III. Model documentation and write-up

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

I am a Co-founder and R&D Director at Visual and AI Solutions (a Nile University spinoff). I previously worked as a research assistant with the Ubiquitous and Visual Computing Group at Nile University, where I published several papers about satellite image analysis between 2016 and 2018. In 2019, I became a research intern at Siemens Healthineers Technology Center, NJ, USA where I worked on development of cutting edge segmentation models that can be used in multiple clinical applications.

2. What motivated you to compete in this challenge?

I wanted to contribute to analyzing tropical storms since it can affect many people's lives.

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

My approach has three stages. In the first stage, ImageNet pretrained CNNs are finetuned on the dataset with different time steps in multiple of 3s. Each 3 time step images are concatenated channel wise and pass through the CNN for feature extraction. The features of each 3 time steps are concatenated and pass through a fully connected layer for final output. Some models are trained (mostly ResNet-50 backbone) with 1, 3, 6, 9, and 12 consecutive time steps and some models with 6 time steps and spacing more than 1 (for example in case of spacing 2 images 1,3,5,7,9,11 are used instead of images 1,2,3,4,5,6). Aside from rotation and flipping, time augmentation is applied by dropping or repeating one of the input images (except for the main image). All models are trained using 224X224 image size, 5 group-folds and test time augmentation is applied on their predictions.

In the second stage, around 200 models are trained on the output predictions of first stage models and taking into consideration a history of 25-30 time steps. The models are combination of:

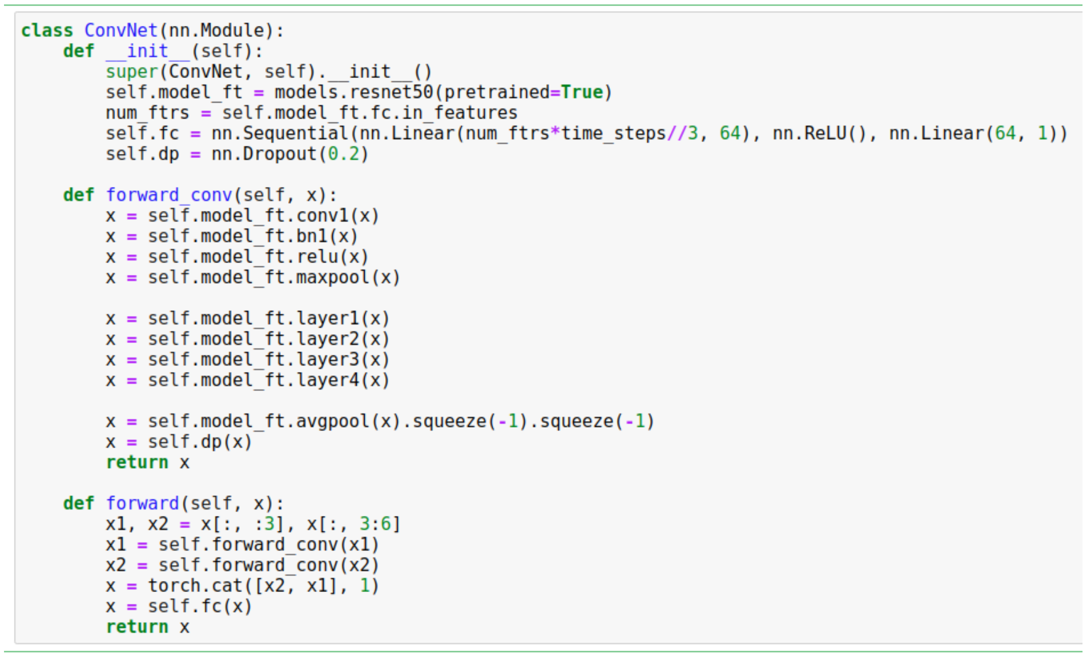
- ***Xgboost***
- ***1D CNN***
- ***LSTM***
- ***GRU***
- ***MLP***
- ***Transformer***
- ***Decision Tree***
- ***Linear Regression***

Each model in the previous list is trained on a different CNN output.

In the final stage, ensemble selection is applied to combine the best group of second stage models.

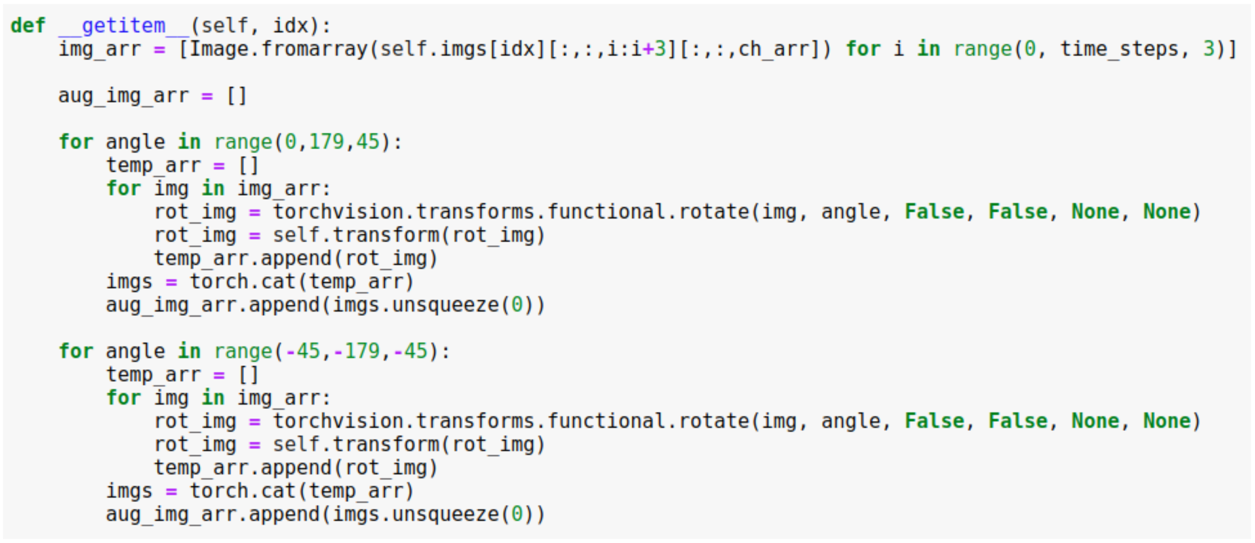
4. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

- ***Multi-step CNN:*** ***The model class of multi-step CNN where each group of 3 images is passed to ResNet-50 instead of a single image repeated 3 times. It improved the results.***



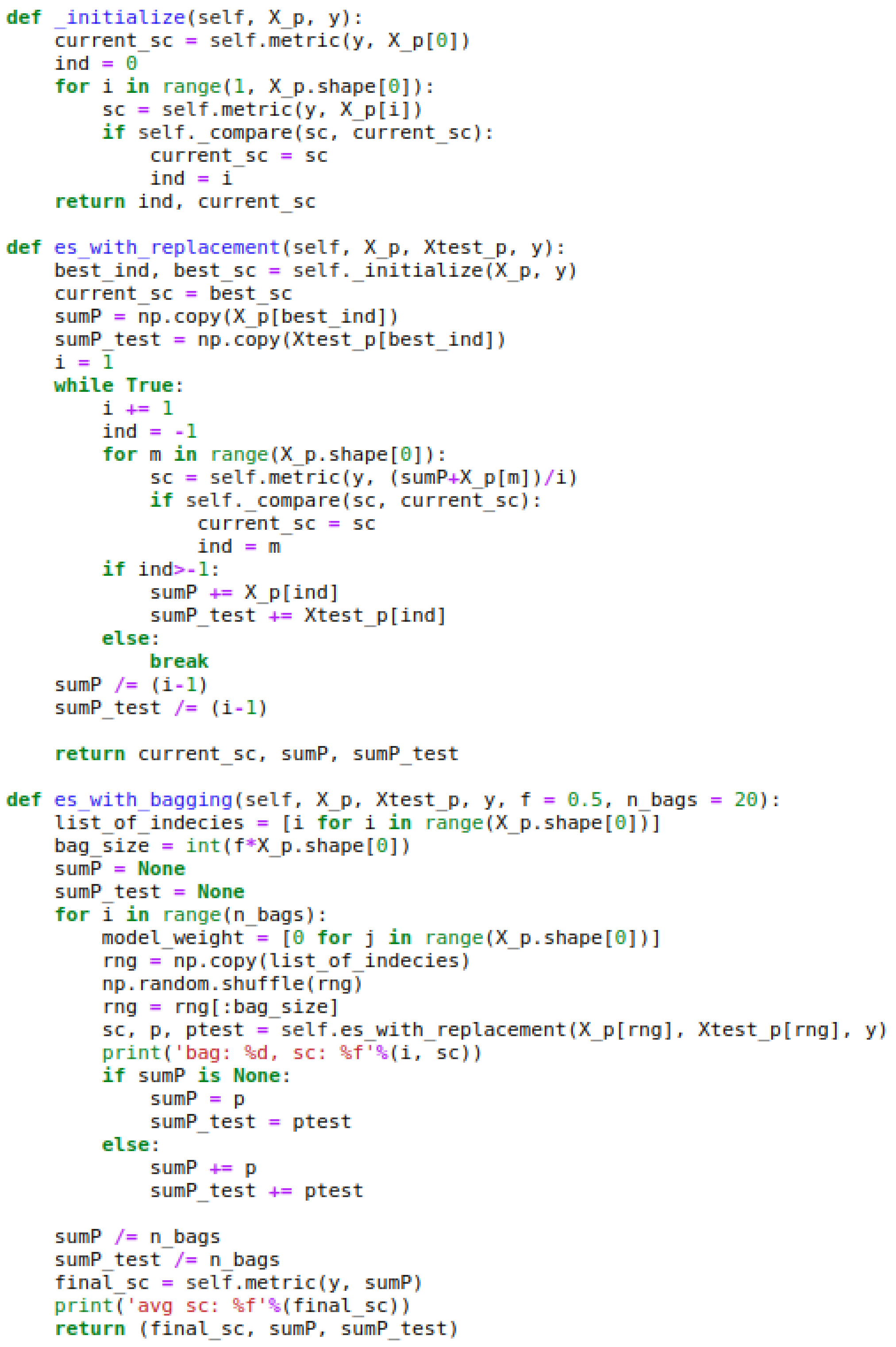
```python
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.model_ft = models.resnet50(pretrained=True)
        num_ftrs = self.model_ft.fc.in_features
        self.fc = nn.Sequential(nn.Linear(num_ftrs*time_steps//3, 64), nn.ReLU(), nn.Linear(64, 1))
        self.dp = nn.Dropout(0.2)

    def forward_conv(self, x):
        x = self.model_ft.conv1(x)
        x = self.model_ft.bn1(x)
        x = self.model_ft.relu(x)
        x = self.model_ft.maxpool(x)

        x = self.model_ft.layer1(x)
        x = self.model_ft.layer2(x)
        x = self.model_ft.layer3(x)
        x = self.model_ft.layer4(x)

        x = self.model_ft.avgpool(x).squeeze(-1).squeeze(-1)
        x = self.dp(x)
        return x

    def forward(self, x):
        x1, x2 = x[:, :3], x[:, 3:6]
        x1 = self.forward_conv(x1)
        x2 = self.forward_conv(x2)
        x = torch.cat([x2, x1], 1)
        x = self.fc(x)
        return x
```

- ***Test time augmentation: applying augmentation during inference improved prediction quality.***



```python
def __getitem__(self, idx):
    img_arr = [Image.fromarray(self.imgs[idx][:,:,i:i+3][:,:,ch_arr]) for i in range(0, time_steps, 3)]

    aug_img_arr = []

    for angle in range(0,179,45):
        temp_arr = []
        for img in img_arr:
            rot_img = torchvision.transforms.functional.rotate(img, angle, False, False, None, None)
            rot_img = self.transform(rot_img)
            temp_arr.append(rot_img)
        imgs = torch.cat(temp_arr)
        aug_img_arr.append(imgs.unsqueeze(0))

    for angle in range(-45,-179,-45):
        temp_arr = []
        for img in img_arr:
            rot_img = torchvision.transforms.functional.rotate(img, angle, False, False, None, None)
            rot_img = self.transform(rot_img)
            temp_arr.append(rot_img)
        imgs = torch.cat(temp_arr)
        aug_img_arr.append(imgs.unsqueeze(0))
```

- ***Ensemble Selection: allowed combining the best group of models in an automated un-biased way.***



```python
def _initialize(self, X_p, y):
    current_sc = self.metric(y, X_p[0])
    ind = 0
    for i in range(1, X_p.shape[0]):
        sc = self.metric(y, X_p[i])
        if self._compare(sc, current_sc):
            current_sc = sc
            ind = i
    return ind, current_sc

def es_with_replacement(self, X_p, Xtest_p, y):
    best_ind, best_sc = self._initialize(X_p, y)
    current_sc = best_sc
    sumP = np.copy(X_p[best_ind])
    sumP_test = np.copy(Xtest_p[best_ind])
    i = 1
    while True:
        i += 1
        ind = -1
        for m in range(X_p.shape[0]):
            sc = self.metric(y, (sumP+X_p[m])/i)
            if self._compare(sc, current_sc):
                current_sc = sc
                ind = m
        if ind>-1:
            sumP += X_p[ind]
            sumP_test += Xtest_p[ind]
        else:
            break
    sumP /= (i-1)
    sumP_test /= (i-1)

    return current_sc, sumP, sumP_test

def es_with_bagging(self, X_p, Xtest_p, y, f = 0.5, n_bags = 20):
    list_of_indecies = [i for i in range(X_p.shape[0])]
    bag_size = int(f*X_p.shape[0])
    sumP = None
    sumP_test = None
    for i in range(n_bags):
        model_weight = [0 for j in range(X_p.shape[0])]
        rng = np.copy(list_of_indecies)
        np.random.shuffle(rng)
        rng = rng[:bag_size]
        sc, p, ptest = self.es_with_replacement(X_p[rng], Xtest_p[rng], y)
        print('bag: %d, sc: %f'%(i, sc))
        if sumP is None:
            sumP = p
            sumP_test = ptest
        else:
            sumP += p
            sumP_test += ptest

    sumP /= n_bags
    sumP_test /= n_bags
    final_sc = self.metric(y, sumP)
    print('avg sc: %f'%(final_sc))
    return (final_sc, sumP, sumP_test)
```

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

    - CPU (model): **AMD® Ryzen 7 3700**
    - GPU (model or N/A): ***RTX 2080 Ti***
    - Memory (GB): ***32G + 64G Swap***
    - OS: ***Ubuntu 18.04***
    - Train + Inference duration: ***107 hours (first stage models) + 5 hours (second stage models) + 2 mins (third stage model)***

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

    The combination of CNN and LSTM.

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

    No

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

    I used Group K-fold based on storm id for local validation on the competition metric

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

    No

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

    No

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?

    Combining Multi-step CNN with LSTM or transformers seems promising but it needs more resources than I have