Texture-based classification for oral cancer detection: Implementation and performance analysis of deep learning approaches

Jo Gay and Hugo Harlin

Advisors: Joakim Lindblad and Nataša Sladoje

ABSTRACT

Texture detection and analysis has been shown to be highly effective in various image classification tasks. Local binary patterns (LBPs) are a promising technique for texture recognition, and a wide range of variations of the original work by Ojala et al. [11] have been published in recent years, some of which utilize LBPs in convolutional neural networks (CNNs).

In this paper, three state of the art texture detection CNN architectures, including two based on LBPs, are adapted to classify images from a dataset consisting of oral cancer cells and healthy cells. Results obtained by Wieslander and Forslid [18] using VGG and ResNet architectures (not specifically designed for texture detection) are used as a reference. Our results indicate that LBP-based CNNs perform better than traditional CNN architectures on this task. The best performing model, published by Juefei-Xu et al. [3], gave an accuracy of 81.03% (f_1-score 84.85%) which is an improvement of 0.5% in accuracy and 9% in f_1-score over VGG.

I. INTRODUCTION

An important element of efforts to reduce cancer mortality is early detection of cancerous cells. Screening programs such as that for cervical cancer are highly effective in preventing advanced stage cancers [12]. One obstacle to the introduction of screening for other cancer types such as oral cancer is the cost associated with manual inspection of the resulting cell samples. Computer assisted analysis of cytology slides would offer a significant reduction in these costs, and recent advances in image analysis techniques make this more feasible. In particular, methods focusing on texture analysis are likely to be useful for discrimination between cancerous and healthy cell samples.

A powerful tool for describing textures is local binary patterns (LBPs), developed by Ojala et al. [11]. LBPs focus on the pattern of variations in intensity between a pixel and other (neighbouring) pixels, instead of using the image intensity values directly. A neural network can then be trained to recognize the range of patterns found in different types of images. Many methods have been proposed which either use LBPs directly, or are inspired by them, and show promising results on a range of different image classification tasks.

In this paper three recently published image analysis methods, suitable for texture analysis and classification, are adapted for oral cancer detection [3],[5],[8], and the results are compared with results from work done on the same dataset by Wieslander et al. [17] using state of the art convolutional neural networks (CNNs), ResNet [1] and VGG[14]. Two of the methods tested here are based on local binary patterns, and the third develops a vector field approach for texture classification.

The performance, measured by F_1 scores, of both of the methods based on LBPs is higher than that obtained with the state of the art CNNs, demonstrating the effectiveness of LBPs for this type of image classification task. Results for the vector field method are not as good, indicating that this method may be less appropriate for classifying single-cell images.

II. BACKGROUND

A. Local Binary Patterns

Local Binary Patterns are an efficient way of extracting features from image data, designed to detect different textures. They were first conceived by Ojala et al. [10] in 1996, following their earlier work on grayscale differences, and they work as follows:

The intensity value of each pixel in an input image is compared with the intensity values of P neighbouring pixels, sampled in a circle with radius R around it. When the central pixel has the lower intensity value, a result of 1 is recorded, otherwise 0. This process results in a binary array of length P, which can be viewed as a pattern index. More formally, in a single channel of an image, with intensity g_x at pixel x, the pattern index for a pixel c is given by

$$LBP_{R,P}(c) = \sum_{i=0}^{P-1} s(g_i - g_c)2^i$$

where the step function $s(x)$ is 1 for $x \geq 0$ and 0 for $x < 0$. These patterns are invariant to any global monotonic grayscale transformation (e.g. differences in illumination), since they only consider the relationship between the intensity of a pair of pixels.

The pattern indices themselves cannot directly act as an input to a convolutional neural network, because there is no relationship between the distance between two pattern indices and the similarity of the patterns. A histogram of all the patterns found in the image can however be used as a feature array for input to a neural network or other classifier.

In an extension [11] to this work, the same authors developed a rotation-invariant version, in which they also introduced the concept of uniform and non-uniform patterns. Since the neighbouring points are arranged in a circle, a substantial degree of rotation invariance can be incorporated by combining patterns which are rotational equivalents of each other, such as 00011010 and 00110100. This does not provide full rotation invariance because the points are at fixed locations (e.g. for $P = 8$, neighbours are located at 0, 45, 90, 135 degrees etc), and possible rotations in between these angles are not fully addressed. However, as well as a degree of rotational invariance it also reduces the number of distinct patterns from 256 to 36 (in the case of $P = 8$), which reduces the noise in the resulting histograms.

The authors also note that in general, a high proportion of the patterns found in natural images come from a specific subset of all the possible patterns, which they term 'uniform' patterns. In each pattern, there is a number of transitions where the bit string changes between 0 elements and 1 elements (and vice versa), and they define a uniform pattern as one with no more than two transitions. So 00011000 is a uniform pattern, with two transitions between 0 and 1, and 00010100 is a non-uniform pattern, with four transitions. All non-uniform patterns are combined into a single category, in order to further concentrate the information in the resulting histograms.

Whilst impressive results have been obtained by using such histograms as feature arrays in deep neural networks, the spatial relationships are lost when forming the histograms.

B. Convolutional Neural Networks for Texture Classification

We are interested in evaluating methods which combine the power of Convolutional Neural Networks and Local Binary Patterns to reach improved classification performance based on texture.

One technique that can be used to incorporate LBPs within deep convolutional neural networks is to implement them as hardcoded, non-trainable filters in convolutional layers. The simple case would be to use a 3x3 filter with R = 1 and P = 4, where the four points on the circle would align with the existing pixels on the grid. If R and P is chosen such that the points on the circle don't align with the center of pixels on the grid, then these values can be calculated by interpolation. This means that standard nxn convolutional filters could be used to extract LBPs up to radius (n-1)/2. The probability of CNNs doing this spontaneously is infinitesimal, due to to the vast search space of all possible filters.

The pattern indexes themselves aren't suitable to be used in CNNs because the indexes aren't ordered or correlated, and performing operations such as averaging or interpolation of indexes is meaningless.

A number of authors have proposed methods based on combinations of fixed and/or binary difference filters which emulate LBPs to a greater or lesser extent and we test two such models here. We also test a third model designed for texture classification but not based on local binary patterns.

The remainder of this section outlines the three original models evaluated in this paper. Details of the oral cancer cell image dataset used in our study are given in section III. In the following section we describe our experimental setup, including data augmentation and how the three models were adapted for classification of the images in this dataset. Our results are shown in section V and we conclude with a discussion and suggested future work.

Juefei-Xu et al.: Local binary convolutional neural networks [3]

In this method, local binary patterns are adapted to form local binary convolutional layers. The authors observe that the calculation of an LBP index can be achieved by the application of P sparse binary filters in parallel, activated through a Heaviside Step function, and then taking a weighted sum of the results, where the weights are powers of 2. The binary filters would consist of all zeros except for the central pixel plus one other (assuming that each point in the circle falls precisely on one pixel, in reality an interpolation of neighbouring pixel intensities would be required). The weighted sum can be calculated by a 1x1 convolutional layer with weights 2^i for $i = 0..P-1$. This reformulation of LBPs as a pair of fixed convolutional layers is illustrated in Figure 1.

The authors then generalize and transform the model in three ways:

1) *Transform pattern indices into weighted sums.* The pattern indices can only be used in a histogram, because similar indices do not imply similar patterns. To overcome this, in step (c), instead of combining the P differences with individual powers of 2 to create a unique index for each possible pattern, the difference maps are combined in a weighted linear sum, with learnable weights for each position $v_0, ..., v_P$.
2) *Rectified linear (or sigmoid) activation.* Instead of binary outputs for each map, obtained through the Heaviside step function, a rectified linear activation is used in step (b). This function is differentiable so allows for gradient descent based optimization.
3) *Random initialization of filters.* Instead of using P filters giving the difference maps between the central pixel and one neighbour (as shown in step (a)), filters are randomly initialized with ± 1 in a fixed proportion of positions, with the remaining ones being zeros. The output from each filter is a feature map containing a combination of neighbouring pixel values and differences.

This results in a two-dimensional feature map of local pattern transformations, of the same size and shape as its input image. These paired local binary convolutional layers (LBC units) can therefore be used as a drop-in replacement for a convolutional layer in any model architecture.

The architecture presented by the authors is based on ResNet [1] and includes several sequential LBC units, each with an accompanying identity connection. Different combinations of the number of LBC units (10, 20, or 75) and

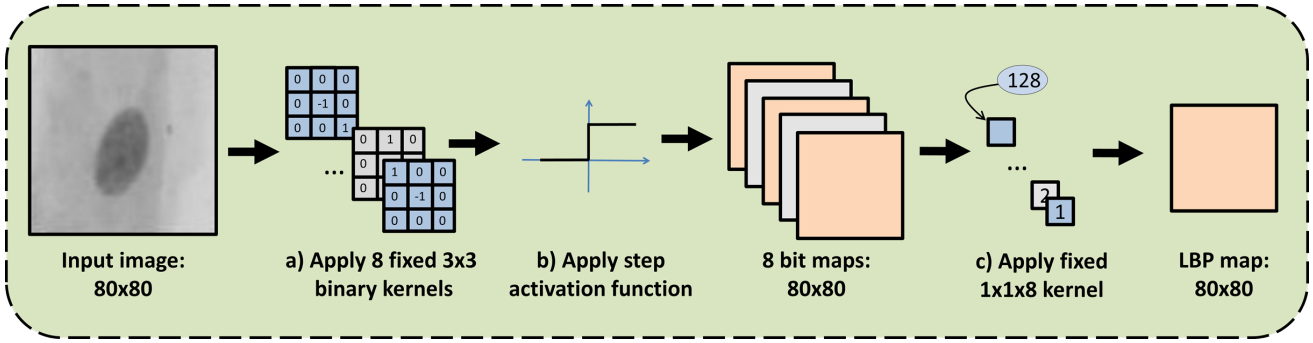


Fig. 1: Implementation of LBPs using two convolutional layers. The 3x3 fixed convolutional filters (a), activated by the Heaviside step function (b), approximate the LBP calculations for $R = 1, P = 8$. The 1x1x8 kernel (c), with weights fixed at different powers of 2, transforms the 8-channel binary output for each pixel in the input image into the corresponding pattern index between 0 and 255.

the sparsity of the fixed filters (10%, 50% or 90% non-zero elements) were suggested in the original paper, depending on the application. These LBCNN layers are followed by a fully connected (dense) layer with 512 nodes, and a fully connected output layer which performs the classification.

The authors show that their LBCNN model can produce results comparable to standard CNN implementations, with a fraction of the parameters, allowing larger and deeper networks to be trained within the same timescales.

Marcos et al.: Rotation Equivariant Vector Field Networks [8]

The method proposed in [8] does not use LBPs but develops a novel vector field network, known as RotEqNet. This is an extension of earlier work by the same authors [9], which proposed a single rotationally invariant convolutional layer, in which each filter is copied R times at different rotations. The filters are unusually large with a size of 35x35 pixels, which the authors argue will be superior to smaller filters in detecting patterns and textures with a broad range of frequencies. The filters are organized in groups, consisting of one 'canonical' filter and rotated copies. Each copied filter has the same weights as its related canonical filter and is only trained in conjunction with it. The maximum activation from any filter in the rotational group acts as input to one or more subsequent fully connected layers for classification. Since the filters are rotated, weights that end up outside a filter when rotated are set to zero. The filters can be thought of as circular filters inscribed in a square.

In the extended version, the angle at which that maximum activation occurred is included alongside the activation value, forming a two-dimensional vector field output. A smaller 9x9 filter size is also used, with multiple layers and using a ReLU activation function. This architecture is illustrated in Figure 2. The authors develop techniques for performing batch normalization and spatial pooling on these 2D vector fields, allowing their RotEqNet layers to be fully incorporated into standard CNN architectures.

Batch normalization is performed only on the magnitude of the vectors, this can be represented as a standard scalar matrix and normalization can be done as usual. The point of batch normalization is to limit the effect of extreme outliers and thus improve performance, and as such normalizing the already bounded angle is meaningless.

Spatial pooling is done similarly to spatial pooling of a scalar map. The magnitudes in the specified non-overlapping window are compared and the vector with the largest magnitude is selected.

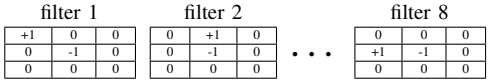
Since only the canonical filter in each RotEqNet layer is trainable, the computational cost of RotEqNet layers is small compared to fully trainable convolutional layers of the same size. Multiple RotEqNet layers (and other layers) can be applied to create a deep neural network architecture which performs comparably to much larger networks in terms of the number of trainable parameters.

Li et al.: Face spoofing detection with local binary pattern network [5]

This model consists of four modules: (1) a series of standard convolutional layers, (2) a local binary pattern extraction module, (3) a custom gate layer, and finally (4) conventional dense layers.

1) Convolutional Module: The first part of the network consists of a convolutional layer with N 3x3 filters, followed by batch normalization. This is followed by a maxpool layer with stride (2,2). The last convolutional layer consists of 2*N 3x3 filters.

2) Local Binary Pattern Extraction Module: In this layer, a set of 8 fixed 3x3 filters are used to find the difference between the central pixel and its 8 neighbours. The center value of all filters is -1, and one other value is $+1$ depending on the filter number. The top left of filter 1 has the value $+1$, for filter 2 the top center value is $+1$, top right for filter 3 and so on. Filter 8 has the value $+1$ at center left. All other filter values are set to zero.



The output from each filter is passed through a batch normalization filter and then a sigmoid activation layer. The filtering, batch normalization, and activation functions are applied in parallel, resulting in eight tensors. The tensors are then added channel by channel, yielding a single tensor

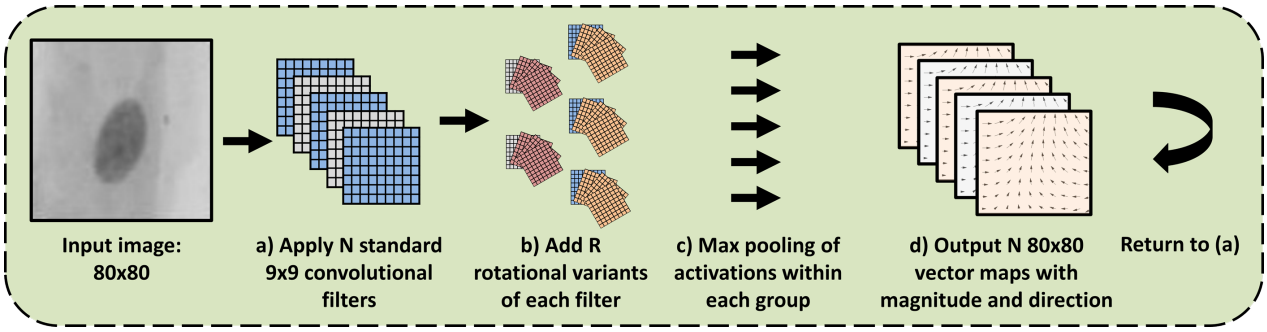


Fig. 2: Rotationally equivariant network (RotEqNet). Each standard convolutional filter (a) acts as the canonical filter for a group of rotationally similar filters (b). The maximum activation value and the responsible rotational angle form a 2 dimensional vector field output (d).

which is passed to the next part of the network, the gate layer.

3) Gate Layer Module: The gate layer consists of passing the output tensor from the LBP layer to eight parallel sets of filters, similar to the architecture of the LBP layer. The input tensor is filtered so that values outside the interval [y-1, y] are set to zero, where y is the filter number (1-8). This yields eight tensors where the values of each tensor are in the range [0,1], [1,2], ... [7,8] for filters 1,2, ... 8. The filtered tensors are then passed through a custom activation function:

$$x^* = \begin{cases} 4\big(x-y\big), & \text{if } x < (y-0.5) \\ -4\big(x-(y+1)\big), & \text{if } x \geq (y-0.5) \end{cases} \quad (1)$$

This is followed by an average pooling layer that reduces each channel dimension to 1x1. In our case the dimension of the channels in the input tensor to the gate layer is 40x40 so an average pooling layer with a 40x40 kernel was used. Finally, the eight filtered vectors are concatenated and sent to the dense layer module. The output from this model can be viewed as the filtered eight-bin histogram of the input tensor.

4) Dense layer Module: The last part of the network consisted of a dense layer with 512 nodes, batch normalization, a ReLU activation layer and finally a dense layer for classification with sigmoid activation.

This method showed promising results on classification of different types of facial images.

III. Data-set

The data-set used in this study consisted of 10274 80x80 grayscale images of cells from six patients, three with cancer and three healthy. Cells were sampled from the mouth of the patient and placed on a glass and photographed. The images were then cropped so that each image showed one cell centered in the image. Out of 15 focus levels available for each image, the one with the best focus was chosen by convolving the image with the Laplacian and calculating the variance [17].

Each image is labelled with the diagnosis of the patient, so all cells from one glass have the same label. Whilst we would not expect to find any cancerous cells within a sample taken from a healthy patient, it is possible that some of the cells taken from a cancer patient may be healthy cells. This may have a limiting effect on the classification power of the methods being tested.

Patient	Glass	Diagnosis	Cells, size 80x80
1	3	Healthy	1965
1	4	Healthy	1851
2	5	Healthy	1382
2	6	Healthy	956
3	7	Healthy	863
3	8	Healthy	738
4	12	Cancer	226
5	36	Cancer	534
6	37	Cancer	963
6	38	Cancer	796

TABLE I: Oral cancer dataset: patients and glasses

Some example cells from healthy and cancer patients are shown in Figure 4.

The division of samples between patients and glasses can be seen in Table I. The glass number is given because it is probable that the images from one glass are correlated with each other, and this needs to be taken into account when partitioning data for training and testing. For this reason the data was split into three folds, as shown in Table II. Each fold is a partition of the full dataset into a training set and an evaluation set, such that the training and evaluation parts each contain both healthy and tumorous cells, and the evaluation set contains no cells taken from glasses used in training. For each fold, a random 20% sub-sample of the training data was chosen as a validation dataset. This was used to prevent over-fitting.

The models were trained on each fold independently and the confusion matrices of each run were added together to produce the final result.

	Glasses for training		Glasses for evaluation	
	Healthy	Tumor	Healthy	Tumor
Fold 1	3,4,5,6	37,38,12	7,8	36
Fold 2	3,4,5,6	36,37,38	7,8	12
Fold 3	3,4,5,6	12,36	7,8	37,38

TABLE II: Partition of glasses in each fold

IV. Experimental Setup

In this section we discuss how the data was pre-processed and augmented for our experiments, and how the three

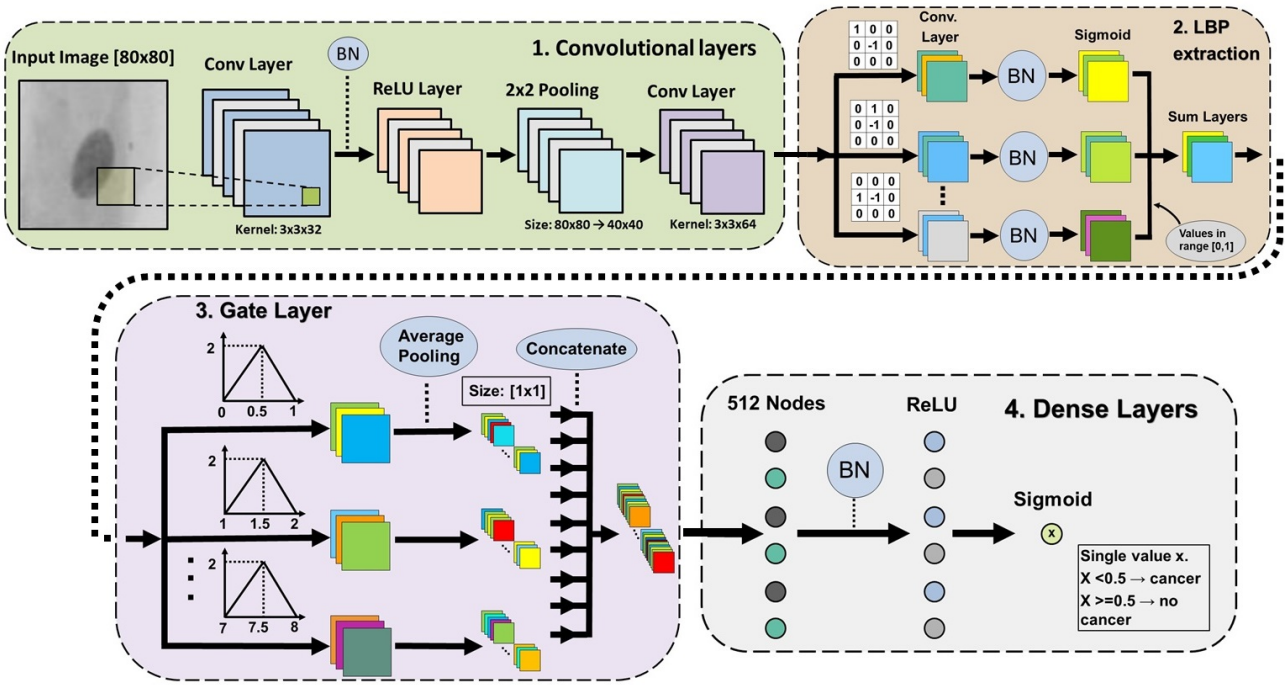


Fig. 3: Architecture of the network proposed by Li et al.[5], adapted to our data set.

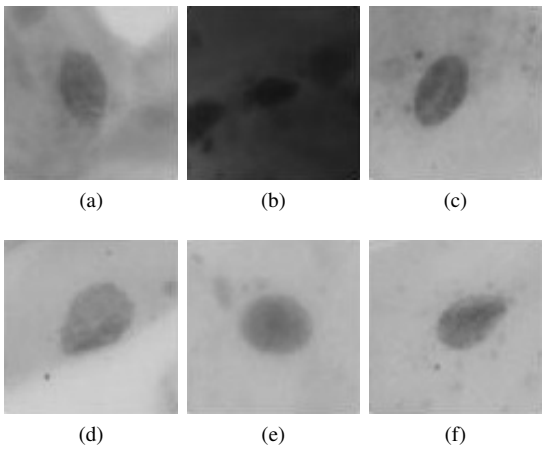


Fig. 4: Sample images, (a) - (c) are from a healthy patient and (d)-(f) are from a patient diagnosed with oral cancer. Note that intensity normalization was used on the images prior to training.

network architectures discussed in the background section were modified to perform optimally on our data set. The original configurations described by the authors [3],[5],[8] were used as a starting point, and hyper-parameters such as the number of filters in each layer, number of layers, and learning rate schedule were tuned. Two different optimizers were tried: the stochastic gradient descent optimizer (sgd) and the Adam optimizer. Unless otherwise specified, parameters were chosen to replicate the original implementations.

A. Data Selection and Augmentation

To reduce the effect of any differences between the lighting or staining conditions in different samples, each image was normalized by subtracting the mean and dividing by the standard deviation. To balance the training data an equal number of healthy and cancer images were selected, using as many images as possible. Since there are more healthy images in each fold, this meant that only a subsample of the healthy images was used.

The training (and validation) data was then augmented prior to training. The images were mirrored in three ways: vertically, horizontally, and both vertically and horizontally. Together with the original images this resulted in four times more data. All images were then rotated 90° counterclockwise, yielding a data set eight times the size of the original set. None of these transformations requires any interpolation, so no artifacts are introduced to the data. The test data was not balanced or augmented.

B. Juefei-Xu et al.: Local binary convolutional neural networks [3]

We adapted the Lua code provided by the authors [2] to implement this model in Python. We added an additional batch normalization layer, after the LBCNN layers (in addition to batch normalization before each LBCNN layer), as the model was unable to train without this. We compared filter sparsities of 0.1, 0.5 and 0.9, and used 10 or 20 LBCNN layers, each with 512 filters operating on 128 intermediate channels. Our average pooling layer uses a filter size of (5x5) per the authors' implementation, but since our original image size is larger, this leads to output feature maps of size (16x16) instead of the (6x6) quoted in the original paper. The best results (shown in Table III) are obtained with 10 layers and sparsity 0.9 (90% of elements non-zero), where an additional standard convolutional layer with 16 (3x3) filters was added after the LBCNN layers and before the fully

connected layers.

C. Marcos et al.: Rotation Equivariant Vector Field Networks [8]

For this model, as well as referring to Matlab code published by the authors, we used PyTorch code which was made available by Anders U. Waldeland at the Norwegian Computing Center [15]. The network architecture implemented, consisting of 3 RotEqNet layers followed by two fully connected layers, was designed for images of size (32x32), so we added an extra RotEqNet layer to adapt this to our (80x80) images.

Results using this setup are shown in Table III. The original parameters include the number of channels (6, 16, and 32 in the first three RotEqNet layers, 32 in the additional RotEqNet layer, and 128 in the fully connected layer); the optimizer (stochastic gradient descent with weight decay 0.01); the learning rate schedule (starting from 0.1 and reducing by a factor of 10 at epochs 20, 40, and 60); and the number of filter rotations used (17). 90 training epochs were used with a batch size of 100.

Improved results were obtained by using the Adam optimizer, with a lower initial learning rate of 0.01, and adjusting the padding in the convolutional layers to remove early padding (thereby shifting the focus of the model towards the centre of the image, based on our knowledge of the image data). This network is quite small compared to the others, and we also tried increasing the number of filters in each layer, from (6, 16, 32, 32, 128) to (64, 64, 64, 64, 64). This increased the accuracy and F-score considerably and it may be worthwhile increasing these further. However, memory constraints mean that batch sizes have to be reduced to accommodate this, which limits the number of filters that can be used in practice.

D. Li et al.: Face spoofing detection with local binary pattern network [5]

The lead author kindly provided us with Matlab code for this model. We referred to this when reimplementing the model in Python using Keras.

Due to the complex architecture of this model, tests were run using a small batch size of 20. The memory requirements for larger batch sizes were not feasible in our system.

Results are shown in Table III. Original parameters include the number of filters used (32 channels in the first module and 64 thereafter) and the optimizer (stochastic gradient descent with momentum 0.9 and weight decay 0.0005). The learning rate decay schedule is quite aggressive in the original paper, with the rate starting from 0.001 and halving in each epoch. An improvement over the original parameters was obtained by decreasing the rate of decay of the learning rate to a factor of 0.8 per epoch, and increasing the initial rate to 0.01.

Several other configurations were tested: changing the number of filters, adding a shortcut connection from convolutional module directly to the dense layers, moving the convolutional module to after the gate layer module, and adding additional convolutional layers. The Adam optimizer was also tried, yielding no improvements.

V. Results

Each of the models described above was implemented in Python and run with hyperparameters (including learning rate schedule, weight decay, momentum, optimizer, number of layers, and number and size of filters) as described by the authors, independently on each of the three training folds listed in Section III. Results from these experiments are shown in Table III, marked as 'original' hyperparameters. Model hyperparameters were then tuned as described in Section IV to maximize performance on the oral cancer data set and the best combinations found are also shown in Table III.

All results given are averages over the three test folds, weighted by the number of samples in the test partition of the fold.

Model	Hyper-parameters	Accuracy	Precision	Recall	F-Score
Wieslander & Forslid VGG	original	80.66	75.04	80.68	77.68
Wieslander & Forslid ResNet	original	78.34	72.48	79.00	75.51
Juefei-Xu et al.	original (20 LBC units, sparsity 0.9)	79.84	79.87	88.29	83.87
Juefei-Xu et al.	with additional convolutional layer (10 LBC units, sparsity 0.9)	81.03	80.99	89.1	84.85
Marcos et al.	original	53.3	71.41	64.4	67.73
Marcos et al.	additional filters	68.37	70.45	78.83	74.41
Li et al.	original	75.08	76.66	83.95	80.14
Li et al.	increased initial learning rate $l_0 = 0.8$, N = 32	80.70	78.97	90.40	84.30

TABLE III: Summary of best results.

In each case, the metrics presented are:

1) Accuracy: The proportion of predictions that were correct
2) Precision: The proportion of healthy predictions that corresponded to healthy cells
3) Recall: The proportion of healthy cells classified as healthy by the model
4) F_1-score: The harmonic mean of the precision p and recall r, $f_1 = 2pr/(p + r)$.

Since the test data is somewhat unbalanced, with an average of 66% healthy cells, the F_1-score is the most appropriate for evaluating results. On this basis, both the Juefei-Xu et al. and Li et al. models show an improvement over the performance of state-of-the-art CNNs, VGG and ResNet, which have not been particularly adjusted for texture-based classification.

VI. Discussion and future work

The experiments presented here show that the use of Local Binary Patterns, in particular the methods proposed by [3] and [5], can increase the classification power of neural networks for the oral cancer dataset. Another state of the art method tested, the rotationally equivariant network of [8], did

not match the performance of the LBP-based models, providing evidence that LBPs, above other texture classification techniques, are an appropriate tool for classifying cells. This corresponds well with our knowledge that pathologists find the fine texture of the nucleus a highly discriminative feature in cancer detection.

Several other methods of adapting LBPs have been proposed, including extended LBPs [19], and recently, multi-radial colour LBPs [13] where the RGB color channels are treated separately with LBP filters of different radius, and the results are concatenated into one output tensor. Another approach, using multidimensional scaling to transform the unordered pattern indices onto an ordered metric space, was proposed by [4] and showed promising results when combined with ResNet50 [1] in the detection of structures in medical images [16]. A comprehensive review of LBP variants [6] found that Median Robust Extended Local Binary Patterns [7] gave the best overall results over a range of different tests. It would be highly interesting to apply this method to our oral cancer cell classification problem.

The classification of the training data is done on patient level and not on cell level, which means that all cells from a cancer patient are labeled as cancer cells and likewise for healthy patients. However, it is possible that not all cells sampled from cancer patients are cancerous. It is therefore possible that healthy cells are labelled as cancerous in the training data, which would have a negative impact on model performance. Labeling the images from the cancer patients individually would eliminate this probability, and improve the quality of the training data.

There is no way for a neural network to realize that images in the training data are mislabeled, and it will instead try to classify these images correctly according to their label. This means that the network, during training, will attempt to find characteristics that allow it to label the mislabeled healthy cells as cancerous. Depending on the frequency of these mislabeled cells, this may be a serious problem: the characteristics/properties that the network learns to classify cells as cancerous do not generalize to new data sets, decreasing the test accuracy. In our case it is likely that this will include properties of the glass, i.e. the larger sample from which individual cells are imaged.

We have suspicions that this is the case for our models, since all of them achieved a high validation accuracy (Juefei-Xu - 96% , Li - 94%, Marcos - 83%) but their accuracy on the test data is considerably lower. As well as by individually labelling the cells, this problem could be mitigated by adding more data, to provide a broader spectrum of variation between glasses in the training data and prevent particular glass properties from becoming associated with a particular label.

It should however be noted that 100% accuracy on a cell level is not required as long as suitable thresholds for diagnosing a sample can be found. When a very low proportion of cells are classified as cancerous, the patient might be assumed to be healthy, and for higher proportions the sample could be flagged for manual inspection. This automated pre-screening could drastically reduce the cost of analysing samples.

The powerful cancer detection ability of texture based CNNs is bound to be an extremely useful tool for medical professionals in the future, decreasing their workload and potentially opening the way to a national screening program for oral cancer.

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[2] Felix Juefei Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks (lbcnn). https://github.com/juefeix/lbcnn.torch, 2016.

[3] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, volume 1. IEEE, 2017.

[4] Gil Levi and Tal Hassner. Emotion recognition in the wild via convolutional neural networks and mapped binary patterns. In *Proceedings of the 2015 ACM International Conference on Multimodal Interaction*, pages 503–510. ACM, 2015.

[5] Lei Li, Xiaoyi Feng, Zhaoqiang Xia, Xiaoyue Jiang, and Abdenour Hadid. Face spoofing detection with local binary pattern network. *Journal of Visual Communication and Image Representation*, 54:182–192, 2018.

[6] Li Liu, Paul Fieguth, Yulan Guo, Xiaogang Wang, and Matti Pietikäinen. Local binary features for texture classification: taxonomy and experimental study. *Pattern Recognition*, 62:135–160, 2017.

[7] Li Liu, Songyang Lao, Paul W Fieguth, Yulan Guo, Xiaogang Wang, and Matti Pietikäinen. Median robust extended local binary pattern for texture classification. *IEEE Transactions on Image Processing*, 25(3):1368–1381, 2016.

[8] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *ICCV*, pages 5058–5067, 2017.

[9] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2012–2017. IEEE, 2016.

[10] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[11] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[12] Peter Sasieni, Alejandra Castanon, and Jack Cuzick. Effectiveness of cervical screening with age: population based case-control study of prospectively recorded data. *BMJ*, 339:b2968, 2009.

[13] Olivier Simon, Rabi Yacoub, Sanjay Jain, John E Tomaszewski, and Pinaki Sarder. Multi-radial LBP features as a tool for rapid glomerular detection and assessment in whole slide histopathology images. *Scientific Reports*, 8(1):2032, 2018.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[15] Anders U. Waldeland. Rotational equivariant networks for pytorch/python. https://github.com/COGMAR/RotEqNet, 2018.

[16] Elisabeth Wetzer, Joakim Lindblad, Ida-Maria Sintorn, Kjell Hultenby, and Natasa Sladoje. Glomerulus detection by fusion of CNN and LBP maps. In *BioImage Computing Workshop*. ICCV, 2018.

[17] Håkan Wieslander, Gustav Forslid, Ewert Bengtsson, Carolina Wählby, Jan-Michael Hirsch, Christina Runow Stark, and Sajith Kecheril Sadanandan. Deep convolutional neural networks for detecting cellular changes due to malignancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–89, 2017.

[18] Håkan Wieslander and Gustav Forslid. Deep convolutional neural networks for detecting cellular changes due to malignancy. Master's thesis, Uppsala Universitet, 7 2017.

[19] Hui Zhou, Runsheng Wang, and Cheng Wang. A novel extended local-binary-pattern operator for texture analysis. *Information Sciences*, 178(22):4314–4325, 2008.