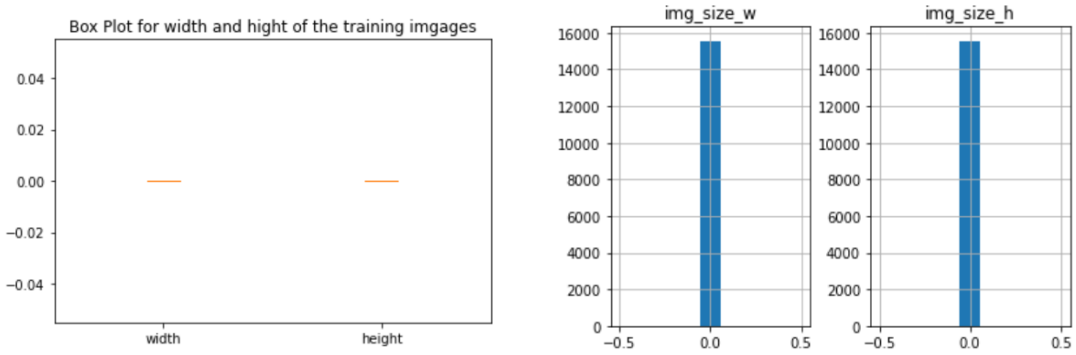
model=Sequential()
model.add(base_model)
model.add(GlobalMaxPool2D())
model.add(Dropout(0.35))
model.add(Flatten())
model.add(Dense(256,activation="elu"))
model.add(Dropout(0.25))
model.add(Dense(CLASS_NUM))

Здесь не совсем понимаю из каких соображений добавлен слой Flatten. В данном случае наличие этого слоя будет эквивалентно его отсутствию. Чтобы это понять достаточно найти размерность тензоров, которые выдает каждый слой сети.



Тут видимо что-то не так с графиками.

Alb.ChannelShuffle(p=0.8)

Операция, приводящая к очень сильному искажению цветов. Нужна ли такая аугментация? Обычно так не делают, но с другой стороны – почему бы и нет? Вообще говоря, хорошо обученная сеть должна распознавать авто независимо от цветов, даже на черно-белых изображениях. Но на практике обесцвечивание снижает качество распознавания (по крайней мере при небольшом количества эпох). Почему – неясно. Может быть и ChannelShuffle, как чрезмерно сильная аугментация, снизит качество. Но это открытый и непростой вопрос, не могу с уверенностью утверждать о правильности или неправильности такой аугментации.

Alb.CoarseDropout (p=0.7, max_holes=2, max_height=10, max_width=20)

Как видно из визуализации, для изображений 320x320 это очень маленький dropout, вряд ли будет иметь сильный эффект.

 # тут я менял руками базовую модель и архитектуру
 # Xception # rescale 21M
 # ResNet50V2 # rescale 23.5M
 # EfficientNetB6 # 41M

Для разных моделей нужна разная нормализация, подробнее это написано в документации. Например, EfficientNet принимает изображения без деления на 255. По-видимому вы забыли это учесть, и все модели принимают ненормализованные изображения.

#model.add(BatchNormalization(axis=1)), # можно было бы добавить, но я же использую елу, с ней не нужно

В статье про ELU действительно сказано следующее: «On CIFAR-100 ELUs networks significantly outperform ReLU networks with batch normalization while batch normalization does not improve ELU networks». Но я бы не был так уверен. После выхода этой статьи (2015 г) появилось много сверточных сетей, использующих BN. Возможно что авторы статьи были неправы, заявив о ненужности BN. Во-первых BN является способом регуляризации за счет стохастичности своей работы (примеры в батче влияют друг на друга), во-вторых впоследствии вышла статья, в которой говорится, что BN улучшает нейронные сети за счет сглаживания ландшафта функции потерь.

```
sub_datagen = ImageDataGenerator(
   #rescale=1.0/255,
   #preprocessing_function=augment
   horizontal_flip=True,
   rotation_range=10,
   shear_range=0.2,
   brightness_range=(0.8, 1.2)
)
```

Непонятно почему на TTA вы делаете аугментации не таким способом, как на трейне. Иногда это может приводить к проблеме «train-test discrepancy» при наличии слоев BatchNorm. В общем, есть шанс, что точность была бы чуть выше, если бы вы делали аугментации тем же способом, что и на трейне. Но это не слишком принципиально.

```
val_datagen = ImageDataGenerator(
   rescale=1./255,
   validation_split=0.5,
)

val_cm_datagen = val_datagen.flow_from_directory(
   train_path,
   target_size=image_size,
   batch_size=BATCH_SIZE,
   class_mode='categorical',
   shuffle=False,
   seed=0,
   subset='validation'
)
```

Не совсем зачем создавать заново val_datagen, только на этот раз делить изображение на 255 (при обучении, судя по коду, оно не делалось).

```
#Confution Matrix and Classification Report
```

У ячеек с этим кодом почему-то нет вывода, саму матрицу не вижу.

```
train_datagen = ImageDataGenerator(
   #rescale=1./255,

train_res_datagen = ImageDataGenerator(
   rescale=1./255,
```

Есть более простой способ. Добавьте нормализацию в саму модель в виде слоя Lambda, и вам не придется делать по два генератора с нормализацией и без нее. Либо можно сделать так:

```
sub_res_generator = (X/255, y for x, y in sub_generator)
```

«*Вариант TTA. Зависит, если на 10 аугментациях все хорошо, то принимаем класс, если еще не точно, то делаем дальше, используя псевдолейблинг*»

Здесь не понимаю что имеется в виду, что значит делаем дальше используя псевдолейбелинг. Псевдолейбелингом называется обучение сети на изображениях из данных для сабмита, такого я в коде не нашел.

Оценивание по критериям

Качество и понятность кода
В целом код хороший, но все же есть несколько проблем, затрудняющих восприятие кода. Представьте, что вы сдаете код тимлиду в компании, соответственно ваша задача – чтобы все было наглядно и понятно. Во-первых большая часть ячеек кода не имеет вывода или вывод некорректен, например не видно confusion matrix, а графики (mean img width) отрисованы с ошибками. Не удалена часть кода из бейзлайна, здесь совершенно не нужная (предсказание на car.jpg). Много раз создаются генераторы, может быть стоило бы обернуть их создание в более лаконичную функцию. И вместо «if False:» можно просто закомментировать ячейку: Ctrl-A, Ctrl+/. Но все это не слишком серьезные проблемы.
Оценка: 3 балла из 3.

Метрика качества
В лидерборде точность 0.96943
Оценка: 2 балла из 3.

Работа с обучающими данными
Использованы аугментации из albumentations, но все же выбор аугментаций ничем не обоснован: ни точностью на валидации, ни научными статьями (хотя бы одним из этих 2 вариантов). Cutout (Alb.CoarseDropout) очень маленький – вряд ли он на что-то повлияет, обычно его делают намного больше. Есть и еще две проблемы. Во-первых непонятно почему при TTA аугментации делаются другим способом, нежели при обучении. Во-вторых генераторы создаются для разных моделей как с нормализацией, так и без – очень много лишнего кода.
Оценка: 1 балл из 3.

Работа с архитектурой модели
Применены три разные модели из keras applications. В интернете есть бенчмарки, можно сравнить точность и размер разных моделей и так выбор уже был бы более обоснованным. Добавлены скрытые полносвязные слои, проведено сравнение точности на валидации.
Оценка: 3 балла из 3.

Работа с процессом обучения
Из раздела с описанием моделей («Пробовал 3 модели - model 1, …») непонятно как соотносятся между собой по точности на валидации модели 1, 2, 3, и также разные сверточные архитектуры. Стоило бы привести графики их обучения. Проведено обучение с постепенной разморозкой слоев, но нет сравнения результата с моделью, которая обучалась бы сразу целиком – может быть ее точность получилась бы выше? Вообще было бы хорошо, если бы процесс обучения и полученные результаты был детально описан в текстовом виде. Потому что по коду кажется, что вы попробовали один вариант и не пробовали никаких других. Так тоже можно делать при нехватке ресурсов, но в этом случае нужно обосновывать этот выбор научными статьями.
Оценка: 1 балл из 3.

Работа с процессом инференса
Делаются предсказания с TTA и ансамблированием. Правда я не нашел сравнения точности с TTA и без него, но предполагаю что вы все-таки это сравнение делали.

Оценка: 3 балла из 3.
+2 бонусных балла за реализацию бустинга
+1 бонусный балл за confusion matrix и выводы из нее

Создание работающего прототипа
Этого пункта у вас не нашел.

Сумма: 3+2+1+3+1+3+2+1=16 баллов