Model documentation and write-up

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

My name is Max Lutz. I'm a graduate from an engineering school in France (master level) specialized in energy and environment. I have been studying data science and machine learning since a bit more than 7 months (started in March 2021). My goal is to work with data to help solving climate change and work on projects involving energy, environment, and society.

2. What motivated you to compete in this challenge?

I was interested in this competition because the data and the objective were linked to the environment. As my objective is to work with data to help solve climate change, it was a very good match. I thought it would be a very good opportunity to learn how to work with satellite data and how to make prediction based on images.

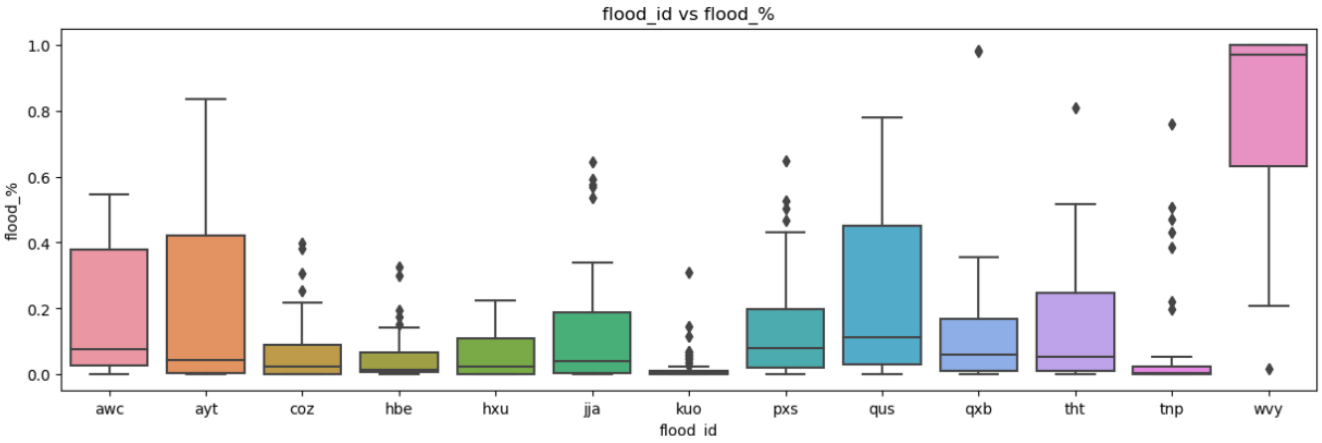
3. High level summary of your approach: what did you do and why?

My best submission includes the following steps:

- Loading the ids of the vv and vh files
- I split the ids into a training set and a test set
    - I choose between 3 different splits which allow me to build 3 models that I use at the end for prediction
- I load the vv, vh, label and additional data from the planetary computer
    - For **vv and vh** images I clip the values outside of the range -30 to 0, then I map these values to 0-255 and convert to uint8
    - For **nasadem** data I clip outside the range 0-255 and then convert to uint8
    - For the rest of the data, I don't apply any transformation.
- I use albumentation to generate new images from the original ones.
    - I use a random transformation from RandomRotate90, HorizontalFlip, and VerticalFlip
    - I append these new images to the original data
- I train a UNET model on the data using diceloss square for the loss function
    - Dice loss square is the dice loss function, but we square the denominator.
    - I tried dice loss and dice loss square and it seemed that dice loss square gave me better result so I kept it.

- I got the idea of testing dice loss square from this article: https://gchlebus.github.io/2018/02/18/semantic-segmentation-loss-functions.html

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



I did a small analysis of the training data to see if there was imbalance between the flood ids. For example the flood wvy has a lot more flood pixel than the flood kuo.

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

```
#load features
cols = ["vv_path", "vh_path", "nasadem_path", "change_path", "extent_path", "seasonality_path", "occurrence_path",
nb_cols = len(feature_path)
for row in range(nb_cols) :
  images = []
  for col in cols:
    with rasterio.open(feature_path.loc[row, col]) as img:
      #load the tif file
      if(col in ["vv_path", "vh_path"]):
          #apply transformation: clip values out of -30;0 range and map them to 0; 255 range then convert to uint8
          images.append(ma.array(np.uint8(np.clip(img.read(1), -30, 0)*(-8.4)), mask = masks[row]))
      elif col == "nasadem_path":
          #clip values > 255 and converto to uint8
          images.append(ma.array(np.uint8(np.clip(img.read(1), 0, 255)), mask = masks[row]))
      else:
          #no transformation, values are already between 0 and 255 and in uint8 format
          images.append(ma.array(img.read(1), mask = masks[row]))
```

This first code is the loading of the data including the data from the planetary computer. For vv, vh and nasadem data, the clipping has two purposes:

- Allowing me to convert the data to uint8 to save RAM space while training the model
- Removing outliers

```python
def DiceLoss_square(y_true, y_pred, smooth=1):
  #create the missing data mask
  mask = tf.math.not_equal(y_true, 255)
  #apply the mask
  y_true = tf.boolean_mask(y_true, mask)
  y_pred = tf.boolean_mask(y_pred, mask)

  y_true_f = K.flatten(y_true)
  y_pred_f = K.flatten(y_pred)
  intersection = K.sum(K.abs(y_true_f * y_pred_f))
  return 1-((2. * intersection + smooth) / (K.sum(K.square(y_true_f),-1) + K.sum(K.square(y_pred_f),-1) + smooth))
```

The definition of my loss function which is based on dice loss but with taking the square of the denominator. This loss function seemed to get me better result compared to dice loss or cross entropy. I got the idea while reading this article ([https://gchlebus.github.io/2018/02/18/semantic-segmentation-loss-functions.html)](https://gchlebus.github.io/2018/02/18/semantic-segmentation-loss-functions.html)

```python
output_predictions = []

for model in models:
    output_predictions.append(model.predict(images)[0,:, :, 0])

output_prediction =  np.mean(output_predictions, axis=0)
output_prediction = ((output_prediction > 0.5) * 1).astype(np.uint8)
```

This piece of code calculates the prediction of my three best models and take the average of their output. Combining my three best models allowed me to go from 0.89 to 0.908 on the public leaderboard.

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

Execution time of training (Google colab):

- CPU (model): single core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)
- GPU (model or N/A): N/A
- Memory (GB): 12 Go
- OS:
- Train duration: 30 min
- Inference duration: not done on google colab

Execution time of inference:

- CPU (model): AMD Ryzen 5 4500U
- GPU (model or N/A): N/A
- Memory (GB): 8 Go
- OS: WIndows 10
- Train duration: not trained on my computer
- Inference duration: 9 min 20 sec

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.)?

Sometimes while training the model, I had problems of RAM overflow and I reduce the batch size to 8. This seemed to solve the problem.

Apart from this, I don't think there is anything to be aware while using my code.

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

I only used pandas and seaborn for EDA. I used albumentation for image augmentation.

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

I mainly used the jaccard score to evaluate my model. At the beginning I used accuracy, but I quickly realized it was not going to give me a good idea of the actual score I could get on the leaderboard. I also used dice score but, in the end, preferred using the jaccard score.

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

- I tried dice loss, cross entropy for the loss function.
- I tried different train/test split but the three I'm using right now seemed to get me the best result.

- I tried different threshold to consider a flood pixel while outputting the tif file (for example 0.6 and 0.8) but 0.5 gave me the best result.

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?

I would probably continue trying to find good train/test splits. I would also probably try different weight while taking the average of my best models. For now, I take 1/3 of model_1, 1/3 of model_2 and 1/3 of model_3. I think the first model performs a bit better than the other two. So, I could try ½ of model_1, ¼ of model_2 and ¼ of model_3.

I did not really think about additional features for training the model.