Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?
   I'm a Business Intelligence/ Data Warehouse developer.

2. What motivated you to compete in this challenge?
   I enjoy working on data science contests, especially where there is a visible positive social impact.

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

   I applied a Multiple Instance Learning approach, where each slide was subdivided into smaller parts that were individually processed. I used the method discussed at https://developer.ibm.com/articles/an-automatic-method-to-identify-tissues-from-big-whole-slide-images-pt1/ to extract useful tissue parts (tiles) from the whole slide images. Using *page* 5 to mask the informative tissue pixels, I generated six datasets with different tile sizes from the 2nd, 3rd and 4th *pages* of the pyramidal TIF files. This was both to increase the dataset and improve generalization by training a different model for each dataset. Tiles were ranked by their usefulness in classifying the slide, based on the scoring formula discussed in the above article.

   During training, the best *N* tiles for each slide were selected and passed through a Convolutional Neural Network to extract feature vectors. The output features were then aggregated for each slide and pooling and classification was done on the slide level.

   Some slides had hundreds of tiles with potentially useful tissue, but I could only fit so many tiles in memory with a reasonably batch size. To allow the model to *see* as much useful data as possible, I randomly sampled the *N* tiles from the top *N_MAX* tiles each epoch (see *config.csv* file). This also doubled as regularization, further enhancing the model's generalization.

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

   4.1 Tile scoring code from https://github.com/CODAIT/deep-histopath. Used to rank slide regions (tiles) by their potential usefulness in classifying the slide.

```
color_factor = hsv_purple_pink_factor(np_tile)
s_and_v_factor = hsv_saturation_and_value_factor(np_tile)
amount = tissue_quantity(tissue_percent)
quantity_factor = tissue_quantity_factor(amount)
combined_factor = color_factor * s_and_v_factor * quantity_factor
score = (tissue_percent ** 2) * np.log(1 + combined_factor) /1000.0
score = 1.0 - (10.0 / (10.0 + score))
```

4.2 Sample *N* tiles from the best *N_MAX* tiles, explained above. Slides with < *N* tiles are padded with a random tile

```
for d in data:
        img = d[0]
        n_available = len(img)
        ixs = []
        choice_range = min(n_available,max_nth_tile)
        if n_available<n_tiles:
          ixs =
list(np.random.choice(range(choice_range),size=n_tiles,replace=True))
        else:
           ixs =
list(np.random.choice(range(choice_range),size=n_tiles,replace=False))
        ixs = sorted(list(ixs))
```

4.3 BCE loss with binned inputs/ targets converged to a better solution than CrossEntropy loss.

```
class BinBCELoss(nn.Module):
    def __init__(self):
        super().__init__()
        self.criterion = nn.BCEWithLogitsLoss()
    def forward(self, inputs, targets):
        targ_bin  = torch.zeros_like(inputs)
        for i in range(targets.shape[0]):
          targ_bin[i,:targets[i]] = 1
        targ_bin = targ_bin.to(device)
        loss = self.criterion(inputs, targ_bin)
        return loss
```

5. Please provide the machine specs and time you used to run your model.
   - CPU (model): Intel Xeon
   - GPU (model or N/A): Nvidia Tesla P100
   - Memory (GB): 16 GB
   - OS: Ubuntu 18.04
   - Train duration: 50 hours
   - Inference duration: 12.5 hours

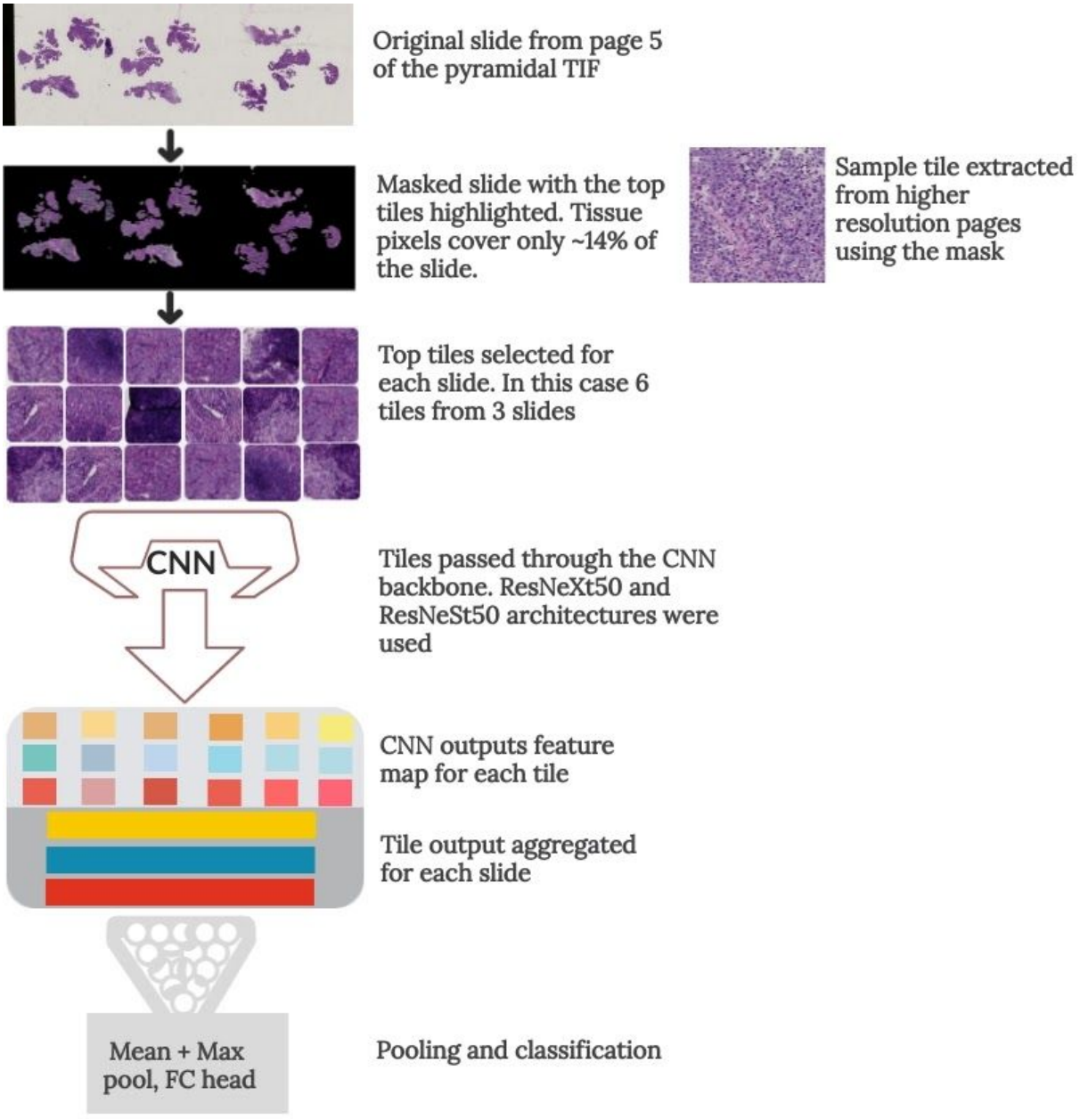
6. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
   Progressive resizing - pre-training on tiles extracted from the lower resolution pages and fine tuning on higher resolution tiles.
   Regression - I thought regression might capture the relationship between the labels i.e. *3>2>1>0*.
   Advanced augmentation - color and pixel level augmentation didn't seem to improve the score so I used basic affine transforms.

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?
   Only the provided metric was used for evaluation.

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.)?
   Tile sampling is based on *numpy.random.choice* so running inference with different *seeds* may produce different results.

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

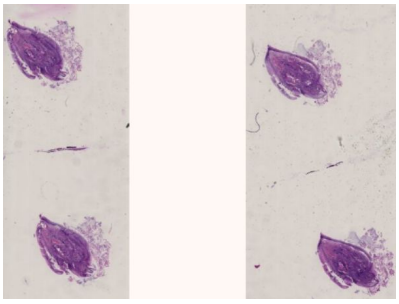


Summary of the process from tile extraction to slide classification

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?

    Attention - Training on the annotated regions quickly converged to a good solution with just a few epochs.This suggests that the annotated images had more useful information compared to the tiles selected with the approach discussed. Adding attention to *tell* the model which tiles to *look* at might improve the score.

    Duplicates preprocessing - some slides contained duplicate or near-identical regions. These were perhaps serial slides of the same tissue combined into one slide, or probably just an artefact of how the data was recorded? Tiles selected from these regions were likely near-identical, therefore not adding any new information to help classify the slide. Identifying and discarding such duplicate regions would allow selection of other potentially useful parts of the slide.



Example slide with duplicated regions