The purpose of this document is to review our team’s performance during Kaggle’s Prostate-cANcer-graDe-Assessment-PANDA-Challenge. Our team ended up placing 654 of 1010, putting our final submission in the top 65%. We had a lot of fun during this competition, and while some of what follows may seem critical, it is not to be taken personally.

There were several techniques and tools that were very useful for the project, including progressive resizing, class binning, and model blending. Progressive resizing is a preprocessing technique that trains a model on batches of smaller images, and then retrains several times using progressively larger images. This leads to a strong model that is originally built upon lower-quality images. Progressive resizing had a strong positive impact the performance of many of our models. Another preprocessing technique that was used in the project was the Python library Cupy, which allowed for faster computing outside of Kaggle kernels. This improved our experiment runtime, but did not necessarily influence the accuracy of any of the models.

Furthermore, running our projects on both Kaggle’s kernels, Colab sessions, and our own machines allowed us to greatly increase the number of experiments we can run, sometimes simultaneously, without having to worry about Kaggle’s limited GPU quota. Since one of the strongest limiting factors in this competition was limited cloud GPU and CPU resources, being able to have outside sources of computational power greatly improved our research speed.

Qishen Ha’s class binning method and loss function BCELogitsLoss (Kaggle forum post <https://www.kaggle.com/c/prostate-cancer-grade-assessment/discussion/155424>) also greatly contributed to the success of the project. This method essentially allowed for each image to be classified as a probability array consiting of all 5 gleason patterns, instead of simply deciding on one. Training a model this way gives more leeway for model improvement, as the model allows for the possibility of an image being classified in more than one category (essentially allowing the ordinality to be visible to a classification model).

Finally, the model blending technique, which predicted results based on the combined probabilities of several models lead to a significant boost in performance and model accuracy. This allowed us to merge several models and get a rough estimate on which combinations would lead to the best results, which increased both performance and work efficiency.

Methods and techniques that were not effective in the project were cycling loss functions and training models on small batch sizes. Certain experiments we ran gave poor results due to low batch size in training (which resulted in some false conclusions early on in the competition). Cycling loss functions caused the models to overfit wildly. In many cases unfreezing pretrained models to train early layers caused overfitting which would not be visible until running the model on test data.

Of course, there was much room for improvement in several areas of the project.

We did not include Efficientnet or Resnext in our final submission as we were not able to get good performance out of those models (many other teams did extremely well with those models). Finding a way to incorporate these models could have lead to very promising results. Additionally, very late in the competition, we found some weird quirks with training models on images of a particular size, then at inference using a different size which the model hadn’t been trained on; this sometimes lead to positive results that warranted further exploration. While blending models (averaging the results on many models) worked well, we never tested model stacking (using the predictions of many models as the input of a final model). Lastly, a few of the notebooks could have used better documentation as much of the code was not straightforward enough to skim through.

The most creative and novel approach in our group was striplication; however we didn’t manage to include it in the final submission. The early results of striplication were a bit too memory and time intensive to include in our submission. Towards the end of the competition, striplication had some promising results, but they required a team effort to implement and we were each too focused on our own models and projects. A remaining untested hypothesis is that we could have trained a model on the striplicated images, then retrained the same model on images that had a computationally cheaper preprocessing method to retain the same performance for less compute on new images.

Overall, the competition was a great experience for all of us. Although we did not quite reach our goal for the competition, we did come across many learning opportunities and were able to pick up on new tools and methods we were not aware of. We had a lot of fun while improving our skills, discussing ideas, and even developing new methods for future competitions.