Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?

My name is xxx. I am a researcher at xxx and computer vision engineer at xxx company. I am professionally working with geospatial data processing using machine learning techniques (especially convolutional neural networks). I am also github active user with popular open-source projects for image segmentation (https://github.com/qubvel) and Kaggle Master.

2. What motivated you to compete in this challenge?

As I am working closely with similar data for me it was interesting to compare my personal professional skills with other participants.

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

For this challenge I have implemented 3 stages pipeline for aerial imagery segmentation.
On first stage ensemble of Unet models trained on noisy labels (tier 1) with hard image augmentations.
On next two stages models trained on noisy train data (tier 1) and automatically labeled by stage 1 models test data (pseudo labels). This allow significantly improve generalization of model for test data.

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

Image augmentation

```
import albumentations as A

train_transform_4 = A.Compose([

   A.ShiftScaleRotate(scale_limit=0.2, rotate_limit=45, border_mode=0,
value=0, p=0.7),
   A.PadIfNeeded(768, 768, border_mode=0, value=0, p=1.),
   A.RandomCrop(768, 768, p=1.),
   A.Flip(p=0.75),
   A.Downscale(scale_min=0.5, scale_max=0.75, p=0.05),
   A.MaskDropout(max_objects=3, image_fill_value=0, mask_fill_value=0,
p=0.1),

   # color transforms
   A.OneOf(
       [
           A.RandomBrightnessContrast(brightness_limit=0.3, 
contrast_limit=0.3, p=1),
           A.RandomGamma(gamma_limit=(70, 130), p=1),
           A.ChannelShuffle(p=0.2),
           A.HueSaturationValue(hue_shift_limit=30, sat_shift_limit=40, 
val_shift_limit=30, p=1),
           A.RGBShift(r_shift_limit=30, g_shift_limit=30, 
b_shift_limit=30, p=1),
       ],
       p=0.8,
   ),

   # distortion
   A.OneOf(
       [
           A.ElasticTransform(p=1),
           A.OpticalDistortion(p=1),
           A.GridDistortion(p=1),
           A.IAAPerspective(p=1),
       ],
       p=0.2,
   ),

   # noise transforms
   A.OneOf(
       [
           A.GaussNoise(p=1),
           A.MultiplicativeNoise(p=1),
           A.IAASharpen(p=1),
           A.GaussianBlur(p=1),
       ],
       p=0.2,
   ),
   post_transform,
])
```

Image augmentation is well known technique in computer vision to enlarge your data synthetically. It helps model not to overfit, each epoch you pass the same image in network, but you modify it with different transforms (brightness, scale, rotation, etc.) Strong augmentations helped a lot for model generalization and improved leaderboard score.

Test images composition (see notebooks/mosaic.ipynb) and boundless prediction scheme.

```
class Predictor:
   def __init__(self, sample_size=1024, cut_edge=128, batch_size=1,
**save_kwargs):
       self.sample_size = sample_size
       self.cut_edge = cut_edge
       self.step = sample_size - (cut_edge * 2)
       self.save_kwargs = save_kwargs
       self.batch_size = batch_size

   def read(self, src, x, y):
       return src.read(window=Window(
           x, y, self.sample_size, self.sample_size),
           boundless=True,
       )

   def write(self, dst, data, x, y):

       if np.all(data == 0):
           return

       h, w = data.shape[1:3]
       write_x = x + self.cut_edge
       write_y = y + self.cut_edge
       write_h = min(h - (2 * self.cut_edge), dst.height - write_y)
       write_w = min(w - (2 * self.cut_edge), dst.width - write_x)
       if write_h < 0 or write_w < 0:
           return
       data = data[:, self.cut_edge: self.cut_edge + write_h,
self.cut_edge:self.cut_edge + write_w]
       dst.write(
           data,
           window=Window(
               write_x,
               write_y,
               write_w,
               write_h,
           ),
       )

   def read_batch(self, src, blocks):

       data = []
       new_blocks = []
```

```
        for x, y in blocks:
            sample = self.read(src, x, y)
            if not np.all(sample == 0):
                data.append(sample)
                new_blocks.append((x, y))

        data = np.stack(data, axis=0) if data else None

        return data, new_blocks

    def write_batch(self, dst, data, blocks):
        for (x, y), sample in zip(blocks, data):
            if not np.all(sample == 0):
                self.write(dst, sample, x, y)

    def _compute_blocks(self, h, w):
        return [
            (x, y)
            for x in range(-self.cut_edge, w + self.cut_edge, self.step)
            for y in range(-self.cut_edge, h + self.cut_edge, self.step)
        ]

    def __call__(self, src_path, dst_path):
        with rasterio.open(src_path) as src:
            profile = src.profile
            profile.update(self.save_kwargs)

            with rasterio.open(dst_path, "w", **profile) as dst:
                blocks = self._compute_blocks(h=profile["height"],
w=profile["width"])
                n_batches = (len(blocks) - 1) // self.batch_size + 1
                for i in tqdm(range(n_batches)):
                    batch_blocks = blocks[i * self.batch_size: (i + 1) *
self.batch_size]
                    batch, batch_blocks = self.read_batch(src,
batch_blocks)
                    if batch is not None:  # check that batch is not
empty
                        prediction = self.predict(batch)
                        self.write_batch(dst, prediction, batch_blocks)

    def predict(self, raster):
        raise NotImplementedError
```

I have assemble test images to big scenes instead of predicting each image separately. It is not always enough context for model to decide is pixel belong to building or not if model “see” only small corner of building on border of an image. So mosaic allow to avoid errors near borders. I have also implemented “Predicto” which allow sequentially read-predict-write patches of data. Algorithm is as follow:

1. read window with shape 1024x1024
2. pass through network and get mask with shape 1024 x 1024
3. cut 128 pixel from each edge where error are more likely appear due to lack of context
4. write result to open file
5. read next window and so on (windows read with overlap to avoid missing data after edge cutting)

This scheme also allow to process large tiff file without loading them to RAM completelyю

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

I have experimented with different optimizers, loss functions, schedulers and network architectures (e.g. try to add attention mechanism, different decoder types).

6. Did you use any tools for data preparation or exploratory data analysis that aren’t listed in your code submission?

I use QGIS software for data visualization and exploration.

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

I have made random split for train and validation sets, however there was no strong correlation between my local score and LB score (may be because test have been made from different scenes), so I rely mostly on LB score. This is not good practice because may lead to overfitting, but I was trying to minimise number of experiments, reduce hyperparameters for tuning and use robust techniques to reduce model variance (such as ensembles and test time augmentation).

8. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

Models have been trained with 0.1 m/pixel resolution (with small deviation about 15%). So it is recommended to resample data before using it for prediction.

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

No

10. 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?

- As a next step instance segmentation could be applied to avoid buildings instances stitching together.
- Roof types classification
- Improving models generalization with new external data

11. Did you learn new ways your machine learning skills could be applied to development (broadly defined by the Sustainable Development Goals) and/or disaster resilience and risk management more specifically? If so, what ways?

No, I have been working with such kind of tasks before. For example, in summer 2019 we [xxx] have been estimating consequences of flooding in Tulun, Russia using similar approach (instance segmentation of satellite imagery).