Model documentation and write-up

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

- I pursued a degree of doctor of veterinary medicine and later I obtained a PhD in the same field. In the meantime, I fell in love with computer vision and AI in general and made a huge professional stride.

- Workaholic. Addicted to Computer Vision Challenges. PhD student with focus on FGVC (Fine-Grained-Visual-Categorization). Experienced team leader from industry. Startup Founder.

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

First of all, we would like to point out that we entered the challenge just 10 days before the deadline. With regards to that, we are not able to clarify or proof all of our assumptions and prognosis that we had while building our system. For most of that we used our intuition and experiences acquired while working on other computer vision tasks / competitions / projects.

To accomplish this challenge we used a well known supervised machine learning paradigm, Convolutional Neural Networks. In the following paragraphs, we will cover our solution together with some references as links to the “Scientific Articles”.

We used commonly known architectures and approaches. Our solution is pretty straightforward. Bells and whistles used to improve the final score are described below.

Train time:

As test time was limited by time and we joined the competition really late with limited resources, we decided to use lightweight convolutional neural network architectures with one exception. Especially,

- EfficientNet B1 and EfficientNet B3 (https://arxiv.org/abs/1905.11946)
- SE-ResNext50. (https://arxiv.org/abs/1709.01507)

For all mentioned architectures we used ImageNet pretrained checkpoints.

- https://pytorch.org/docs/stable/torchvision/models.html
- https://github.com/Cadene/pretrained-models.pytorch
- https://github.com/lukemelas/EfficientNet-PyTorch

While training we used:

- No validation set.
- Cyclical Learning Rates / One cycle policy
    - https://arxiv.org/pdf/1708.07120.pdf

    - https://arxiv.org/pdf/1506.01186.pdf
    - Warm start and Cosine annealing
- Input size with same aspect ratio - 512x384
- Horizontal flip data augmentation

Two training strategies - random sampling and chunk sampling (we divided training set in chunk comprised of one or more seasons)

Test time:

- weighted ensemble (6 forward passes in total):
    - EfficientNet B1 (chunks)
    - EfficientNet B3 (random)
    - SE-ResNext50 (random) + horizontal flip
    - SE-ResNext50 (chunks) - 2 checkpoint (after season 10 and 9)
- Sequences (Observations) - mean for animals and gmean for empty

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

These are the 3 parts that in combination enabled us to train the models quickly (15 days on a single 1080Ti) with fast convergence and no overfitting (1 epoch)

- Cyclical Learning Rates / One cycle policy
- Lightweight models
- No augmentations but horizontal flip (0.5)

4. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    - Adjusting probabilities by priors based on day-time / season …Some species are active during the night and others not. However, including the priors for the date and time was not always helpful so we decided to drop it.
    - Background Subtraction for "animal like" region proposals
    - meta-learner (NN) with CNN probabilities and cyclic DateTime features
    - Fine-tuning on season -> 9+10

5. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
    - We used raw data without any preparation.
    - We have done only basic EDA (class frequency across date and time) and visualized errors in a few experiments.

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

    - Log Loss as primary metric and in initial stages we tracked accuracy but it was not very helpful.

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.)?

   - We trained our models on 1080Ti and 2070 so any GPU with 11GB or 8GB memory should work.

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

   - Yes we do have, examples are provided in the repo with code
   - Zeeeeeebra image motivated us to commit to the task!

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?

   - Inference time should be reduced further to maybe 2 h. Everyone knows that bigger models and ensembles help but this comes with the cost for both training and inference time. Instead limiting inference time to be able to ensemble 3-4 lightweight models or use only 1 bigger would produce more creative solutions.