# 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 working as System Architect in having around decades of experience and involved in development of framework targeted towards Telecom Data Processing. Since last 4-5 years I am exploring different ML techniques and got interested in Deep Learning since last 2 years. In enrich my ML knowledge by participating in different competitions on various competition platforms like Kaggle and Driven Data. I have completed couple of courses on Coursera and takeb PG diploma in data science.
   I like to design, explore algorithms, and ultimately code them.

2. What motivated you to compete in this challenge?

   For past few months I have been participating in competition hosted on DrivenData. Most of the competition are trying to solve certain problems around our environment and nature. So contributing towards this noble cause is one of the motivation. Other motivations are technical in nature like satellite images, noisy labels, raw data set and semantic image segmentation as it is one of my favorite areas in deep learning.

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

   **Training Methodology:** I have used GroupKFold to create 5 folds with location as group. I have used commonly used augmentations (from Augmentation library) like
   - Random Crop to size 384x384
   - Horizontal Flip, Vertical Flip,
   - ShiftScaleRotate, GridDistortion
   - Custom Augmentation: In low cloud coverage image (coverage < 0.3), clouds are added from other low cloud coverage images. This augmentation included in the final submission and improved the score.

   Loss Used: XEDiceLoss, (0.5 * CrossEntropyLoss) + (0.5 * DiceLoss).
   LR Scheduler: OneCycleLR
   Optimizer: Adam
   Batch Size: 24

   **Inference Methodology:**
   - Simple average of predictions from all the models with threshold of 0.5 to convert prediction into binary.
   - I have used Horizontal flip and vertical flip augmentation during test time

   My approach has three Parts:

- **How to filter images with wrong labels/mask:** This is required as some images are having wrong labels/mask. This is also highlighted in competition discussion forum. I think removal of such erroneous samples is required, otherwise model would erroneously try to identify non-cloud portion of image as cloud and vice versa. The filtration is based on manual inspection of images with low validation IoU score (<0.5). I have used prediction of one 5-fold UNET (with Efficientnet-b3 backbone) to determine the validation score. I have mainly focused on images with mask representing
    - No cloud (Mask sum is 0)
    - Whole image has clouds (Mask sum is 512*512)

- **How to choose backbone for U-NET or any other Segmentation model:** Main factor for this decision is the inference time which is 4 hour. I am training the models based on 5 folds and I want to choose model with inference time (for 5 models corresponding to 5 folds) in 2:00 to 2:30 hours in range. Based on these criterions, some experiments and number of model parameters (as available in reference GITHUB repos), I have chosen 3 models
    - UNET with tf_efficientnet_b1 backbone
    - UNET with tf_efficientnetv2_b2 backbone
    - SegformerForSemanticSegmentation-B1 (From HuggingFace)

  I want to use 1 transformer based segmentation model with intuition that with its global attention it will comfortably identify images with no cloud or all cloud. Segformer can adapt to different inference time image size, other transformer based segmentation models like based on Swin transformer can't adapt to different sizes. CV and LB score of the three models follows following relation:

  SEGFORMER-B1 < UNET with tf_efficientnetv2_b2 backbone < UNET with tf_efficientnet_b1 backbone

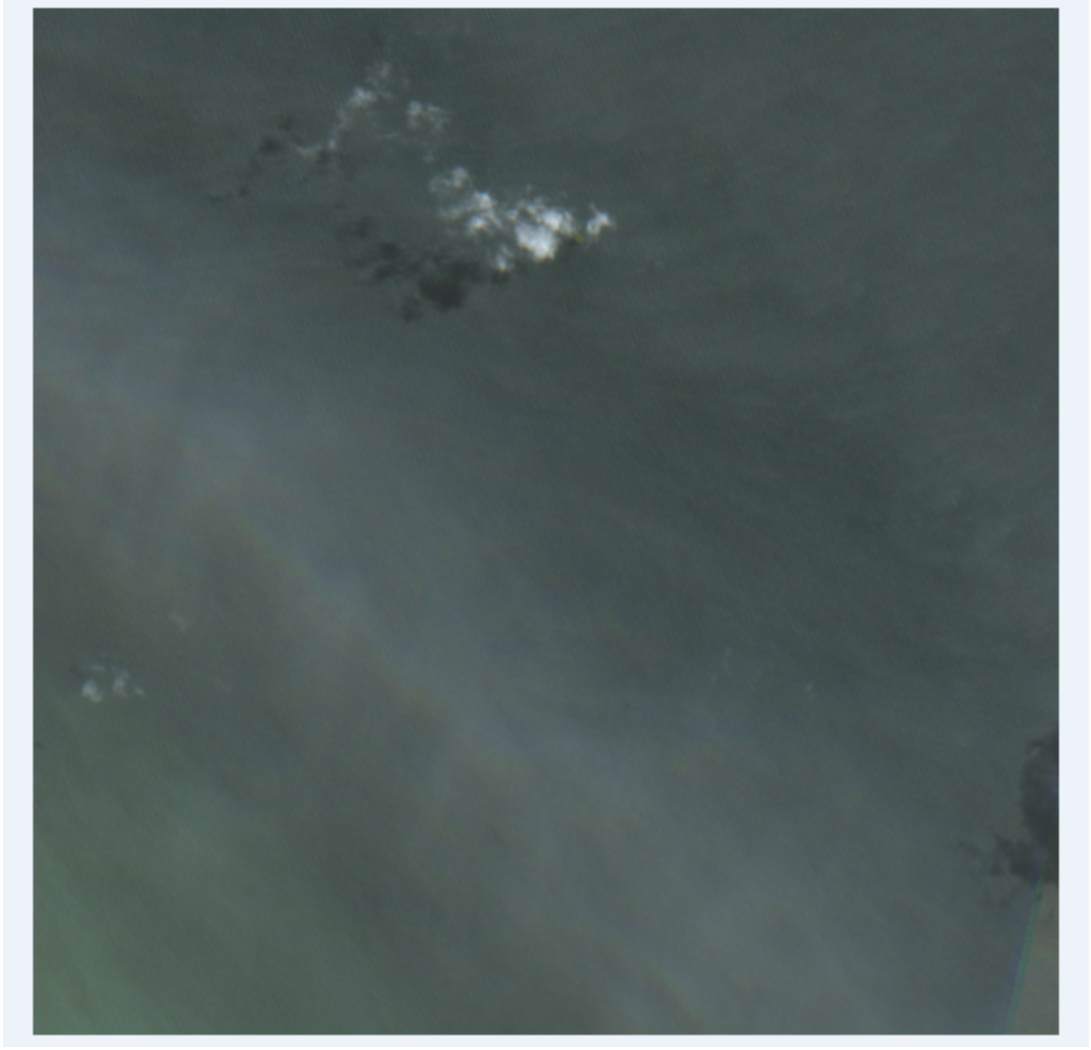
- **What all models to include for the ensemble:** This is based on public LB score (due to inference time restriction), I have tested 2 ensembles
    - UNET with tf_efficientnet_b1 backbone and UNET with tf_efficientnetv2_b2 backbone
    - UNET with tf_efficientnet_b1 backbone and SEGFORMER-B1 (only 3 folds are included due to inference time restriction)

  UNET with tf_efficientnet_b1 backbone and SEGFORMER-B1 have better public and private LB score.

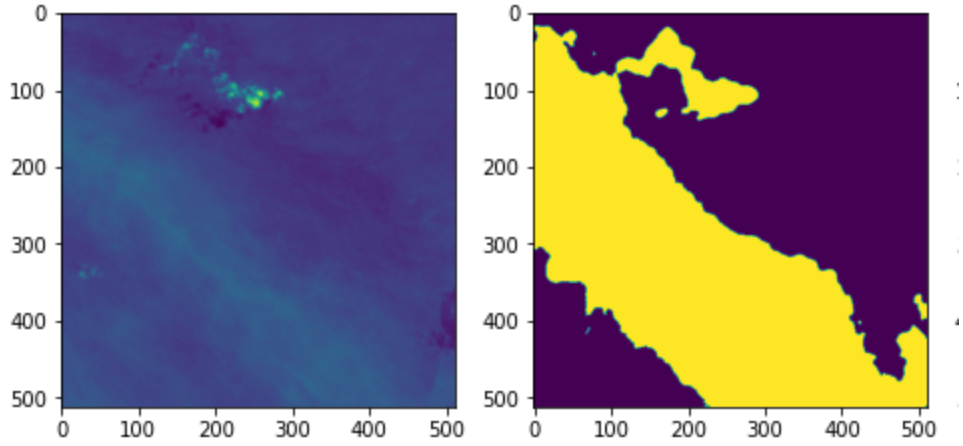
| Ensemble | Public Score | Private Score |
|---|---|---|
| UNET with tf_efficientnet_b1 backbone and UNET with tf_efficientnetv2_b2 backbone | 0.8983 | 0.8965 |
| UNET with tf_efficientnet_b1 backbone and SEGFORMER-B1 | 0.8985 | 0.8974 |

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

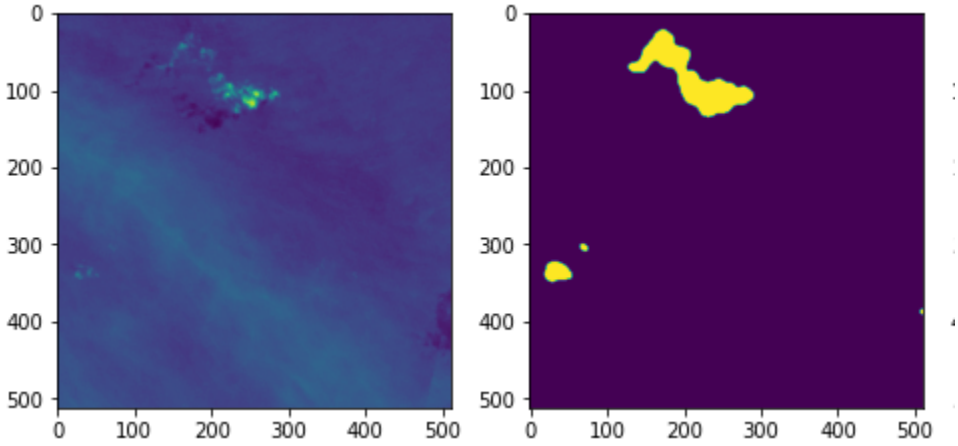
During submission generation I inspected segmentation output of chip-id **agrp**. The prediction varies a lot for this chip_id for different models. I think such chips are challenging to predict.
Following is the image from visual band



Following is the output from models used for best submission




Following is from some other submission:




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

**TTA inference:** Horizontal flip and vertical flip augmentation during test time has improved the predictions.

```
images = torch.stack([images, torchvision.transforms.functional.hflip(images),
                      torchvision.transforms.functional.vflip(images)], 0)
    n, bs, c, h, w = images.size()
    images = images.view(-1, c, h, w)
# prediction for test batch

mask = model(images)

probs1, probs2, probs3 = torch.split(mask, bs)
probs2 = torchvision.transforms.functional.hflip(probs2)
probs3 = torchvision.transforms.functional.vflip(probs3)

mask =  (1/3)*probs1 + (1/3)*probs2 + (1/3)*probs3
```

```
mask = mask.sigmoid()
```

**Filtering chips with mask in error and then folds are created:**

**Augmentation used during training:**

```
import albumentations as A
training_transformations = A.Compose(
  [
    A.ShiftScaleRotate(shift_limit=0.0625,rotate_limit=15,p=0.5),
    A.GridDistortion(p=0.35),
    A.RandomCrop(384,384,p=1),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.GaussianBlur(p=0.25),
  ]
)
```

**Custom Augmentation**: Following code is randomly select one image (with low cloud cover), select cloud (based on mask) from the image and pasting it in the original image. This way cloud from 1 image are merged into another.

```
def merge_low_chips(self,ref_chip_id,ref_img,label,p=0.5):


    if p < np.random.rand(1) and ref_chip_id in self.low_cloud_covers_chips.chip_id.values:
      rand_ix = np.random.randint(0,len(self.low_cloud_covers_chips))
      imgs = []
      chip_id = self.low_cloud_covers_chips.iloc[rand_ix].chip_id
      for b in self.bands:
        pth = f'{self.data_path}/{chip_id}/B0{b}.tif'
        with rasterio.open(pth) as img_file:
          img = img_file.read(1).astype(float)
          img = (img/2**16).astype(np.float32)
          imgs.append(img)
      x= np.stack(imgs,axis=-1)
      lpth = f'{self.label_path}/{chip_id}.tif'
      with rasterio.open(lpth) as lp:
        y = lp.read(1).astype(int)

      mask_ix = y==1
      for i in range(3):
        ref_img[:,:,i][mask_ix] = x[:,:,i][mask_ix]
      label[mask_ix] = y[mask_ix]
    return ref_img,label
```

6. Please provide the machine specs and time you used to run your model.

c2-deeplearning-pytorch-1-10-cu110-v20211219-debian-10 image on GCP

- CPU (model): Intel(R) Xeon(R) CPU @ 2.20GHz
- GPU (model or N/A): NVIDIA Tesla V100
- Memory (GB): 32 GB
- OS: Debian 10.11
- Train duration: ~ 1 hour 15 minutes per fold, ~6-7 hours Total
- Inference duration: ~4 hours (2:00 -2:30 hours per model) based on competition runtime environment

Most of the experiments is performed on setup with following configuration:

- CPU (model): AMD Ryzen7 3800XT 8-core Processor
- GPU (model or N/A): NVIDIA RTX 3060 12GB
- Memory (GB): 32 GB
- OS: Windows 10 Pro
- Train duration: ~ 12 hours per fold
- Inference duration: ~4 hours (2:00 -2:30 hours per model) based on competition runtime environment

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.)?
   While training for multiple folds encounters SHM out of memory error. It better to train model with 1 fold at a time.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   No, mostly tools are in conda env file as provided for competition

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

   Same as provided metric. I think important part is metric is calculated globally for the validation set and not per image.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    Following things are tried which are not part of final workflow:
    - Pseudo labelling by downloading images from **Microsoft's Planetary Computer** with same long/lat as given for training data but with different timestamp.
    - ConvNext Segmentation (Custom Made not included due to time constraint)
    - UNet with efficientnet-b3 backbone (not included due to time constraint)
    - DeepLabV3Plus with efficientnet-b3 backbone (not included due to less IoU)
    - Segformer-b3 (not included due to time constraint)

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?
    - Will explore pseudo labelling more.
    - Fine tune the custom augmentation, different way of mixing two images can be explored.
    - Explore data from other bands not provided as part of competition.