Model documentation and write-up

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

   Most of my experience (about 5 years) is development of high load services using such technologies as Hadoop, Spark, Cassadra, Kafka.

   last 2 years I work as a CV engineer for embedded devices in noema.tech.

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

   1. Reproduce baseline.

   2. Change model, trainable layers, amount of images.

   2.1 Change loss function to the metric function.

   3. Found the model from https://keras.io/applications/ which give me the best score (it was InceptionResNetV2)

   4. Training several model and submit ensemble.

   5. Look at losses of each class and found that empty is the biggest

   6. Extract background from seq of images and trained binary classify (empty/ non-empty)

   7. Trained lgbm classifier as a second level model using predictions of each image + prediction of background classifier

   8. Build merged DNN based on InceptionResNetV2 which take background and mean of images simultaniously

   9. Update lgbm classifier using all model that I trained before.

   DNNs trained on seasons 1-8 (9,10 for validation), lgbm trained jn season 9 (10 for validation).

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

1. *np.sum([np.abs(img - imgs_mean) for img in imgs], axis=0)* extract backgrounds which help to reduce loss of empty class

2.

```
params = {
                'boosting_type': 'gbdt',
                'objective': 'binary',
```

```
                'metric': {'binary_logloss'},
                'num_leaves': num_leaves.get(clazz, 32),
                'max_depth': max_depth.get(clazz, 6),
                'learning_rate': learning_rate.get(clazz, 0.009),
                'feature_fraction':  feature_fraction.get(clazz, 0.6),
                'bagging_fraction': bagging_fraction.get(clazz, 0.8),
                'bagging_freq': bagging_freq.get(clazz, 5),
                'scale_pos_weight':1,
                'lambda_l1':lambda_l1.get(clazz, 0.2),
                'lambda_l2':lambda_l2.get(clazz, 0.2),
                'min_child_samples':min_child_samples.get(clazz, 25),
}
gbm = lgb.train(params,
                    lgb_train,
                    num_boost_round=3000,
                    valid_sets=[lgb_train, lgb_val],
                    early_stopping_rounds=20)
```

searching parameters and training second level classifier.

3.

back_input =
model_back.layers[-4].output

mean_input =
model_mean.layers[-4].output

x = Concatenate(axis=3)([back_input,
mean_input])

x = Conv2D(filters=1024, kernel_size=3, padding='valid')
(x)

x = GlobalMaxPooling2D()
(x)

x = Dense(54, activation="sigmoid")
(x)

model = Model(inputs=[model_back.input, model_mean.input], outputs=x)

Concatenation of pretrained background and normal models and training as a single one.

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

   On boosting step we can use average scores from all other cameras around several seconds. If zebra walks near a camera we can expect it will be detected again soon. Of course we do not know which scores to average but it does not matter, anyway it improves result.
   I did not use it because leaderboard has a sparse data and it should not work properly there. It is risky to make such algorithm without test. But I tested it on a full season 10 and it works great. On season 10 holdout it improves from 0.00265 to 0.00242.

5. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   No

6. How did you evaluate performance of the model other than the provided metric, if at all?
   I looked at each class (zebra, empty...) metric separately
   metric==loss, so I did nothing else.

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.)?
   No

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

No

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?

On production we know which camera provides images.
So we can use information from previous images and so:
- we can clearly define background of each camera (separately for day and night time)
- we can use previous predictions to build better second level classifier
- we can create a map and predict movement of animals, so if we know that lion runs to another camera it is a good feature.