXt (no HRV)**: 0.093 4h, 0.067 1h
   - Surprisingly good for just using power and site metadata

The Conclusion

- Contributions: a bunch of useful tricks (custom features, stacking cropped images, custom activation function)
- Observations: for multi-modality, a carefully tweaked transformer seems to work better than more convenient PerceiverIO
- Potential improvements:
  1. Add more data sources
  2. Windowed attention
  3. Make model bigger, train on more data