Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?
   I'm working as Head of Computer Vision at a multi-format retailer in Russia. Also Kaggle Grandmaster (rank 7). I am leading a team of 15 CV engineers and developers. Tasks that the team cover both, R&D in the ML and CV domains as well as envelop the whole solution from shaping hardware architecture to integration with the data warehouse.
   https://hackernoon.com/interview-with-kaggle-grandmaster-lead-data-scientist-at-dbrain-artur-kunzin-28f516a91e3

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

   I used a dataset from Pavel Pleskov, which was reduced to 512 on the wide side. He used PIL and ANTIALIAS interpolation.

What worked in and how my solution looks like:

1. swsl_resnext50, wsl_resnext101d8. The first convolution and the first BN are frozen during all stages of training.
2. pytorch-lightning, apex O1, distributed
3. WarmUp, CosineDecay, initLR 0.005, SGD, WD 0.0001, 8 GPUs, Batch 256 per GPU
4. loss / metric torch.nn.MultiLabelSoftMarginLoss
5. Progressive increase in size during training. Wide side resize: 256 -> 320 -> 480 for resnext50, 296 -> 360 for resnext101.
6. During training, resize to ResizeCrop size on the wide side -> RandomCrop with ResizeCrop / 1.14 size. Moreover, the crop is not square, but rectangular with the proportion of the original image. During inference, resize to ResizeCrop and that's it.
7. From augmentations: flip, contrast, brightness. With default parameters from albumentations
8. TTA: flip
9. Averaging within one series - gmean
10. TTA prediction and model averaging - gmean

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

   Image preparation during train and inference:

```
img = Image.open((osp.join(self.path, row["file_name"])))
# sample_id = row["file_name"]
# resize to 512 longest size:
```

```
w, h = img.size
ratio = max(h / self.long_side, w / self.long_side)
# want them to be divizable by 16
new_w = int((w / ratio) // 16 * 16)
new_h = int((h / ratio) // 16 * 16)
img = img.resize((new_w, new_h), resample=Image.LANCZOS)
```

augmentation:

```
def train_aug(height=224, width=224):
    return alb.Compose(
        [
            alb.RandomResizedCrop(height=height, width=width, always_apply=True,
interpolation=cv2.INTER_LANCZOS4),
            alb.CLAHE(p=0.1),
            alb.ToGray(p=0.2),
            alb.RandomBrightnessContrast(p=0.6),
```

average predictions during test:

```
outputs = []
# for model in models:
for i in range(2):
    imgs = batch[f"images{i+1}"][0]  # .type(torch.FloatTensor).cuda()
    mirror = torch.flip(imgs, (3,))
    imgs_mirror = torch.cat([imgs, mirror], dim=0).type(torch.FloatTensor).cuda()
    output = torch.sigmoid(models[i](imgs_mirror))
    arr = output.cpu().numpy()
    model_predict = gmean(arr, axis=0)
    outputs.append(model_predict)
mean_arr = np.array(outputs)
preds = gmean(mean_arr, axis=0)
```

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

   What did not work in this competency:

   - All other ways to read images (opencv, jpeg4py, dali), except for those used by Pavel. I got my first result from using dali data loader and was very happy. But then I was struggling to achieve the same local score on LB.
   - Sampling.
   I tried to take rare classes more often, emptiness less often. The score worsened. I also tried to speed up the whole train and throw 70% of the easiest samples, where the loss is already almost 0, also did not work.

- Imagenet-style RandomCrop.
Which default in torchvision and dali. I had hard time to understand how to choose the parameters in order to set the same scale as it would be for the test. As a result, I switched to Albumentations and everything turned out.
- Small resolution.
For a very long time (like a couple of days) I experimented with the size of the input image in the region of 224 and could not break through the loss of 0.0040. Then I increased it and everything worked out.

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

   Nope. I used a dataset from Pavel Pleskov as is.

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

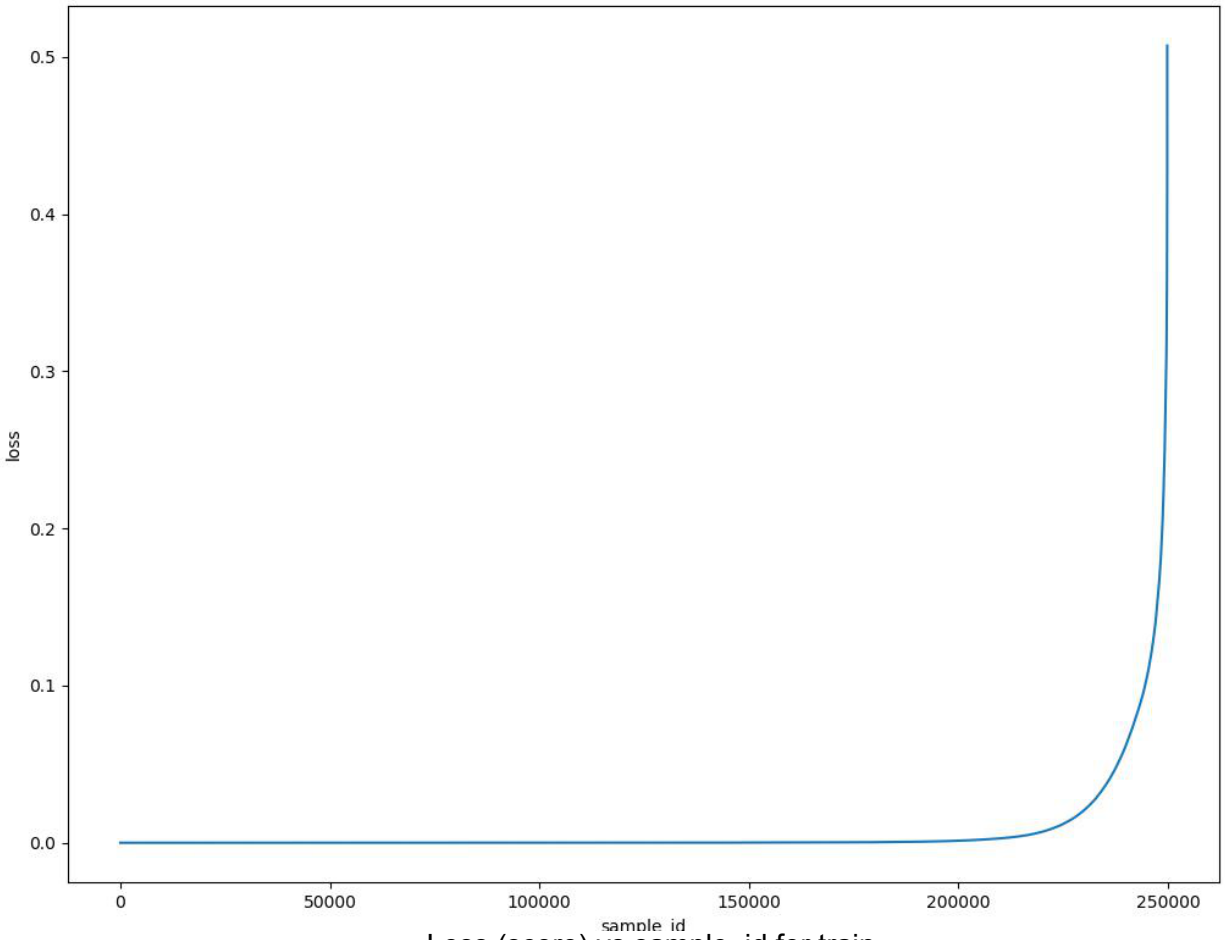
   torch.nn.MultiLabelSoftMarginLoss for both score and loss

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.)?

   I used apex with O1 optimization. But It works fine on regular fp32 inference. During training I used 8x Tesla V100 32Gb, so I used large batch size. Didn't check with smaller batch size.

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

Samples with the worst loss (score)



Loss (score) vs sample_id for train

9. 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?

   - Bigger resolution
   I believe largest efficientnet on large resoluiton might be interesting for ensemble. Didn't try it due to 512 pixel wide side on Pavel's dataset
   - External data for rare classes
   If it's legal for next competition, it might be helpful for real usage of this solution