Competition Overview and Postmortem

Sunday, September 13, 2020 9:32 AM

- [NOTE 9/13/20: To be addended after competition winners publish their code and I can review it. At this point I've only seen the one solution from the end of competition webinar from leading teams.]

Competition Overview

- This was a pretty straightforward binary image classification task classifying venomous vs non-venomous snakes https://www.aicrowd.com/challenges/ai-for-good-ai-blitz-3/problems/snake
    - The only wrinkle is that in reality what the model was learning was a multiclass classification before then separating those classes into the two binary classes
        - i.e. venomous: rattlesnake, cobra, black mamba; non-venomous: python, garter, king
    - But unless I'm very mistaken these sufficiently deep NNs should handle this perfectly fine.
        - Which is not to say that there aren't different approaches which could take further advantage of this (I was thinking about clustering approaches)
- My models performed worse on the test set than on the training and validation sets suggesting that either I was having some overfitting (I think unlikely) or there was some aspects of the test set which weren't well reflected in the training data
    - I'm very curious to see if the competition winners did anything drastically different than me. In the webinar the solution presented was extremely similar to mine

Model Overview

- For this competition I boosted a bunch of CNN models together with a gradient-boosting metaclassifier
- The CNNs were all fastai transfer learning models with different architectures and trained on three different cv-splits. All models used imagenet weights for the pre-trained models
    - Models used were [resnet34, xresnet34, resnet50, xresnet50, densenet121, densenet201]
        - At the time of the competition fastai didn't have efficientnet easily accessible, otherwise I would have used it
- Models were trained using fastai's factory fine tuning method with the following parameters:
-
```
models=[resnet34,xresnet.xresnet34,resnet50,xresnet.xresnet50,densenet121,densenet201]

for model in models:
  cvdict=cv_splits('/content/All_Data/',3,0.15,True)
  for cvfold in range(3):
    train_val_dirs('/content/',cvdict,cvfold)

    dls = ImageDataLoaders.from_folder('/content/',train='train', valid='val',item_tfms=Resize(460),
                                       batch_tfms=aug_transforms(flip_vert=True, max_rotate=45.0, size=224))

    learn = cnn_learner(dls, model, metrics=[accuracy],cbs=[ShowGraphCallback()])

    learn.fine_tune(freeze_epochs=4, epochs=7, base_lr=1e-2)

    datestr=datetime.datetime.now(pytz.timezone('US/Eastern')).strftime("%y%m%d_%H%M")
    learn.export(mount_path+model_path+datestr+'_'+str(model).split()[1]+'_cv'+str(cvfold)+'.pkl')
```
- NOTE: The aug_transforms function has a ton of augmentations, the arguments shown are the ones where I'm changing from the default.
    - I would therefore characterize this model as having extensive augmentation
- After training I exported the models for later use
- For boosting I ran every training and test image through every classifier and trained a gradient boosting machine on the training data and then made predictions with it on the test data.
- This is where I got fucked. Something happened, possibly with fastai's get_image_files function or possibly with Colab's varied backends, but the predictions (which were saved in a simple numpy array, and not a labeled dataframe or something smart) got out of order between different

models which made using the boosting model impossible and I had to rerun all of the predictions over again using a robustly deterministic ordering of the images

  - Because the competition was so fast, I ran out of time and compute to do as much test-time augmentation (tta) as I wanted to and in the end only used between 2-4 rounds of tta for each image. I think being in the 4-10 range is much better and that this likely hurt me quite a bit.

Postmortem

- The boosting classifier reached f1=0.98 on a validation set but only achieved an f1=0.865 on the test set which put me in 16th place. The first place solution had f1=0.906. https://www.aicrowd.com/challenges/ai-for-good-ai-blitz-3/problems/snake/leaderboards
  - All things considered an f1 difference of .041 isn't a lot. From a competitive standpoint it's huge, but from the perspective of the classifier's efficacy it's fairly small
- The boosting classifier didn't have a true validation set, because the individual NNs had been trained on the data. To really measure the performance of the boosting classifier a true test set would have been needed, but I didn't have time to do that experiment AND train the models on as much data as possible.
  - Just means that the f1=0.98 isn't the whole story
- I think I would have benefited a lot from further time and compute to do more tta
- I didn't cv optimize pretty much any hyperparameters, I just found ones that worked very well and stuck with them. This may have helped slightly, but I think it was right to not spend time on this
  - The learning curves from training showed convergence and no overfitting, so the only thing I might have gotten was slight better accuracy
- I would have liked to use some better CNN architectures (efficientnet, others?) and explored other pretrained weights.
- I really suffered here from time and compute constraints, it probably would have helped me out a lot to try and optimize around those considerations