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?

I'm a machine learning engineer at Mapbox where I often solve ML problems related to automatic map improvements. I have been participating in data science competitions for 3 years and consider that as my hobby.

2. What motivated you to compete in this challenge?

I worked with satellite/aerial imagery in a lot of challenges and at my work. Building a robust model that can work with off nadir imagery to predict building footprints is a challenging task that cannot be solved without a 3D dataset, which are usually not available. In this challenge the dataset had full information to train an end-to-end model which can transform RGB images to 3D.

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

While meta-architecture is mostly the same as provided baseline there are crucial differences that allow to increase single model score by 10%.

Encoder

I replaced Resnet34 with a very powerful encoder EfficientNet V2 L which has a huge capacity, receptive field and in addition to that it is less prone to overfitting.

During my experiments I also tried EfficientNet B7 which worked a bit worse and ResneSt-200. ResneSt-200 based model worked really great on validation split but performance on the leaderboard was 2% worse than with EfficientNets. That led me to a conclusion that there is an implicit leak in the dataset.

Decoder

UNet like encoder with more filters and additional convolution blocks for better handling of fine grained details. The resolution of some images are not very high (50cm per pixel) and pixel level details are important to obtain a high score.

Training

Downsampling images with already low resolution (30-50cm/pixel) will certainly harm performance so original image resolution (2048x2048) should be used in training.

Training on full images was not a option and I optimized pipeline to use

- mixed precision training
- random crops

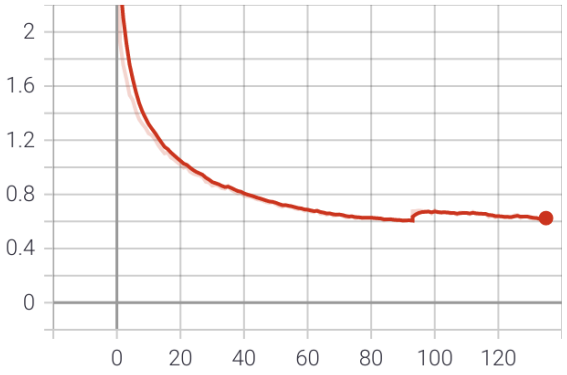
Loss functions

MSE loss would produce imbalance for different cities, depending on building heights. I decided to train a model with a R2 loss for AGL/MAG outputs which reflects the final competition metric and is also more robust to noisy training data.

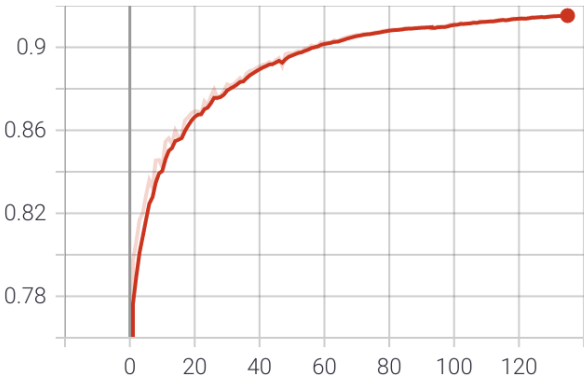
4. Do you have any useful charts, graphs, or visualizations from the process?

Train loss/validation score

loss
tag: train/loss



r2
tag: val/r2



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

I used the following loss function to optimize the final metric and also because MSE is very noise sensitive.

```
def r2_per_city(output_full, targets_full, target_mean_full, city_ohe):
    total_loss = 0
    num_cities = 0
    # calculate loss for each city independently
    for c_i in range(4):
        city_idx = city_ohe[:, c_i] > 0 #
        if torch.sum(city_idx * 1.) == 0:
            continue
        output = output_full[city_idx] # select predictions for this city
        target = targets_full[city_idx] # select targets for this city
        target_mean = target_mean_full[city_idx] # select mean target for
this city
        diff = torch.squeeze(output) - target
        not_nan = ~torch.isnan(diff)
        target = target.masked_select(not_nan)
        output = torch.squeeze(output).masked_select(not_nan)
        target_mean = target_mean.masked_select(not_nan)
        total_loss += r2_loss(output, target, target_mean) # compute r2 loss
        num_cities += 1
    # average city losses
```

```
return total_loss / num_cities
```

6. Please provide the machine specs and time you used to run your model.
    - CPU (model): Ryzen Threadripper 3970x
    - GPU (model or N/A): 4 x RTX A6000
    - Memory (GB): 256G
    - OS: Ubuntu 20.04
    - Train duration: 2 days for each model, 4 days overall
    - Inference duration: 20 mins

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.)?

To train this model at least 2 GPUS with 24gb memory are required. 10gb GPU memory is enough to run inference.

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?

I incorporated evaluation code that computes the final metric and used it to select the best checkpoints. For EfficientNets overfitting was not a problem but for ResneSt models it was.
To build a robust validation split on the training set

- predicted features for each image with pretrained CNN
- used cosine similarity between features and used agglomerative clustering
- some images repeated 20 times in the train set, though with small gsd/angle differences

This approach also showed that there is a data leak between train and test sets as scores on correct validation splits are lower than on the leaderboard. Visual inspection showed that some images overlap with the training set.
Robust validation did not help to improve LB scores as EfficientNet was not overfitting at all and I guess longer training would produce better results. So I left models with standard kfold split validation.

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

    In one experiment I added metadata (one hot encoded city and gsd) as additional inputs to the decoder, but that did not improve validation scores, though that could improve LB scores. Unfortunately I relied only on validation and did not check if it works better on the test set.

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?

- Wisely use metadata during training and inference
- Normalize images to the same gsd resolution that would certainly help to improve validation and LB scores.
- Try other decoders
- Try transformers like Swin-L
- Try techniques that produce state of the art results on monocular depth estimation tasks (like AdaBins)