Abstract

The Kaggle-competition "PANDA challenge" provided us with data that does not have anything to do with the pandas we know. This challenge was about "Prostate cANcer graDe Assessment". We got pathological data and had to predict the state of prostate cancer. Since we were not very familiar with designing and creating a new network structure, we made use of what is called "Transfer-Learning". We selected well-known pre-trained models, like ResNet and DenseNet, and tried to adapt these to our problem case. A huge amount of data, limited training times, and special properties of the data were things we had to take care of in this project.

1. Introduction

Prostate cancer is the second most common cancer among males worldwide. Currently examining biopsy under a microscope is the gold standard in the detection and grading of prostate cancer. Thereby the Gleason grade is determined by pathologists on hematoxylin and eosin (H&E) stained tissue specimens based on the architectural growth patterns of the tumor. A different grading system is the ISUP grade on a 1-5 scale, which is important when considering how a patient should be treated.

However, the grading suffers from variability among pathologists and depends not only on the pathologists themselves but also on the quality of the image-slides.

The need for an automatic system for grading prostate cancers is undoubtedly useful, especially in relieving the burden from pathologists and giving a second opinion apart from those already observed by the professionals.

2. Method

2.1 Dataset

The dataset for this project is provided by a Kaggle-competition. The training set consists of up to 10.616 whole-slide images of digitized H&E-stained biopsies originating from two centers, available in tiff format. This is one of the largest public whole-slide image dataset available.

The whole training dataset requires 383.62 GB of memory.

In Figure 1 there is an example image slide presented. As we can see the whole slide is very large and even in small image patches, there is valuable detailed information that is used to classify the cancer state of the sample.

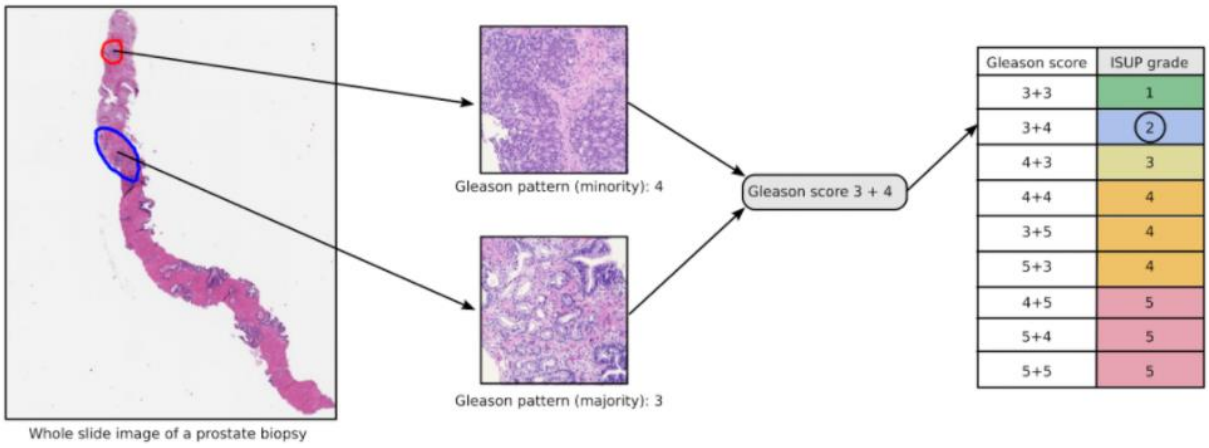

Figure 1: "An illustration of the Gleason grading process for an example biopsy containing prostate cancer. The most common (blue outline, Gleason pattern 3) and second most common (red outline; Gleason pattern 4) cancer growth patterns present in the biopsy dictate the Gleason score (3+4 for this biopsy), whoch in turn is converted into on ISUP grade (2 for this biopsy) following guidelines of the International Society of Urological Pathology. Biopsies not containing cancer are represented by an ISUP grade of 0 in this challenge." (from challenge description)

The labeling of the data is given by a .csv file as can be seen in the excerpt in Figure 2. For an image ID, there is information on the provider, the ISUP-grade, and Gleason-score.

Two different research groups, namely the Radboud University Medical Center and the Karolinska Institute provide the data.

	image_id	data_provider	isup_grade	gleason_score
0	0005f7aaab2800f6170c399693a96917	karolinska	0	0+0
1	000920ad0b612851f8e01bcc880d9b3d	karolinska	0	0+0
2	0018ae58b01bdadc8e347995b69f99aa	radboud	4	4+4
3	001c62abd11fa4b57bf7a6c603a11bb9	karolinska	4	4+4
4	001d865e65ef5d2579c190a0e0350d8f	karolinska	0	0+0

Figure 2: Excerpt from the given .csv file

For better understanding, here are some descriptions of the column titles (from the description of the Kaggle-competition):

- **image_id**: ID code for the image.
- **data_provider**: The name of the institution that provided the data. Both the Karolinska Institute and Radboud University Medical Center contributed data. They used different scanners with slightly different maximum microscope resolutions and worked with different pathologists for labeling their images.
- **isup_grade**: Train only. The target variable. The severity of the cancer on a 0-5 scale.

For simplicity reasons, we only consider the ISUP grade for further analysis.

2.2 Preprocessing

2.2.1 Generating Image Patches

In Figure 3 we can see a whole slide of one example image. Since we want to use a neural network approach, we normalize the inputs for our neural networks in a certain manner. Therefore, we use a sliding window approach to detect 16 image patches out of the whole slide that have a low ratio of white space inside.

An example result of this selection approach can also be seen in Figure 3.

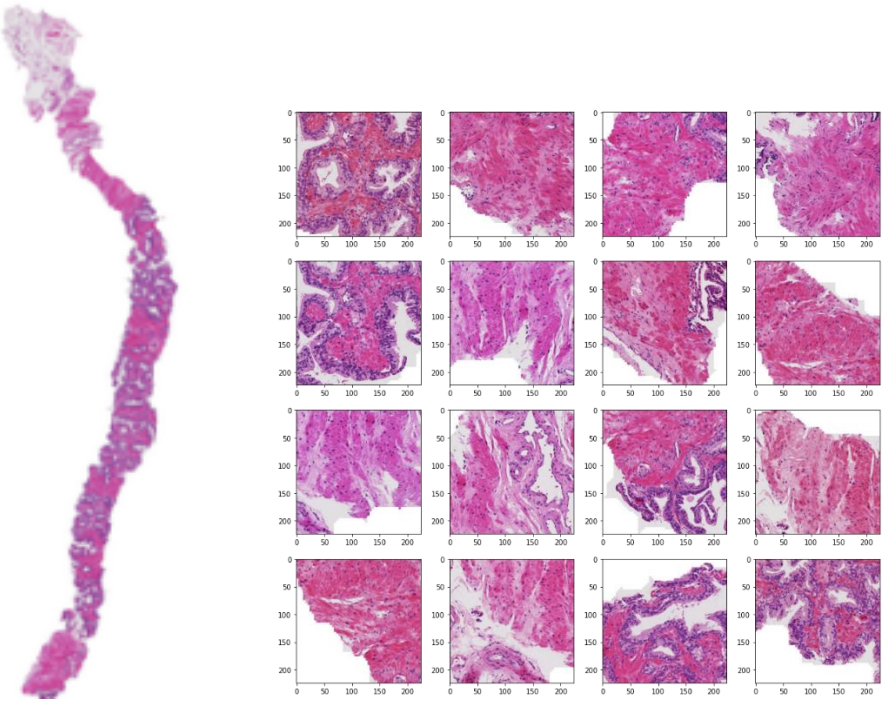

Figure 3: Full image with id_0 (left), 16 patches of the image with id_0 (right)

2.2.2 Data Augmentation

Data augmentation is often used in computer vision to generate more data and increase the diversity of the training set since more data will allow neural networks to be trained in a more generalized way. In image processing, images can be rotated, mirrored, or modified without the risk of altering the original object.

We apply different random transformations on the input image patches while training our network without needing to save all the generated images from the transformations. We only add random jitter, horizontal, and vertical flips because we do not want to alter the medical data too much in the risk of adding artifacts to the highly detailed data.

In Figure 4 and 5 there are examples for these transformations. Note that the jitter applied here is a little exaggerated for presentational reasons.

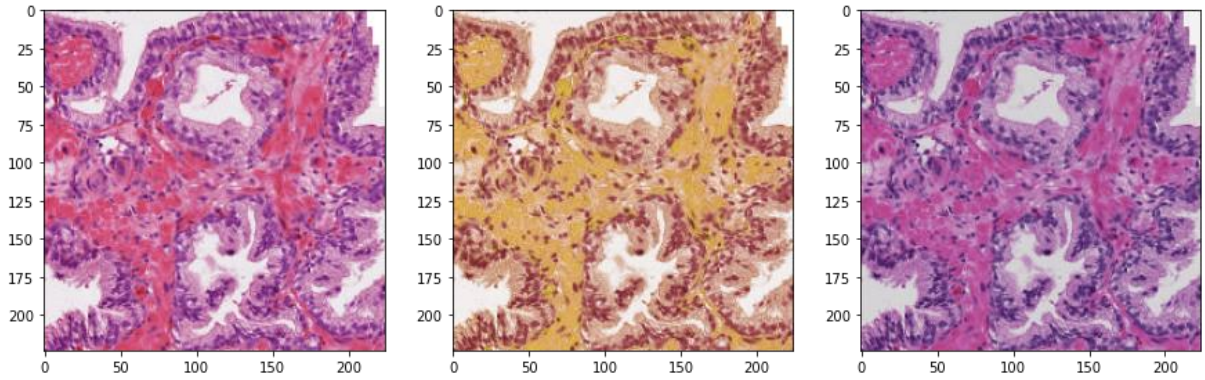

Figure 4: original patch (left), random jitter (middle), another random jitter (right)

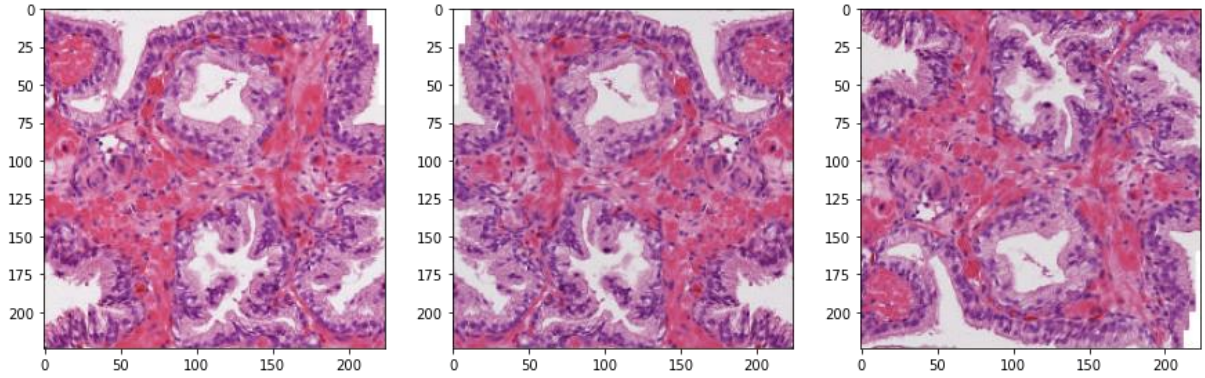

Figure 5: original patch (left), horizontal flip (middle), vertical flip (right)

2.3 Training

For training, we use the provided data to train in Kaggle Kernels and split the whole training dataset into 70% for actual training and 30% for validation. We use the image patches to feed into a model for prediction, loss, and optimization and randomly applied data augmentation on the input patch. Since we only use image patches for the prediction of one large image slide, it is important to note some considerations:

For each of the 16 selected patches of an image, a prediction for all 6 classes is performed by the forward pass of a neural network, i.e. we get a vector for 6 entries describing how well the provided image patch fits each cancer classification state. This vector is built for all 16 patches of one image slide. Using all result vectors of one image slide we calculate the mean for each class and find the maximum value of the mean vector as the prediction for the whole image slide. This prediction and the actual target label are then used for loss criterion and optimization.

We get the following shapes for input and output: RGB for 224x224 pixels and 16 image patches [224,224,3,16] → 16 forward passes of the CNN → 6 predictions for each patch [16,6] → the mean vector of the predictions [1,6] → and the maximum of the predictions max([1,6]) → for our final prediction label (0 to 5) which is compared to the target or used as a classification of an image.

2.3.1 Neural Network Architectures

In Figure 6 we can see the usual architecture of a convolutional neural network (CNN). CNNs use convolutional layers which have the benefit of weight sharing for image-inputs. This means fewer parameters and better generalization than in fully connected networks.

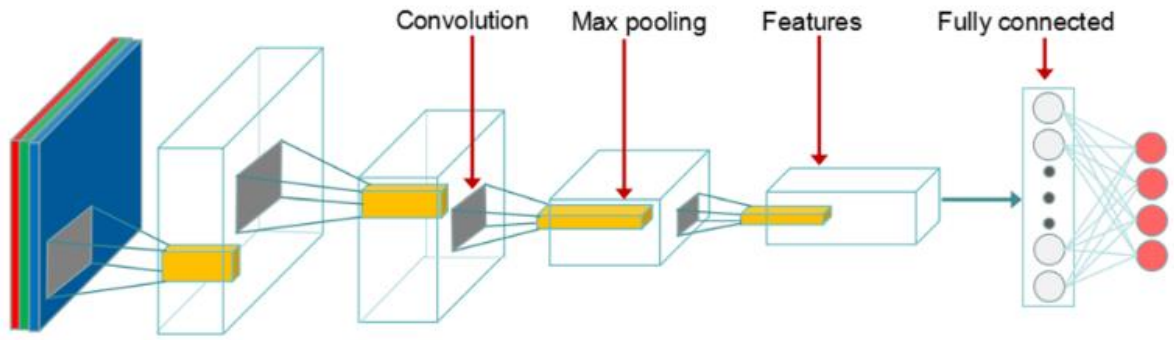

Figure 6: Architecture of ResNet50 model [https://www.researchgate.net/publication/334767096_Residual_convolutional_neural_network_for_predicting_response_of_transarterial_chemoembolization_in_hepatocellular_carcinoma_from_CT_imaging]

In "Transfer-Learning" a pre-trained model is used as an initial network which is then adapted to a certain application as we can see in Figure 7. We use 2 popular networks that perform greatly on the ImageNet database. We modify the last classification layer in both network structures to the output of 6 classes.

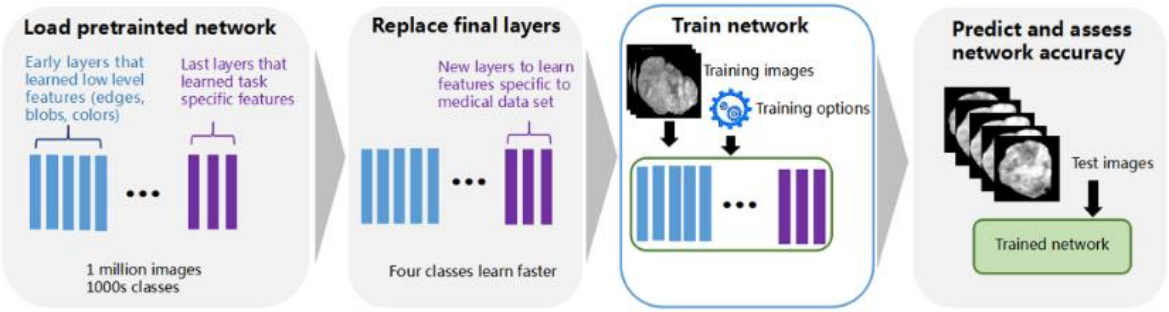

Figure 7: Basic Principle of Transfer-Learning

2.3.2 ResNet50

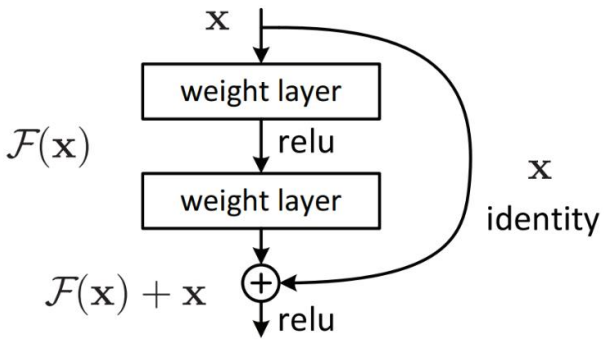

Figure 8: Residual block (from the original paper - https://arxiv.org/abs/1512.03385)

The ResNet uses skip connections for better gradient propagation from later layers to earlier layers as it can be seen in a building block in Figure 8. These skip connections are in fact beneficial for deeper network structures. ResNet50 model was pre-trained on a million images from the ImageNet database and can classify images into 1000 object categories.

2.3.3 Dense-Net-161

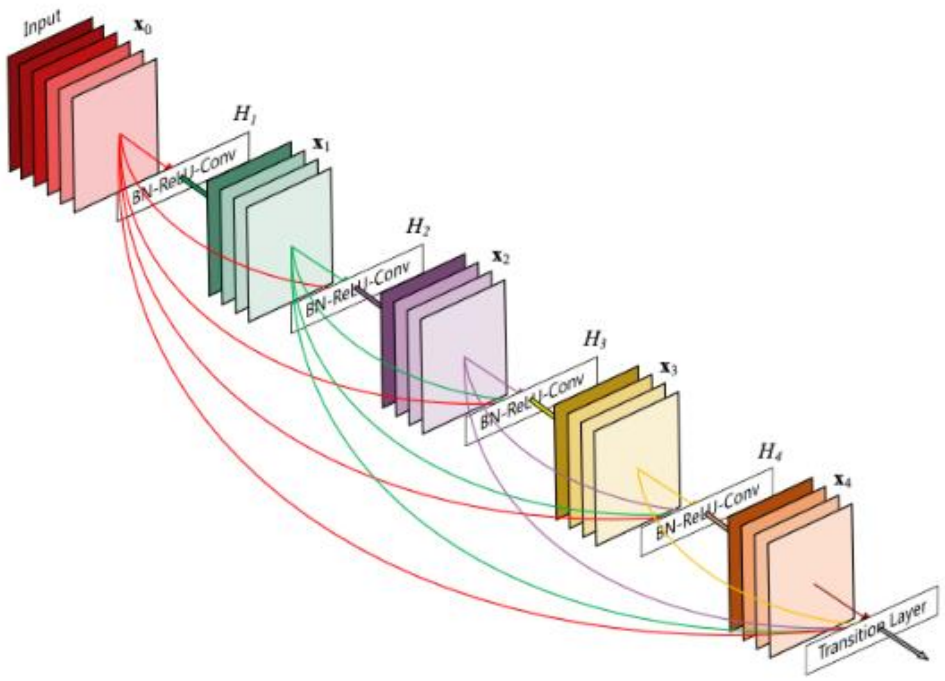

Figure 9: Architecture of DenseNet161 model(from the original paper - https://arxiv.org/abs/1608.06993)

Another structure we test is the DenseNet structure. This structure increases the information flow even further. Dense blocks have connections from any layer to all subsequent layers.

We train both network structures for a few epochs on the full dataset as well as only a portion of the whole data since the Kaggle has a time restriction on training time.

2.3.4 Optimizer

Optimizers are used to change the properties of a neural network, such as weights and learning rates, with the aim of reducing losses. The way how the weights or learning rates of the neural network should change to reduce the losses is defined by the selected optimizer.

To find out which optimizer is the best for our problem, we tested different optimizers from PyTorch using the full dataset, a batch size of 8 images per epoch, and 1 epoch per training for each test using the ResNet50.

In the following are some implementation examples to show the optimizer settings used for testing:

- optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)
- optim.Rprop(model_ft.parameters(), etas=(0.5, 1.2), step_sizes=(1e-06, 50))
- optim.RMSprop(model_ft.parameters(), lr=0.01, alpha=0.99, eps=1e-08, weight_decay=0, momentum=0, centered=False)
- optim.Adam(model_ft.parameters(), lr=0.01)
- optim.ASGD(model_ft.parameters(), lr=0.01)

Table 1: Results of the optimizer tests with identical boundary condition

Optimizer	Train loss	Train accuracy	Valid. loss	Valid. accuracy	Time
SGD	3.7806	0.7871	1.5830	0.3617	238m 16s
RMSprop	4.5528	0.6499	2.7392	0.2763	235m 10s
Rprop	11.0897	0.6518	13.3808	0.2776	261m 27s
Adam	5.0758	0.6713	2.7250	0.3074	239m 6s
ASGD	3.9290	0.6659	1.6808	0.2901	248m 11s

Based on the results, we decided to use the SGD Optimizer for training our network.

3. Project results

The networks performed quite similar and while training our networks we came across major problems:

3.1 Problems of our approach:

1. The large size of the dataset: The dataset we use has roughly 10k whole slide images, which is very difficult to train through the dataset given that we use GPU from Kaggle that times out every 9 hours. This leads to long training time and can be very difficult to track the training progress. One epoch using the whole dataset needed about 4 hours.
2. Variation of images in the dataset: the sources of the dataset come from different laboratories with different staining techniques and microscopes, which results in a variation of colors among the images. So far, we just directly use directly the image without any color normalization, and this could partly affect the training process.
3. Random tile selection: Because of the large size, we have to divide each whole slide images into smaller tiles and aggregate the output by averaging the outputs. This approach is particularly common when training with whole slide images. However, the way we select is quite random. We just run the sliding window and select the tiles with high ratio of cell area with respect to background, which could lead to situations where tile's label is different from image's label.

3.2 Suggestion for improvements

1. The large size of the dataset can be an advantage for Deep Learning techniques, but it could at the same time demands a proper computational capacity that can process such a large amount of data. Given the shortage of resources, we believe it is sensible to run experiments on a smaller size of the data to check if there is any computational errors and then run cross-validation on the whole scale of the dataset.
2. Instead of classifying 6 classes at once, it is worth trying to train only 2 classes: benign and malignant and another classifier for stages within the malignant classes. This is because the gap between the labels is large, and by training a binary classifier first

could potentially improve the accuracy by increasing the true positives. In fact, knowing whether somebody actually has cancer or not should be of greater importance than knowing the stage of cancer he or she is in. This approach is also based on the routine the pathologists do, which is to detect whether one has cancer and then grade the stages.

3. As already explained above, the variation in color could potentially influence the training and it is thus recommended by us the do the color normalization for all the images in the dataset.
4. A better way of image patch selection would be beneficial for the actual grading part. If there would be a way, like for example using segmentation, to find cancerous places in the images, these would represent the whole image slide even better and a classification of only these regions would be benificial in runtime as well as precision in prediction.

4. Conclusion and Learnings

We were able to implement the base-structure of different networks to classify prostate cancer and learned a lot about the usage of huge amounts of data, pre-processing, and large images.

The online course could be easily integrated into our lecture time. Furthermore, it contained very informative and well summarized papers, which taught the topic of artificial intelligence in a very understandable way. During the exam phase, however, we could only spend your remaining free time on the project, which slowed down the progress of the individual online course.

Our project "ProsGrade" was the final phase of our TechLabs experience. Most members of our team had only theoretical knowledge about artificial intelligence before they enrolled at TechLabs, mostly through their studies. Although we come from different fields, we all share a common interest in artificial intelligence and could contribute different ideas for our approach.

Our mentor was very helpful and helped us to define realistic goals. Nevertheless, we miscalculated the amount of free time we were able to invest in our project during the exam phase.

In the end, our limited possibilities became an obstacle, so we did not have enough training time for optimization of our approach and could not achieve very good results. Regardless of the results, we acquired a lot of practical knowledge together and learned really much from each other!

5. Information about the project team

Everything from this blog and the entire library can be found in the following Github Repo.

We are Master students at the RWTH University Aachen:

Luca Fahrendholz: Major in General Mechanical Engineering, LinkedIn

Thong Nguyen: Major in Computer Engineering, LinkedIn

Christina Prigge: Major in Automation Engineering, LinkedIn

Tuan Truong: Major in Biomedical Systems Engineering, LinkedIn

Judith Lefkes: Mentor, LinkedIn