Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?

Currently I'm doing research at the ___ in Computer Graphics, using Machine Learning for problem solving: posing 3D characters via gesture drawings. I got my Master in Mathematics at the ___. Lately, I fall in love with machine learning, so I was enrolled in ___ 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 the competitions

2. What motivated you to compete in this challenge?

I like competitions and to solve new problems, at least new to me. I've never worked with 3D images so in this challenge I learned new things, how to store, read and process such data efficiently.

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

The best models for 2D image recognition are 2D convolutional neural networks (CNNs). Most likely this will be true for 3D. So the 3D CNNs were chosen, namely ResNets. Thanks to provided region of interest on video one can crop it to reduce significantly disk and, hence, ram usage. To deal with different spatial resolutions of frames they were resized to 160x160. Depth dimension was zero-padded till the longest depth of all samples in the batch. As dataset is highly imbalanced balanced sampler with ¼ ratio of positive (stalled) classes was applied. This also makes training faster. Heavy ResNet101 model was trained with binary cross entropy loss with standard spatial augmentations like horizontal and vertical flips, distortions and noise. Finally, deep learning is data-hungry so the model was trained on full tier 1 dataset and all stalled samples with crowd score> 0.6 from tier 2 dataset. The predictions of five different snapshots on various epochs of the same model were averaged.

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

1) lmdb library to efficiently store and read data which reduced I/O bound during training.
to store:
```
import lmdb
env = lmdb.open(
    str(root),
    map_size=map_size,
 )

 with env.begin(write=True) as txn:
     txn.put(key.encode(), pickle.dumps(features[key]))
```

```
to read:
```
```
env = lmdb.open(
    str(root),
    readonly=True,
    lock=False,
    readahead=False,
    meminit=False,
)
with env.begin(write=False, buffers=True) as txn:
    feature = pickle.loads(txn.get(fname.encode()))
```

2) Resize all frames to the same spatial resolution and zero-pad in depth dimensional to be able to put all samples into one batch.
to resize
```
import albumentations as A
size=160
rsz = A.Resize(size, size)
```
to zero-pad
```
def collate_fn_3d(x):
    x, y = list(zip(*x))
    max_len = max(map(len, x))
    x = torch.stack([
        torch.nn.functional.pad(_x, (0, 0, 0, 0, 0, max_len - len(_x)))
        for _x in x
    ])  # [b, seq, h, w]

    return x.unsqueeze(1), torch.tensor(y).float().unsqueeze(1)
```

3) Balanced sampler with ¼ ratio of positive (stalled) samples which makes training faster
```
if index % 4 == 0:  # pos
    item = self.df_pos.iloc[index//4]
else:  # neg
    item = self.df_neg.sample(n=1).iloc[0]
```

5. 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: 2 weeks
- Inference duration: 40 minutes

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

I tried to improve second part of baseline model 2D CNN + LSTM. LSTM was replaced by 1D CNN and Transformer, but the score was the same as with LSTM (0.59 score). New neural network ResNeSt works not better than Efficientnets. Surprisingly, but test time augmentations worsen the score. One thing I find is interesting that width and height has positive signal, i. e. if we train some model (linear or gradient boosting) with these two features we can get 0.16 Matthew correlation coefficient on local validation. But I didn't try it on leader board. Furthermore it is useless in production.

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

-

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

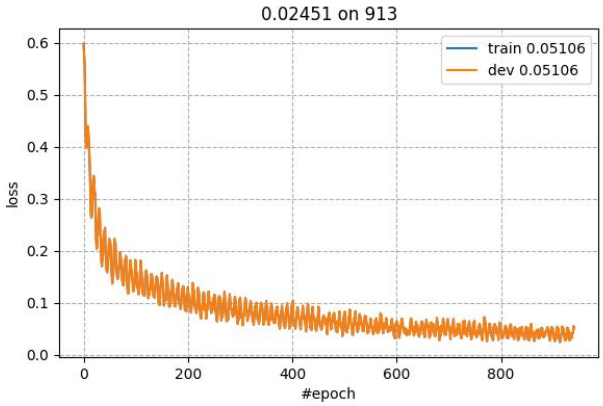
-

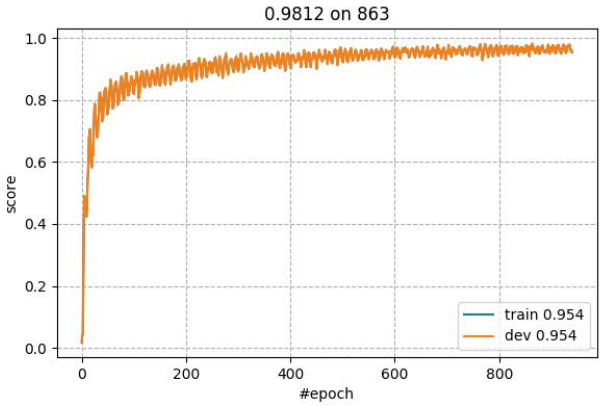
9. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

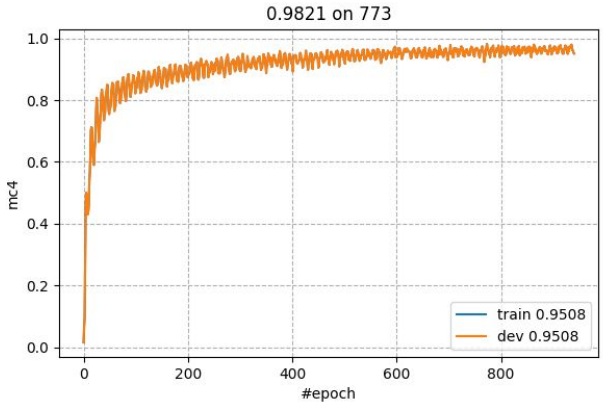
To preprocess data you need 200Gb of RAM. To train the model you need 1 GPU with 32Gb VRAM and about 2 weeks of heat release. If you encounter out of memory (OOM) error on GPU, try to resume training with argument --resume <PATH_TO_LAST_CHECKPOINT>.

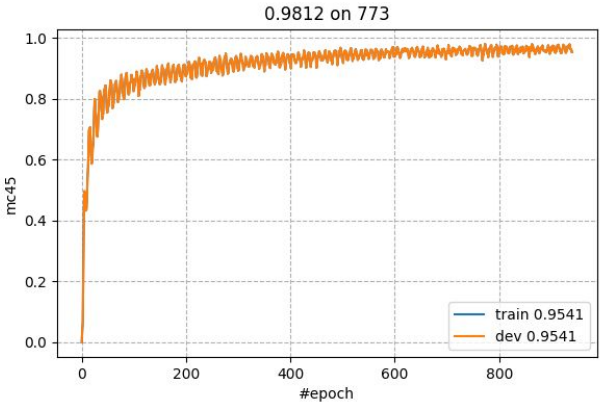
10. Do you have any useful charts, graphs, or visualizations from the process?

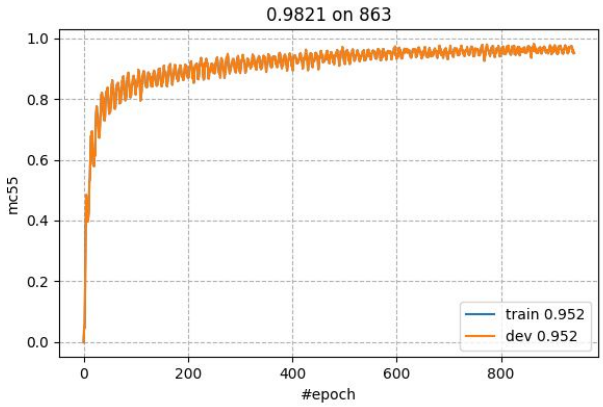
Here are the learning curves

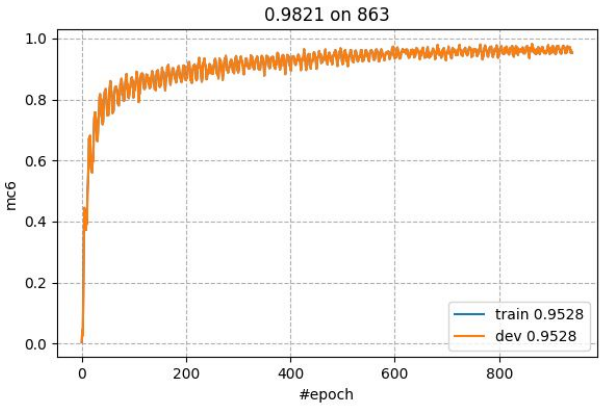

0.02451 on 913
train 0.05106
dev 0.05106
loss
#epoch

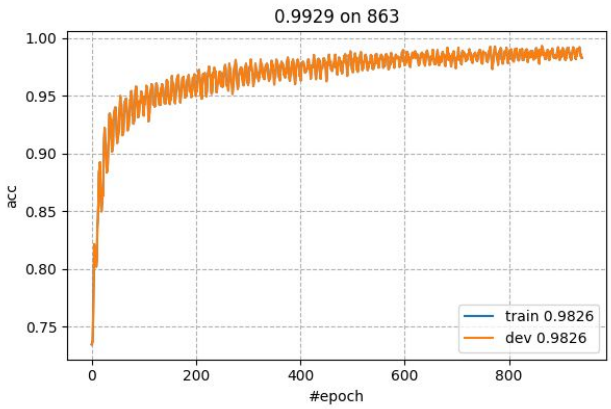

0.9812 on 863
train 0.954
dev 0.954
score
#epoch


0.9821 on 773
train 0.9508
dev 0.9508
mc4
#epoch


0.9812 on 773
train 0.9541
dev 0.9541
mc45
#epoch


0.9821 on 863
train 0.952
dev 0.952
mc55
#epoch


0.9821 on 863
train 0.9528
dev 0.9528
mc6
#epoch


0.9929 on 863
train 0.9826
dev 0.9826
acc
#epoch

where `loss` is a binary cross entropy loss, `mcT` is a Matthew correlation coefficient with threshold T and `acc` -- accuracy with 0.5 threshold.

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?

I've already tried all ideas I came up. One thing I would like to do is to optimize the model in terms of computational resources, use smaller CNN, lower spatial resolution and less data.