Machine Learning vs. Cancer: An In-depth Analysis

By
Group Paranormal Distributions:

Naomi Warren (z5114203)
Yuying Chen (z5180254)
Yathin Suresh (z5312394)
Matthew Harrison (z5336191)
Jun Hyeok Im (z5259480)

INTRODUCTION:

Histological images are a key data source for identifying and classifying tumours in tissue samples (Komura & Ishikawa 2018), but manual classification of each image is a labour-intensive task that needs to be performed by a qualified pathologist. Hence, automation of this task via machine learning methods would be highly favourable.

We have been provided with a labelled dataset of 858 histological images with 1024x1024 pixels defined over 3 colour channels (red, green, blue). These images each belong to one of four classes, with one non-cancerous class (Class 0) and three classes of different tumour types (Classes 1-3). We present here a selection of machine learning methods to address this multi-class supervised learning problem, and analyse their performance with particular reference to their f-1 score.

There are several main issues involved in this task: a) loading the dataset; b) reducing the size/complexity of each image instance while retaining its information ('feature extraction'); and c) learning the relationship between these features and the output classes ('class prediction'). Part a) was addressed using PyTorch's DataLoader class. For parts b) and c), we developed several models. The first set compared performance of average pooling vs. Haralick's textural features for feature selection using Logistic Regression, Gradient-Boosted Decision Trees or Balanced Random Forests. The second involved a Residual Neural Network, which combines both feature selection and class prediction within the one algorithm.

The Residual Neural Net yielded the most accurate classifications with a f-1 score of 0.89 on our validation set.

EXPLORATORY DATA ANALYSIS:

Missing Values & Anomaly Detection:

Inspecting the training set reveals that there are no missing values or rows within our dataset. There appears to be considerable variety in the way the tissue is presented (e.g. 100% confluence of small cells over the image vs. clustered cells scaffolded around empty 'pockets' vs. red patches significantly (~20-100x) larger than the average cell size), but given the lack of provided

domain information, we have assumed that these do not represent anomalies and reflect true information about the tissue.

Class Distribution:

The training dataset contained 858 samples, with 407 (47%), 225 (26%), 189 (22%), and 37 (4%) images belonging to Classes 0, 1, 2, and 3, respectively (Supp. Fig. 1). This results in a balance between non-cancerous (class 0) and cancerous (classes 1-3) instances, but a clear imbalance within the different tumour classes.

Feature Extraction/Manipulation:

Image data is extremely information-rich, as it contains not only scalar features at each pixel (namely, colour intensities) but also spatial information. In addition, histological images tend to have a high degree of resolution in order to capture the shape of very small objects (cells). With <1000 image instances, it is clear that this problem easily succumbs to the curse of dimensionality. Even reducing the spatial information by compressing each image into a single vector would result in $1024 \times 1024 \times 3 \sim 3$ million features, which would perform poorly both computationally and statistically. More astute methods for dimensionality reduction are hence required.

One of the simplest transformations is **grey scaling**, which only retains a weighted average of the different colour intensities for each pixel, reducing the data to a third of its original size. Grey scaling can be very effective in a histologic context (Komura & Ishikawa 2018), since tissue samples are monochromatic unless treated with multiple stains. Indeed, on inspection the data provided appears to have no more than two stains (red and purple), and these still appear distinct when converted to greyscale (Supp. Fig. 2). We constructed average colour histograms for a random subset of each class, and found that colour was relatively consistent across classes (barring class 2; see Supp. Fig. 3), with a dominant red channel but all colours having the same distribution shape (a bell curve), suggesting grey-scale compression would not lose major information.

Another simple method to reduce dimensionality is **pooling**, which reduces an N by N region of intensity values into a single value, causing the image matrix to be compressed by a factor of N squared. Pooling can be done in a variety of ways, including averaging (e.g. mean or

median) or min/max pooling. This method preserves global spatial information but loses local spatial information, where the information granularity is determined by N. From inspection, cells in the histological samples are small with respect to the size of the image, and hence, significant pooling might preserve the shape of cell clusters or scaffolds, but would lose information about the boundaries and shapes of individual cells. Hence, any downsizing via pooling with this dataset ought to be approached with caution.

Given all of the above, the most appropriate avenue for extracting meaningful but low dimensional features in this data space appears to be by looking at texture and local shape context. Texture features are a tried and tested method for classification of cancer types in histological images (Rexhepaj et al. 2013).

METHODOLOGY:

Initial loading of the dataset was performed with PyTorch's DataLoader to handle the large files. We then explored a range of common models for classification, increasing in complexity, to determine the simplest method able to accurately learn and describe the data.

In order to train each model and select appropriate hyperparameters, we first split our data into a training and validation set (see Table 1 for details), with the validation set used for tuning. The split was performed with an 80:20 ratio and shuffling. An attempt was made to split the data into a training, validation and test set with ratio 60:20:20, but this caused models to consistently underfit. As a result we chose not to keep any data for testing, since there was separate test data provided. These models were hence evaluated by their f-1 score on our validation dataset.

Due to the class imbalance mentioned above, we sought to preserve the class distribution in both the training and validation sets (percentages in Table 1).

Table 1: Distribution of classes within training and validation sets.

	Class 0	Class 1	Class 2	Class 3	**Total**
Train	325 (47%)	175 (26%)	156 (23%)	30 (4%)	686 (100%)
Validation	82 (48%)	50 (29%)	33 (19%)	7 (4%)	172 (100%)
Total	407	225	189	37	858

Feature Extraction

All classifiers apart from RNN required some form of feature extraction before applying the machine learning algorithm.

We used average pooling as a baseline feature extraction method as it downsizes the number of features but leaves them in the same domain/range, meaning that the transformation itself should not impose a high degree of bias. However, our logistic regression model was not able to handle more than a 16x16 new image matrix (i.e. 256 features, Supp. Fig. 4), which required 64x64 averaging, resulting in a significant loss of local information. This was performed using the skimage.measure.block_reduce function.

The second approach to feature extraction was to focus on texture using Haralick's feature extraction. The Haralick texture fundamentally relies on the computation of the Grey Level Co-occurrence matrix (GLCM). The GLCM records pairs of adjacent pixel values that occur in an image over the whole image. There are 4 unique variations to the GLCM depending on the direction of the adjacency. These four matrices are then normalised and parsed through functions which in tandem yield a 13-dimensional vector that describes prominent image features such as contrast, entropy and energy. In the context of our problem, we used Mahotas' mahotas.features.haralick function that allows us to compute the 13-dimensional vector from image input.

Examining the train set transformed to haralick features (Supp. Fig. 5), we can see the magnitude of features vary greatly, which we know affects logistic regression models heavily when using L2 penalty. Therefore, the dataset is normalised using sklearn's StandardScaler fitted in our splitted training dataset and the validation set transformed when using logistic regression.

Modelling

1) **Logistic Regression**

The first model implemented to compare average pooling vs Haralick features was logistic regression. Both models were trained under L2 penalty with the use of the 'lbfgs' solver, since the dataset is multi-class.

Due to the dataset's multiclass nature, each model trained 4 separate weight vectors. In the case of Haralick Texture the vector was a 13 dimensional vector, requiring the model to calculate 52 weights in total. For average pooling, the feature vector contained 256 dimensions, ultimately calculating 1024 weights in total. After these weights had been learned from the training set, class predictions were assigned to unseen instances in an unbiased manner such that the class with the highest probability was assigned to that instance.

To maximise both models' performance, the hyperparameter *C* was tuned using a grid search across the values [0.0001,0.001,0.01,0.05,0.1,0.5,1,1.5,2]. The 'best performance' as measured by f1-score occurred with $C = 1$ (Supp. Fig. 6), as seen in Figure 1.

2) Gradient Boosting Method

In order to improve upon the results of logistic regression, Gradient Boosting Decision Trees were selected as a viable model. With the Haralick textures impressive information retention, the involvement of a model with low bias and high variance to help further improve the high baseline prediction accuracy set by logistic regression. On the other hand despite average pooling having a poor performance with logistic regression, perhaps a powerful model such as Gradient Boosting could leverage its large feature space to yield an accurate model. The performance of both models were maximised by hypertuning the following parameters: Estimators, Learning Rates and Max Depths. Each parameter was varied using a grid search, with model performance evaluated using the f1-score (Supp. Fig. 7). The final selected parameters are as follows: number of estimators = 80, learning rate = 1.0 and max depth of 1.

3) Balanced Random Forests

The algorithms applied so far did not address the class imbalance inherent in the data, and hence several methods were applied to determine if this could improve model performance. Generally speaking, a model will pay equal attention to every sample within a dataset – meaning that, if there are far more samples belonging to a specific class, the model will spend more time learning from that class than from any other. This can be addressed by assigning greater weight to the misclassification of samples belonging to the lower frequency class, as this ensures they are still able to contribute information to the model rather than being overwhelmed by the additive effect of many samples from the high frequency class.

The implementations below involve either directly assigning weights to balance the contribution of classes to the loss function, or performing subsampling to synthetically increase the proportion of the dataset that belongs to the low frequency class. The models made use of the RandomForestClassifier from the sklearn.ensemble package, BalancedRandomForestClassifier from the imblearn.ensemble package, and the SMOTE implementation in imblearn.over_sampling. Each model was only trained on Haralick features, as this consistently yielded better results in the previous two approaches.

First, a baseline Random Forest model was created using a RandomForestClassifier.

Then a second RandomForestClassifier was trained with the optional parameter class_weight set to 'balanced_subsample'. This generates a weight for each sample inversely proportional to its class's frequency: classes with fewer samples have higher weight, and classes with more samples have lower weight. For a balanced *subsample*, the proportional class frequencies are calculated on a tree-by-tree basis, since each tree is built on a different bootstrapped dataset, rather than based on class frequencies in the initial data set.

The third method used a BalancedRandomForestClassifier. This randomly undersamples every class but the minority class when bootstrapping, in order to synthetically balance the classes for each bootstrapped dataset given to a tree in the forest.

The final method involved a combination of over- and under-sampling, as this combination has been shown to be more effective than either alone. The Synthetic Minority Oversampling Technique or 'SMOTE' was used to oversample the minority class. As the name suggests, this creates new synthetic data points that belong to the minority class and adds these to the dataset. The new data points are created by selecting an existing sample from the class, and then translating it in the direction of its *k* nearest neighbours. Hence, the new sample is close to but not an exact duplicate of a true sample, and is presumed to represent a probable observation from this class. This is then followed by undersampling, using the BalancedRandomForestClassifier.

Hyperparameter tuning was performed for several parameters by performing a grid search. For each value of the hyperparameter in the grid search, 5 models were made, and their average performance (assessed using the weighted average f1-score) was graphed (Supp. Fig. 8 & 9). The best values were found with max_features = 3 and n_estimators = 1000.

Residual Neural Network:

An alternative to neighbour averaging using pooling is down-sizing via **Nearest-Neighbour scaling**, as implemented by Pytorch's torch.nn.functional.interpolate function. It was primarily used in our ResNet model to reduce the computational load on memory and enable our model to be trained within a reasonable amount of time. Since we wanted to preserve as much local information as possible, we used a scale factor of 0.25 to produce images of size 256 x 256 pixels, which was the maximum size that was practicable for training the ResNet model with the given memory constraints.

Justification and Architecture

The necessity of feature extraction in this task made a convolutional neural network an attractive choice for classifying the images and the complexity of the task necessitated a deeper neural network than would be reasonable using the standard convolutional neural network architecture. For this reason, a residual neural network (ResNet) was desirable, as the skip connections between layers allowed increased complexity of the model with fewer vanishing/exploding gradient issues.

Our ResNet consisted of an input layer, three "double-layers" with skip connections between, and an output layer. The double-layers consisted of two convolutional layers separated by batch-normalisation layers followed by a Rectified Linear Unit (ReLU) activation layer. The output layer was a 2-D average pooling layer followed by a fully-connected layer.

Early tests showed that the ResNet was overfitting – training accuracy would reach 100% swiftly while validation accuracy remained in the 55%-70% range. To rectify this, dropout was applied to the input layer and the double-layers. This increased the number of epochs needed to train from around 30 to around 70 and reduced the training accuracy to be in line with the validation accuracy - in the 70% to 80% range.

Hyper-parameter Tuning/Training:

The hyper-parameters of the ResNet we focused on tuning were batch size and learning rate. Due to the time it took to train the network, it was not possible to fully inspect the results of each

combination of batch size and learning rate – instead, we varied batch size in in the range [1, 50, 100, 150, 200, 250, 300] and learning rate in the range [0.001, 0.005, 0.01, 0.05, 0.1].

The batch size range was selected to investigate the differences between Stochastic Gradient Descent (where batch size = 1) and Mini-Batch learning. Batch size = 300 was the upper limit due to memory constraints.

The learning rate was selected with a lower bound of 0.001, the lowest reasonable learning rate that would still allow the model to train quickly enough to be useful, and an upper bound of 0.1, as any higher caused the error of the model to oscillate wildly.
It was found that a learning rate of 0.01 was optimal, and from there, batch sizes were tested to find which would give the best model. A batch size of 300 was desirable both for its higher accuracy and more consistent results (Supp. Fig. 9).

K-Fold Cross Validation
As seen in the images above, even with the reduced variability associated with the higher batch size, the accuracy still fluctuated during training. In an attempt to reduce the unpredictability of our model associated with these fluctuations, k-fold cross validation was implemented. However, the increased training time required with even 5 folds made this technique unviable, and instead, we continued to use the train-validate split as described in previous parts.

RESULTS:

F-1 score justification:
The F-1 score was selected as the evaluation metric of our model for a variety of reasons. The main driving factor however is the F-1's ability to compensate for class imbalance. This is due to its formula which gives equal weight to Precision and Recall via the Harmonic mean. This is useful in the context of our dataset due to the imbalance of Class 3. Note that the table below was generated with respect to the F-1 score. Further note that the final model was selected purely based on its predictive power. In medical imaging applications, models are only used to flag

potential complications, and would be further diagnosed by a human expert. Hence we deprioritised model explainability (cost of false negative is a lot greater than poor explainability).

Table 2: Summary of F1-Scores using different features and/or classifiers.

Validation f-1 Scores	Logistic Regression	Gradient Boosted Trees	Balanced Random Forest	ResNet
64x64 Neighbour Averaging	0.34	0.68	-	-
Haralick Features	0.86	0.88	0.83	-
Grayscale 256x256	-	-	-	0.89

Logistic Regression Results:

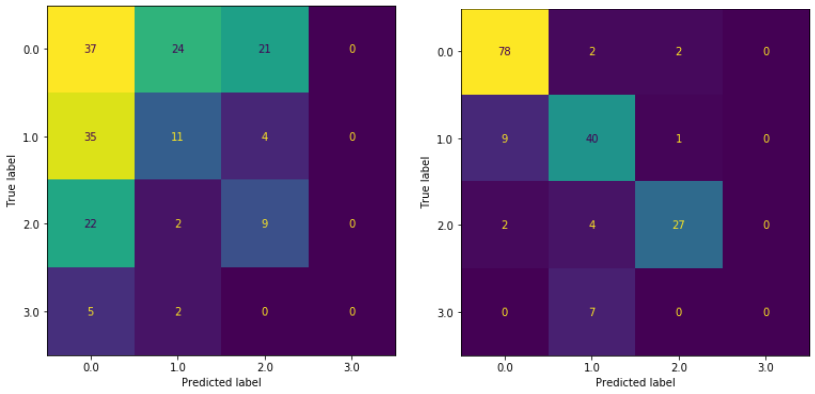


Figure 5: Confusion matrices, Nearest Neighbour Averaging (left) and normalised Haralick Features(right) for logistic regression on the validation set.

As expected, the grayscale neighbour averaging method performed poorly, with 0.61 training score and 0.34 validation score. However a logistic regression model with L2 normalisation (C=1.0) trained on Haralick features improved on this, with f-1 scores of 0.82 for training and 0.86 for validating (0.82 and 0.75 respectively without normalisation). This shows the model did not overfit to the training set, suggesting that L2 normalisation led to effective generalisability. However, neither of these logistic regression models made predictions for class 3, true or otherwise. This is most likely due to the class imbalance in the dataset.

Gradient Boosted Trees Results:

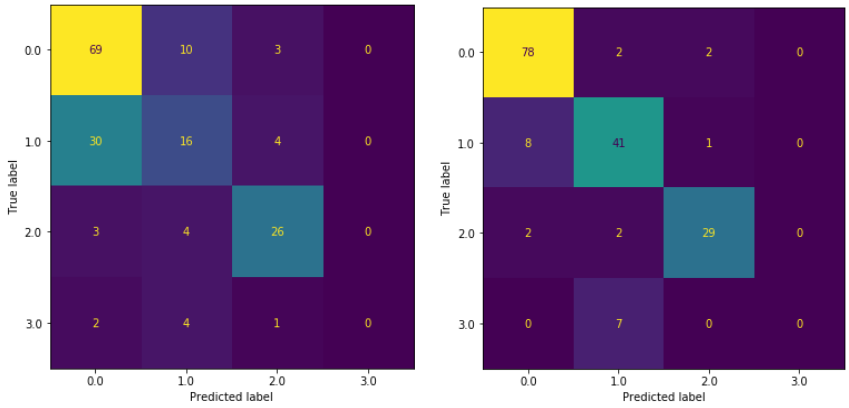


Figure 7: Confusion matrices, Nearest Neighbour Averaging (left) and Haralick Features(right) for gradient boosted trees on the validation set.

Similar to the results of Logistic Regression, the Gradient boosting classifier trained on 64x64 neighbour averaging features gave 1.00 training f-1 score and 0.68 validation score, a clear sign of overfitting as this method has 256 uninformative features. Despite this, the validation score of the Gradient Boosted Classifier was significantly better in comparison to Logistic Regression, which can be attributed to its baseline higher predictive power, On the other hand the Gradient boosting classifier trained on unnormalised Haralick features yielded an impressive 0.95 training f-1 score and 0.88 validation score (equal to using normalised data). Which can also be attributed to the same reasoning mentioned previously.

Balanced Random Forest Results:

Table 3: F1-Scores for each RF Classifier, by class and averaged

Class	**F1-Score**			
	Std RF	**class-weighted**	**undersampled**	**SMOTE/undersampled**
0	0.89	0.90	0.87	0.89
1	0.76	0.77	0.73	0.80
2	0.75	0.77	0.72	0.80
3	***0***	***0***	***0.26***	***0.17***
Macro average	0.60	0.61	0.64	0.66
Weighted Average	**0.79**	**0.80**	**0.77**	**0.82**

The baseline Standard Random Forest Classifier gave a weighted average f1-score of 0.79 on the validation data set. On a class-by-class basis, the f1-score was highest for the largest class (class 0) with 0.89, while the smallest class, class 3, was not predicted for any of the validation set, resulting in an f1-score of 0.

The Random Forest Classifier with bootstrapped class weights had approximately the same performance, with f1-scores improved by only 0.01-0.02 points, and again no samples predicted for class 3.

In contrast, the Balanced Random Forest Classifier, which performs undersampling, did assign samples to class 3, at the cost of recall for classes 0 and 1. The f1-score for classes 0-2 decreased while the f1-score for class 3 increased to 0.26. This resulted in an overall weighted average f1-score of 0.77.

Finally, the Balanced Random Forest Classifier combined with preceding SMOTE oversampling produced high f1-scores for classes 0-2 and a moderate f1-score of 0.17 for class 3, resulting in a weighted average f1-score of 0.82. This was the highest weighted average f1-score produced by a Random Forest method, and was still able to make predictions for the minority class (class 3).

These results are summarised in Table 3. For a visual representation, see Appendix (Supp. Fig. 10).

DISCUSSION:

In our comparison of feature extraction methods, we found that significant average pooling performed very poorly, both with logistic regression and gradient boosted classifiers. Hence, we determined that the important information for the classification task was at a local level within these images, rather than a global level. The average pooling approach is inappropriate for the problem, as the patterns that dictate the labels are independent of the global location. For example, a pattern that signals cancer can appear in different locations of the image, but they would be equally important. This naive approach cannot learn that information. Residual neural networks overcome this exact challenge.

Exploration of various Random Forest classifiers revealed the inherent difficulty in classifying samples given a significant class imbalance. Overall, the best RF model according to the weighted average f1-score was the SMOTE/Balanced RF classifier, which combined oversampling of the minority class and undersampling of the majority classes to address this imbalance. However, even with these methods employed, the f1-score for the minority class (class 3) was still quite poor, with only 0.25 precision and 0.14 recall. The model that was best able to classify instances from class 3 was the Balanced RF classifier which performed undersampling without SMOTE oversampling, but this came at the cost of misclassifying instances from the other classes. Given that the purpose of these models is to classify samples as tumorous or non-tumorous, this trade-off could be worthwhile, since the cost of misclassifying a sample as non-cancerous, when the patient in fact has a tumour, is significantly higher than the cost of incorrectly classifying a sample as having cancer. However, while the SMOTE/Balanced RF does not perform as well as the Balanced RF in predicting class 3 cancer types, it still predicts these samples as belonging to a cancer type, that is, to either class 1 or class 2. Since we do not know the cost of misclassifying a given cancer type as another cancer type (e.g., classifying class 3 cancer as class 1 cancer), we cannot meaningfully discriminate between models on this point. As such, the SMOTE/Balanced RF classifier is still deemed to be the best classifier out of the RF methods since it has the highest weighted-average f1-score, despite the better performance of the Balanced RF on class 3.

The overall comparison of the 4 models generated has allowed us to derive a variety of insights. Despite ResNet having the highest f-1 score, the models trained on Haralick features had similar scores, and hence, when considering model explainability, logistic regression and gradient boosted trees on Haralick features may be preferred over the ResNet. This is because the complexity of Neural Networks make it difficult to understand the way it has selected for features, whereas Haralick features are 13 explicit, well-defined representations of the image, such as contrast and gradient. It is very easy to run a feature importance test on these models and understand what is driving the predictions. This could be performed by adding random noise to a single feature at a time on a validation set, and determining how well the model performs on these noisy datasets. If noise in a specific feature compromises the performance of the model, that feature should be considered important.

CONCLUSION:

Ultimately after training various models on our 80:20 split dataset, we came to the conclusion that the Residual Neural Network was the model that offered the most prediction power with a f-1 score of 0.89. Despite our models impressive predictive power, there were still areas of improvement that we could have targeted given enough time. This includes decomposing the images into hematoxylin and eosin stained sections, a popular feature method extraction seen in published research. Another area of improvement would be to implement bagging in order to compensate for the lack of class 3 images present within the dataset.

Improvements specific to the Residual Neural Network that we would implement would be reducing variability using decaying learning rates over subsequent epochs and potentially increasing the complexity of the model to allow it to learn the problem more deeply. Increased complexity in the model may result in overfitting. Potential solutions for overfitting include artificially increasing the training set by applying image transformations to the data and adding more regularisation (weight decay, dropout, etc.).

REFERENCES:

Komura D, Ishikawa S, Machine Learning Methods for Histopathological Image Analysis, Computational and Structural Biotechnology Journal, Volume 16, 2018, Pages 34-42, ISSN 2001-0370, https://doi.org/10.1016/j.csbj.2018.01.001. (https://www.sciencedirect.com/science/article/pii/S2001037017300867)

Zheng, W., Jin, M. The Effects of Class Imbalance and Training Data Size on Classifier Learning: An Empirical Study. SN COMPUT. SCI. 1, 71 (2020). https://doi.org/10.1007/s42979-020-0074-0

Rexhepaj E, Agnarsdóttir M, Bergman J, Edqvist PH, Bergqvist M, et al. (2013) A Texture Based Pattern Recognition Approach to Distinguish Melanoma from Non-Melanoma Cells in Histopathological Tissue Microarray Sections. PLOS ONE 8(5): e62070. https://doi.org/10.1371/journal.pone.0062070

Sarwinda, D., Paradisa, R. H., Bustamam, A., & Anggia, P. (2021). Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer. Procedia Computer Science, 179, 423-431. https://doi.org/10.1016/j.procs.2021.01.025

Pytorch. (2022). *Python API Reference*. Retrieved from Pytorch:
https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader
https://pytorch.org/hub/pytorch_vision_resnet/
https://pytorch.org/docs/stable/generated/torch.nn.functional.interpolate.html

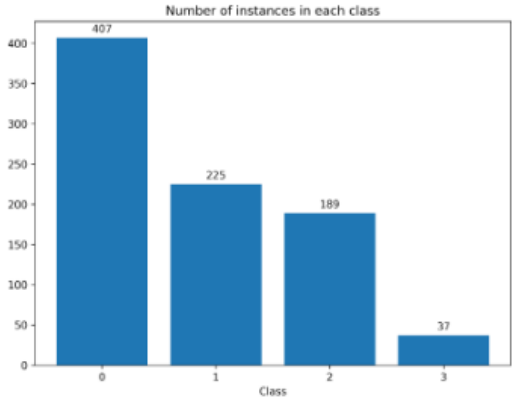
SciKit Image. (2022). *API Reference*. Retrieved from SciKit Image: https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.block_reduce

SciKit Learn. (2022). *API Reference*. Retrieved from SciKit Learn:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

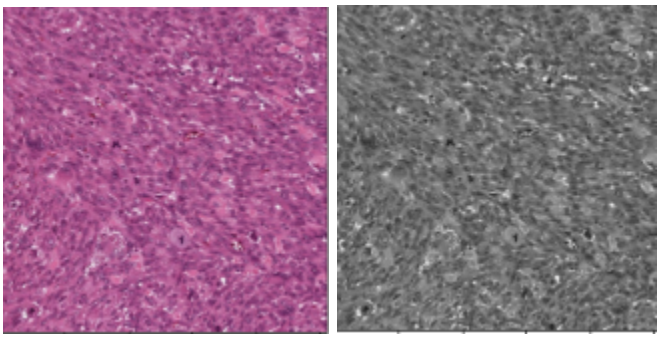
https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

Mahotas. (2022). *API Reference*. Retrieved from Mahotas: https://mahotas.readthedocs.io/en/latest/features.html#haralick-features

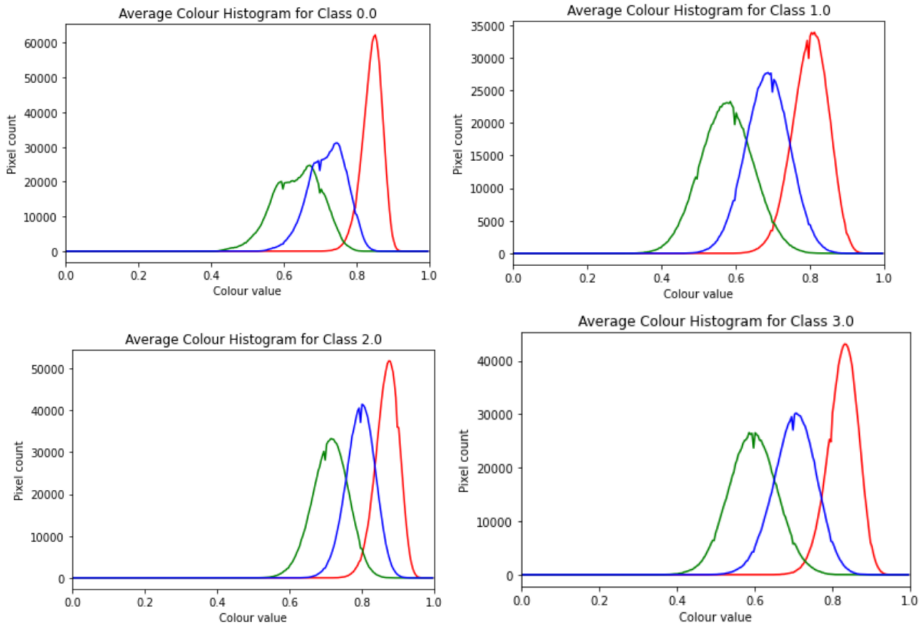
APPENDIX



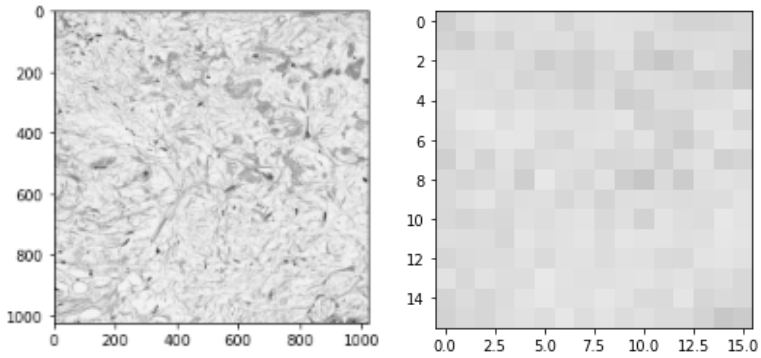
Supplementary Figure 1: Distribution of classes in the provided dataset.



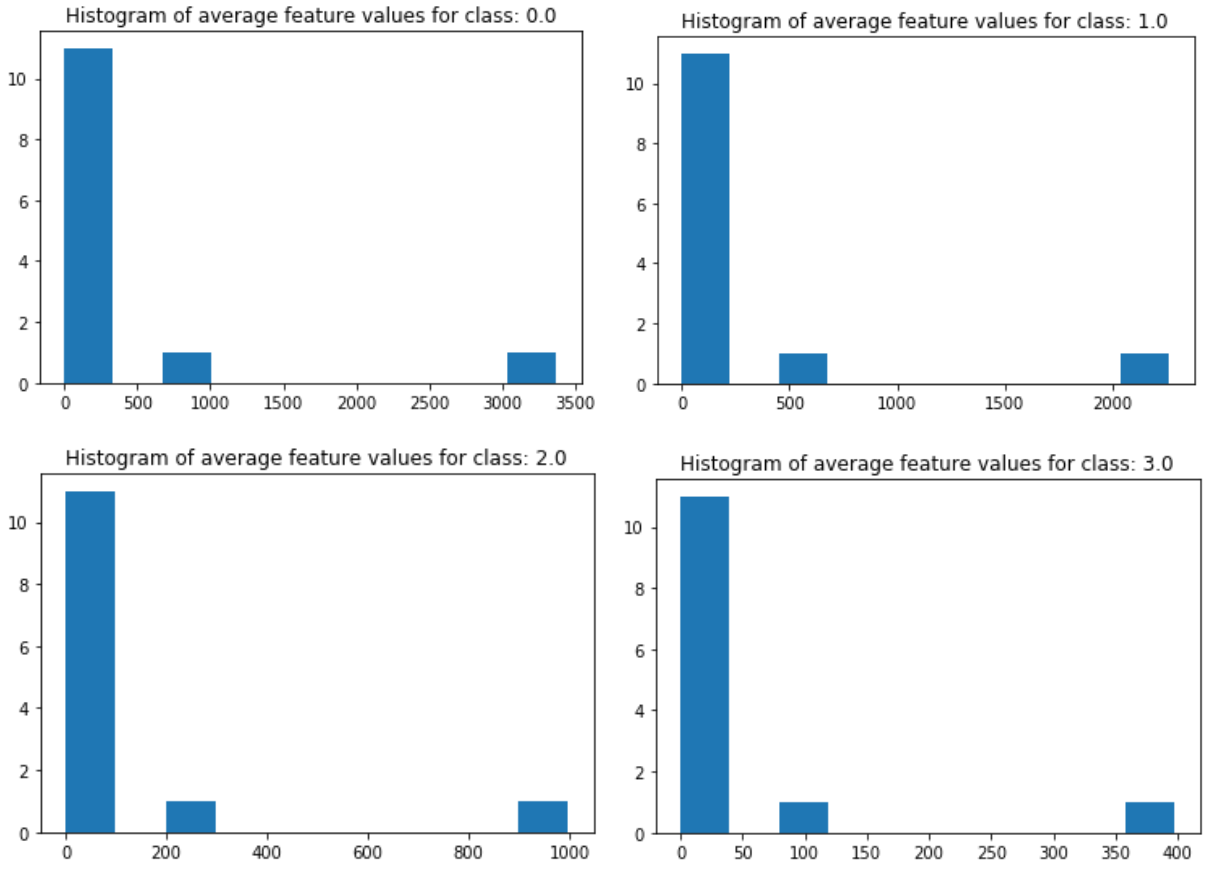
Supplementary Figure 2: A comparison of an image from our dataset in both its original 3-colour channels (left) and grey-scale (right).



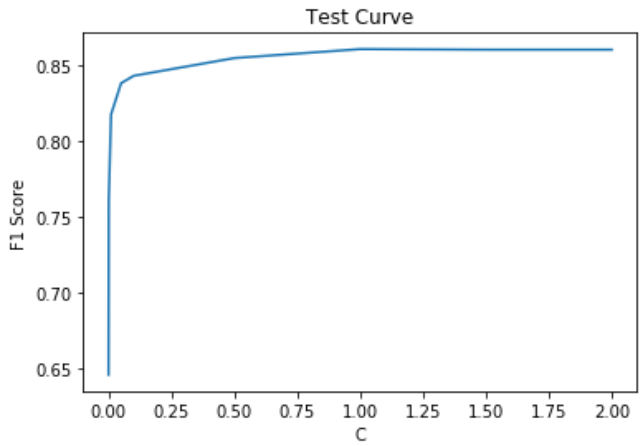
Supplementary Figure 3: Average colour histograms for a random subset of each class



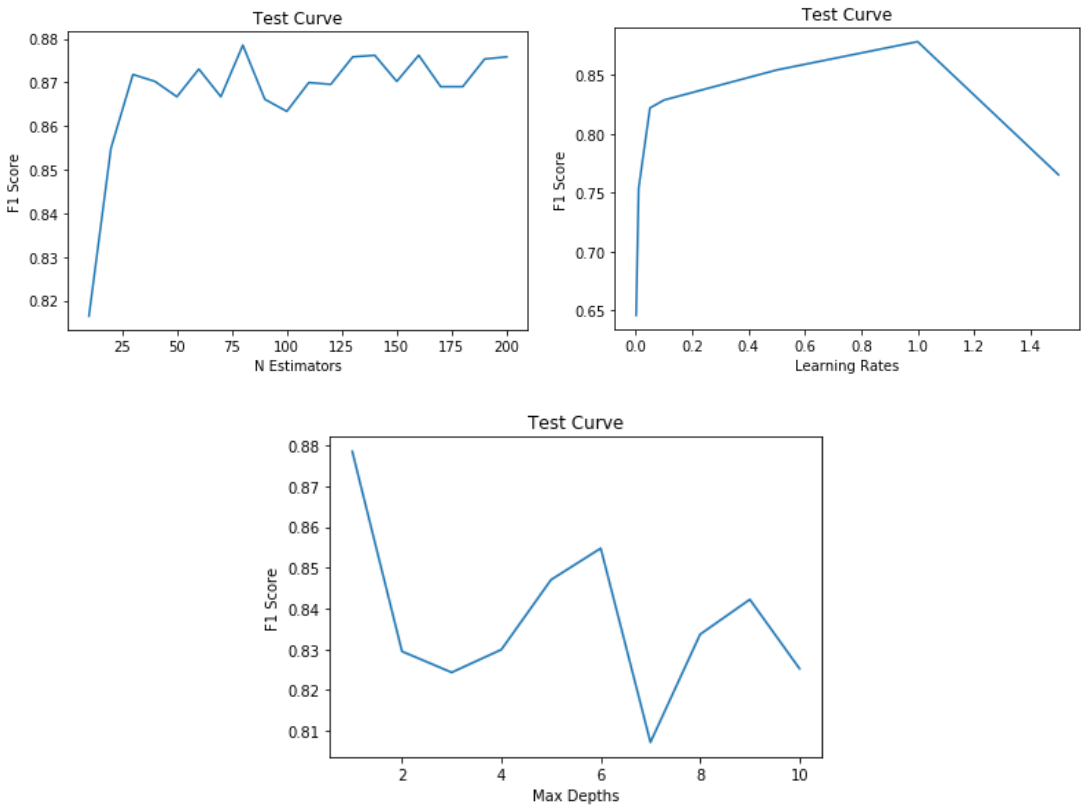
Supplementary Figure 4: A comparison of an image from our dataset in both original dimension (left) and simplified dimension (right)



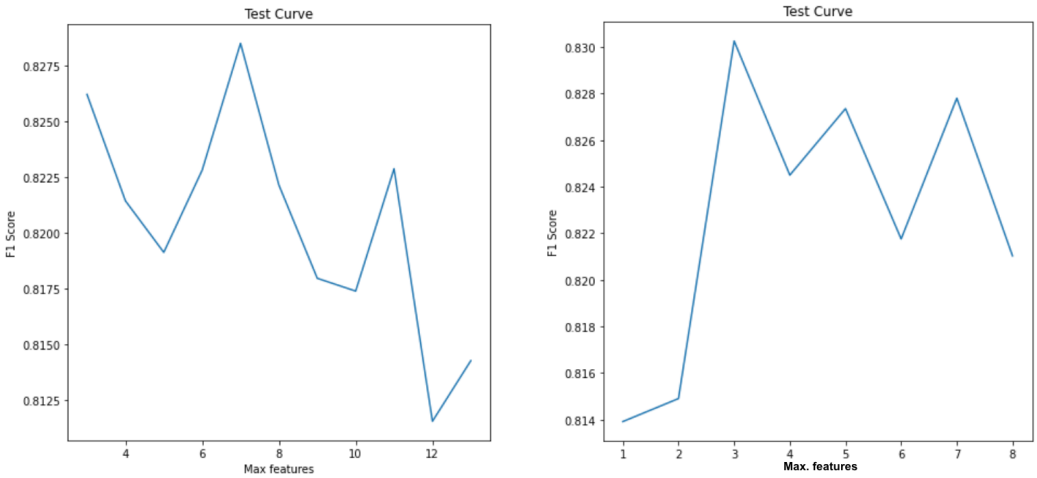
Supplementary Figure 5



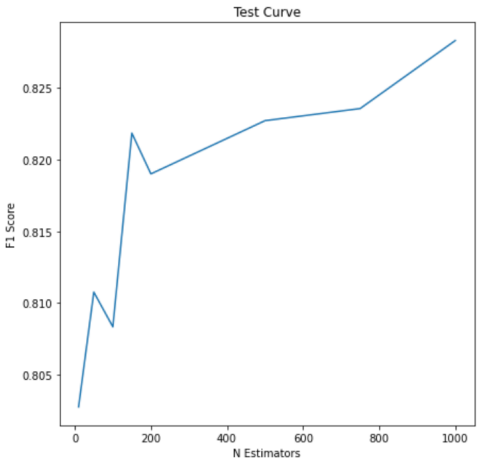
Supplementary Figure 6: Finding the optimal C value for normalised Haralick features.



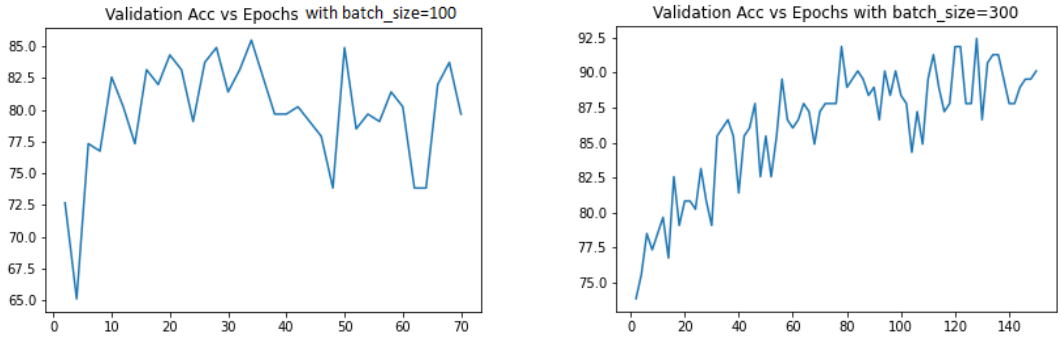
Supplementary Figure 7: Parameter tuning for gradient boosted classifier on Haralick features.



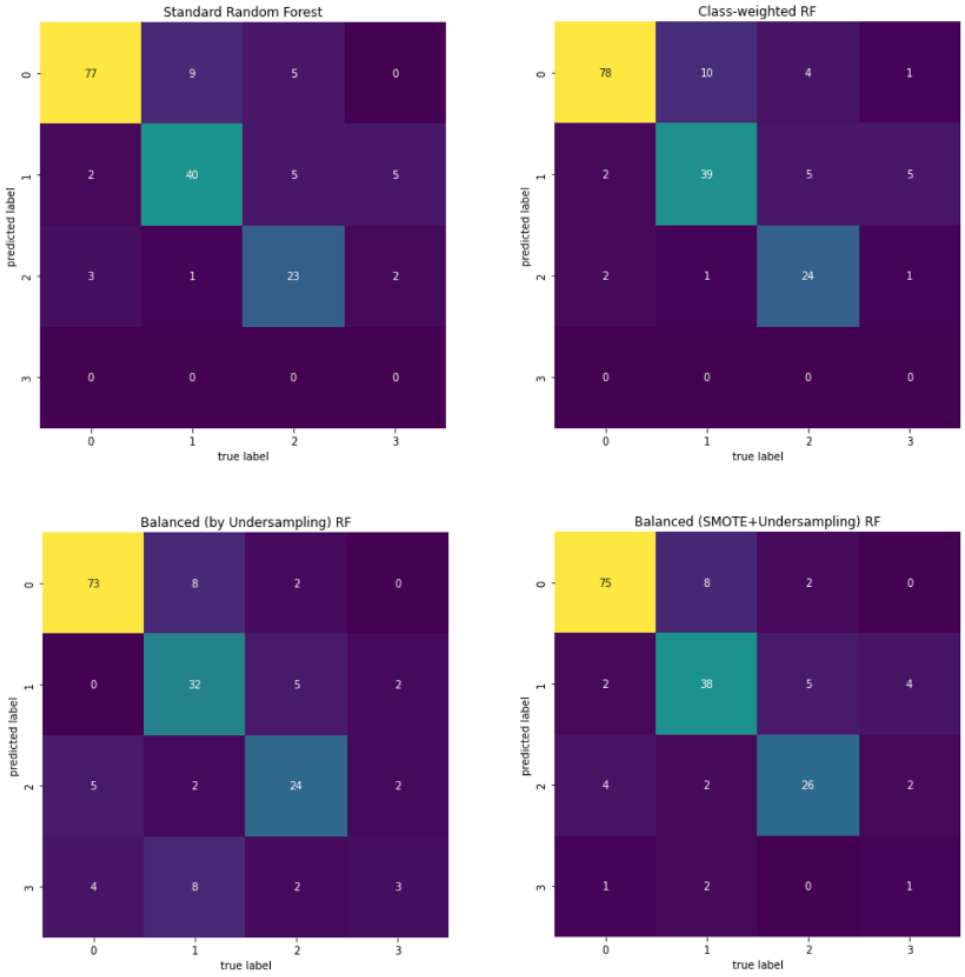
Supplementary Figure 8: Performance of SMOTE/Balanced Random Forest Classifier with different values for the max. features per tree.



Supplementary Figure 9: Performance of SMOTE/Balanced Random Forest Classifier with different values for the number of trees (estimators) within the forest.



Supplementary Figure 10: Batch Size hyper-parameter tuning for ResNet with batch_size=100(left) and batch_size = 300(right).



Supplementary Figure 11: Confusion matrices of validation set performance with the Random Forest Classifiers.