Добрый день! Сразу отмечу одну проблему: было бы лучше, если бы репозитории был readme-файл, иначе непонятно куда смотреть и какой файл за что отвечает. Если вы добавите этот проект в портфолио, то потенциальному работодателю может не понравиться, что вы не документируете репозиторий и код.

**000_split_dataset.ipynb**

```
X_train, X_valid, y_train, y_valid = train_test_split(imgs_list,
class_list, test_size=0.2, random_state=142)
```

Все же советую писать X_val, y_val, так больше принято в ML , а valid – это слово с совсем другим значением, поэтому может запутать.

Не совсем понял смысл выполняемых действий. У вас получается, что папки train и valid помешаются внутрь папок с классами, а зачем? Если ваша задача – разделить данные на train и val, то это можно сделать без копирования файлов, стандартными средствами такими как параметр validation_split в ImageDataGenerator, как показано в бейзлайне. Если вы используете pytorch, то можно делать так, как показано здесь, используя SubsetRandomSampler.

**001_Train_baseline/analayze_results.ipynb**

Я правильно понял, что результаты экспериментов вы сохраняете в csv? Хорошая идея, так действительно легче сравнивать, объединяя в таблицу.

Только из таблицы ничего не понятно, почему-то эпохи нумеруются как 0, 1, 4, 36, …. Надо было добавить какие-то комментарии.

```
In [5]: df_concat.sort_values(by='valid_loss')
```

Out[5]:

| | Epoch | train_loss | train_acc | lr | batch_size | e |
|---|---|---|---|---|---|---|
| **26** | 0 | 0.541 | 0.815 | 0.0043 | 32.0 | 7 |
| **27** | 1 | 0.166 | 0.943 | 0.0043 | 32.0 | 7 |
| **30** | 4 | 0.214 | 0.928 | 0.0043 | 32.0 | 7 |
| **62** | 36 | 0.033 | 0.990 | 0.0043 | 32.0 | 7 |
| **71** | 45 | 0.017 | 0.994 | 0.0043 | 32.0 | 7 |

**001_Train_baseline/main.py**

```
######## Upload model and weight ########
```

Скорее download, а не upload, вы ведь скачиваете модель.

```
trial.suggest_float("lr", 1e-4, 1e-2, step=step)
```

Здесь советую использовать логарифмическую шкалу: параметр log=True. То есть логарифм learning rate выбирается из равномерного распределения.

```
trial.suggest_int("batch_size", 32, 80, step=16)
```

batch_size не слишком сильно влияет на результат, а вот на скорость обучения влияет существенно. Поэтому обычно batch_size выбирают максимально возможным, чтобы сеть помещалась в память.

```
trial.suggest_int("epochs", 20, 120, step=10)
```

А зачем выбирать количество эпох как гиперпараметр, если можно обучать сеть до тех пор, пока loss на валидации не перестанет падать.

Можно было еще подбирать аугментации. Например использовать suggest_int('rotate', 0, 1) для того, чтобы определить, будет ли осуществляться случайный поворот изображения.

```
valid_dataloader = DataLoader(valid_dataset, batch_size=8
```

Для валидации можно взять большой размер батча, так как валидация – это инференс, а при инференсе тратится меньше памяти, чем при обучении, так как для расчета не нужно запоминать выходы предыдущих слоев.

```
GradScaler()
```

Вы же не используете float16, тогда зачем GradScaler()?

Кстати, можно было использовать pytorch-lightning, с ним код получается проще и короче, чем со стандартным pytorch.

**002_Run_model/run_model.ipynb**

```
torchvision.transforms.Resize((224, 224))(img)
```

Можно было попробовать больший размер изображения, точность могла бы стать выше.

В целом интересная работа тем, что выполнена на pytorch, потому что все в основном используют tensorflow. Но комментарии в коде бы не повредили, а то не очень понятно что там происходит, приходится очень подробно смотреть чтобы понять. Комментарии бы облегчили понимание.