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.

My name is Azin Al Kajbaf. I am a Ph.D. candidate in the Department of Civil & Environmental Engineering at the University of Maryland, and my area of research/academic focus is “Disaster Resilience.” My research involves the application of machine learning and statistical methods in coastal and climate hazard assessment. My research goal is to leverage data science and engineering to enhance the prediction and assessment of natural hazards in support of more robust risk analysis and decision-making.

My name is Kaveh Faraji. I am a Ph.D. candidate at the University of Maryland, and I work in the area of Disaster Resilience. My main research is focused on risk assessment of natural hazards such as flood and storm surge. I am employing geospatial analysis and machine learning approaches in my research.

2. What motivated you to compete in this challenge?

We implement machine learning and deep learning methods in our research projects. We are interested to learn more about these approaches and their real-world application. Competing in these challenges allows us to gain experience with different machine learning and deep learning techniques that we can potentially employ in our research projects as well.

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

MATLAB solution: Our best private score with the MATLAB only solution was MAE: 1.948.

Step 1: We imported a publicly available pretrained 3D network (ResNet 3D 18) using MATLAB "importONNXLayers" function.

Step 2: We used MATLAB for preprocessing videos, and we cut the frames out of videos at the timestamps that the distance needed to be estimated.

Step 3: We used MATLAB "imageDatastore" and "Image Processing Toolbox" (used to augment the frames of videos) to prepare the data for Deep Learning Toolbox.

Step 4: We trained the network using MATLAB Deep Learning Toolbox. We only considered two random partitions (80% training/ 20% validation) due to time limitation.

Step 5: We post-processed the predictions of the trained networks (results of the two random partitions) from the previous step and combined them using MATLAB Regression Learner toolbox (Boosted Trees). The final submission file is the output of this step.

PyTorch+MATLAB Solution: The best combined result of using these two programs was MAE: 1.859.

This solution is based on combining the outputs of MATLAB and PyTorch using MATLAB Regression Learner toolbox (Boosted Trees). For PyTorch, we used ResNet (2+1)D pretrained network and modified it to work with four channels (RGB+ depth) instead of three. To obtain depth, we used "Monodepth2" that is mentioned on the competition website.

4. Do you have any useful charts, graphs, or visualizations from the process?
   The following gif, shows the frames that are used as input for training the networks. We used bounding box information for masking the image on the timestamps of interest.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   The code in section 10.1 is a helper function in the main code that is used for cutting timestamps out of original videos. It is responsible for preprocessing the videos effectively.

   In addition, the code in section 10.2 is presented here which is another helper function in the main code. This function is responsible for augmenting the videos which is an important part since it helps with limitation of training data and improves the predictions accuracy.

10.1 Video pre-processing

```
function output = readvideo(filename, metadata, folder)
    % Load video
    vr = VideoReader(filename);
    H =vr.Height;
    W = vr.Width;


    [~, name] = fileparts(filename);
    idx = contains(metadata.video_id,name);
```

```
videoMetadata = metadata(idx,:);

n_Frames = height(videoMetadata);
vid_len=vr.Duration;



for i = 1:n_Frames
    % Preallocating the output video arrays
    output = uint8(zeros(180,320,15,3));
    t = videoMetadata.time(i); % Extracting timestamp
    file = fullfile(folder, [name num2str(t) '.mat']); % Generating file name
    time_point=[-5,-2,-1.5,-1,-.75,-.5,-.25,0,.25,.5,.75,1,1.5,2,5];
    if ~isfile(file)
        vr.CurrentTime = t;
        for jj=-7:7
            try
                vr.CurrentTime = (t+time_point(jj+8));
                f = readFrame(vr);
            catch
                if (t+time_point(jj+8)) <= 0
                    vr.CurrentTime = 0.1;
                    f = readFrame(vr);
                elseif (t+time_point(jj+8)) >= vid_len
                    vr.CurrentTime = (vid_len-0.1);
                    f = readFrame(vr);
                else
                    disp([name, string(t)])
                end


            end


            if ~isnan(videoMetadata.x1(i))

                if jj==0
                    % Extracting corners of bounding box
                    x1 = videoMetadata.x1(i);
                    x2 = videoMetadata.x2(i);
                    y1 = videoMetadata.y1(i);
                    y2 = videoMetadata.y2(i);

                    % Masking using bounding box
                    mask = poly2mask([x1 x2 x2 x1]*W,[y1 y1 y2 y2]*H,H,W);
```

```
                        maskedFlow = bsxfun(@times, f,...
                            cast(mask, 'like', f));
                        f=maskedFlow;
                    end


                    f = imresize(f,'OutputSize',[180 320]);
                else


                    f = imresize(f,'OutputSize',[180 320]);
                end


                output(:,:,(jj+8),:) = uint8(f);
            end




            if isfolder(folder)
                save(file,'output')
            else
                mkdir(folder)
                save(file,'output')
            end
        end
    end
end
```

10.2 Loading processesd mat files

```
function data = matRead(filename,augment_img)
inp = load(filename);
f = fields(inp);
source=inp.(f{1});
if augment_img


    rng("shuffle")
    random_Seed=randi([1, 10^7],1);
    rand0=rand;


    if rand0>=0.5
```

```
        rand1=rand;
        rand2=rand;


        for ii=1:size(source,3)
            rng(random_Seed)
            img=squeeze(source(:,:,ii,:));


            tform = randomAffine2d('XTranslation',[-25 25],'YTranslation',[-25
25]);
            outputView = affineOutputView(size(img),tform);
            img = imwarp(img,tform,'OutputView',outputView);
            tform = randomAffine2d('Scale',[1.0,1.3]);
            outputView = affineOutputView(size(img),tform);
            img = imwarp(img,tform,'OutputView',outputView);
            tform = randomAffine2d('XReflection',true,'YReflection',false);
            outputView = affineOutputView(size(img),tform);
            img = imwarp(img,tform,'OutputView',outputView);


            if rand1 >= 0.75
                img = jitterColorHSV(img,'Hue',[0.0 0.15]);
                img = jitterColorHSV(img,'Saturation',[-0.4 0.1]);
                img = jitterColorHSV(img,'Brightness',[-0.3 0.1]);
                img = jitterColorHSV(img,'Contrast',[1 1.4]);
            end


            if rand2 >= 0.7
                img=repmat(rgb2gray(img),[1 1 3]);
            end


            source(:,:,ii,:)=img;


        end
    end
end
%% changing the axis
source=permute(source,[3,1,2,4]);
data = source;
end
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Intel Core i7
   - GPU (model or N/A): 2xRTX 2070 and Tesla P100
   - Memory (GB): 64 GB
   - OS: Windows and Linux
   - Train duration: Less than 24 hours for 20 epochs
   - Inference duration: Less than 30 minutes

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.)?
   No.

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?
   In addition to MAE, we used mean squared error (MSE).

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    We used optical flow and 2D CNN, but the results were not promising.

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?
    As explained in PyTorch+MATLAB solution in question 3, we modified the pretrained network to work with four channels (RGB+ depth) instead of three. If we were to continue this project, we would have added more channels that reflect depth. These depth channel inputs can potentially be created with Monodepth2 that are trained using different datasets. Also, we could use other models for computing depth, such as MiDaS.