Post competition

This is a compilation of things that could have been done better.

* You can add external data to your validation set and compute two CV scores. One using the external data and one without using the external data. Obviously, you should believe the one not containing the external data more, however, it can help to see how well the model generalizes on different distributions of data.
* Have a nice CV strategy and believe more ur CV score than the public score.
* When ensembling models based on OOF files, (ie ensembling oof predictions), it is important to have the same splits for all the oof models ensembled. Ie, this means using the same n\_splits and random\_seed for all models: klearn.model\_selection.KFold(n\_splits = 5, shuffle = True, random\_seed = 42). You should not have 5 folds for one models and 15 folds for another, since there you contaminate the CV score of the ensemble.
* Higher EffNets work better, lvl 0, 1, 2 where not helping the ensemble at all.
* Image size is also important. 128, 192, 256 were decreasing the scores.
* The top solution combined tabular and single cnn before doing the ensemble of models. Instead of ensembling all cnn and then combining with the tabular.
* Data augmentation is important:

|  |
| --- |
| transforms\_train = A.Compose([  A.Transpose(p=0.5),  A.VerticalFlip(p=0.5),  A.HorizontalFlip(p=0.5),  A.RandomBrightness(limit=0.2, p=0.75),  A.RandomContrast(limit=0.2, p=0.75),  A.OneOf([  A.MotionBlur(blur\_limit=5),  A.MedianBlur(blur\_limit=5),  A.GaussianBlur(blur\_limit=5),  A.GaussNoise(var\_limit=(5.0, 30.0)),  ], p=0.7),   A.OneOf([  A.OpticalDistortion(distort\_limit=1.0),  A.GridDistortion(num\_steps=5, distort\_limit=1.),  A.ElasticTransform(alpha=3),  ], p=0.7),   A.CLAHE(clip\_limit=4.0, p=0.7),  A.HueSaturationValue(hue\_shift\_limit=10, sat\_shift\_limit=20, val\_shift\_limit=10, p=0.5),  A.ShiftScaleRotate(shift\_limit=0.1, scale\_limit=0.1, rotate\_limit=15, border\_mode=0, p=0.85),  A.Resize(image\_size, image\_size),  A.Cutout(max\_h\_size=int(image\_size \* 0.375), max\_w\_size=int(image\_size \* 0.375), num\_holes=1, p=0.7),   A.Normalize() ])  transforms\_val = A.Compose([  A.Resize(image\_size, image\_size),  A.Normalize() ]) |

* Keep batch size >= 36 (may have to use multi-gpus)
* Instead of using Binary Cross Entropy loss, they used 9 classes w/ Cross Entropy Loss per each model.
* Should use a warmup epoch in order to avoid saving a model in the first epochs.
* Tqdm library for the loops is quite useful.
* When ensembling different models that use the same oof strategy, it is usually better to ensemble them using a weighted mean average rather than a simple average. This can be achieved using a Bayesian optimisation. There is another technique called forward selection OOF ensemble, which not only weights the models but also select the best combination of models that maximize the cv score.