Introduction

The goal of this notebook is to describe my journey after almost a month of work competing on Kaggle's competition **CellSignal: Disentangling biological signal from experimental noise in cellular images**. The full proposal can be viewed in the README of the github repository.

As a reminder, the goal is to classify images of cells under one of 1,108 different genetic perturbations, and thus eliminate the noise introduced by technical execution and environmental variation between [drug] experiments. This is a multiclass classification challenge applied to a healthcare related topic.

This notebook will contain the following sections:

- The journey
- Upcoming attempts
- Reflexions
- Conclusion

Although I think this notebook is sufficient to allow me to graduate from the nanodegree program, I definitely plan on continuing to compete in this competition after graduation since there is still two months to go.

The journey

Technical limitations disclaimer

Since I started this competition a month ago, after my proposal had been accepted, I ran into several issues preventing me to experiment all the things I wanted to before graduation.

First, I needed to upload the competition's data on FloydHub because this is where I have the most GPU credits, which is definitely needed for this competition. During this process, I corrupted the data and it took me a whole week to realize this was the problem. My models didn't want to train because of that, and the stacktrace and errors were not comprehensible. I was then back to square one after a week and a half of work.

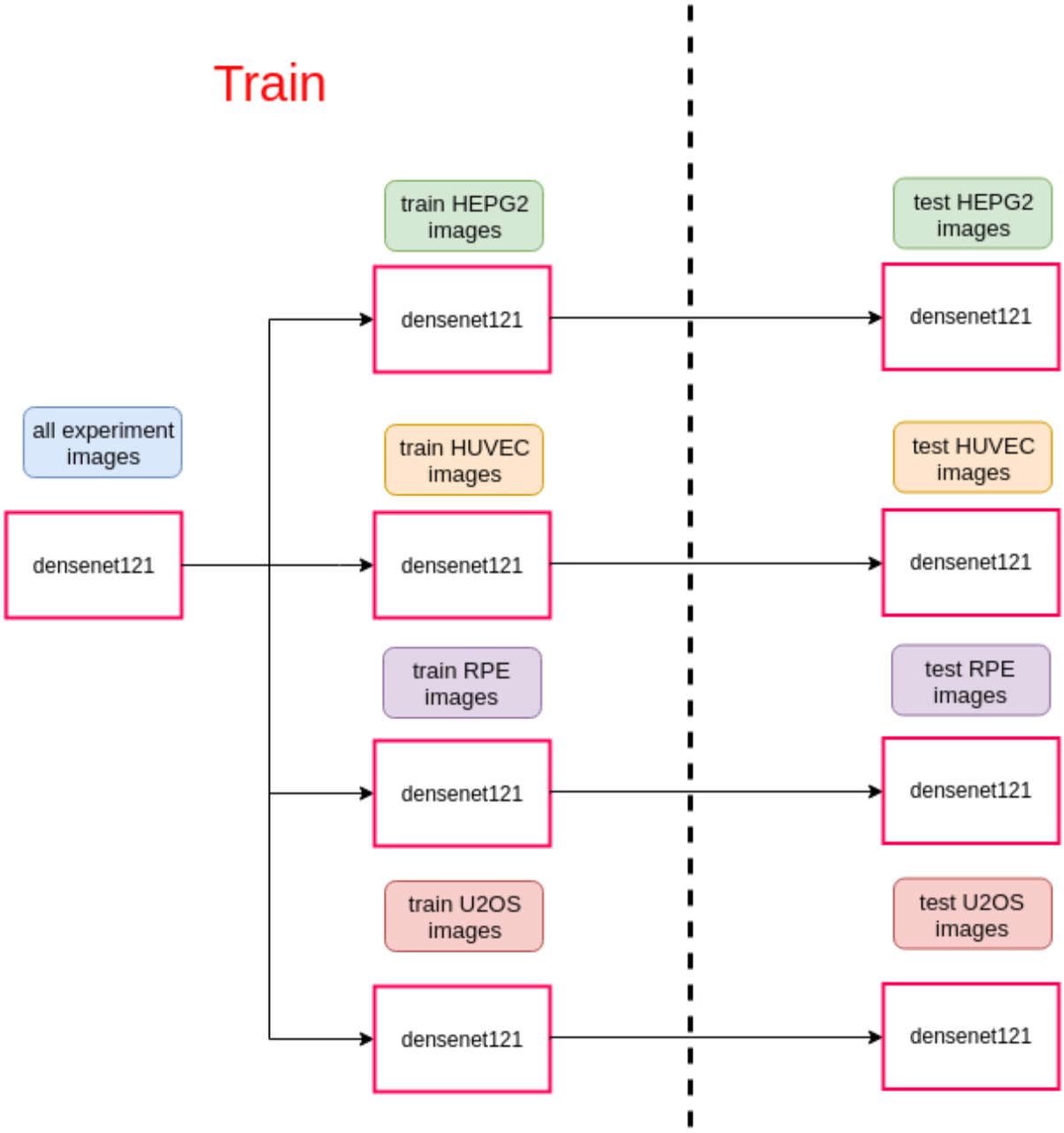
I then tried a second time to upload the data on FloydHub, but for unknown reasons, this time it was crashing at the middle of the almost-24-hour-long uploading process. I lost about three days trying to upload the data, to finally give up.

This is at this moment that I started to work directly through Kaggle's kernels. However, Kaggle's kernels have limited resources and do not allow to train more complex models such as ResNet-152 or any Densenet - which appear to be one of the key in obtaining superior scores according to some of the competition's discussions.

Furthermore, kernels on Kaggle have a 9 hours computing time limit. This forces you to divide work across multiple kernels because the time to train models for this competition is quite long - 1 hour for the base model, 5+ hours for the ResNet-50, and most likely way more for more complex models with a lower learning rate (because more epochs before convergence).

Also, since some discussions vent the merit of adopting a two-step model approach (a step to train the model on all the cell images, and a second step to train models on one of the four cell types (HEPG2, HUVEC, RPE, U2OS), training time can become exponentially long.

> A two-step model training approach - credits phalanx



Actions will be taken to solve all of those technical limitations in the upcoming weeks, but I couldn't have known these limitations ahead of time as a first time Kaggler. This will be discussed in the **Upcoming attempts** section.

Base Model

As suggested in the proposal, the base model (see notebook) was a very basic CNN model trained over 2 epochs on 6-channel 512x512 images. Here is the overall architecture:

Data: 6-channels images, both sites. Input resolution: 512x512

Base model: Basic CNN model

```
Net(
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
  (conv1): Conv2d(6, 16, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bia
s=False)
  (conv2): Conv2d(16, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bi
as=False)
  (fc1): Linear(in_features=65536, out_features=2216, bias=False)
  (fc2): Linear(in_features=2216, out_features=1108, bias=False)
)
```

Loss: Cross Entropy Loss

Optimizer:
Adam w/ lr=0.0003

Epochs: 2

Framework: PyTorch

This gave a cross-validation score of 0.0011 (.1% accuracy), and a test score of 0.002 (.2% accuracy). This score is a bit better than chance since we have 1108 classes. An accuracy reflecting chance would be 1/1108, which is equivalent to ~0.09% accuracy.

ResNet-50 - a better model

Despite the technical limitations encountered, I succeeded in training a ResNet-50 model with the 6-channel 512x512 images (see notebook). The first convolutional layer was modified to accept 6 channels instead of 3. The rest remained similar to the base model architecture, although we added early stopping and a learning rate scheduler decreasing the learning rate at each epoch. Here is an overview:

Data: 6-channels images, both sites. Input resolution: 512x512

Base model: ResNet-50 accepting 6-channel images as input and outputting 1108 possible classes.

Loss: Cross Entropy Loss

Optimizer:
Adam w/ lr=0.0003 + learning rate scheduler w/ gamma=0.95

Epochs: 15 epochs maximum

Framework: PyTorch

This gives us a cross-validation score of 0.27 (27% accuracy), and a test score of 0.125 (12.5% accuracy). This score is way better than chance since we have 1108 classes. An accuracy reflecting chance would be 1/1108, which is equivalent to ~0.09% accuracy.

Disclaimer: you will actually see `10.4%` in the ResNet-50 notebook - it is because I ran it twice and the second time it was with a poorer initial learning rate and gamma. The best learning rate and gamma attempted thus far are `lr=0.0006` and `gamma=0.9`. Sadly, I only saved the second notebook with `lr=0.0003` and `gamma=0.95` - but saved the best weights of the first notebook.

Also, it seems that the lost function may not be the best indicator of our model performance. We obtained the minimum loss at the 6th epoch, but it resulted in a lower accuracy score compared to the 10th epoch: 8.4% vs 10.4%.

ResNet-50 (step 2)

As discussed in the **Technical limitations disclaimer** , many discussions and kernels on the competition's website talk about the relevance of taking a two-step approach to training our model: 1) train the model on all cell images, 2) train four seperate models on one of the four cell types (HEPG2, HUVEC, RPE, U2OS).

I attempted such an implementation in the following notebooks based on xhlulu's work:

- https://github.com/MichelML/ml-cellsignal/blob/master/my_notebooks/cell_line_model_hepg2.ipynb
- https://github.com/MichelML/ml-cellsignal/blob/master/my_notebooks/cell_line_model_huvec.ipynb
- https://github.com/MichelML/ml-cellsignal/blob/master/my_notebooks/cell_line_model_rpe.ipynb
- https://github.com/MichelML/ml-cellsignal/blob/master/my_notebooks/cell_line_model_u2os.ipynb

However, after aggregating the submission files of these notebooks and submitting the result on Kaggle, our score from a one-step approach to a two-step approach dropped by 0.03%. This is surely due to a bad implementation on our part.

First, there may be overfitting due to the fact that we were resplitting the data after the second step, which may cause validation data to enter the training data. Second, in xhlulu's work, they perform data augmentation through flipping and rotational changes, which may help the model generalize better.

Upcoming attempts

Many things remain to be attempted to make the accuracy increase.

Technical limitations

- Succeed at uploading competition's data on FloydHub
- Gain more computing resources once the data is uploaded on FloydHub
- Try more complex models, more epochs, and lower learning rate.

Two-step approach

- Implement the two-step learning approach successfully.
- Add data augmentation to this process to improve generalizability.

Data augmentation

- Scaling
- Translation
- Rotation (at 90 degrees)
- Rotation (at finer angles)
- Flipping
- Adding Salt and Pepper noise
- Lighting condition
- Perspective transform
- GANs to generate additional fake images (source)

Training speed

- Pre-compute images to tensors to avoid on-the-fly image loading.
- Reduce image size.
- Use 3-channel images instead of 6.

Image classification tricks

- See Bag of Tricks for Image Classification with Convolutional Neural Networks by Sanyam Bhutani.

Domain specific tricks

- See biological tips here
- See conversation around the use of negative and positive controls here and here.

Reflexion

I realized through this capstone that Kaggle's competitions are very powerful. There is nothing better than being part of competitions like this one and learning from experts from all over the world. The only thing I regret now is not taking part in such competitions earlier in my learning journey. It forces you to truly learn all the important concepts - given that you want to win the competition. I plan on continuing to participate in this kind of competitions in the future, hoping to grab the rank of Kaggle Master before 2020.

Conclusion

After reading a lot of papers, blog posts, kernels, and discussions, and trying various model architectures (which I'll talk about later), my current best solution for this 1108-class classification problems is `12.5%` based on the model weights of the 12th epoch.

Although much improvement remains to be made, a score of `12.5%` is actually way better than chance since we have 1108 classes. It allows me to rank in the top 25% of the competition thus far. An accuracy score reflecting chance would be 1/1108, which is equivalent to ~0.09%.

After my Udacity graduation, my goal is to get a silver medal in this competition.

Thanks

A heartfelt thanks to all the reviewers, mentors, and Udacity staff for this amazing journey.

All the best.