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.

2. Currently, I'm doing research at the University of Montréal in Computer Graphics, using Machine Learning for problem-solving: Learning-based Posing of 3D Characters via Bitmap Sketches. I got my Master's in Mathematics at Novosibirsk State University, Russia. Lately, I fell in love with machine learning, so I was enrolled in the 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.
3. What motivated you to compete in this challenge?
   I become addicted to machine learning contests. The interesting part of this challenge are extremely high resolution images and ordinal labels. I also hope it advances medical research.
4. High level summary of your approach: what did you do and why?
   The main challenge here is to work with extremely high-resolution images (e.g. 150,000 x 85,000 pixels). One can reduce the resolution by downsampling them to reasonable sizes, such that a deep neural network (DNN) could work with them but it hurts the performance. On the one hand, we need quite large images with sufficient information. On the other hand, we should be able to pass them into DNN as an input.
   One can notice that most of the image has a monotone background (usually white). The simple method to reduce the size is to select the tiles/crops based on the number of tissue pixels. First, divide the image into N x N grid. Calculate the sum or mean of pixels' intensity and then stack top K tiles/crops into one image.
5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   1) Make grid, compute the sum, sort by pixels' intensity, and take top k tiles/crops

```python
def get_tiles(img, tile_size=256, n_tiles=36, mode=0):
    h, w, c = img.shape
    pad_h = (tile_size - h % tile_size) % tile_size + ((tile_size * mode) // 2)
    pad_w = (tile_size - w % tile_size) % tile_size + ((tile_size * mode) // 2)

    img = np.pad(
        img,
        [[pad_h // 2, pad_h - pad_h // 2], [pad_w // 2, pad_w - pad_w // 2], [0, 0]],
        constant_values=255,
    )
    img = img.reshape(
        img.shape[0] // tile_size, tile_size, img.shape[1] // tile_size, tile_size, 3
    )
    img = img.transpose(0, 2, 1, 3, 4).reshape(-1, tile_size, tile_size, 3)

    n_tiles_with_info = (
        img.reshape(img.shape[0], -1).sum(1) < tile_size ** 2 * 3 * 255
    ).sum()
    if len(img) < n_tiles:
        img = np.pad(
            img, [[0, n_tiles - len(img)], [0, 0], [0, 0], [0, 0]], constant_values=255
        )

    idxs = np.argsort(img.reshape(img.shape[0], -1).sum(-1))[:n_tiles]
    img = img[idxs]

    return img, n_tiles_with_info >= n_tiles
```

2) Stack tiles/crops into one image

```python
def concat_tiles(tiles, n_tiles, image_size, rand=False, transform=None):
    if rand:
        idxes = np.random.choice(list(range(n_tiles)), n_tiles, replace=False)
    else:
        idxes = list(range(n_tiles))

    n_row_tiles = int(np.sqrt(n_tiles))
    img = np.zeros(
        (image_size * n_row_tiles, image_size * n_row_tiles, 3), dtype="uint8"
    )
    for h in range(n_row_tiles):
        for w in range(n_row_tiles):
            i = h * n_row_tiles + w

            if len(tiles) > idxes[i]:
                this_img = tiles[idxes[i]]
            else:
                this_img = np.ones((image_size, image_size, 3), dtype="uint8") * 255

            if transform is not None:
                this_img = transform(this_img)

            h1 = h * image_size
            w1 = w * image_size
            img[h1 : h1 + image_size, w1 : w1 + image_size] = this_img

    if transform is not None:
        img = transform(img)

    return img
```

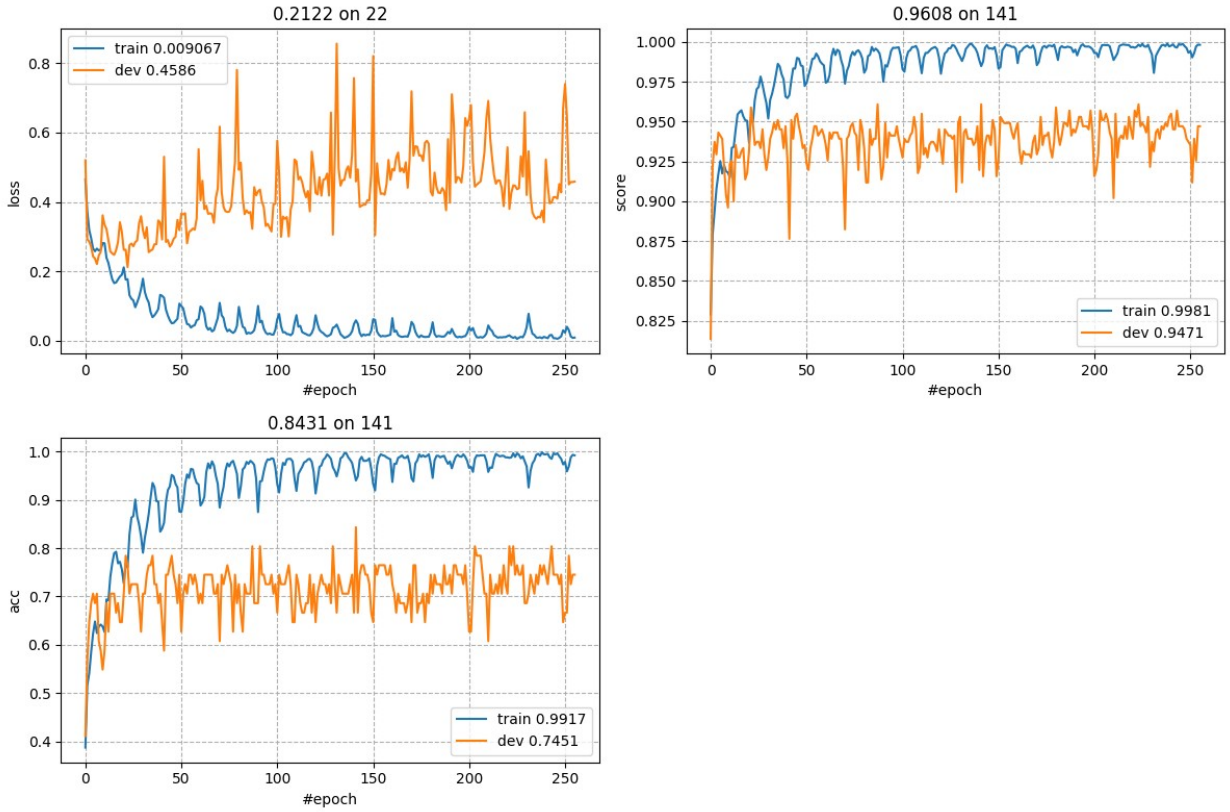
3) Convert one hot label to ordinal one

```python
label = np.zeros(self.config.network.num_classes, dtype="float32")
label[: item[self.config.data.labels].values.argmax()] = 1.0
```

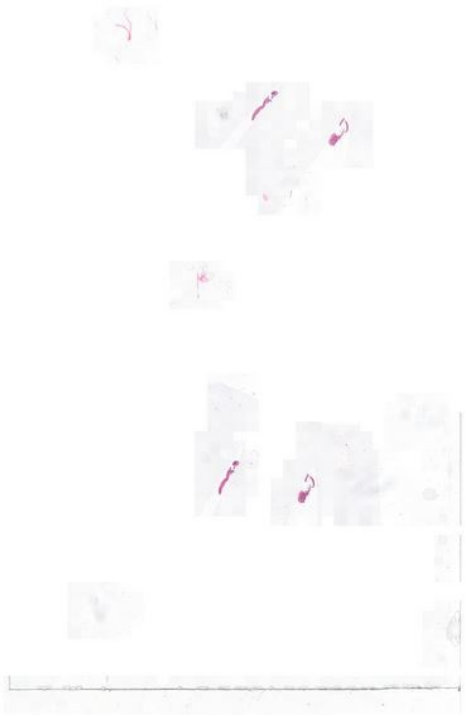
6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Nvidia Tesla V100 32GB
   - GPU (model or N/A): Nvidia Tesla V100 32GB
   - Memory (GB): 755Gb
   - OS: CentOS 7
   - Train duration: 2 weeks
   - Inference duration: 10 minutes

7. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
   I trained heavier models like resnets50, se-resexts-50, efficientnet-b3 on different downsampled resolutions (page=2, page=3) with a different number of tiles and its sizes. But all these things train slower in terms of computational resources, but cross-validation was higher.
8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   No

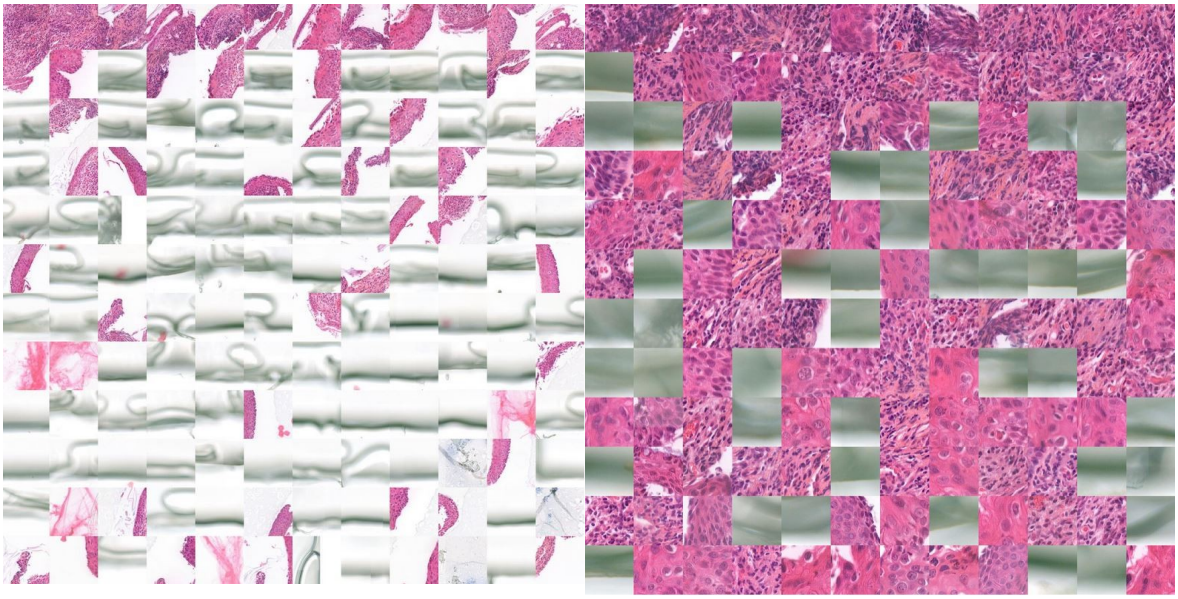
9. How did you evaluate performance of the model other than the provided metric, if at all?
   “Hard” accuracy, i. e. simple accuracy metric
10. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?
    20Gb VRAM
11. Do you have any useful charts, graphs, or visualizations from the process?



12. 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?
    Train with adaptive downsampling resolution during training. Some images are extremely large even with the page=4 (18384 x 17008 pixels) and the tissues are very small, like in the figure below with (a lot of white pixels on the left side and small tissues on the right)



If we use page=4 (image to the left), we get a lot of noisy crops. Using less downsampling, e.g. page=1 (image to the right) we can capture more useful information about tissues. Based on the original image size, we need to choose a page number and don't use a fixed one.



Another approach may be a binary segmentation of tissues on low-resolution images. But this implies to collect another training dataset.