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?

I'm doing research at the University of Montréal in Computer Graphics. I got my Master in Mathematics at the Novosibirsk State University, Russia. Lately, I fell in love with machine learning, so I was enrolled in Yandex School of Data Analysis and Computer Science Center. This led me to develop and teach the first open Deep Learning course in russian. Machine Learning is my passion and I often take part in competitions.

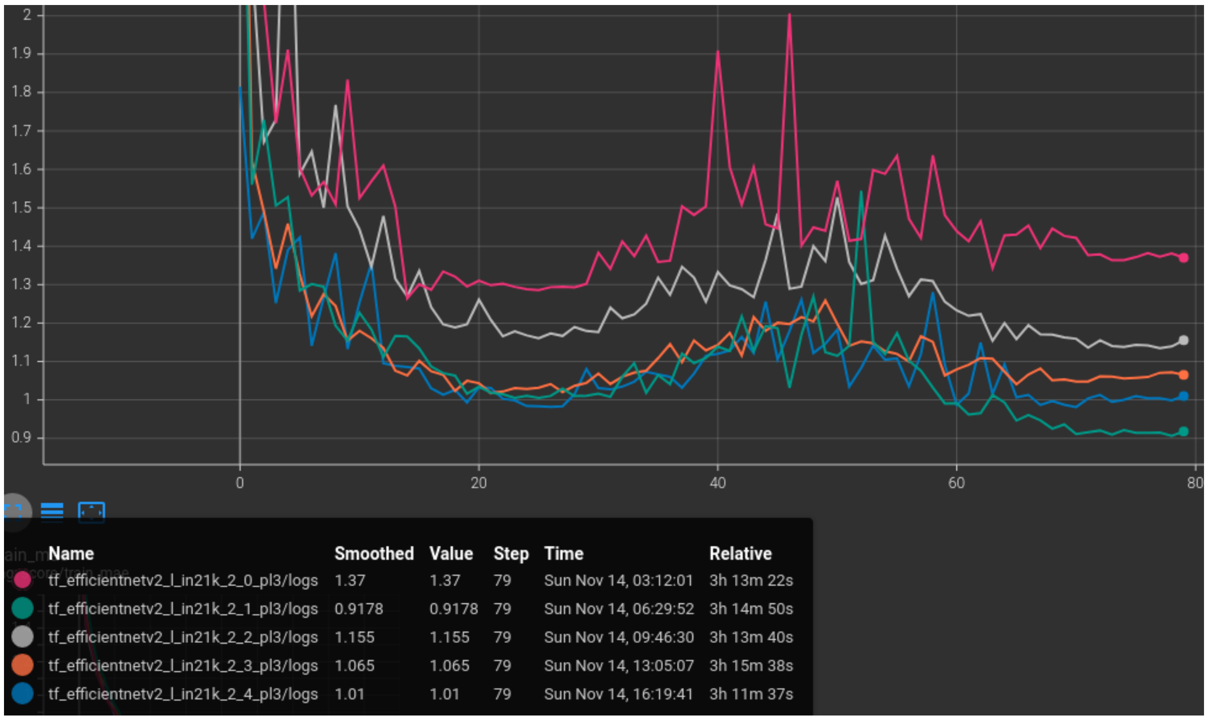
2. What motivated you to compete in this challenge?

I like competitions and to solve new problems.

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

   I've trained new `efficientnetv2` CNN models with MAE loss. To capture the motion I've stacked a sequence of 5 frames channel-wise in one array. The frames are downsampled to `270x480` size. The model's output is a distance. I've trained the model with heavy augmentations for 80 epochs with the `AdamW` optimizer and `CosineAnnealingLR` scheduler. To reduce the predictions' variance I've ensembled models from different folds using the simple average. Then I've added these predictions as pseudo labels to the train set and finally retrained models.

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

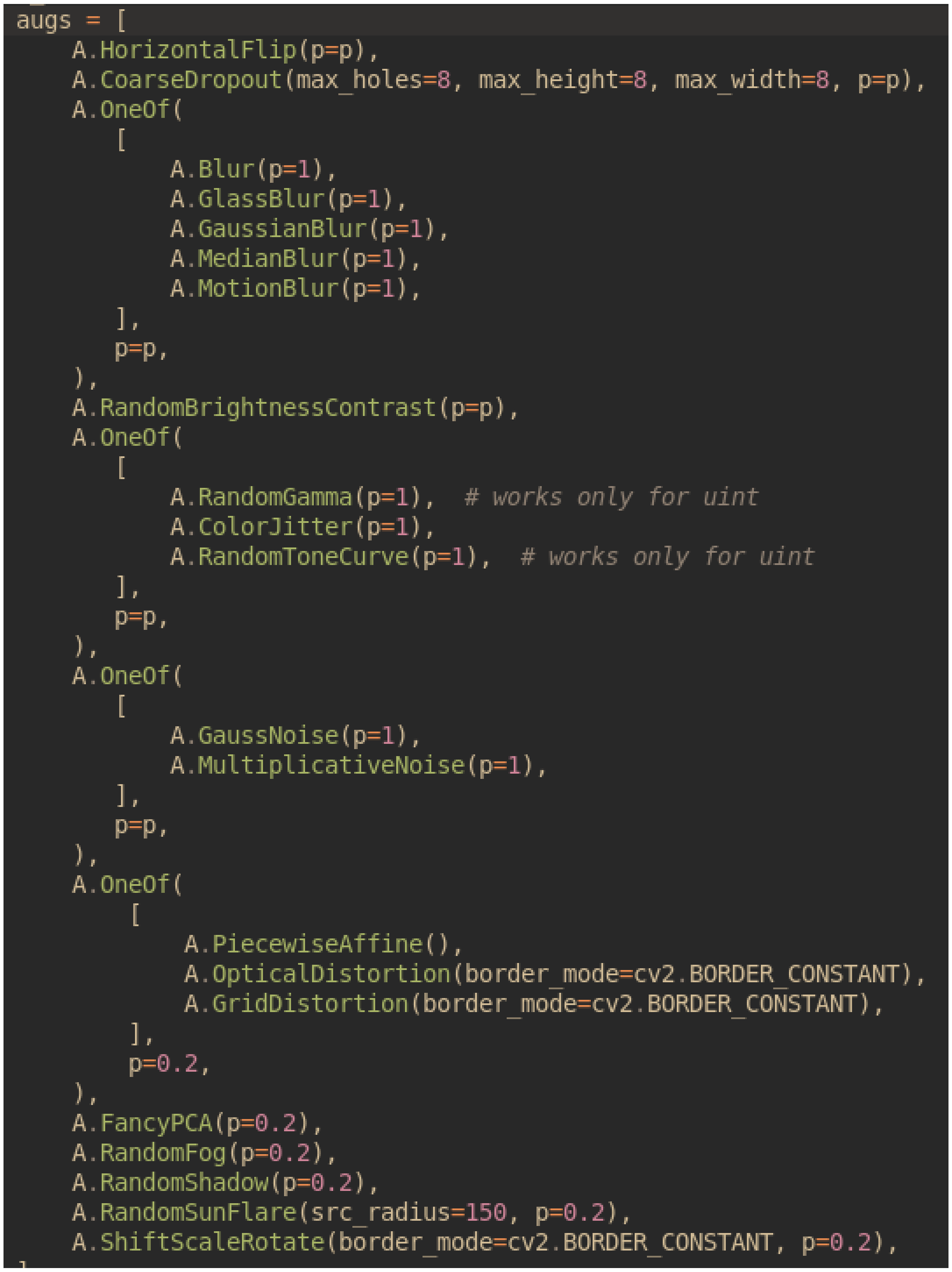


	Name	Smoothed	Value	Step	Time	Relative
●	tf_efficientnetv2_l_in21k_2_0_pl3/logs	1.37	1.37	79	Sun Nov 14, 03:12:01	3h 13m 22s
●	tf_efficientnetv2_l_in21k_2_1_pl3/logs	0.9178	0.9178	79	Sun Nov 14, 06:29:52	3h 14m 50s
●	tf_efficientnetv2_l_in21k_2_2_pl3/logs	1.155	1.155	79	Sun Nov 14, 09:46:30	3h 13m 40s
●	tf_efficientnetv2_l_in21k_2_3_pl3/logs	1.065	1.065	79	Sun Nov 14, 13:05:07	3h 15m 38s
●	tf_efficientnetv2_l_in21k_2_4_pl3/logs	1.01	1.01	79	Sun Nov 14, 16:19:41	3h 11m 37s

MAE on validation folds

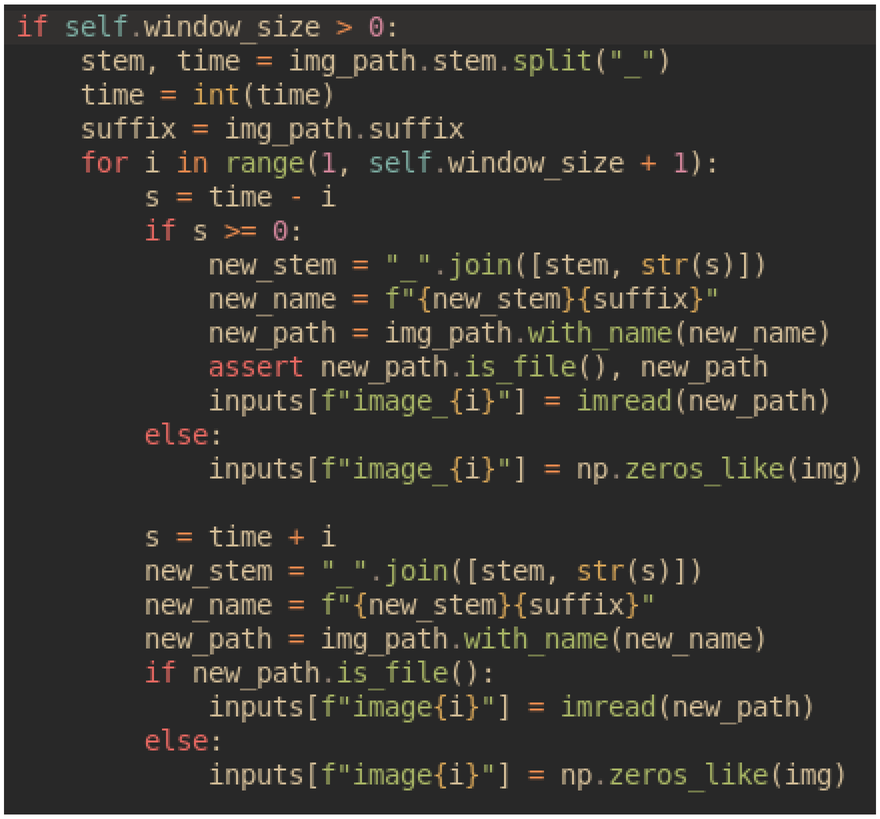
5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

    Heavy augmentations



```
augs = [
    A.HorizontalFlip(p=p),
    A.CoarseDropout(max_holes=8, max_height=8, max_width=8, p=p),
    A.OneOf(
        [
            A.Blur(p=1),
            A.GlassBlur(p=1),
            A.GaussianBlur(p=1),
            A.MedianBlur(p=1),
            A.MotionBlur(p=1),
        ],
        p=p,
    ),
    A.RandomBrightnessContrast(p=p),
    A.OneOf(
        [
            A.RandomGamma(p=1),  # works only for uint
            A.ColorJitter(p=1),
            A.RandomToneCurve(p=1),  # works only for uint
        ],
        p=p,
    ),
    A.OneOf(
        [
            A.GaussNoise(p=1),
            A.MultiplicativeNoise(p=1),
        ],
        p=p,
    ),
    A.OneOf(
         [
             A.PiecewiseAffine(),
             A.OpticalDistortion(border_mode=cv2.BORDER_CONSTANT),
             A.GridDistortion(border_mode=cv2.BORDER_CONSTANT),
         ],
         p=0.2,
    ),
    A.FancyPCA(p=0.2),
    A.RandomFog(p=0.2),
    A.RandomShadow(p=0.2),
    A.RandomSunFlare(src_radius=150, p=0.2),
    A.ShiftScaleRotate(border_mode=cv2.BORDER_CONSTANT, p=0.2),
```

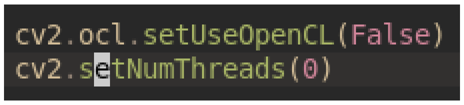
Stacking channel-wise the sequence of frames into one array



```
if self.window_size > 0:
    stem, time = img_path.stem.split("_")
    time = int(time)
    suffix = img_path.suffix
    for i in range(1, self.window_size + 1):
        s = time - i
        if s >= 0:
            new_stem = "_".join([stem, str(s)])
            new_name = f"{new_stem}{suffix}"
            new_path = img_path.with_name(new_name)
            assert new_path.is_file(), new_path
            inputs[f"image_{i}"] = imread(new_path)
        else:
            inputs[f"image_{i}"] = np.zeros_like(img)

        s = time + i
        new_stem = "_".join([stem, str(s)])
        new_name = f"{new_stem}{suffix}"
        new_path = img_path.with_name(new_name)
        if new_path.is_file():
            inputs[f"image{i}"] = imread(new_path)
        else:
            inputs[f"image{i}"] = np.zeros_like(img)
```

Disabling opencv threading reduces training time



```
cv2.ocl.setUseOpenCL(False)
cv2.setNumThreads(0)
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
   - GPU (model or N/A): Nvidia Tesla V100 32GB
   - Memory (GB): 755Gb
   - OS: CentOS 7
   - Train duration: 5 days on 5 GPU V100
   - Inference duration: 40 min

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.)?
   At least 24 GB VRAM is required for inference (I've tested on 1 Nvidia V100 32GB).

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?
   Only MAE metric on validation folds

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    Data augmentation using linear interpolation of distances between two frames harms the performance.

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?
    Use depthmaps from pretrained model as additional features.