Prostate Cancer Grade Assessment

Yi Hou
Stanford University
yihou@stanford.edu

Abhishek Bharani
Stanford University
abharani@stanford.edu

Neeraj Mathur
Stanford University
mathurn@stanford.edu

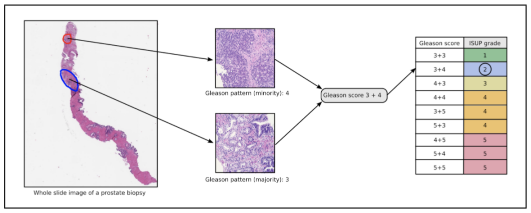
Abstract

Recent studies have shown that deep learning systems can achieve pathologist-level performance, however, these systems were not tested with multi-center datasets at scale [3]. We present a computational approach based on deep convolution neural networks for prostate cancer histology image classification using the multi-center dataset on Gleason grading, applying image segmentation using residual connections to identify cancerous tissues.

1. Introduction

With more than 1 million new diagnoses reported every year, prostate cancer (PCa) is the second most common cancer among males worldwide that results in more than 350,000 deaths annually. The key to decreasing mortality is developing more precise diagnostics. Diagnosis of PCa is based on the grading of prostate tissue biopsies. These tissue samples are examined by a pathologist and scored according to the Gleason grading system. We are looking to develop models for detecting PCa on images of prostate tissue samples, and estimate severity of the disease using the multi-center dataset on Gleason grading.

The grading process consists of finding and classifying cancer tissue into so-called Gleason patterns (3, 4, or 5) based on the architectural growth patterns of the tumor (Figure 1). After the biopsy is assigned a Gleason score, it is converted into an isup grade on a 1-5 scale [3]. The Gleason grading system is the most important prognostic marker for Prostate Cancer, and the isup_grade has a crucial role when deciding how a patient should be treated. The deep learning systems have shown some promise in accurately grading PCa, however these systems were not tested with multi-center datasets at scale. Our work here will improve on these efforts using the most extensive multi-center dataset on Gleason grading yet.


Gleason pattern (minority): 4
Gleason score 3 + 4
Gleason pattern (majority): 3
Whole slide image of a prostate biopsy

Gleason score	ISUP grade
3+3	1
3+4	2
4+3	3
4+4	4
3+5	4
5+3	4
4+5	5
5+4	5
5+5	5

Figure 1. The most common (blue outline, Gleason pattern 3) and second most common (red outline, Gleason pattern 4) cancer growth patterns dictate the Gleason score (3+4 for this biopsy), which is converted into an isup_grade (2 for this biopsy). Biopsies not containing cancer are represented by an isup_grade of 0.

2. Related Work

Some of the deep learning algorithms have recently been shown to exceed human performance in detecting various types of cancer. Besides other categories, we have observed that Image Classification [22] [11] [16] [21] [2], Medical Object Detection [12] [9] [19], and Transfer Learning [8] [7] are primarily being used in detecting various kinds of cancer and these techniques are also used on different types of pathology images.

1) Image Classification One of the notable work [5] using image classification techniques where the authors have outlined the development of a CNN that matches the performance of dermatologists at multiple diagnostic tasks. Here the authors have demonstrated the classification of skin lesions, trained end-to-end from images directly, using only pixels and disease labels as inputs. They have tested the performance of the model against 21 board-certified dermatologists and found that the model has performed at par.

2) Medical object detection: One of the study [12] using object detection techniques which couple of years ago set the state-of-the-art performance had proposed a Computer Aided Detection (CAD) system based on object detection framework, Faster R-CNN. This system detects and classifies malignant or benign lesions on a mammogram without any human intervention. This method set the state-of-the-art classification performance on the public INbreast database.

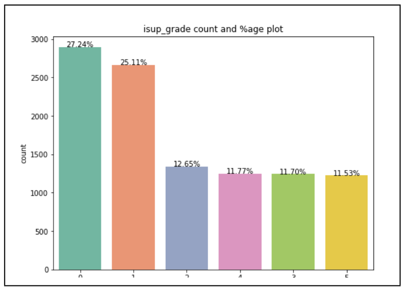


Figure 2. Distribution of isup grade across dataset

3) Transfer learning: The study [8] performed back in 2016 analyzed the effect of initializing networks with pre-training on the ImageNet dataset and then fine-tuning on mammography images. The proposed end-to-end model to classify pre-detected breast masses from mammograms has provided state-of-the-art results of the time. More recently, the study [7] used pre-trained deep CNN feature extractors to improve the classification accuracy and accelerate the learning process. Here five deep CNN architectures are used as features extractors, namely InceptionV3, Inception-ResNetV2, Xception and two VGGNet models for classification of breast cancer.

3. Dataset and Features

The dataset consists of around 10,616 whole-slide images of digitized H&E-stained biopsies originating from Radboud University Medical Center and Karolinska Institute. This multi-center (multiple center across these two institutions) test set is graded (labeled) by expert uropathologists. For tile-based inputs, we split our dataset into 10,000 training and validation images and 516 biopsy-labelled test images (we removed 100 images as they don't having corresponding masks).

Each image contains one, or in some cases two, thin tissue sections cut from a single biopsy sample. Prior to scanning, the tissue is stained with haematoxylin & eosin (H&E). The samples are made up of glandular tissue and connective tissue. The glands are hollow structures, which can be seen as white "holes" or branched cavities. The appearance of the glands forms the basis of the Gleason grading system. The glandular structure characteristic of healthy prostate tissue is progressively lost with increasing grade.

We had a total of 10,600 biopsy images with distribution of isup grade as shown in Figure 2. Input image was in multi-page tiff format with different magnification rate of 16x, 8x and 4x, in order to pre-process whole slide images we choose images with 8x resolution and applied different pre-processing to get best results. First, we created 16-tile images from one biopsy image. Second, we removed the white space portion of image and generated a glued image

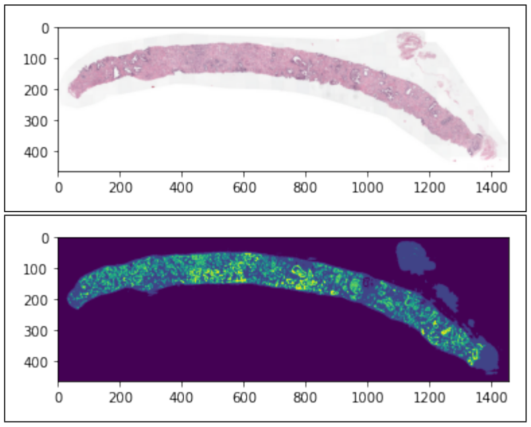


Figure 3. Sample Image and mask for isup grade 5 prostate cancer

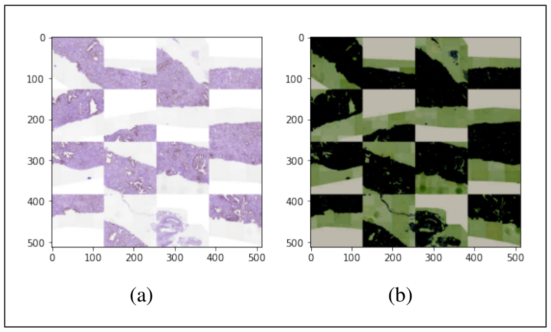


Figure 4. Cropped biopsy image of isup 5 down-sampled by 16 (a) before normalization (b) after normalization

using sliding window patch of 512^2 pixels with 50 percent overlap. For U-Net model to created one-hot encoding of target mask based on different label value for each class of tissue. Apart from this, calculated the mean and standard deviation and applied to each image, random-resize and crop, rotation, squeezing to generate augmented images.

3.1. Data Pre-processing & Feature Extraction

Before splitting the dataset to train/val/test sets, we checked the resource from two data centers. We first unified the Gleason score notations as Radbound uses "negative" for Gleason score category (0+0). Then, we corrected the mapping issue between Gleason score and isup_grade. Finally, we found some image IDs having empty slides and we pruned them out. For each isup_grade, we reserved 20% for the test dataset and then split the rest by 4:1 for training and validation usage.

The biopsy image dimensions are quite large (typically between 5000 and 40000 pixels in both x and y). Each slide has 3 levels, corresponding to a downsampling rate of 1, 4 and 16. The dimensions of each level differ based on the original image. Biopsies can be in different rotations, which has no clinical value and is only dependent on how the biopsy was collected in the lab. There are also noticeable color differences between the biopsies, which is caused by different laboratory procedures.

The first challenge is how to preprocess these large images. Dataloader in PyTorch has a lot of good features including batching, shuffling and loading the data in parallel. To utilize these features, we customized the Dataloader class to load and re-scale an image on the fly. But such dataloader hit a problem during the training. To generate a batch of 4 images, it takes 7 seconds. We analyzed the elapsed time of image loading and resizing, we found that the image resizing took around 1 or 2 seconds, which varied by the original image dimensions. To alleviate the effect of resizing, we picked the lowest level of each slide, resized it to 512x512x3 and saved in PNG format. By loading those images of smaller dimensions, the data batching can be done within one second.

Given the customized dataloader, we started with a baseline two-layer FC network by feeding a full image. The validation accuracy plateaued after 4 epochs and always stayed at 27 (i.e. 464 out of 1692 is correctly classified). By analyzing those 464 samples, all labels of "isup 0" are classified. Furthermore, we tried different combinations of learning rates and batch sizes, but they all had the same issue. One theory is that each image contains a large fraction of empty, which makes FC network hard to extract valued information for training. So we need a way to zoom into each image and crop a patch which contains more biopsy information. We zoomed in images by slicing the original images to multiple patches by padding [17]. As patches with more biopsy parts have less pixel sum values. We sorted those patches and took the smallest patch as the representative of each image. The image statistics were also collected during image cropping, which is used to zero-center and normalize cropped images.

For ResNet50, we pre-processed the input image using sliding window of 512 pixels with 50 percent overlap. We use library sliding window to get the patches, after we generate the sliding window patches we remove the white space by looking at pixel values and only keeping the tissue portion of the image. We generate more images using the sliding window fig 5. Since there is considerable class imbalance, we applied weighted random sampling with train-valid split of 0.25 and .10 for test set.

For U-Net, we pre-process the target mask into one-hot encoded mask as shown in (Figure 6)

4. Methods

4.1. Baseline

We used the two-layer FC network (50 * 100 with softmax) as our base-line model. We first used the whole biopsy image (i.e. 3x512x512) as the input. With the learning rate of $3e-3$ and SGD optimizer, we got the loss value of 1.82, which is a little worse than the random guessing result of 1.79. Though the validation accuracy is 27%, only "isup 0" get predicted correctly. We also tried Adam optimizer and smaller learning rate with step scheduler. But the training loss would not become better than 1.79.

4.2. ResNet18 Model

The flattened 1×512 output features from the final residual block is fed to two different headers according to the input formats. For the concatenated-tile input in figure 7 (a),

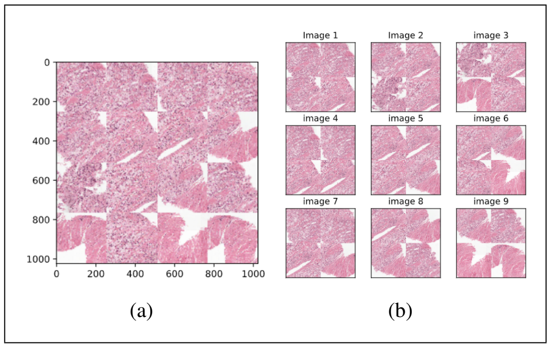


Figure 5. Image Generated using sliding window (a) Input image (b) sliding window patches generated

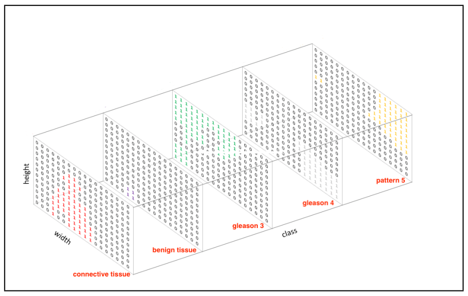


Figure 6. one-hot encoding for each Gleason pattern and two tissues

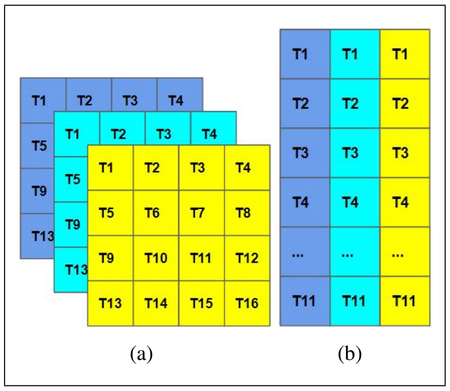


Figure 7. (a) Input Batch of Concatenated Tiles (b) Input Batch of Streamed Tiles

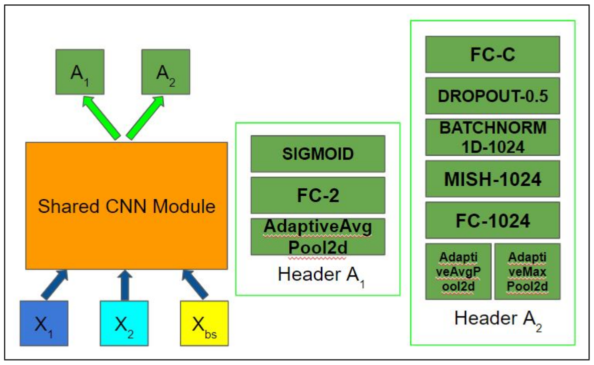


Figure 8. MTL Model & Two Customized Headers

we used the linear layer of dimension 512×6 (num classes $\bar{6}$). For the streamed-tile input in figure 7 (b), we added a customized header which expanded to a dimension 1×1024 and then went through a extra batch norm and dropout layer of dimension 1×512. The six logits by the last FC layer. Each training batch was preprocessed as following:

1. Load tile sequences randomly from a mini-batch of isup-weighted images with the pixel value range $0 - 255$
2. Randomly flip and rotate each tile with probability .5 and degree less than 15. The blank space are filled with a pixel value of 255
3. Shuffle the tile order per each image ID
4. Normalize using the pixel-wised mean and standard deviation. Since concatenated-tile and streamed-tile inputs use different number of tiles, their means and stds are calculated separately.

The ResNet18 model[10] we used is adapted from He et. al's paper [6], obtained from Pytorch model zoo . It contains an additional conv layer at the stem and 4 residual blocks of two 3×3 conv layers. The first layer is additionally followed by max pooling, while the last also includes global average pooling. In the residual blocks that do not change the dimensionality of inputs, identity shortcut connections are used.We also had to limit batch size and the number of tiles after we did parallel cross-validation. The ResNet18 is straightforward but sensitive to the bias in the data distribution. Next, we propose a two-stage model, in which ResNet18 is used as feature extraction but shared by multiple linear headers.

4.3. Multi-Task Learning Model

In the PCa dataset, negative samples (isup_grade 0 and 1) accounts for 52% . While, it is more important to get positive samples (isup_grade 2, 3, 4 and 5) predicted correctly. Such imbalanced data distribution affects the predication accuracy for the high risk group. To alleviate this effect, we came up two-stage classification. The first-stage will do a coarse classification and identify whether a given sample has a cancer or not. Based on that result, the second-stage will do a fine classification to predict the risk level for the cancerous sample. This two-stage model can be implemented via multi-task learning (MTL) by having two tasks. We start off simultaneously having one neural network (i.e. ResNet18) learn two tasks at the same time. And then sum over the losses of two tasks. We use the so-called hard parameter sharing architecture[15]. There is a shared lower module B (i.e. ResNet18) that encodes the feature representation for all tasks. Each task has a specific output prediction headerA_i. To train the neural network, loss function is sums over losses from each individual task $\sum_{i=1}^{k} l_i(\hat{Y_i}, Y_i)$. In our PCa classification, we have X_i, Y_i as input for $i = 1, \ldots, k$, where k is the size of a mini-batch, X_i corresponds to a preprocessed biopsy image, and Y_i corresponds to the isup_grade of X_i. For our two-stage classifier 8, two task headers, A_1 and A_2, corresponding to the binary classification in stage-1 and the multi-classification in stage-2. For the output of the header A_1, $\hat{Y_{i1}}$, we measure the focal loss (FL)[20] between $\hat{Y_{i1}}$ and Y_i. Similarly, we apply the cross entropy (CE) loss between $\hat{Y_{i2}}$ and Y_i where $\hat{Y_{i2}}$ is the output of the header A_2. To train the whole model, we combine these two losses,

$$L = \sum_{i=1}^{k} (FL_i(\hat{Y_{i1}}, Y_i) + CE_i(\hat{Y_{i2}}, Y_i)) \tag{1}$$

4.3.1 Focal Loss

In MTL ResNet18, we applied the focal loss to address the imbalanced data distribution on stage-1. One notable property of FL is that examples that are easily classified ($p_t \gg .5$) incur a loss with non-trivial magnitude. Given most of our input images are negative, these easily classified samples compromise the majority of the loss and dominate the gradient, which makes it hard to differentiate between easy/hard examples. FL can reshape the loss function to down-weight easy examples and thus focus training on hard negatives. More formally, we get FL by adding a modulating factor $(1 - p_t)^\gamma$ to the cross entropy loss, with tunnable focusing parameter $\gamma \geq 0$.

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{2}$$

$$p_t \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases} \tag{3}$$

4.3.2 Kappa Score

For validation and test phase, we evaluate our models in a supervised fashion using quadratic weighted kappa. The kappa score measures the agreement between two outcomes. This metric typically varies from 0 (random agreement) to 1 (complete agreement). In the event that there is less agreement than expected by chance, the metric may go below 0. The quadratic weighted kappa is calculated as follows. First, an $N \times N$ histogram matrix O is constructed, such that $O_{i,j}$ corresponds to the number of isup grade i (actual) that received a predicted value j. An N-by-N matrix of weights, w, is calculated based on the difference between actual and predicted values:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{4}$$

An N-by-N histogram matrix of expected outcomes, E, is calculated assuming that there is no correlation between values. This is calculated as the outer product between the actual histogram vector of outcomes and the predicted histogram vector, normalized such that E and O have the same sum. From these three matrices, the quadratic weighted kappa is calculated as:

$$kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \tag{5}$$

4.4. U-Net

The grading system for gleanson score recognizes three categories: 3, 4, and 5. A biopsy image can have different Gleason patterns, but the score is composed of the two of the most frequently occurring pattern, as judged by the pathologist. The minority or second pattern must account for at least 5 percent of the total area to be included, e.g. if pattern 3 is less than 5 percent, the Gleason score 4 + 3 will instead be 4 + 4. Further, the highest grade should always be part of the score. For example, a biopsy that contains 60% Gleason 4, 37% Gleason 3 and 3 percent Gleason 5 should get a score of 4 + 5 = 9. If we can calculated the number of pixels for each gleanson pattern and calculate percent-wise distribution, we can easily predict the gleanscore mapped to isup-grade.

To predict pixel-wise mask, we applied U-Net [13] convolution neural network for image segmentation. The Gleason pattern grading of prostate images are addressed in our work by generating pixel-level semantic segmentation map. The best well known architecture for this task is U-Net. The input images x are resized during training process to 256^2, this is to avoid memory issues while training the model. Prostate images of cancerous tissues have two different data providers Radboud and Karolinska, both had different labeling mechanism (semantic map had 6 classes for Radboud and 3 classes for Karolinska). Concretely, the defined labels are : Background (non-tissue), healthy and cancerous tissue. We used pixel level soft-max to obtain probability map. A pixel with higher probability for across the classes was assigned the corresponding class. The model architecture used is shown in figure 9. In order to improve performance of U-Net model we used residual blocks as mentioned in the paper [2] . The residual blocks composed of 3 convolution filter of size 3x3 and output of first connection is connected in a skip connection with result of batch-norm, ReLU and two other filters.

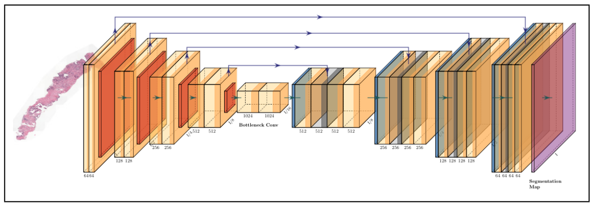


Figure 9. U-Net Architecture Used

4.4.1 Dice Loss

Most commonly used loss function for image segmentation is pixel-wise cross entropy loss , it evaluates the predictions for each pixel and then averages over all pixels, but since our image have imbalanced labels (there is more white area in the image than issue area). Also, each pixel can belong to one of the class labels (out of 6 or 3), we used combination of Dice loss and binary cross entropy loss to loss calculation. Dice loss makes a balance between intersection and union of predicted and target mask. With respect to the neural network output, the numerator is concerned with the common activation between our prediction and target mask, where as the denominator is concerned with the quantity of activation in each mask separately. This has the effect of normalizing our loss according to the size of the target mask such that the soft Dice loss does not struggle learning from classes with lesser spatial representation in an image.It's appropriate for imbalanced dataset and is defined as :

$$Dice\ Loss = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 \sum_i^N g_i^2} \tag{6}$$

where p_i and g_i are one-hot-encoded predicted and target labels(mask) , i denotes one of N classes Background, Healthy or Cancerous tissue.

4.5. Residual U-Net

More recent developments in image models almost use the trick of residual connections, and most of the time, they are just a tweak of the original ResNet. The residual blocks

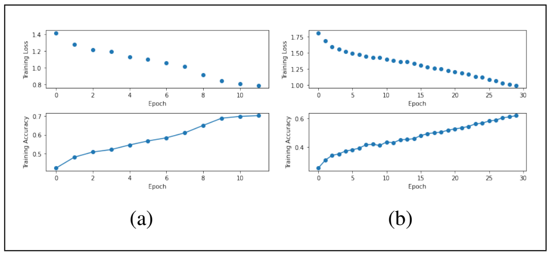


Figure 10. Training loss and accuracy history (a) Concat-ResNet18 (b) Stream-ResNet18

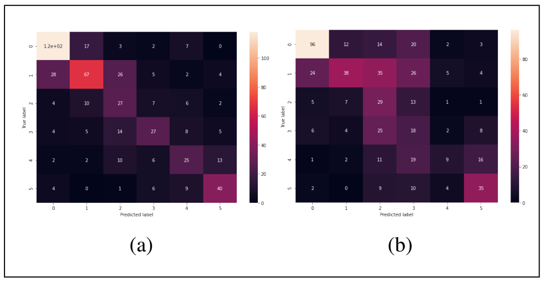


Figure 11. Confusion matrix for 516 test samples with (a) Concat-ResNet18 (b) Stream-ResNet18

is a configuration of convolution filters with skip-additive connections [4]. We replaced every occurrence of conv(x) with x + conv(x), where conv is the function from the previous chapter that adds a second convolution, then a ReLU as mentioned in paper [2]. The identify mapping configuration is used. We optimized using same dice loss used in U-Net model.

5. Experiments/Results/Discussion

Table 1. 4-Fold CV Evaluation Results

Model	*Batch Size*	*# tiles*	*Avg Val Acc*	*epoch time (min)*
Concat ResNet18	16	16	0.6014	93.5
Stream ResNet18	16	11	0.4931	21.8
MTL ResNet18	14	12	0.4645	24.8

Table 2. Test Accuracy

Model	*Macro Acc (isup-2,3,4,5)*	*Kappa Score*
Concat ResNet18	0.5021	0.4993
Stream ResNet18	0.3840	0.4586
MTL ResNet18	0.4430	0.8543

5.1. Concat/Stream ResNet18

Given 16 tiles were cropped from each biopsy image, we experimented those tiles in two different ways. And the linear layer of ResNet18 needed to be modified correspondingly. In the concatenate format, we utilized all 16 tiles of an image. As each tile is $128 \times 128 \times 3$, we concatenated them by four rows and four columns[7]. Then the concatenated image is $512 \times 512 \times 3$. The concatenation happens inside the dataset function. But we found that this became the bottleneck during the training as it took about ten seconds to load a batch of 16 samples. To speed up the batch loading, we saved concatenated images to the drive in advance. But this makes us lose the flexibility when doing transformation, as we can only transform per image instead of per tile. And we also saw 7% performance drop by using pre-concatenated images. To alleviate concatenation bottleneck without losing transform flexibility, we came up with the streamed-tile format. Therefore, we streamlined the tiles [7] and passed them through the conv layer. Then we concatenated all tile features from one image before feeding to the linear header. Now we can use any number of tiles (i.e. $[1, 16]$) without the constraint of being a square shape. Furthermore, we truncated the whole FC layers and replaced with seven new layers including AdaptiveConcatPool2d [1], Mish, BatchNorm1d and Dropout ($p_{drop} = .5$).
Both ResNet18 were trained with Adam with a learning rate of $1e-4$. We used the learning scheduler of ReduceLROnPlateau with a factor of 0.1 and patience of 2. For streamed ResNet18, we picked the number of tiles to 11. We used cross entropy loss for both models. Besides, we experimented with the SGD optimizer and learning rate of $1e-3$. But a learning rate larger than $1e-3$ caused the loss value to vibrate or blow up. And it took longer time for SGD to converge or achieve the similar performance as Adam. Considering that our training set has more negative samples, we applied weighted sampler to alleviate such imbalance effect. Table 1 shows that training concat-Resnet18 (i.e. using concatenated-tile input) takes four times longer than stream-Resnet18 (i.e. using stream-tile input). By running 4-fold cross validation, we found that concat-Resnet18 stopped learning after epoch 12 and could achieve the best validation accuracy of 60%. Figure 10 is the loss and accuracy by training on all samples for 12 epochs. Figure 11 is the confusion matrix of concat-ResNet18 on 516 test samples. We can see that isup_grade 0 mostly got classified correctly, which was expected according to the imbalanced data distribution. The model was bias to predicting negative samples, because most mis-classified positive samples are located at the bottom-left triangle. For example, 10 samples of isup_grade 4 were predicted as isup_grade 2, and 14 samples of isup_grade 3 were wrongly classified as isup_grade 2. Though we applied weighted sampler to address the imbalance training data, it still had an impact on our train-

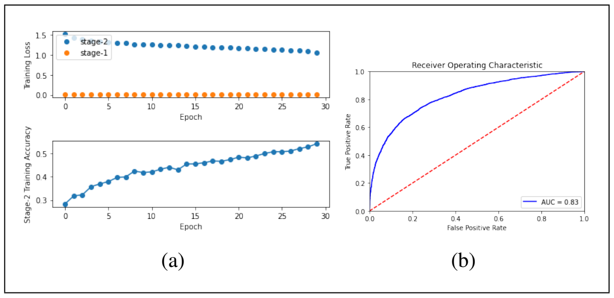


Figure 12. (a) Stage-2 loss and accuracy with MTL ResNet18 (b) Stage-1 RoC & AUC with MTL ResNet18

ing model. The kappa score of 0.49 means that most samples are predicted correctly and the mis-classified labels are close to the true label. This is highlighted on the diagonal of the confusion matrix.
Training on stream-ResNet18 is much faster than concat-ResNet18. During 4-fold cross validation, the validation accuracy plateaued after 15 epochs. Figure 10 (b) shows the curve of training loss and accuracy on all training samples for 30 epochs. The macro accuracy of stream-ResNet18 is lower because it gets less negative samples predicted correctly. Though stream-ResNet18 has a lower test accuracy, we still get a kappa score of 0.45 close to concat-ResNet18's. As kappa score is calculated only on positive samples, it indicates that the both models get similar accuracy in terms of prediction of positive samples. And from the confusion matrix Figure 11 (b), we can see that most correctly predicted isup_grades are along the diagonal. Stream-ResNet18 has a similar bias issue as concat-ResNet18, where it tends to mis-classify positive samples to smaller isup_grades. But the similar kappa scores implies that it is enough to use less number of tiles to judge positive samples. This also reflects the steps that a pathologist might carry out to obtain a clinical impression, as only a few of tiles contain the cancerous tissue. Another reason why stream-ResNet18 get less accuracy is that we pass tiles through ResNet18 individually where the tissue contents on the margin area are used less compared to the concatenated format. But stream-ResNet18 is a good alternative in terms of training acceleration.

5.2. MTL ResNet18

According to [5], the partitioning algorithm has demonstrated to be more effective in skin diseases classification than one trained directly on target classes. We use two stage classifier, first distinguishes cancer v/s non-cancer, and then we give a perceived severity for each predicted cancer case. In both stages, we still utilized ResNet18 as our base CNN module. In stage one we use sigmoid layer and in stage two we use customized header for each isup grade 2,3,4,5 (instead of 6 isup grades).

Considering the long training time, we used the same stream input as we used in stream-ResNet18. This vanilla architecture showed good training results, where the validation accuracy of stage one is close to 80% and the stage two could also achieve 53%. But we found a performance drop after we cascaded the results from two stages, which is worse than the results of stream-ResNet18 where we did direct classifications. We reasoned that there were two situations causing this bad cascaded performance. One is that we might predict correctly in the first stage but fails the prediction on the second stage. The other is we get correct prediction in stage two but fails on stage one. The two situations increase the failure rate of the predictions. The root cause is that we train two models individually and, hence, the outputs from two stages are nearly independent.
To create the association between outputs from two stages, we took the advantage of the MTL architecture and merged the CNN modules of two classifiers into shared one. In order to get outputs for two stages, we attached two headers to the shared CNN module and trained them simultaneously. Same as stream-ResNet18, we trained MTL ResNet18 using streaming tiles as the input. MTL ResNet18 was trained with Adam with a learning rate of 1e-4 as larger learning rate caused the fluctuation of the training loss.The learning scheduler of ReduceLROnPlateau was used with a factor of 0.5 and patience of 1. We also experimented with weight decay on both Adam and SGD optimizer, but they resulted in worse validation accuracy. For the outputs from stage one, focal loss was used with gamma value of 2. For stage two's outputs, we applied cross entropy to compute the loss value.
Since MTL-ResNet18 also used tiles in a stream way, it inherits the training time advantage. We trained the model on all training samples for 30 epochs. Figure 12 (b) is the RoC metric of the stage-1 header, i.e. the classification between positive (cancer) and negative (no-cancer). Before the false positive rate (FPR) of .3, the true positive rate increases very fast. But after FPR of .3, it needs t sacrifice more FPR to achieve less improvement on true positive rate (TPR). To compare the performance of MTL-ResNet18 with stream-ResNet18, I picked a threshold value of .33 to make both models have similar FPR value which we emphasis on. In figure 13 (a), we can see that both models have similar performance in terms of true negative rate (TNR) and false negative rate (FNR). But MTL resolved the bias issue much better as less positive samples are miss-classified to lower grades. And it gets more samples with higher isup grades predicted correctly, which is comparable to concat-ResNet18's. The kappa score of 0.85 indicates that, though it gets some positive samples predicted wrongly, its predictions are very close to the ground truth. Figure 13 (b) is the confusion matrix with a threshold value of .43. It shows an-

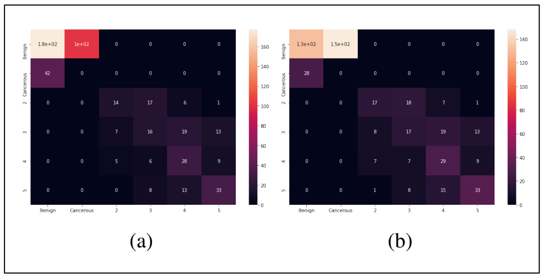


Figure 13. MTL ResNet 18 Confusion matrix for 516 test samples (a) threshold=.43 (b) threshold=.33

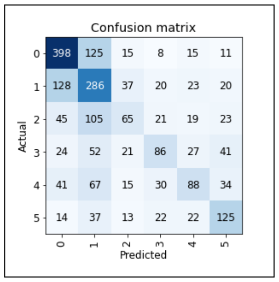


Figure 14. ResNet50 Confusion Matrix

other advantage than stream-ResNet18 that we can sacrifice FNR for higher TNR and less FNR with less impact on TPR by increasing threshold value.

5.3. ResNet50

Initially, last layer of the pre-trained model was randomly initialed and then fine-tuned for our dataset for few epochs. We then unfreeze the model to step over all the parameters of the model.To find out ideal learning rate for our training, we used learning rate finder [18] by running one epoch with very low learning rate (like $10e^{-8}$) and changed it at each mini-batch until it reaches a very high value. We take the learning rate where there is most steep slope. We used a batch-size of 32 images with each image of size 224^2 generated by removing white spaces in the image. After 25 epochs we obtained below results figure 14 plotted on confusion matrix. The most confusion is among isup grade 0 and 1 but model does well on other isup grades which are cancerous.

Table 3. Tests Accuracy Sample size 506 Images

Model	***Test Acc***
ResNet50(without glued image)	0.4915
ResNet50(with glued images)	0.6725
U-Net(resized 256^2 images)	0.6932

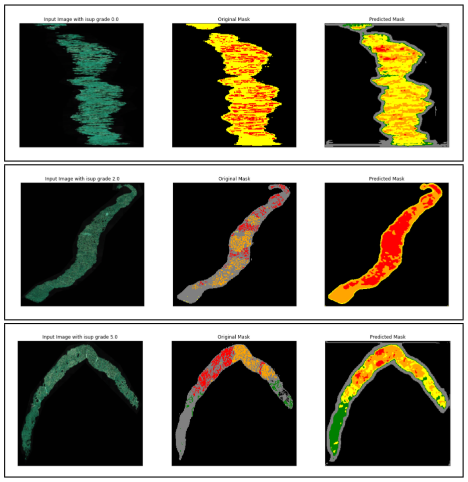

Input Image with isup grade 0.0
Original Mask
Predicted Mask
Input Image with isup grade 2.0
Original Mask
Predicted Mask
Input Image with isup grade 5.0
Original Mask
Predicted Mask

Figure 15. Segmented mask generated from U-Net model for three different isup_grades.

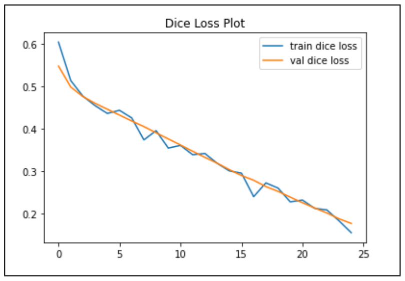


Figure 16. Train and Validation Dice loss

5.4. U-Net and Residual U-Net

In this experiment, we tried to learn segmentation map where each pixel contains a class label. Input image was of size 256^2 , by one-hot encoding the class labels we created output map for each possible classes. We collapsed the segmentation map by taking argmax across each pixel vector and by overlaying the predicted mask on input image to identify regions of cancerous tissue. Predicted mask on three different isup grade are shown in figure 15 , the predicted mask v/s original mask. For most of the isup grade , the predicted mask contains exact same number of cancerous pixels as in target pixels. We used learning rate finder [18] to pick the best learning rate for our training. Using short skip connections mentioned in [2] allow us to convergence faster when training and allow for deeper models to be trained.

6. Conclusion/Future Work

Concat-ResNet18 was easier to implement as the input is square in dimension size, and we don't need to customize the existing linear header. Stream-ResNet18 can resolve the slow data loading issue occuring on concat-ResNet18.

Compared to concat-ResNet18, stream-ResNet18 has less accuracy as it uses tile images less effectively. While, both concat-ResNet18 and stream-ResNet18 suffers from the bias impact caused by the imbalanced data distribution. MTL-ResNet18 demonstrates that it not only resolve the training time headache, but also alleviate the bias effect from the training samples. And MTL-ResNet18 can achieve comparable accuracy to concat-ResNet18 in terms of positive samples. Its two-header architecture add an extra nob for user to trade off TNR, FNR and FPR without impacting TPR. Figure 12 shows that it is hard to train such model as the loss curve of stage-2 decreases very slow and the curve of stage-1 is almost flat. Additional future work could combine the provided mask images and learn on target tiles directly. We might also try focal loss on stage-2 of MTL architecture to improve the micro accuracy for each positive isup_grade.

Transfer learning using Resnet50 gave good results and generating more augmented images does help model generalize better. The biopsy images are in random shape with un-fixed position of cancerous tissue. The processing time to generate each 20 augmented images was approximately 2.5 mins and it was a challenge given gcp resource issue. Using U-Net, we were able to extract the mask with about 70 percent accuracy. There were some issues building up Residual U-Net model initially and we disucssed it office hours. Also, the training data requires more cleaning as some gleanson score calculated does not match the labeled gleason score provided. There are some images labeled with gleason score $5 + 3$ but there is no gleason 5 pattern present.

7. Contributions & Acknowledgement

Abhishek contributed on

- Transfer learning for resnet50, glued image pre-processing and augmentation.
- Implementing the new residual U-NET paper [2] and U-NET [14]
- We U-NET we looked at the code at git repo : https://github.com/usuyama/pytorch-unet but residual U-NET we actually implemented the paper ourself.

Yi and Neeraj contributed on

- Data collection and pre-processing
- Baseline training and evaluation
- Concat-ResNet18, Stream-ResNet18 and MTL ResNet18
- Git repo : https://github.com/yh36/CS231nProstateCancer.git

We would like to thank CS231N instructors Fei-Fei Li, Ranjay Krishna and Danfei Xu , as well as TAs for their general guidance and support. We would also thank Google Cloud for granting us education credits and GPU instances for our training use. For project guidance, we would like to thank Jon Braatz.

References

[1] Custom fastai modules, 2019. `https://github.com/fastai/fastai/blob/master/fastai/layers.py#L176`.

[2] A. C. F. L.-M. V. N. Amartya Kalapahar, Julio Silva-Rodríguez. Gleason grading of histology prostate images through semantic segmentation via residual u-net. *CoRR*, abs/1512.03385, 2020.

[3] K. Challenge. Prostate cancer grade assessment (panda) challenge, 2020. https://www.kaggle.com/c/prostate-cancer-grade-assessment.

[4] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal. The importance of skip connections in biomedical image segmentation. *CoRR*, abs/1608.04117, 2016.

[5] K. B.-N. R. e. a. Esteva, A. Dermatologist-level classification of skin cancer with deep neural networks. Technical Report 1476-4687, Nature, Feb 2017.

[6] S. R. J. S. Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. Technical Report arXiv:1512.03385, ArXiV, Dec 2015.

[7] S. H. Kassani, P. H. Kassani, M. J. Wesolowski, K. A. Schneider, and R. Deters. Breast cancer diagnosis with transfer learning and global pooling. *arXiv preprint arXiv:1909.11839*, 2019.

[8] D. Lévy and A. Jain. Breast mass classification from mammograms using deep convolutional neural networks. *arXiv preprint arXiv:1612.00542*, 2016.

[9] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[10] P. Napoletano, F. Piccoli, and R. Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors (Basel, Switzerland)*, 18, 01 2018.

[11] K. Nazeri, A. Aminpour, and M. Ebrahimi. Two-stage convolutional neural network for breast cancer histology image classification. In *International Conference Image Analysis and Recognition*, pages 717–726. Springer, 2018.

[12] D. Ribli, A. Horváth, Z. Unger, P. Pollner, and I. Csabai. Detecting and classifying lesions in mammograms with deep learning. *Scientific reports*, 8(1):1–7, 2018.

[13] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[14] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*

(MICCAI), volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[15] C. R. Sen Wu, Hongyang R. Zhang. Understanding and improving information transfer in multi-task learning. Technical Report arXiv:2005.00944 [cs.LG], ArXiV, May 2020.

[16] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride, and W. Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific Reports*, 9(1), Aug 2019.

[17] R. Singh. Panda - eda + better visualization+simple baseline, 2020. https://www.kaggle.com/rohitsingh9990/panda-eda-better-visualization-simple-baseline.

[18] L. N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015.

[19] N. Tomita, B. Abdollahi, J. Wei, B. Ren, A. Suriawinata, and S. Hassanpour. Attention-based deep neural networks for detection of cancerous and precancerous esophagus tissue on histopathological slides. *JAMA network open*, 2(11):e1914645–e1914645, 2019.

[20] R. G. K. H.-P. D. Tsung-Yi Lin, Priya Goyal. Focal loss for dense object detection. Technical Report arXiv:1708.02002, ArXiV, Aug 2017.

[21] J. W. Wei, L. J. Tafe, Y. A. Linnik, L. J. Vaickus, N. Tomita, and S. Hassanpour. Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks. *Scientific reports*, 9(1):1–8, 2019.

[22] N. Wu, J. Phang, J. Park, Y. Shen, Z. Huang, M. Zorin, S. Jastrzebski, T. Févry, J. Katsnelson, E. Kim, S. Wolfson, U. Parikh, S. Gaddam, L. L. Y. Lin, K. Ho, J. D. Weinstein, B. Reig, Y. Gao, H. Toth, K. Pysarenko, A. Lewin, J. Lee, K. Airola, E. Mema, S. Chung, E. Hwang, N. Samreen, S. G. Kim, L. Heacock, L. Moy, K. Cho, and K. J. Geras. Deep neural networks improve radiologists' performance in breast cancer screening. *IEEE Transactions on Medical Imaging*, 39(4):1184–1194, 2020.