

Classifying Tumor Colon Tissue



Intro/Background:

Pathological analysis plays a role in identifying cancer cells at an early stage within small tissue samples. Pathologists face the daily task of reviewing numerous tissue slides, which can be tiresome and time-consuming. In the field of clinical diagnosis, the primary objective for pathologists is to accurately measure cancer cells and their distribution in tissue samples. To simplify this process and facilitate the identification of malignant cell regions throughout an image, methods for classifying pathological tissue sections are sought, and deep learning methods are being explored for this purpose.

The Dataset:

The dataset consists of images of tissue slices taken from colonoscopy (data source: <https://medfm2023.grand-challenge.org/> (task 2)). The dataset had two classes; images of tissue with tumor tissue present, and images of tissue without tumor tissue. The dataset consists of a total of 10009 images, where 6494 are normal tissue and 3515 are tumor tissue. The data was collected from a total of 396 patients. All images have a resolution of 1024x1024 pixels and are colored. Information on the quality of the labeling of the images was not provided, but it was made by pathologists and it's considered to be correct.

See example of the images below.

	<table><tr><th colspan="2">Normal Tissue</th></tr><tr><td>#Sample</td><td>6494</td></tr><tr><td>#Slide</td><td>249</td></tr></table>	Normal Tissue		#Sample	6494	#Slide	249		<table><tr><th colspan="2">Tumor Tissue</th></tr><tr><td>#Sample</td><td>3515</td></tr><tr><td>#Slide</td><td>147</td></tr></table>	Tumor Tissue		#Sample	3515	#Slide	147
Normal Tissue															
#Sample	6494														
#Slide	249														
Tumor Tissue															
#Sample	3515														
#Slide	147														

Data pre-processing:

The data was first loaded into the file. Due to memory limitations, the whole dataset was not loaded. Only 2000 images of each class in grayscale were used for training our models. This makes the used dataset balanced between the two classes, but also means that not all data is used.

For usage of pretrained models in MLP model training, we had to use colored images considering the pretrained model we chose required 3 channels. This requirement increased the memory consumption. Hence, we ended up using 750 images of each class in rgb in this case.

The memory limitations also meant that a reduction in image resolution was necessary. The images were downsampled to 128x128 and 256x256. Both resolutions were used in later models, as we thought it would be interesting to see how much the resolution affects the results.

The method uses 3-fold cross validation, so the data is split into 3 equal parts, and each $\frac{2}{3}$ of the data is used for training and each last $\frac{1}{3}$ for validation for a total of three times..

Model Architecture:

In our study, we tried to employ two powerful convolutional neural network models, VGG16 and ResNet50, to analyze and classify tumor images within the dataset. Both models are characterized by their deep architecture but differ in their design principles

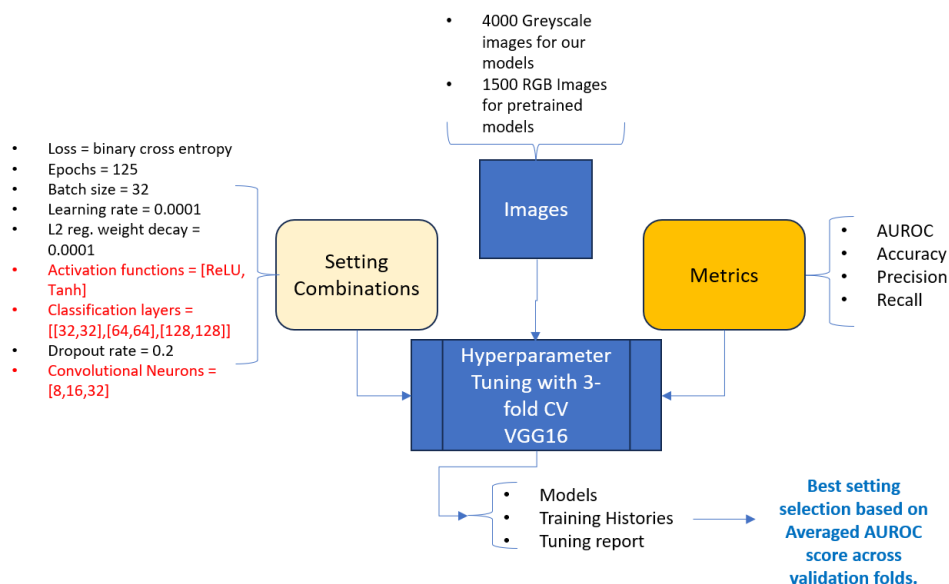
VGG16 follows a straightforward approach, stacking multiple convolutional layers with small (3x3) filters, interleaved with max-pooling layers for downsampling. The architecture is relatively simple yet effective for image classification tasks.

On the other hand, ResNet50 introduces a more sophisticated structure by incorporating residual connections. This innovative design helps address the challenges of training deep networks, allowing the model to learn residual functions. This is particularly advantageous when dealing with intricate patterns and features in medical imaging datasets.

By leveraging both VGG16 and ResNet50, we aimed to harness the strengths of each model for a comprehensive analysis of tumor images, exploring the nuances of their respective architectures to enhance diagnostic accuracy and sensitivity.

Training Pipeline:

In this project, from preparation of the images to generation of results, we have followed the following pipeline:



After the preparation of the images, we input the images, metrics, and hyperparameter setting combinations (red colored hyperparameters) into hyperparameter tuning. The hyperparameter tuning produced three main outputs which are Models, Training Histories, and Tuning report.

Models are the keras files that are saved at the epoch where the model achieved the highest AUROC score on validation data throughout the training. Training histories were saved at the end of each setting combination and fold. Tuning report was the report that kept information such as model settings, training time, epochs that achieved the lowest losses and the highest AUROC scores, and fold based AUROC score, accuracy, precision, and recall. The best setting is chosen based on the averaged AUROC score across the validation folds.

Results:

During our attempts with training of ResNet50 and VGG16, we realized that the ResNet50 architecture we implemented was computationally more expensive. For that reason, we decided to experiment with VGG16 architecture only. We used 2 approaches for hyperparameter tunings. These were training without pretrained model and with pretrained model approach. Both approaches used 128x128 and 256x256 images with 3 cross validation.

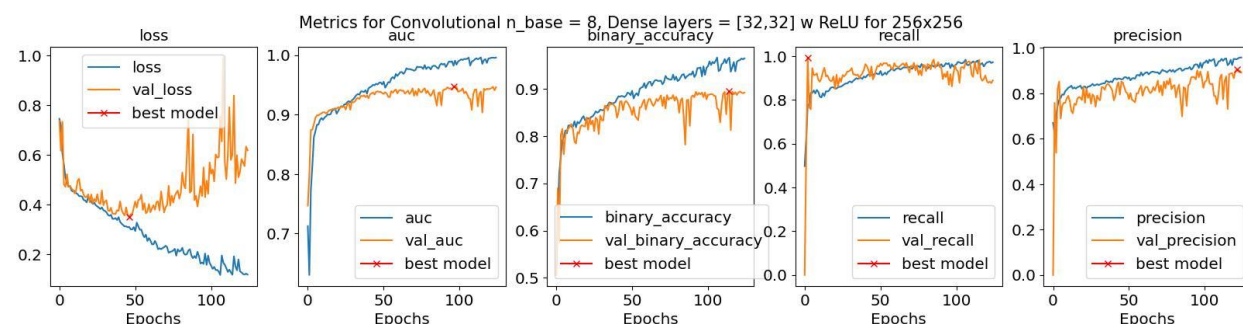
At the end of tuning operations, 54 models (18 setting x 3 folds) were trained for each image size for without pretrained approach and 18 models (6 settings(convolutional neurons were excluded) x 3 folds) were trained for each image size for pretrained model approach.

Based on the average AUROC scores calculated over validation folds, the following settings were the best for corresponding image size and approach:

Image Size	Approach	Avg. AUROC Score	Act. Func.	Conv. Neurons	Class. Layers	Avg. Training Time
256x256	No pretraining	0.930	ReLU	8	[32,32]	7.3 mins
128x128	No pretraining	0.933	ReLU	16	[128,128]	4.18 mins
256x256	With pretraining	0.961	ReLU	X	[32,32]	0.58 mins
128x128	With pretraining	0.941	Tanh	X	[128,128]	0.53 mins

According to the results, we see that the highest scores are achieved mostly with ReLU activation function. When the image size gets bigger, the network prefers to have less complex data. That might be because the model gets overtrained on training data and complexity should decrease in the model architecture to avoid it. When we compare the pretrained and not pretrained models, the pretrained models are winning both in terms of speed and scores achieved.

Considering we used smaller data portions than the other competitors used, we are not able to compare the scores we achieved in the challenge.



In the above, the metric plots of the best setting for the non-pretrained model trained on 256x256 images can be seen. For further results, we recommend you to check *Project-Result and Report Analysis.ipynb*.

Codes and How to Run:

The project consists of 5 jupyter notebook files(4 for CNN training, 1 for result and report analysis), and 2 python files *Models.py* containing model architectures, and *Utils_project.py* containing functions for data loading/processing, and result analysis. In order to run the models and check the results, *Project-Result and Report Analysis.ipynb* file can be executed in the server under the '/home/group_8' path. Considering models, training histories, and reports are needed, this location is suggested.