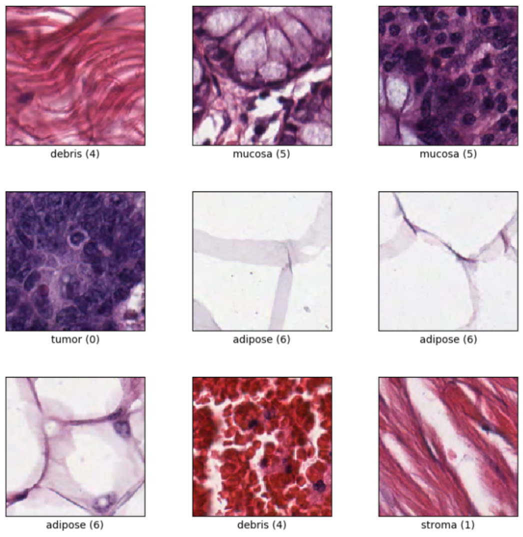
Physics Dataset Practice Problems

Week of August 3, 2020

4 Colorectal Cancer Histology

Histology is the study of tissues at the microscopic level. Histology is enormously useful for understanding tumor growth and patient prognosis. Modern imaging technology allows pathologists to digitally capture and store images of healthy and malignant tissues. This week, you'll be using a convolutional neural network to sort histological images into 8 classes. The compressed `histology` directory contains 8 subdirectories with a total of 5000 images belonging to various histological classes. This dataset and many other useful datasets can be found using the `tensorflow-datasets` module (https://www.tensorflow.org/datasets/overview).



4.1 Data Visualization

As always, it is useful to visualize the dataset before beginning. Visualize one example from each of the histological classes with `matplotlib.pyplot`'s `imshow` function.

4.2 Data Preprocessing

When training neural networks, it is often useful to augment the training data with random shifts, rotations, shearing, and other techniques. Fortunately, `keras` has many of these preprocessing techniques built into objects such as their `ImageDataGenerator`.

Instantiate an `ImageDataGenerator` object from the `tensorflow.keras.preprocessing.image` module and implement some augmentation procedures of your choosing. Make sure to use the `rescale` and `validation_split` keyword arguments to normalize and split the dataset.

Following instantiation of the data generator, use the `.flow_from_directory()` method of the generator to create training and validation generator objects. `.flow_from_directory()` will automatically load images from several subdirectories within the master directory and create training labels based on their subdirectory. Use the keyword arguments `target_size`, `batch_size`, `class_mode`, and `subset` to specify your training procedure.

Information regarding `ImageDataGenerator`s can be found in the Keras documentation at https://keras.io/api/preprocessing/image/.

4.3 Initialize a Model

Use a 3-layer convolutional neural network to classify the images based on their histology. Consider using `tensorflow.keras.layers.Dropout` layers after your convolutional layers. Dropout layers, randomly "turn off" neurons during training, which helps prevent overfitting of the model.

Specify the input to match your target size specified in the data generators, and use an 8-neuron Dense layer for the network output. Use `softmax` activation for this layer, since the outputs are categorical.

Visualize your model with the `.summary()` method.

4.4 Train the Model

Compile and train your model. Use the `'categorical_crossentropy'` loss function to assess model performance. Plot the loss and accuracy for the training and validation data over the course of training.

Call the built-in `next` function on your validation generator to get a batch of validation samples. Visualize the samples and their predicted/true classes according to your model.