**COCS2673 Computational Machine Learning**

**Assignment 2**

**First Draft**

Histopathology (or histopathology) refers to the study and diagnosis diseases related to the tissues and involves observing tissues under a microscope (Royal college of Pathologists).

Histopathologists, doctors involved in the study and observation of tissues, provide a diagnostic service for cancer; they handle, and study tissues and cells removed from suspicious lumps, identify the nature of the abnormality, and categorize it as either malignant or benign. If the diagnosed as malignant, they provide information to about the type of cancer, its grade and, for some cancers, its responsiveness to certain treatments (Royal college of Pathologists).

From this, we can come to understand how critical this process can be.However**,** detection and classification of cell nuclei in histopathology images of cancerous tissue is a challenging task due to cellular heterogeneity. Deep learning approaches have shown some promising results on histopathology images in various studies.

In this project, we attempt to create two separate machine learning model based on Deep Neural Networks to aid in predicting i) whether based on the given images, a cell is cancerous or not and ii) the cell type of the provided image.

METHODOLOGY

We have followed similar trajectory for both the models, with differences in the parameters and functions we have used.

Part 1 – Predict whether cell is cancerous or not

First we load both the mainData and extraData. mainData has 6 columns with 9895 rows of data. For the first part since we will only be predicting whether cell is cancerous or not, we will be dropping all the other column except for `patientID` and `isCancerous`.

`isCancerous` contains the acutal diagnosis, with 1 indicating cancerous and 0 indicating the opposite (benign). The split of benign to cancerous is a 35:65 ratio.

[histogram of cancerous data split]

We split the data into training and test splits, with a further split on the training data into training and validation splits.

Models

We first started out with a base Neural Network model to see how it would perform and then move on to more complex models based on the needs. For the base model, we used the following arguments:

* Input shape of 2187 dims
* 1 input layer with 256 internal nodes
* 1 output binary layer
* Loss – Binary Cross Entropy
* Metric – binary\_accuracy
* Sigmoid activation function

The results were pretty good for a base model, however there was problem of overfitting.

[fitHistory image]

Therefore, the next step was to perform some regularization. Here, we use l2 regularization with a default lambda value of 0.01. The final result wasn’t any different from the previous baseline model, however, the overfitting was still persistent. Hence, we apply dropout regularization.

[fitHistory image l2 regularization]

Here, there was definitely a improvement, but the test data seems better than the training data. Since we’re getting close, the next step would be to develop a full blown neural network.

Baseline VGG

The first model was to develop a baseline VGG model. Here, we have 3 blocks, each with 3 layers of convolution, activation and pooling. These blocks will allow us to downscale the image with the use of the max pooling layer.

[code for VGG model]

[fitHistory img]

There was a massive improvement here, as we can observe from the graph. While were a few hiccups during the initial training, the overall end result was considerably better than the baseline models. But there was still a slight issue of overfitting. So next thing to do was to tweak this baseline VGG model. The following are the observations we made:

* Image size is small as is, therefore, scaling it down won’t be too beneficial.
* Reduce the number of epochs from 150 to 50
* Reduce the number of overall convolutions
* Given the number of images, data augmentation is not required
* Use dropout regularization along with l2

[code for VGG 2]

[fitHistory img]

Here, we can see that the overall performance is much better.

Part 2 – Predict cell type

The model development for predicting the cell type is almost exactly the same as Part 1, except due to the requirement, we used Categorical Cross Entropy for the loss function, categorical\_accuracy for the accuracy metric and for the output layer of the Neural Network, we used `relu` instead of `sigmoid` as the final classification result involves 4 prediction classes instead of 2.