EDA

> есть авто не относящиеся к классу `car`

Это как понять? :)

> кропним target'ы

Не совсем понимаю что это означает. Под «таргетом» обычно понимаются целевые данные, то есть то, что нужно предсказать, в нашем случае номер класса. Как его можно кропнуть?

> не разобрался какая модель использует GPU, чтобы ускорить процесс

Вряд ли какая-то конкретная модель использует GPU, а какая-то нет. Скорее это зависит от настроек всей библиотеки в целом – будет использоваться GPU или нет.

> в среднем 1.24 секунды на картинку (итерацию)

Не толькоь обучение, но и предсказание (инференс) делается не отдельными картинками, а батчами. Поэтому дело еще в размере батча: чем он больше, тем меньше времени уйдет на одну картинку.

> хотя встречаются излишне (неточно) обрезанные \ проводить дополнительный tuning модели не будем (да и не умею)

Такие модели обучаются на очень большом датасете, и вряд ли тюнингом здесь можно еще что-то улучшить. Тем более откуда взять размеченные данные для тюнинга? Скорее можно взять не одну, а несколько моделей и усреднить их предсказания (то есть выдаваемые ими маски).

> Отсеяно - 2596 изображений

Здесь хотелось бы видеть визуализацию ответов модели center_net_resnet50_v1b_voc и отсеянные изображения, и главное алгоритм отсеивания. Может быть это прояснит почему отсеяно так много изображений.

```
axs[1].boxplot(classes_dist, vert=False)
```

Строя бокс-блот полезно помечать что отложено по осям, т. е. что этот бокс-плот означает. Здесь по-видимому это распредение классов. Но вообще такой график пожалуй излишний, поскольку график слева более информативен.

> Идея оказалась нежизнеспособной: средняя точность на валидации 40-50% против 70-80% с некропнутыми изображениями\ (См. удаленный код в Module-7_[Addon]_Getting_Cropped_Image.ipynb)

Посмотрел этот файл, но не нашел там самого алгоритма кропа. Там только создание генераторов с использованием функции augmentations, которая нигде не определена. Видимо часть кода вы удалили. В общем, для понятности здесь стоило указать конкретный алгоритм кропа, как именно вы создавали и обучали модель и какой получился график точности на валидации. Тогда была бы необходимая информация, на основе которой можно было бы строить предположения. Точность на валидации 40-50% нереалистично низкая, поэтому причиной скорее всего является какая-то ошибка.

> Хотелось найти способ сохранить соотношение сторон картинки при прохождении через from_dataframe(target_size), но выяснилось, что данный функционал все еще в разработке с 2018 года

Как вариант можно использовать tf.data.Dataset или заранее обработать изображения на диске так, чтобы они имели одинаковое соотношение сторон (например методоп crop или pad).

```
transform = A.Compose([
```

Здесь можно еще добавить cutout. Это хороший способ повысить точность на валидации. Однако у всех аугментаций есть и обратная сторона: они могут понижать точность на валидации из-за проблемы, называемой batch-normalization train-test discrepancy (можете поискать в интернете, например здесь https://arxiv.org/pdf/1906.03548.pdf , раздел 3.1). Особенно cutout связан с этой проблемой. Поэтому к концу обучения стоит уменьшать степень аугментаций.

```
val_generator = datagen.flow_from_directory(directory=TRAIN_IMG_PATH,
```

У вас валидация делается с аугментациями. Так не должно быть. Представьте, что одну модель вы обучили на слабых аугментациях, другую на сильных. Если валидация делается с аугментациями, то модель с сильными аугментациями покажет более низкую точность на валидации. Если же делать валидацию нормально, без аугментаций, то все может быть наоборот. Поскольку валидация влияет на выбор моделей и гиперпараметров, важно делать ее без аугментаций.

Вижу что вы хорошо разобрались в устройстве головы сети и провели несколько интересных экспериментов.

Step 1
Defrost a half of the pre-trained model weights

А вы пробовали обучать сразу всю модель, без постепенной разморозки? Возможно результат был бы лучше.

```
TensorBoard(log_dir=LOG_PATH, histogram_freq=1)
```

С tensorboard разобрались, это очень здорово. Мало кто его использует.

```
def test_augmentations(image):
… A.ShiftScaleRotate(p=0.75,
```

По идее в TTA нужно делать преобразования с p=1

```
for a in range(5):
    # TTA predict
    print(f'TTA loop {a+1}:')
```

Можно как вариант делать TTA больше раз на сомнительных изображениях. Если на изображении все 5 ответов совпали – дальше не продолжаем, а если не совпали – делаем еще больше итераций TTA.

```
for i, j in enumerate(models4ensemble):
        model = keras.models.load_model(j)
        print(f'-> Predicting of Model {i+1}:')
```

Получается что вы делаете предсказания на 4 моделях (models4ensemble) и затем усредняете их с весами 0.1, 0.2, 0.5, 0.5? Идея очень хорошая, но было бы полезно сначала посчитать точность каждой из моделей по отдельности. Вдруг какая-то из них совсем плохая?

> Был проведен обзор (доступных) работ участников верхних строк лидерборда Kaggle соревнования и вяты в работу pretrained и output (head) модели наиболее часто встречающиеся и показывающие наилучший результат

То что все в лидерборде так делают не означает, что это лучший вариант. Все просто копируют бейзлайн. Наилучший результат наверняка достигается на более современных моделях, чем те, что есть в keras.applications. Можно было бы например взять например модель efficientNetV2 из tensorflow_hub.

> Сколько раз запарывались расчеты из-за обрыва сессии...

Есть еще платные платформы для облачных вычислений на GPU, такие как vast.ai.

> Что можно еще сделать помимо увеличения разрешения input images:

Вы ведь использовали модель для сегментации автомобилей из gluon? Этой моделью можно было бы обрезать фон со всех изображений, ведь на фоне сверточные сети переобучаются.

В целом отличная работа!