Deep learning for computer science

Doctor Jonathan Rubin

Colorectal histology classification
by Stas Vinogradov
Spring 2021 – סמסטר ב ,א"ל תשפ"שנה

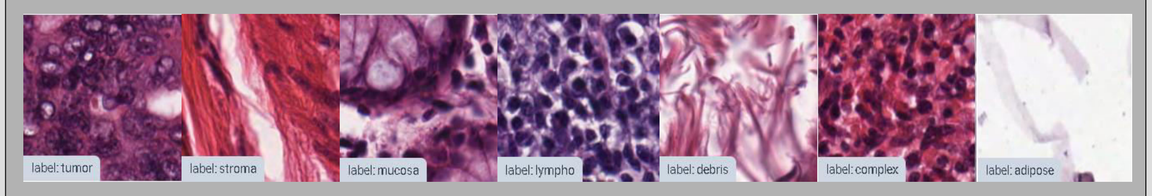
מכון טכנולוגי חולון
Holon Institute of Technology

Introduction

Work Environment: Jupyter notebook in google colab
Programming language: Python
Project goal:Classification of histopathology images by a deep learning model and visualizing the results on a large colorectal histology images

Project Summary

The main propose of the project is to train a model to be able to classify histopathology, to recognize colorectal symptoms
Based on 8 classes,7 types of tissue and a background.



Part 1- loading the data
From the keras dataset 5000 images of tissue sized 150x150 in RGB,classified into 8 classes,then process the data for ease of use

Part 2 - training the models
Several CNN models will be tried, all the models will be trained on 150x150x3 processed images, and will classify them based on 8 classes, the one with the highest accuracy will be chosen

Part 3 - Feature Visualization
Plotting and displaying a confusion Matrix from the prediction of the best model and using a VGG-16 feature extractor on the second to last layer to extract a 4096-D features representation,
using the TSNE algorithm to reduce the dimension and display the predictions

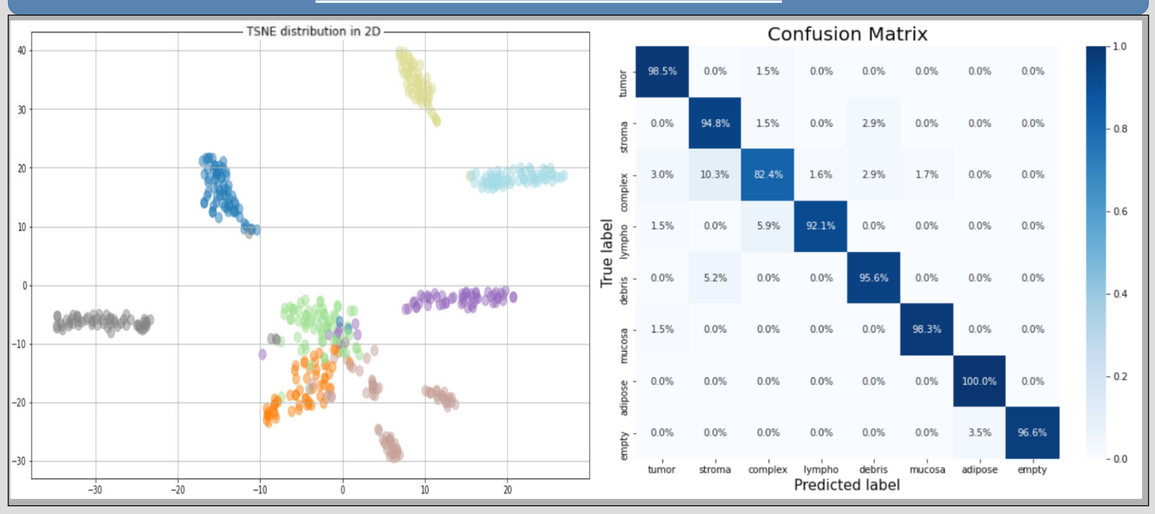
Part 4 – Prediction Visualization
Using the best model, slice the large 5000x5000 biopsy images into 150x150 patches with a 50% overlap, then color the patches and tumor heatmap based on the classification of the model, And finally combining the images back to size 5000x5000 and displaying the results side by side.

Prediction Visualization

Help Sources

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html
https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html

Feature Visualization



The Models

1) **VGG like-** the first model is a basic **ConvNet,** the model has a 4 block structure, the first 3 blocks are made of two conv2D and max pooling, the last block has a dense(512) and a dropout layer(0.5) after the flatten layer.

2) **VGG-like + data augmentation-** a more advanced **ConvNet** model, still with a 4 block structure, with added batch normalization after every conv2D and dense layer,
the training data was also augmented in-order to increase data size for more accuracy

3)
VGG16 transfer learning- using **VGG16** with “imagenet” weights as the base model, without the “top”, replacing it with two dense layers(512 and 256) with dropout(0.5) after each one, in the learning process the base is frozen and the new top weights are the only one’s updated.

VGG16 Fine tuning- after training the new **top**, the base is unfrozen and the model is trained again at 10% the transfer rate

Both the transfer and fine tuning train on augmented data, allowing it to train on a larger sample size to increase accuracy

Model	#Parameters	Optimizers	batch-size	epochs	accuracy
VGG-like	21,525,288	Adam	64	20	0.7720
VGG-like+data augemntation	21,529,128	Adam	128	30	0.8700
VGG16 transfer/fine tune	19,042,888	RMSPOP, SGD	128	20	0.9459

Results-Third model