Dhruv Adha

May 5, 2024

AI Precision Lab—Kits-21 Project Report

Problem Statement

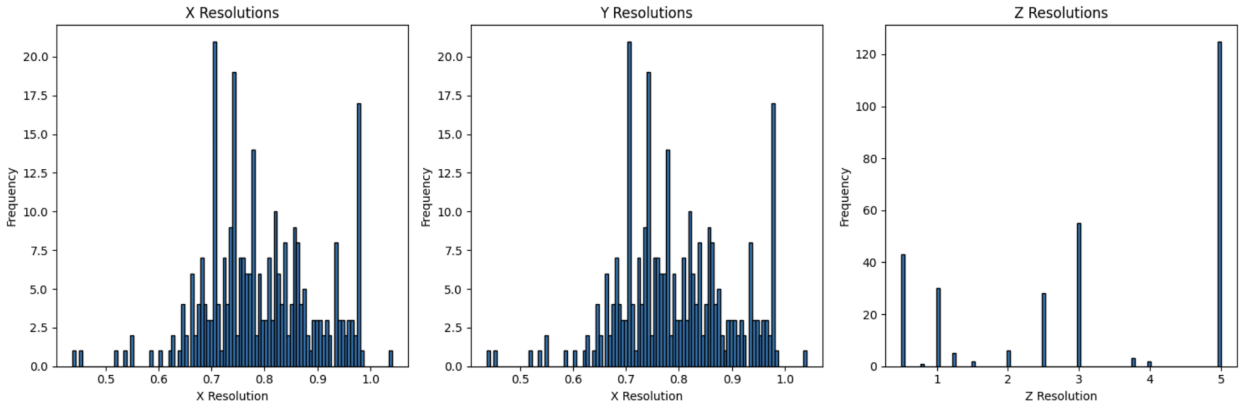
This research project aims to create a model to detect the grade of Clear Cell Renal Cell Cancer (CCRCC) using 3D CT scans of the abdominal region. CCRCC is most commonly classified using the ISUP grade or T-stage, so this project will utilize the joint grade: a positive label is assigned if both the ISUP and T-stage grade are greater than or equal to 2.

Data

The data was taken from the 2021 Kidney and Kidney Tumor Segmentation challenge (KiTS21), consisting of 300 3D CT scans of "patients who underwent partial or radical nephrectomy for suspected renal malignancy between 2010 and 2020 at either an M Health Fairview or Cleveland Clinic medical center" (2021). Segmentations were applied to each by experts and reviewed by medical trainees. The CT scan images were stored as NIFTI files.

Preprocessing

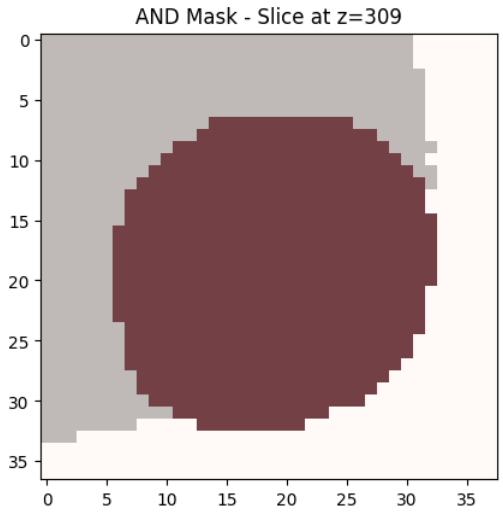
The initial analysis of the dataset revealed inconsistencies in the dataset, specifically in the z dimension and the x, y, and z resolutions.



So, the resolutions were resampled to a (1, 1, 5) isotropic resolution. This step is essential for the model to learn correct patterns in the images using images that are in the same format.

Next, to highlight the section of the image that corresponds to the kidney, only the CT image intensities between -150 to 200 were considered. All other intensities were set to 0. Then, by scaling the intensities to be in the range (-1, 1), the model will be able to perform simple calculations with standardized numbers.

After these preprocessing steps, the data size for each patient was still too large to enter into the model for good efficiency. Through visualizing the modified 3D CT scan in 3D Slicer, it was clear that a large portion of the scan didn't include the kidney and tumor regions so we decided to crop for the tumor region. For each mask, we found the minimum and maximum x, y, and z coordinates of the tumor region. Then, by cropping the image using those dimensions with a small buffer, we were able to find the smallest cuboid region that completely encompassed the tumor region. The image below depicts a slice of the cropped mask with the tumor region highlighted in red and the kidney highlighted in gray.



We used these cuboid coordinates to crop the original image for only the useful data that corresponds to the tumor.

Finally, we rescaled both the cropped image and the full mask to (112, 112, 112) to be compatible with our model. These represent the two channels used in the input of the model.

PACE Cluster

We first tried to run our model locally on Google Colab with a GPU but quickly realized that we did not have enough compute power. Learning that Dr. Shiradkar had access to credits on the PACE cluster, I attended a few workshops to understand how to interact with it. I created a

sbatch file requesting an Nvidia V100 GPU for our tasks, varying the time allocation depending on the number of epochs I was running the model for. Setting up an environment on the cluster came with many challenges due to version issues and specific constraints on the packages available.

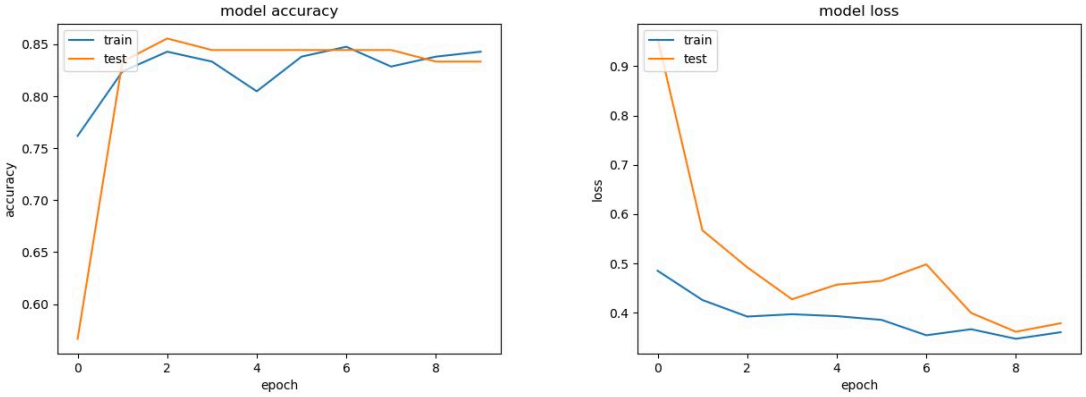
Model

We began with a Decision Tree Classifier and a 2D Resnet, but soon decided to use a 3D Resnet to best take advantage of the 3-dimensional relationship between slices of our 3D scans. The 3D Resnet is a powerful CNN with 150+ layers with a skip connection that overcomes the problem of vanishing gradient.

Our dataset of 300 images was first split for the holdout set using 30% of the cases. Then, the remaining 70% was split into a 70% training set and 30% test set. To reflect the size differences in the training and test sets, a training batch size of 8 and a test batch size of 6 was used. We used Pytorch's pretrained r3d_18 model, the CrossEntropyLoss criterion, and the Adam optimizer. After some testing, we settled on 0.0001 as the best learning rate for good incremental changes through the epochs. Below are the initial results after running with 10 epochs:

```
Epoch 10/10

 1/14 [=>............................] - ETA: 42s - loss: 0.1633 - binary_accuracy: 0.9375
 2/14 [===>..........................] - ETA: 39s - loss: 0.1721 - binary_accuracy: 0.9375
 3/14 [=====>........................] - ETA: 45s - loss: 0.2176 - binary_accuracy: 0.8958
 4/14 [=======>......................] - ETA: 38s - loss: 0.2342 - binary_accuracy: 0.8906
 5/14 [=========>....................] - ETA: 33s - loss: 0.3154 - binary_accuracy: 0.8500
 6/14 [===========>..................] - ETA: 28s - loss: 0.3464 - binary_accuracy: 0.8438
 7/14 [=============>................] - ETA: 24s - loss: 0.3367 - binary_accuracy: 0.8482
 8/14 [===============>..............] - ETA: 19s - loss: 0.3153 - binary_accuracy: 0.8594
 9/14 [=================>............] - ETA: 16s - loss: 0.2952 - binary_accuracy: 0.8750
10/14 [===================>..........] - ETA: 13s - loss: 0.3259 - binary_accuracy: 0.8625
11/14 [=====================>........] - ETA: 10s - loss: 0.3218 - binary_accuracy: 0.8636
12/14 [=======================>......] - ETA: 6s - loss: 0.3615 - binary_accuracy: 0.8438
13/14 [=========================>....] - ETA: 3s - loss: 0.3532 - binary_accuracy: 0.8462
14/14 [==============================] - ETA: 0s - loss: 0.3605 - binary_accuracy: 0.8429
14/14 [==============================] - 53s 4s/step - loss: 0.3605 - binary_accuracy: 0.8429 - val_loss: 0.3787 - val_binary_accuracy: 0.8333
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])
```



The initial results seemed very promising with a high accuracy and low loss. However, when we looked into what the model was actually doing, we realized that it predicted a negative class (0) for almost every case in the test set. This happened because our dataset included 250 cases from the negative class and only 50 from the positive class.

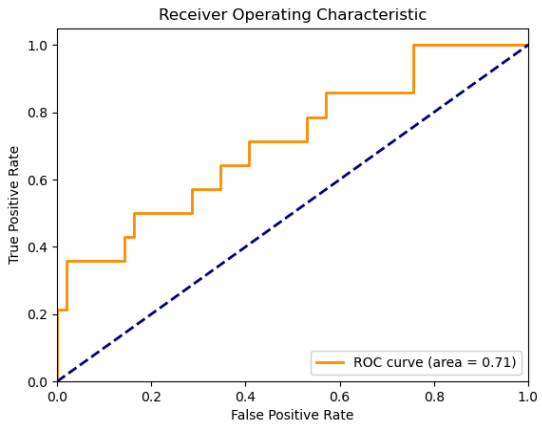
Due to this bias in the model, we attempted a series of changes to try to improve it. First, a dropout layer with dropout rate of 0.5 was added in the 3D Resnet 18 model to prevent overfitting. By setting the parameter 'weight_decay' in the optimizer, a regularization term is added to the optimization process to penalize large weights and further prevent overfitting. We also added layer freezing up to layer 3 since it can be useful when applying a pre-trained model to a new task. I then scaled the class weights so that the minority positive class can be better represented/considered in the model by assigning the weight of a class to the square root of the size of the inverse class:

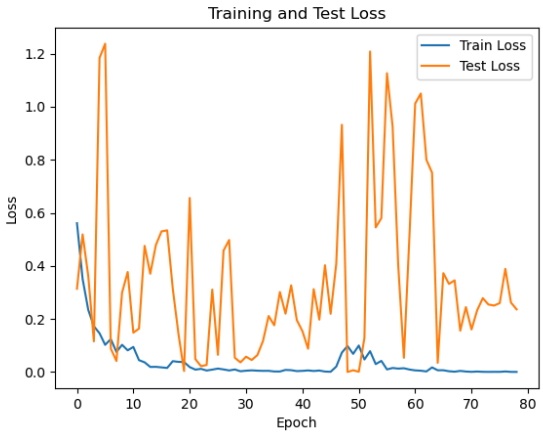
```
class_weights = [sum(class_counts) / c**0.5 for c in class_counts]
```

Finally, early stopping was implemented to prevent overfitting due to extra epochs after the model finds a decent local minimum for the loss.

For better understanding of the performance, I plotted other statistics such as sensitivity, specificity, and the f1 score as shown in the results below. The model was run with 100 epochs and early stopping with patience 30.

```
Epoch 74/100, Train Loss: 0.0008, Test Loss: 0.2546, Train Accuracy: 1.0000, Test Accuracy: 0.7937, F1 Score: 0.3810, Precision: 0.5714, Recall/Sensitivity: 0.2857, Specificity: 0.9388
Epoch 75/100, Train Loss: 0.0010, Test Loss: 0.2507, Train Accuracy: 1.0000, Test Accuracy: 0.8095, F1 Score: 0.4000, Precision: 0.6667, Recall/Sensitivity: 0.2857, Specificity: 0.9592
Epoch 76/100, Train Loss: 0.0009, Test Loss: 0.2599, Train Accuracy: 1.0000, Test Accuracy: 0.8095, F1 Score: 0.4545, Precision: 0.6250, Recall/Sensitivity: 0.3571, Specificity: 0.9388
Epoch 77/100, Train Loss: 0.0023, Test Loss: 0.3889, Train Accuracy: 1.0000, Test Accuracy: 0.7937, F1 Score: 0.4348, Precision: 0.5556, Recall/Sensitivity: 0.3571, Specificity: 0.9184
Epoch 78/100, Train Loss: 0.0008, Test Loss: 0.2618, Train Accuracy: 1.0000, Test Accuracy: 0.8254, F1 Score: 0.4762, Precision: 0.7143, Recall/Sensitivity: 0.3571, Specificity: 0.9592
Epoch 79/100, Train Loss: 0.0007, Test Loss: 0.2365, Train Accuracy: 1.0000, Test Accuracy: 0.8254, F1 Score: 0.4762, Precision: 0.7143, Recall/Sensitivity: 0.3571, Specificity: 0.9592
Stopping early
Hold-out Test Set: Accuracy: 0.8556, F1 Score: 0.0000, Precision: 0.0000, Recall/Sensitivity: 0.0000, Specificity: 1.0000, AUC: 0.6573
```





Despite once again seeing good accuracy and a decent AUC score, it is clear that the model continues to be biased as shown by the low sensitivity and high specificity. In other words, the model is able to predict a high percent of negative classes correctly but a low percentage of positive classes. The inconsistent test loss shown above is also worrying and depicts that nothing is really being learned by the model.

In the future, we plan on taking a step back to better understand the cause of the bias. It seems that no amount of minor adjustments such as adding dropout and regularization will significantly improve the bias in the model. The first step for this is plotting the Grad-CAM map to highlight the sections of the image being most considered by the model. We will then possibly return to the preprocessing or model training steps to address the bias.

Through this project, I was able to dip my toes in the deep learning development process, from the many preprocessing data steps to building a model and understanding hyperparameters that hopefully improve the model. While this project hasn't yet produced an accurate and unbiased model that is able to classify renal cancer grades, I think I learned an important lesson in machine learning—the importance of data preprocessing and a balanced and thorough dataset. I am looking forward to the next few weeks, where I will hopefully be able to understand the specific reasons behind the bias and hopefully develop an improved model.

References

Brownlee, Jason. “A Gentle Introduction to Dropout for Regularizing Deep Neural Networks.” MachineLearningMastery, 6 Aug. 2019, machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.

“Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value.” PSU Online Stat, online.stat.psu.edu/stat507/book/export/html/692. Accessed 5 May 2024.

“The 2021 Kidney and Kidney Tumor Segmentation Challenge.” Kits21, kits-challenge.org/kits21/.