BACKGROUNG

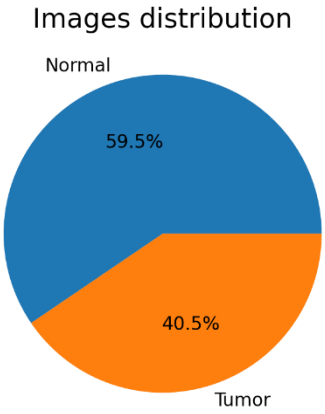
Microscopic examination of suspected tissue is regarded as the gold standard for the clinical diagnosis of cancers. Normal and tumorous tissues differ for several characteristics: cell density, cell size, cell heterogeneity, vasculature integrity and tissue morphology. If the histological type is different from what is usually found in the tissue being examined, it can point to a cancerous phenotype.

Even though diagnosis made by the doctor is the current best tool for cancer and tumor diagnosis, it is time consuming and leaves room for human error. In principle, analysis of histological images could be automated and deep learning algorithms may be able to accurately detect cancerous lesions, allowing the physician to take care of other medical matters.

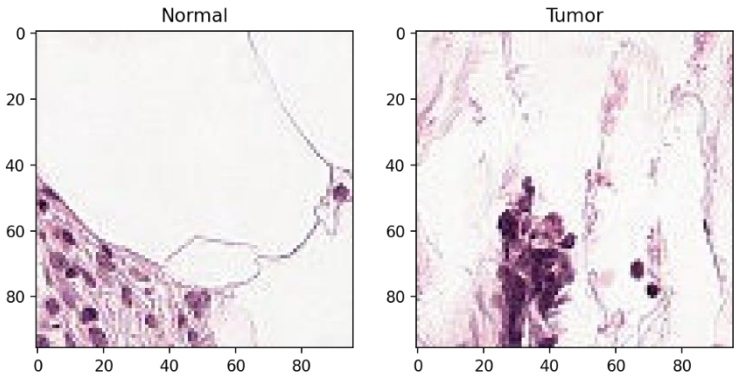
The goal is this project is to build and tune up a deep learning algorithm for the detection of cancerous lesions in histological images.

DATA OVERVIEW AND EDA

The dataset is available on Kaggle (https://www.kaggle.com/c/histopathologic-cancer-detection) and contains 220025 color (RGB) images (96x96 pixels). Most of the images, 59.5%, are showing normal tissue. The two classes are fairly balanced, thus we can avoid over-sampling or under techniques that handle unbalanced data sets.



The image below shows an example of normal and tumor tissue. While the normal tissue appears well organized, the tumor cells are mangled up and the normal structure of the tissue is missing.



MODELING

When I began this project, I was determined to train all models from scratch. Even though, transfer learning usually the best course of action in visual science, I wanted to experience how the number of layers, filters, kernel sizes affected the performance of the model… basically I wanted to fool around. This approach would have not been possible if I didn't have a powerful desktop specifically designed for deep learning applications. A big thanks to Nvidia for that.

Initially (part 1), I implemented a vanilla approach: split the all dataset into training and testing (80/20 split) and utilized the whole training set for training and the whole test set for validation. Later (part 2), I decided to split the training set into 5 folds and train 5 separate models (4 folds for training, one for validation). The predictions of the 5 models on the test set were ultimately averaged to obtain the final estimate.

Part 1 – Vanilla approach

Images were divided in training and validation sets with a 80/20 split (175780 training images, 44245 images for validation). Several models were designed trained for 50 epochs and tested. You can find the details of each model here. The best two model delivered an AUC of 0.9914 and 0.9905.

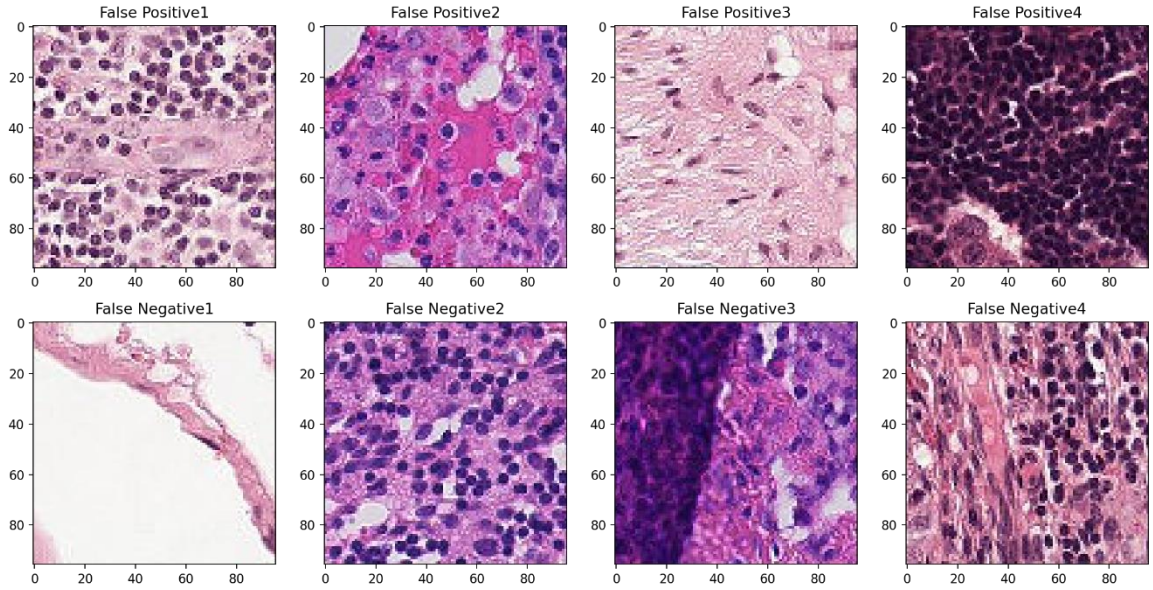
	VALIDATION ACCURACY	VALIDATION AUC
MODEL1	0.943	0.9841
MODEL2	0.9611	0.9914
MODEL2B	0.9534	0.9897
MODEL3	0.9597	0.9905
MODEL4	0.8499	0.9579
MODE5	0.9302	0.9834

Part 2 – K fold approach

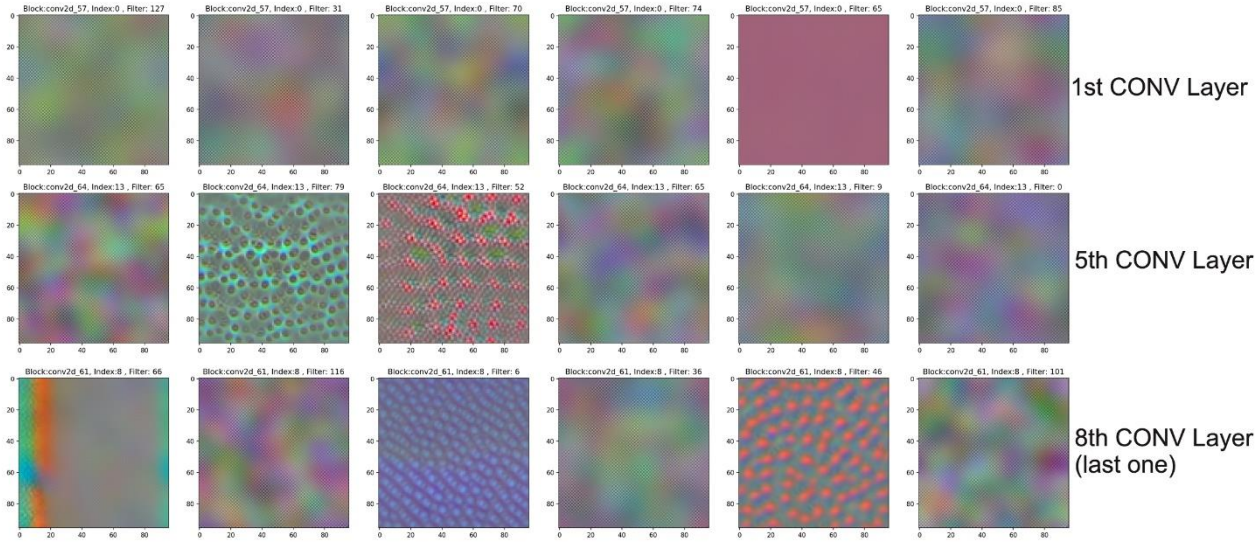
Next, I utilized a K-fold validation approach. The training set was split in 5 chunks: 4 were used to for the train of the model, while the 5th one was used to test it. The 5 trained models where than evaluated on the validation set. The 5 predictions were averaged to obtain the overall prediction. For some model I also used test time augmentation. The table below summaries the performance of all the models I built and tested. For details on the architecture of the CNN please refer to the notebook here.

5 Folds Models	Epochs	TIME TEST VALIDATION	Validation Balanced Accuracy	Validation AUC	Misclassified Images
Model K1	75 with LR decreasing at 50 (0.95 per epoch)	NO	0.9334	N/C	1425
Model K2	90 with LR decreasing at 50 (0.95 per epoch)	NO	0.9329	N/C	1430
Model K3	80 with LR decreasing at 50 (0.95 per epoch)	NO	0.9334	N/C	1417
Model K4	100 with LR decreasing at 50 (0.93 per epoch)	YES	0.967	0.9981	716

Time augmentation has a major impact on the prediction power of the model. Out of 44125 images, Model K5 misclassified only 716 of them. Next, I checked how the misclassified images looked. The figure below shows eight randomly picked misclassified histological images. It is not clear me why/how they were not classified properly by the Model K4. I could argue that for the false positive 1, the lack of tissue in the image may have tricked the classifier somehow. But I can't provide any reasonable explanation for the misclassification of the remaining 7 images.



I was curious to see what the different filters learned during training. The Keras-vis package is the ideal tool to answer that question. The figures below are a visual representation of few filters of the 1st, 5th and 8th convolutional layers. As we move deeper into the network, the convolutional filters learn more complex features in the images. In the last row, cell-like structures are clearly identifiable. What is interesting to note is that some filters in the last layer learned just as much as the filters in the first layer did, and thus, probably didn't contribute at all to the prediction. If that's the case, removing these filters would save computational resources and, potentially leading to faster predictions.



FUTURE DIRECTIONS

Proper diagnosis of tumor is critical for delivering the most appropriate therapy to the patient. Therefore, it is imperative that the ML model be 100% accurate. The final model is undoubtedly great but requires further improvements in order to substitute an experienced histopathologist. The model could be better by dropping few data samples for the overly sampled data class (normal tissue), improving the resolution of the images and deepening the network.