LU TP 19-37
March, 2019

Two subtypes of lung cancer classification from histopathology images based on deep learning

Xiaochen Shen

Department of Astronomy and Theoretical Physics, Lund University

Master thesis (30 credits) supervised by Mattias Ohlsson

LUND
UNIVERSITY

Abstract

Adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC) are the most common forms of lung cancer. These two subtypes of lung cancer are usually classified by visual inspection clinically. Our aim is to design an effective strategy based on convolutional neural networks to classify histopathology slides of these two types of lung cancer. With augmentation of the histopathology slides, different classifiers were trained and three ensemble learning methods were compared in this project. Finally, we determined which training strategy that produced the best result.

In the case of limited samples, we find that the combination of transfer learning and ensemble learning greatly improves the classification accuracy for whole-slide images of lung tissue. The optimal strategy achieved 94.2% accuracy with 120 training cases and 86.3% accuracy with 80 independent test cases. We consider this comprehensive strategy remarkable in solving the classification problem with the different kinds of lung cancer.

Popular science description

Artificial Neural network (ANN), as the name suggests, is a mathematical model to simulate the brain. Therefore, its development is more or less inspired by the research of brain science. In the 1960s, when Hubel and Wiesel studied the local sensitive and directional selection of neurons in the cat's primary visual cortex, they found that a unique network structure can effectively reduce the complexity of the traditional ANN. Subsequently, many scientific researches were inspired by this work and built the basic structure of convolutional neural network (CNN). Thanks to massive reductions in both the amount of computation and the number of parameters, CNN has achieved unprecedented success in the field of computer vision. Now, CNN has become one of the research hotspots in many scientific fields, such as autonomous driving and natural language processing. While, with the development of ANN, various technologies, such as transfer learning and ensemble learning, have been proved effective in many aspects of computer vision.

Since visual inspection of histopathology slides is one of the main methods used by pathologists to assess the stage, type and subtype of tumors in the medical field, this naturally leads to introduce CNN into the task of medical image classification. The use of machine learning in medicine can not only improve the accuracy of medical image classification, but also reduce the workload of physiologists. This study expects to build a powerful deep CNN when the number of the medical image is limited, so that the model can achieve better classification accuracy with two main subtypes of lung cancer, LUAD and LUSC. Usually, in a large tissue section image, the cancer cells cover only part of the image. So when the large image is segmented into tiles for training, we need to select tiles which contain really important tumor information to train the neural network. As a result, we adopted a multi-level training process to filter the unimportant tiles, allowing our model to capture the really important cancer features from the informative images. Based on this idea, only a small part of images in the database can be used to get a classification accuracy rate higher than 85%. Besides, because many medical images have some degree of similarity, we also hope to extend the remarkable strategy to other related medical image classification tasks.

Contents

1 Introduction

In the past few years, deep learning has been applied extensively in computer vision research [1]. These studies have achieved outstanding results in image classification and image recognition tasks. At the same time, many advanced techniques for Artificial Neural Networks (ANNs), such as transfer learning, ensemble learning, and different regularization methods, have greatly improved the performance of ANNs, and got excellent achievement in a wide range of fields.

At present, visual inspection of histopathology slides is one of the main methods used by pathologists to assess the stage, type and subtype of tumors in the medical field. Over the years, an increasing number of histopathology slides have been collected in the medical databases. This naturally leads to introduce deep learning techniques into the task of medical image classification [2].

Lung cancer is one of the most malignant tumors with the fastest growth in morbidity and mortality and the greatest threat to population health and life [3]. Data from the American Cancer Society and the Cancer Statistics Center indicates that in 2018, new cases of lung cancer reached 234030. In addition, from 2011 to 2015, the mortality rate of lung cancer is as high as 43.4% [4].

There are two main types of lung cancer: about 80% to 85% of lung cancers are non-small cell lung cancer (NSCLC) and about 10% to 15% are small cell lung cancer (SCLC). The different types of NSCLC are characterized by cancer cells that grow and spread in different ways. The two most common forms of NSCLC are lung adenocarcinoma (LUAD), lung squamous cell carcinoma (LUSC), accounting for roughly 40% and 30% of all lung cancer cases, respectively [5]. Because LUAD and LUSC differ in cell of origin, location within the lung and growth pattern, they are considered as distinct diseases [6]. Therefore, the classification of lung cancer type by the tumor slide images is a key diagnostic process.

The goal of this study is to design a deep learning-based cancer classification system that classifies tumor slide images into LUAD and LUSC based on limited samples. Compared with the classification by visual inspection, an automatic classification system provides a quick and efficient way to solve the tasks. Besides, as an adjunct, it can also help the pathologists to improve the accuracy of cancer identification to a certain degree.

The tissue section image usually has large size. In order to better train the CNN, we need to segment these large images into small tiles and let the neural network extract features from them. However, cancer cells usually cover only part of the tissue section image. In this case, if all of the tiles are labeled as consistent with the label of the tissue section image they came from, it may cause the CNN to extract the wrong features from the tiles which do not contain any important cancer features, which will ultimately limit the final classification accuracy and robustness of the model. Therefore, in this study, we designed a multi-level training strategy to filter the unimportant tiles. After cleaning the data in this way, combined with the visual analysis of the classification results, we found that the

newly trained model achieved higher classification accuracy, which also means that the network extracted the truly important features from the images. In this process, ensemble learning and transfer learning are used to further improve the accuracy and stability of the final classification.

2 Data

2.1 Data Introduction

High-resolution microscopy images of living tissues provide detailed information about the morphology of normal and diseased tissue [2]. We aim to make a correct classification about the type of cancer from the information contained in these medical images.

The whole-slide images of lung tissue obtained from The Cancer Genome Atlas [7] is used as the dataset for this study. Currently, the database contains about 500 whole-slide images of LUSC and LUAD, respectively. Some of these images, although from cancer cases, are normal lung tissues, which do not contain any cancer features. The database also includes some metadata about the cases, such as the magnification. This information is helpful to analyse whether the classification results are biased and how to standardize images.

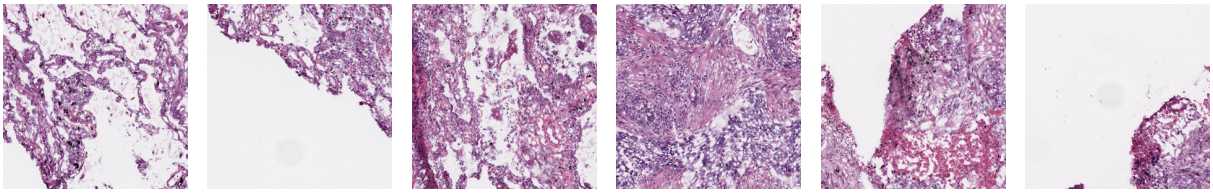


Figure 1: The six tiles were taken from different locations within the same sample. Each tile contains a percentage of valid features.

The size of the original image is too large (the lengths range from 5000 to 100000 pixels). In order to better classify with Convolutional Neural Network (CNN), each original image was segmented into small tiles with 299×299 pixels, as shown in Figure 1. Then, these tiles from the original images are used for training, validation and testing. The size of the tiles was chosen for two reasons. First, this size meets the input requirements of most pre-trained models for transfer learning. Second, sufficient cancer features can be retained on most tiles. Besides, since the original images have different magnifications, it is necessary to choose a correct scaling factor before the segmentation to confirm that each tile contains morphology features with the same scale.

2.2 Data selection

Each tile usually contains white pixels with different proportions, such as the tiles covering the edges of the tissue, which include a large percentage of white pixels and do not provide any features of disease. Therefore, it is essential to calculate the white ratio of each tile and remove the tiles with a significant white fraction. To calculate the white ratio, the tiles were converted from color images to gray scale. If the value of a pixel is larger than 204, it considered as a white pixel. In this way, after calculating the white ratio of each tile, those tiles with white ratio higher than 70% were filtered out.

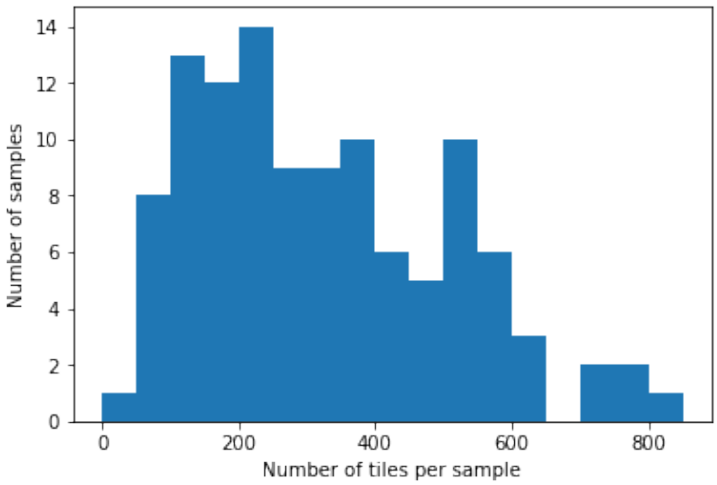


Figure 2: Distribution of the number of tiles per slide for the 120 slides in this project.

The number of tiles from the different samples varies by the original image size, see Figure 2. Large images can produce more than a thousand tiles, while small images might only generate a few dozen tiles. For our model to treat the features from each sample equally, some rules are employed to limit the number of tiles from a specific sample. These rules can avoid the model over-fitting to the samples which contain a large number of tiles. Here, we downloaded 120 samples (60 of them belong to LUSC and 60 of them belong to LUAD) as training sets and validation sets. Rules for data selection were set as follows:

1. Segment the original images into small tiles, the size of each tile is 299×299 pixels.
2. Exclude tiles with white ratio higher than 70%.
3. If a sample contains more than 150 tiles, randomly pick 150 tiles into the dataset. Otherwise, pick all the tiles from this sample.
4. If the number of tiles belonging to each type of lung cancer differs greatly, it is necessary to augment the data of the cancer class which contains fewer tiles to balance the classes.
5. After data augmentation, pick 80% of the tiles from the dataset randomly as the training set and the others as the validation set.

The training set is used to train the model, and the validation set can provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. To verify the performance of the trained model on the new samples, we also downloaded 80 independent samples randomly (40 of them belong to LUSC and 40 of them belong to LUAD) as test set, which can provide an unbiased evaluation of the final model fit on the training set.

2.3 Data Augmentation

There are three main benefits with data augmentation: When data is limited, more valid data can be obtained by appropriate augmentation. Even with a large amount of data, augmentation can also prevent the model from learning unrelated features in the images and thereby, improve the model performance fundamentally. For imbalanced classes, data augmentation offers an effective method to create a balanced dataset.

For medical images, a lot of features are irrelevant with the essential features for classification, such as brightness, hue, saturation, contrast and uneven stain within a certain range. This means that even if these features are changed, the important features of cancer for classification should remain the same. Therefore, by randomly perturbing these irrelevant features in advance to get more varieties and use these enhanced images to train the network, the model will extract important features more accurately and get better classification results.

TensorFlow's image library ($tensorflow.image.random\_X$) [8] was chosen to augment data in this project. Implementation details are given in Table 1:

Table 1: Implementation details for data augmentation.

Function	Parameter
`random_brightness`	max delta = 0.25
`random_saturation`	lower = 0.875, upper = 1.125
`random_hue`	max delta = 0.04
`random_contrast`	lower = 0.7, upper = 1.3
`random_flip_left_right`	-
`random_flip_up_down`	-

Tensorflow provides four functions to adjust the brightness, saturation, hue and contrast of the images randomly within the set ranges respectively. Two functions are also provided to randomly flip images. By perturbing these variables, we were able to get more images without changing the important features of the cancer for final classification.

3 Theory

3.1 Artificial Neural Network

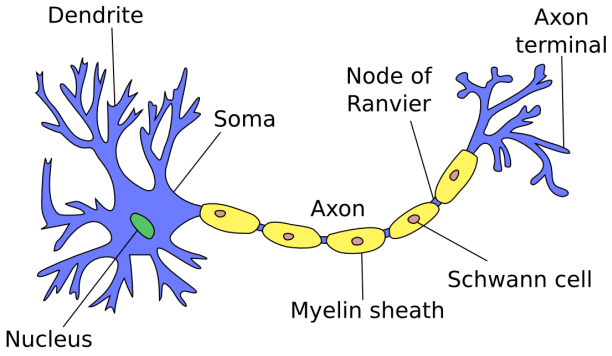


Figure 3: The basic structure of the pyramidal cortical neuron [9], which inspires the structure of ANNs.

Artificial neural network (ANN) is a mathematical model built by simulating the construction of biological neural networks [10], see Figure 3. With the advent of the digital age, data in different fields have been accumulating rapidly. With the further improvement of computing power and machine learning algorithm, ANN is gradually introduced into various fields [11] [12]. In many cases, ANN has proven to be effective in solving some repetitive, tedious tasks.

The basic component of ANN is the artificial neuron. The artificial neuron can receive information from the incoming signals as the inputs and map these signals to output through an activation function. The activation function does a kind of transformation to the input, which allows the model with multiple layers to better handle complex tasks. Under the action of the activation functions, the output value will map to a given range.

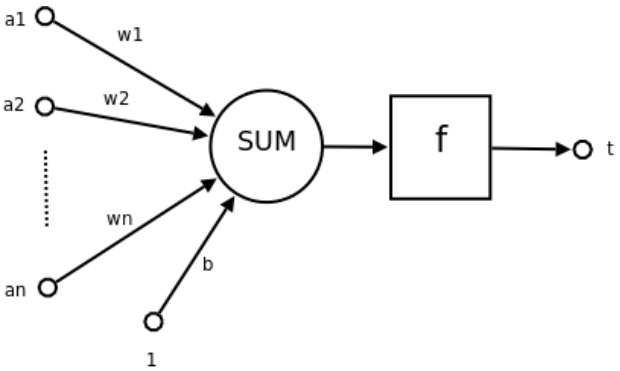


Figure 4: A perceptron with n-dimensional input. The components a_1, a_2, ..., a_n of the n-dimensional input are weighted by the weights w_1, w_2, ..., w_n, summed and added to the bias b. The result is passed into an activation function f to produce the output t [14].

The perceptron is one of the simplest feed-forward neural networks [13]. Figure 4 shows a perceptron consisting of one input layer and one neuron. The input signals are connected to the neuron by weighted connections respectively. When the input signals flow into the neuron under its own weight, the neuron will map the total received value to a final result by the activation function.

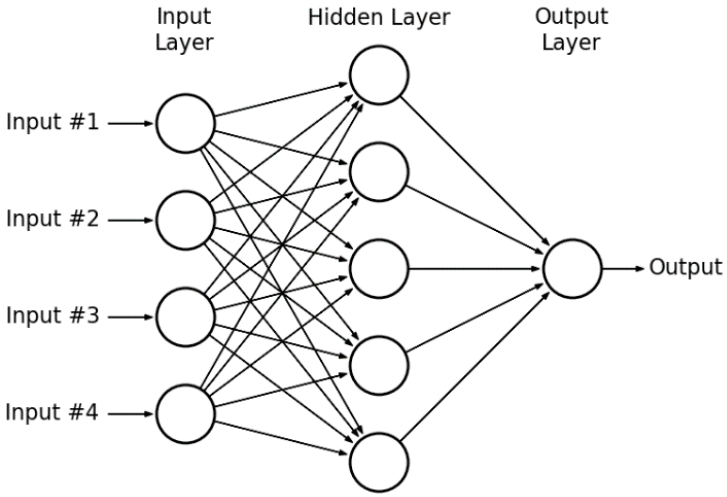


Figure 5: A typical MLP consists of an input layer, a hidden layer and an output layer. The inputs are weighted by connection weights and passed to the nodes in the next layers, get a final output value [15].

For more complex non-linear separable problems, the multilayer perceptron (MLP) can be adopted. A MLP consists of an input layer, some hidden layers and an output layer, see Figure 5. Except for the input layer which is determined by the input vector, each layer contains a certain number of neurons. In this case, the non-linear activation functions are introduced to add non-linear factors for the network. Therefore, in a neural network with multilayer structure, after a neuron receives multiple inputs, these signals are weighted with corresponding weights and pass through the activation function to produce an output, which will be fed forward to the nodes in the next layer or output as a final result.

The process of obtaining information from data and storing the acquired knowledge in the weights is called the learning process. It can be done by back propagation (BP) algorithm. For supervised learning, the error between the output result with the expected result is measurable by a proper loss function. Then, the gradient of the loss function can be calculated layer by layer for the weights and fed back to the optimization method to minimize the loss function by updating the weights in each layer. After this process has been repeated many times, the result will be close to what is expected.

3.2 Convolution Neural Networks

Convolutional neural networks (CNN) is one of the representative network structures in deep learning. It can be considered as a MLP with a special structure. CNN can take advantage of the 2-dimensional structure and the high correlation between adjacent pixels of the image, thus avoiding the one-to-one connection between all pixel units.

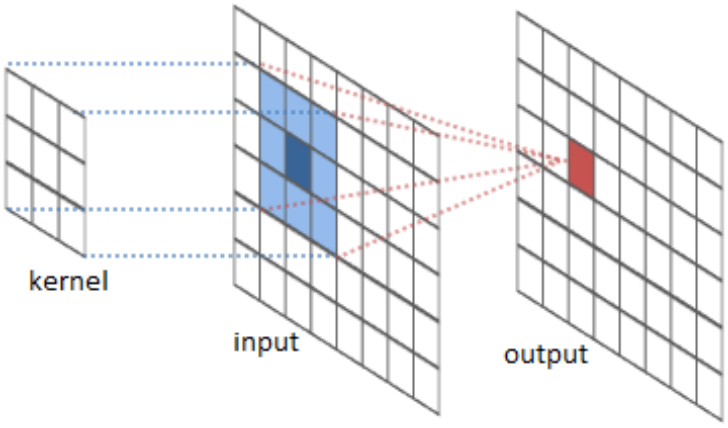


Figure 6: An image is convoluted with a 3×3 convolution kernel [17].

In image analysis, the data that needs to be processed is a discrete, two-dimensional form. So the convolution formula for the discrete variables can be written as:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \quad ,$$

where $g(x,y)$ represents the filtered image and f(x,y) represents the input image. The function $w(s,t)$ is the weighting function, which is also called the convolution kernel. By training the convolution kernels, they can extract the features from an image that are considered to be the most relevant to the classification. In general, a convolution kernel covers only a small area of an image, see Figure 6. So the kernel needs to slide on the whole image with a certain stride, which is the number of pixels the kernel shifts over the input image at a time, to extract features in different positions.

The two important properties of CNN are sparse interactions and parameter sharing [16]. For an image, the correlation of the adjacent pixels is relatively important, while pixels with far distances are weakly correlated. Besides, the convolution kernels should be the same for the same feature in different positions of the image. Based on these ideas, the concept of shared weights is proposed, and the number of parameters in a CNN is greatly reduced. Therefore, small convolution kernels can save the essential features in an image efficiently.

For multi-channel input convolution, such as a color image with three RGB input channels, it usually has three kernels (one for each input) instead of one. As shown in Figure 7, the

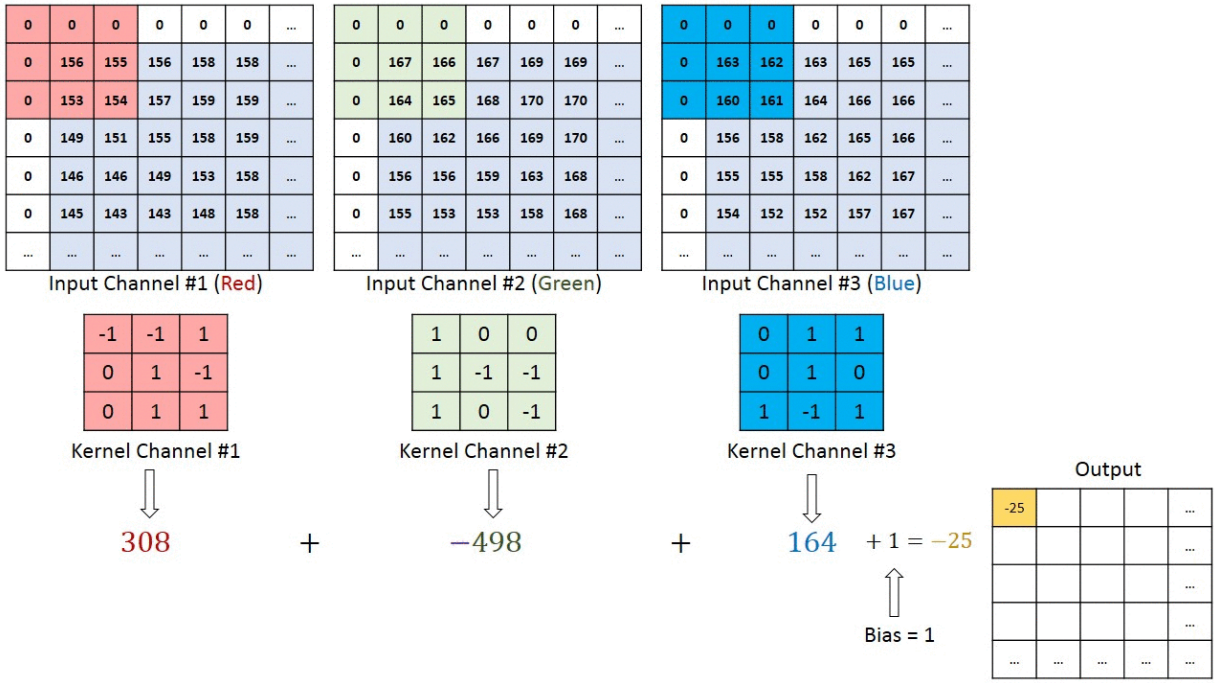


Figure 7: Convolution with three input channels and one output channel [18].

convolution kernels over the image pixels and perform convolution operation to obtain the feature maps. During this process, the feature maps from the three channels are summed to produce a final feature map as an output.

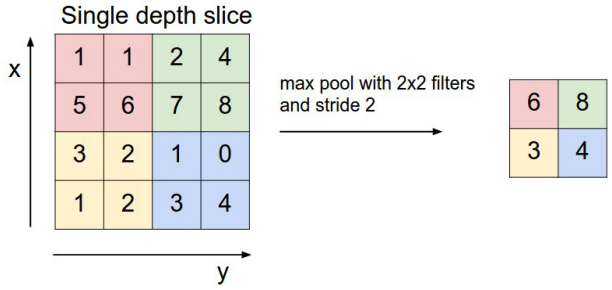


Figure 8: Example of max pooling with 2×2 filters and stride 2. According to the rules of max pooling, the maximum value is extracted from the area covered by the pooling kernel [19].

After the convolution layer, a pooling layer is usually inserted to further simplify the feature map. Max pooling is one of the most common technologies in CNN. Like the convolution kernel, the pooling kernel also needs to slide over the entire image to complete the pooling. As shown in Figure 8, when an area on the image is covered by a max pooling kernel, the maximum value in the area will be used as the output value. The obvious benefit of the pooling layer is compressing the size of the feature map. For an image, max pooling

helps to make the representation approximately invariant under a small translation. This property is important when we care more about the feature itself than the location of the feature. Besides, it also expands the size of the receptive field by keeping the important features and compressing the image size.

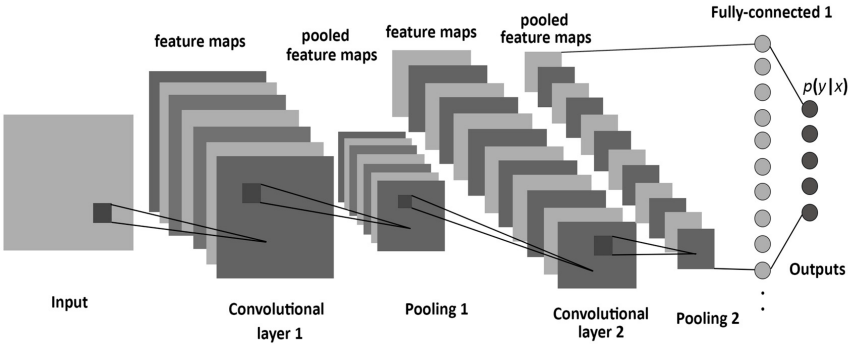


Figure 9: The structure of a CNN. It usually consists of convolution, pooling, and fully-connected layers [20].

In a complete CNN architecture, multiple convolution layers and pooling layers are usually used. Since the convolution and pooling layers only output feature maps from the original image, to get the final result, we need to transform the feature map into a feature vector and input it to the fully connected layers. At the last layer, the probability corresponding to each class will be output. A typical CNN structure is shown in Figure 9.

3.3 Optimization algorithm

In supervised learning, the targets corresponding to the input values are given. By comparing the targets with the output values, we can obtain an error. Optimization algorithms help us to minimize the error function $E(x)$ used in the model to realize the learning process.

To implement this algorithm, it is necessary to find an effective way to measure the error between result and target. For the binary classification problem, the binary cross-entropy function [21] is usually chosen as the error function, which can be derived from probability theory:

$$L = -\sum_{i=1}^{N} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \quad ,$$

where $\hat{y}$ is the output of a network and y is the corresponding desired output [22].

For the case where the training data is large, a mini-batch stochastic optimization algorithm for finding the weights or coefficients of machine learning is usually adopted: split the training dataset into some small batches and use them to calculate the error and update the weights by back propagation algorithm individually. This method for optimizing a differentiable error function is called stochastic gradient descent (SGD) [23].

It is also important to select the right batch size. In general, large batch size allows faster convergence, but it tends to converge to sharp minimas of the training functions, which can lead to poorer generalization. Small batch size can offer a regularizing effect, but the total training time can be very long, since it is difficult to converge [16]. Besides, the choice of batch size also depends on the data and GPU memory. Therefore, we usually need to try different batch sizes to balance the classification accuracy and training time.

Adam is another optimization algorithm that computes adaptive learning rates for each parameter from estimates of first and second moments of the gradients, which is extended from SGD [24]. All the models in this study were trained using Adam as the optimizer. It makes the error have faster convergence speed and achieves better performance with final accuracy.

3.4 Batch Normalization

Another technique used in this project is batch normalization (BN) [25]. In the process of training the neural network, the distribution of the input training data will change due to the change of network parameters. This phenomenon is called internal covariate shift. So, to reduce the internal covariate shift, BN normalizes the data at every training step or at some interval. The general advantage of BN is that it allows us to use much higher learning rates and be less careful about initialization. After data processing in batches, BN can be implemented with the following steps:

1. Calculate the mean and variance of the input for each batch.

$$\text{Batch mean: } \mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\text{Batch variance: } \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

2. Normalize the inputs for each batch.

$$\overline{x_i} = \frac{x_i - \mu^B}{\sqrt{\sigma_B^2 + \epsilon}} \quad ,$$

   where ϵ is a small number added to variance to avoid dividing by zero.

3. Scale and shift in order to obtain the output of the layer.

$$y_i = \alpha\overline{x_i} + \beta \quad ,$$

   where α and β are scale and shift coefficients learned during training.

3.5 Regularization techniques

One of the major problems that occur during neural network training is over-fitting. This phenomenon usually shows that the model performs well on the training set, but dose badly on a new dataset. For enhancing the generalization of the model, early stop and dropout technologies are adopted in this project to prevent over-fitting.

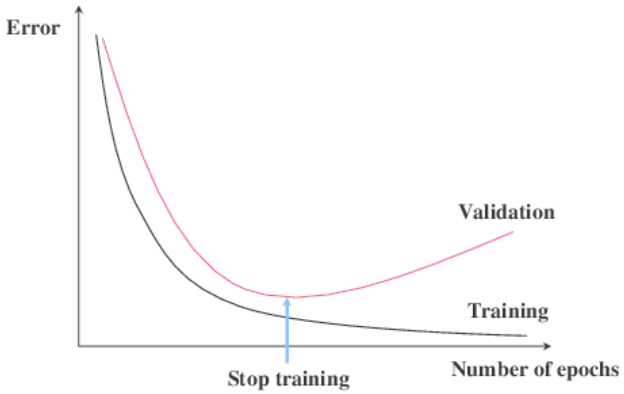


Figure 10: The principle of early stopping is stopping training at the point when the minimal validation loss is achieved [26].

Early stopping is a simple technique implemented during the training. After randomly dividing the dataset into a training set and validation set, one can get a generalized model by returning the parameter that minimizes the validation loss, which is the loss on the validation set during the training of the model, see Figure 10. Based on this idea, the validation loss is treated as an essential indicator to stop training [27]. Although the model does not fully converge on the training set, it enhances the performance of the model with an independent dataset.

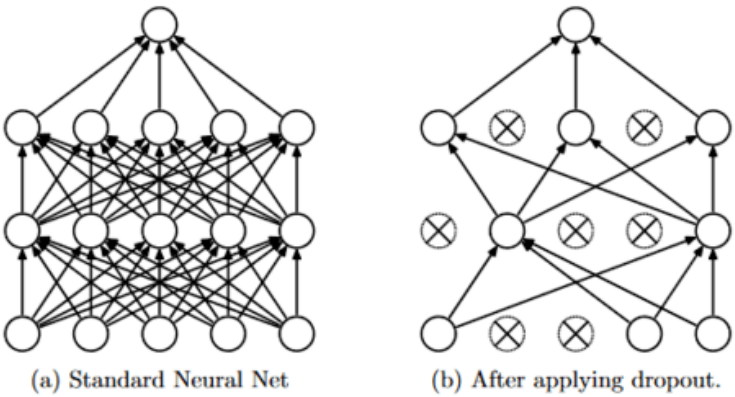


Figure 11: The principle of dropout is randomly invalidating some neurons in the training phase [28].

Dropout [29] is another extensively used method to prevent over-fitting. It relieves the complex co-adaptations between neurons and reduces inter-neuronal dependence to some extent by providing a way of approximately combining many different neural network architectures efficiently. The implementation of dropout is randomly deleting some nodes of the hidden layer at each training batch, as shown in Figure 11. Therefore, applying dropout to a neural network amounts to sampling a thinned network from it. The thinned network consists of all the units that survived dropout.

3.6 Model Selection

For model selection, it is usually important to compare multiple models with different architectures and parameters, then select an optimal model as the final one. Because of the training process and the selection of the dataset have a certain degree of randomness, we should select an optimal model in a statistical sense.

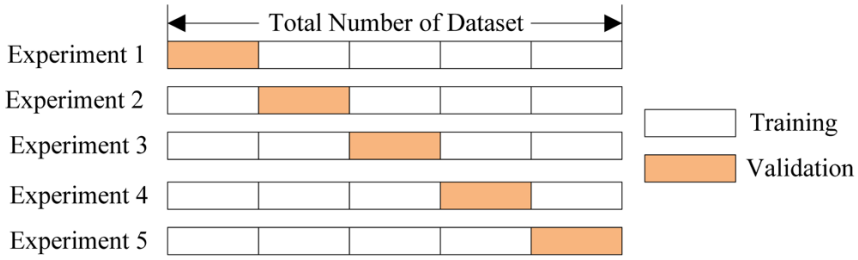


Figure 12: Example of 5-fold cross-validation. We divide the data into 5 pieces, each being 20% of the full dataset. Each subset is used as a validation set once and the remaining 4 subsets are used as the training set [30].

Cross-validation is a statistical analysis method used to verify the performance of a classifier [31]. The basic idea is splitting the original data into some subsets randomly. Then, use one of the subsets as a validation set and the other part as the training set. The classifier should train with the training set, and the trained model be validated by the validation set to evaluate the accuracy of classification. K-fold cross-validation divides the original data into K groups equally, as show in Figure 13. Each subset is used as a validation set once and the remaining $K-1$ subsets are used as training sets. After training, K models with the same architecture will be obtained, the average classification accuracy on the validation set of the K models can be used as the evaluation index of the model. By comparing the cross-validation indicators of models with different configurations, the optimal model can be determined. In this project, we used 5-fold cross-validation for model selection.

3.7 Performance Measures

The receiver operating characteristics (ROC) curve and Area Under the ROC Curve (AUC) are typically used to evaluate the performance of a binary classifier [32,33]. In a ROC curve, the horizontal axis represents the false positive rate (FPR), which can be calculated as (1

− specificity). The vertical axis represents the true positive rate (TPR), which is also called sensitivity. Here, specificity means the fraction of actual negatives that are correctly identified and sensitivity means the fraction of actual positives that are correctly identified.

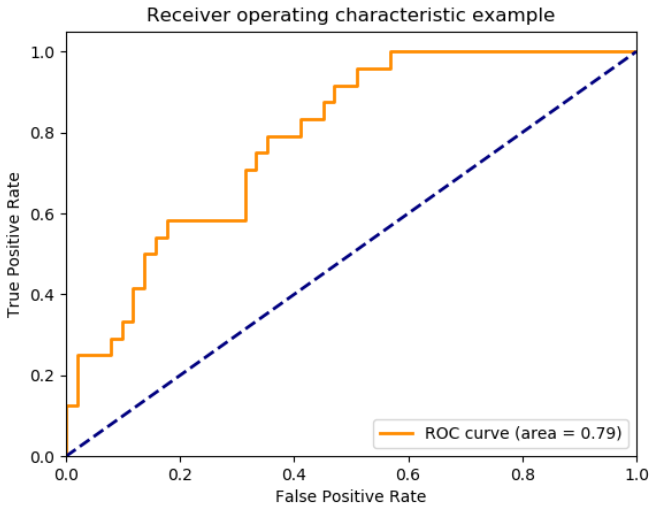


Figure 13: An example for a ROC curve (orange) and the computed Area Under the Curve (AUC) [34].

The sigmoid function was used as the activation function of the output layer in this study when we trained binary classifiers. Because the output of the classifier is a probability value, a discrimination threshold is needed to determine the output class of the classifier when plotting the ROC curve. Generally, we're going to take a series of thresholds from high to low, and classify these samples with the different thresholds many times. When a test sample has a higher probability than the threshold, it is considered as a positive sample. Otherwise, it is negative. After the ROC curve is obtained with these different choices of thresholds, the performance of the classifier can be evaluated by calculating the value of the AUC, see Figure 13. The AUC value is used as the evaluation criterion. It quantifies the ROC curve with a numerical value.

With these definitions, we can assert that the closer the ROC curve is to the upper left corner, the better the performance of the classifier. The diagonal line in the ROC graph represents a result of a random guessing strategy, meaning that the half of all the samples were randomly guessed as positive, and the other samples were guessed as negative.

3.8 Transfer learning

Transfer learning refers to reuse the model parameters trained on a large dataset to the specific task. It can also be treated as a method to initialize weights. Usually, these pre-trained models have consumed a lot of time and computer resources during training. Considering that many data contain similar features, such as edge features in the image,

the trained parameters from one pre-trained model usually can be shared with the new model in some way to improve the learning efficiency [35].

The benefits of transfer learning is mainly reflected in rapid convergence, higher accuracy and simplified tuning process compared to the model we built ourselves or the model without the pretrained weights. In our case, first, we don't need to fine-tune the learning rate to make the model converge quickly. Second, since some general features are saved in the pre-training weights, the model will have high accuracy at the beginning of training, and this result will further improve by training with our data. In this study, we transferred a model based on the $VGG16$ architecture with pre-trained weights on $ImageNet$ [36], then, use our images to train the network.

The transfer learning strategy used here is not only to replace and retrain the model on top of the CNN, but also fine-tune the weights of the pre-trained model by continuing the back-propagation. It is possible to keep some of the earlier layers fixed and only fine-tune some higher-level layers of the $VGG16$ model. The principle for this training strategy is that the earlier features of the pre-trained model contain more general features, such as edge detectors, that should be useful to medical images, but later layers of the model becomes gradually more particular to the details of the specific images contained in the training dataset. Therefore, for the pre-trained $VGG16$, the parameters in all the layers but the last eight ones were frozen during training. In this way, it not only speeds up the training of the model but also ensures that the training can achieve higher accuracy.

Figure 14 shows the architecture of $VGG16$. This structure is characterized by the use of multiple small 3×3 convolution kernel stacks to replace the large convolution kernel, which improves the network depth under the condition that the receptive fields are the same. This change not only reduces the number of parameters, but also increases the network depth and makes the network have better capability of nonlinear approximation. After we obtain the final feature map of an input image from the last convolution block, we flatten it as a feature vector and input the dense layer, which connects every neuron in one layer to every neuron in another layer, for final classification.

Figure 15 shows our artificial network architecture based on $VGG16$ and all of the techniques mentioned above. The size of the input layer of $VGG16$ was modified to be consistent with the size of training tiles. The batch normalization and the dropout layer were inserted after each dense layer in the red circle in Figure 14.

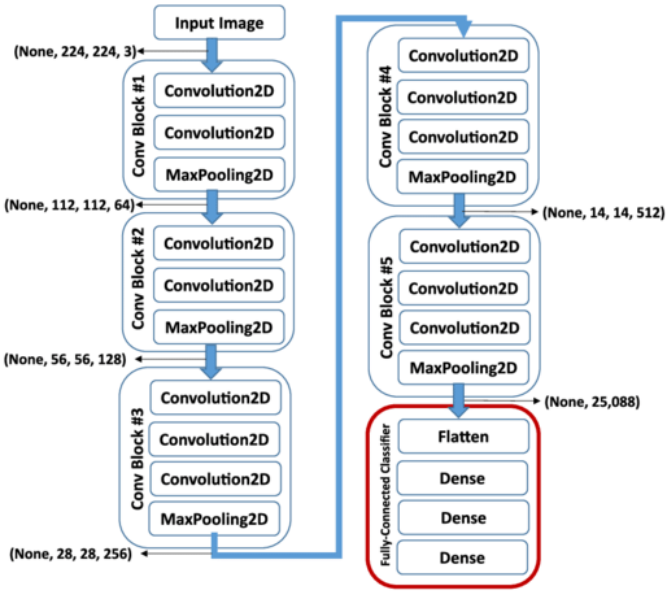


Figure 14: Architecture of VGG16 [37]. In this example, the size of input image is 112×112 with three color channels. The Numbers in black brackets show the size of the feature map at different stages.

Layer
VGG16
Flatten 1
Dense 1
Batch Normalization 1
Dropout 1
Dense 2
Batch Normalization 2
Dropout 2
Dense 3

Figure 15: Transfer learning architecture based on VGG16.

In the following, all the classifiers used in the different strategies have the same architecture with different parameters.

3.9 Ensemble Learning

Ensemble Learning accomplishes learning tasks by building and combining multiple classifiers. Compared to a single classifier, it generally has superior generalization performance and solves the memory issue caused by training a single classifier with a large dataset [38, 39].

Boosting is one of the ensemble learning strategies for primarily reducing bias and also variance in supervised learning. The main idea of Boosting is to increase the weight of the samples that were misclassified by the previous classifier and use this new data set to train the next classifier. Thus, future classifiers will focus more on the cases which are misclassified by the former classifiers.

There are many ways to use boosting. For our task, it is more important to determine the classes of cancer based on the whole-slide image than a single tile. Therefore, when we use the boosting method to train a new classifier, the data proportion were adjusted based on the misclassified samples. This means that the number of tiles from misclassified samples has been increased when training the new classifier.

4 Ensemble Learning Strategies

Three strategies that achieved good classification results are compared in this project. Unlike some of the tasks which only focus on the results at the tile level, the classification results at the sample level and the average classification accuracy for each sample are also important indicators in this research. Since the results of the different classifiers in our strategies show pretty close values at the tile or sample level sometimes, it is important to consider different indicators from different levels comprehensively to determine the best strategy.

All of the three strategies have a three-level structure. The key idea is to use the classifiers at the first two levels to filter out the tiles that do not contain any important cancer features from the original dataset. Then, use the selected tiles containing significant cancer features to train the three-class classifier at the last level (including tiles with two types of lung cancer and normal lung tissues) to achieve a more accurate classification.

4.1 The First Ensemble Strategy

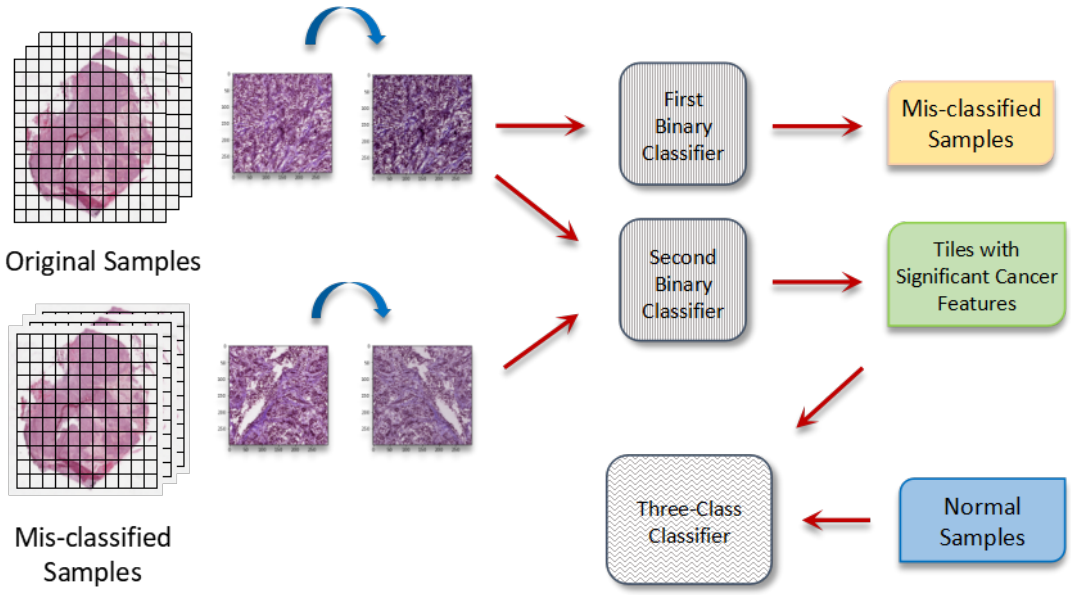


Figure 16: The training process of the first strategy.

The first strategy is shown in Figure 16. This strategy has a three-level structure. The first two levels are composed of binary classifiers and the third level is composed of one three-class classifier. The binary classifier at the first level was trained with the tiles which are randomly augmented from the original tiles. Based on this, the misclassified samples are screened out. After re-segmenting the misclassified samples in a different way, a boosting technique was adopted to train the classifier at the second level. The final three-class classifier was trained by the tiles containing important cancer features, which

were selected by the second binary classifier, and the tiles from the normal samples.

There are two main disadvantages with a single binary classifier for classifying the tiles to LUSC or LUAD. First, the classification result from a binary classifier might be different from the real situation even if it has high validation accuracy. Since the coverage of cancer features (such as cancer cells) do not reach 100% in a whole-slide sample, some of the tiles from the same sample may not contain any cancer features. Because only two types of cancer labels based on the whole-slide samples were used in the dataset, the model tends to make some misjudgements when the tiles are normal tissue. Therefore, we need to use a binary classifier to find the tiles containing important cancer features and build a three-class classifier based on these tiles.

Second, screening valid tiles with a single binary classifier to construct a new dataset to train the three-class classifier may result in loss of important cancer features. A kind of misclassification situation generated by a single classifier is that almost all the tiles from a sample are misclassified (the results section shows more details about this). If we construct a new dataset based on this result, we may not obtain any tiles containing valid features from this kind of misclassified samples, which will reduce the richness of some valid features.

The binary classifier in the second level is to improve the second disadvantage. Here, we used the method of boosting in ensemble learning to train the new model by increasing the proportion of misclassified samples from the first level. It should be noted that the extra tiles selected for boosting are not based on the mistake at the tile level, but based on the classification result of the whole-slide samples. This means that when half of the tiles in a sample are misclassified, we will record this sample and re-segment the sample in a different approach. Then, we will add these new tiles from the misclassified sample to the previous dataset and train a new model.

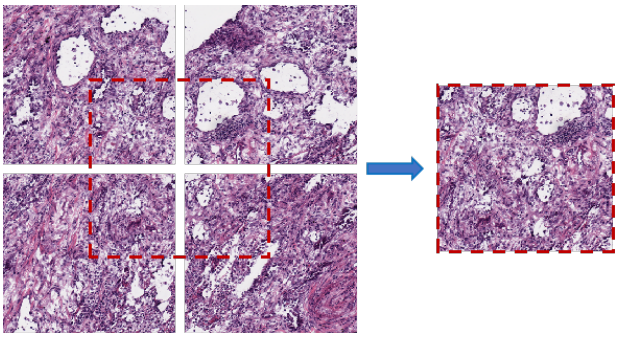


Figure 17: With a different segmentation approach, each new tile contains part of the information from the four original tiles.

The reason of re-segmenting the misclassified samples is to obtain richer feature information. If we simply train the second binary classifier by doubling the misclassified samples, it usually results in severe over-fitting. But once the new tiles, which were generated by a different segmentation method, are mixed with the old tiles, the over-fitting does not occur.

In this study, we re-segment the samples by changing the starting point of the segment window. As shown in Figure 17, for the first tile in the upper left corner of a sample, the new segment window moves to the right and down with half of the tile length. In this way, each new tile contains the features from the four original nearest tiles without changing the magnification and size. In addition, some alternative methods for augmenting the tiles from the misclassified samples are also worth trying, such as rotating the tiles randomly. This kind of enhancement method can provide tiles from more angles without changing the features of cancer.

Based on the classification results of the second-level classifier, we can filter out the tiles which are misclassified and assume the correctly classified tiles contain significant cancer features. In this process, those tiles whose classification results do not have a clear tendency were also excluded: we only saved tiles with classification confidence greater than 55% (if the result obtained by the binary classifier is between 0.45 and 0.55, the tile is rejected.).

It is still important to consider the tile distribution. When selecting tiles from different samples, the previous data selection rules still need to be followed. At this point, a smaller dataset which is relatively balanced and has more generalized cancer features was obtained.

Mixing the cancer tiles from the second-level classifier and the healthy lung tissue samples, the third-level three-class classifier was trained with this new dataset. The ratio of cancer tiles to the normal tiles is about $1:1$.

4.2 The Second and Third Ensemble Strategies

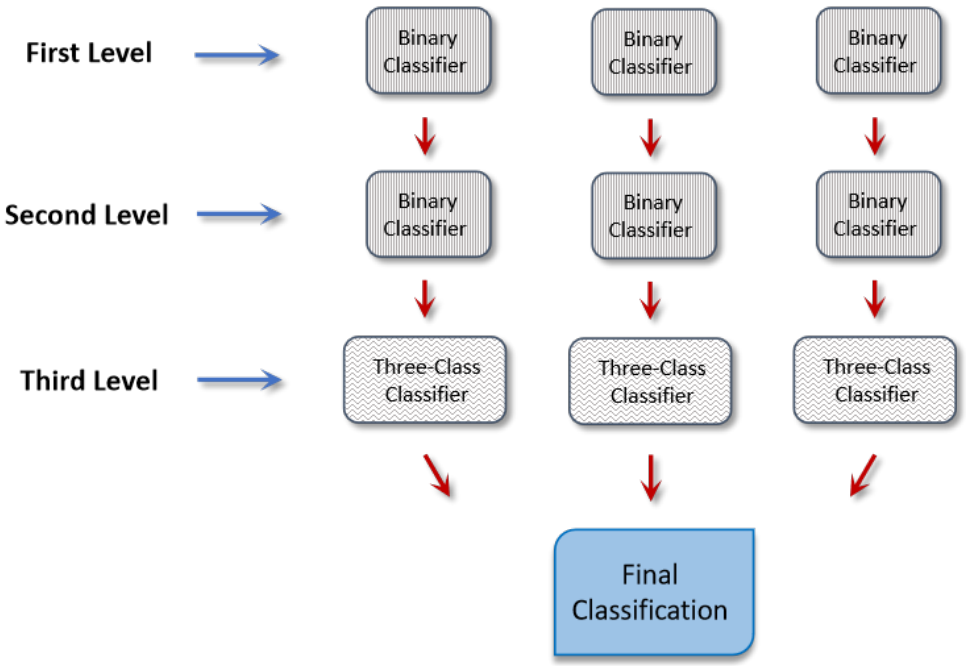


Figure 18: The training process of the second strategy. The main principle is to train three independent three-class classifiers using the first strategy and get the final result by averaging the results from the three independent classifiers.

Based on the first strategy, the ensemble method was introduced to build the second strategy, see Figure 18. In this strategy, we used the same process as in the first strategy, as shown in Figure 16, to train three independent three-class classifiers. The classification results of these three independent three-class classifiers were averaged to determine the final class of each tile.

In the third strategy, we improved the second strategy. Based on the process of the second strategy, the classification results from the second-level binary classifiers were averaged. With these results, we can save the tiles which are classified correctly as our dataset, see Figure 19. The reason for this step is to pick more reliable tiles and eliminate more accidental cases. As before, tiles with a classification accuracy close to 50% were also excluded, even if they were correctly classified.

The next process is similar to the first strategy: mixing the selected tiles with normal tiles, using the new dataset to train the three-class classifier.

It should be pointed out that for the tiles selected by the classifiers at the second level, there is a certain degree of class imbalance. In the third strategy, the selected tiles were augmented as two different datasets. For the first dataset, we perturbed and mirrored the selected tiles randomly based on the method described in the data augmentation section.

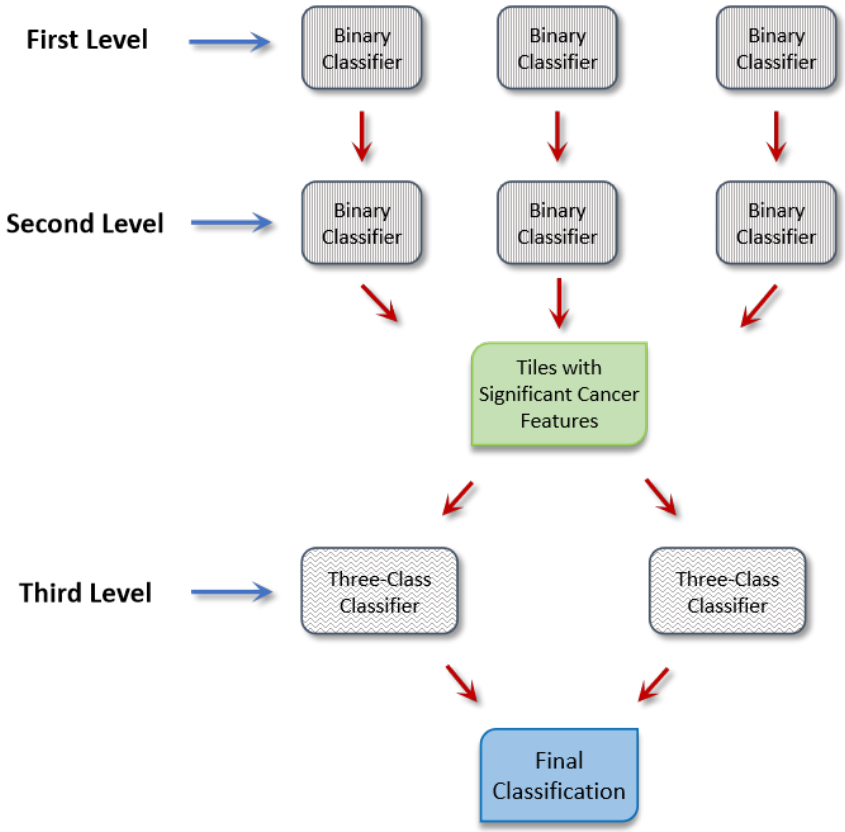


Figure 19: The training process of third strategy.

Then, we used this dataset to train the first three-class classifier. For the second dataset, we balance the classes. It means some tiles from the class with fewer cases were randomly selected and put into the dataset again. After the classes were balanced, we perturbed and mirrored these tiles and used them to train our second three-class classifier. The test results on the independent test set show that these two three-class classifiers are different. So in order to get more stable performance and better generality, we average the results of the two three-class classifiers.

5 Implementation Details

5.1 Software Library

The programming language used in this project is PYTHON (version 3.6.6). The design, training and performance evaluation of neural networks were implemented using the KERAS library. KERAS is a high-level neural network Application Programming Interface (API), which is based on TENSORFLOW, THEANO and CNTK backends. The characteristics are to be able to quickly and efficiently achieve the ANN construction and convenient testing.

5.2 Hyperparameters

The VGG16-based binary classifiers have the layers shown in Figure 14.

Here, Dense 1 and Dense 2 use the Relu function as the activation function and Dense 3 uses the Sigmoid function for binary classification. The number of nodes in Dense 1 and Dense 2 are 2048 and 1024 respectively. Half of the nodes are dropped with all of the dropout layers. Adam is used as the optimizer to train the neural network. When training all the binary classifiers, binary cross-entropy function was used as the loss function. The learning rate is 0.0001 and the batch size is 16.

The VGG16-based three-class classifiers adapted similar layers and hyper-parameters as the binary classifiers. The difference is that the activation function of Dense 3 is Softmax, which is a function for multiple classifications. When training the models, the batch size is 8.

In the section of results and analysis, we compare the results of the neural network structure designed by ourselves with the neural network based on transfer learning. The architecture designed by ourselves is shown in Table 2. The size of the convolution kernel is (3×3) for each convolutional layer. The convolution layer from 1 to 5 contains 32, 64, 128, 256 and 512 convolution kernels, respectively. The size of the pooling kernel is (2×2) for each pooling layer. The number of nodes in Dense 1 and Dense 2 are 1024 and 512 respectively. Half of the nodes are dropped with all of the dropout layers.

Table 2: Architecture of the binary classifier without transfer learning.

Layer
Convolution layer 1
Batch Normalization 1
Max pooling layer 1
Convolution layer 2
Batch Normalization 2
Max pooling layer 2
Convolution layer 3
Batch Normalization 3
Max pooling layer 3
Convolution layer 4
Batch Normalization 4
Max pooling layer 4
Convolution layer 5
Batch Normalization 5
Max pooling layer 5
Flatten 1
Dense 1
Batch Normalization 6
Dropout 1
Dense 2
Batch Normalization 7
Dropout 2
Dense 3

6 Results and Analysis

6.1 Improvement of transfer learning

Table 3: Validation accuracy of two neural network models.

Model	Validation accuracy
First Architecture	73.5%
Second Architecture	82.8%

For the binary classifiers, two network architectures were compared in this study. The first architecture is shown in Table 2, which was built by ourselves. The second architecture is based on transfer learning with the pre-trained $VGG16$ model, as shown in Figure 15.

The tiles from 120 samples were divided into training set and validation set according to the process mentioned in the data selection section. The results in Table 3 are the average validation accuracy with 5 independent classifiers (5-fold cross validation).

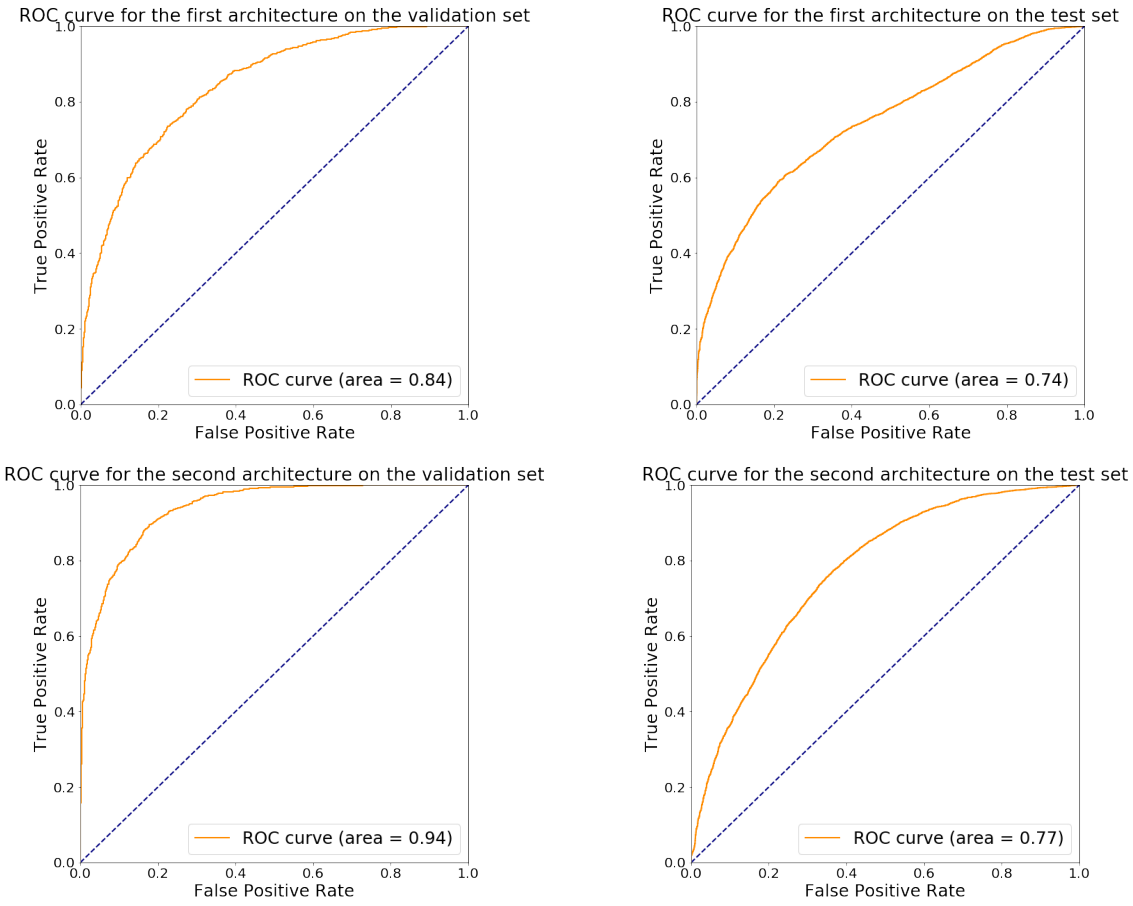


Figure 20: ROC curve of the two architectures on the validation and test sets.

Figure 20 shows the ROC curves of the first and second architectures on the validation dataset and independent test sets. These data indicate that transfer learning works much better than the best structure of the classifier we have found so far. Therefore, all of the other models in this study are based on transfer learning techniques.

6.2 Results from Ensemble Learning

When comparing the different strategies of ensemble learning, there are four main indicators for measuring the pros and cons of these strategies.

- Validation Accuracy at Sample Level.
- Test Accuracy at Sample Level.
- Average Validation Accuracy at Tile Level.
- Average Test Accuracy at Tile Level.

The first two indicators only provide information about the classification at the sample level. The last two indicators reflect the average performance of the classification at the tile level on a single sample. Before training, the tiles from 120 samples were mixed and randomly generated the training set and validation set based on our selection rules. Therefore, when we calculate these indicators with these 120 samples at the different levels, the training accuracy actually refers to the overall accuracy of the training set and validation set. But all of these tiles used to calculate the accuracy were not disturbed with color. In addition, all of the test accuracies were calculated from 80 independent samples. The importance of different indicators varies according to the different roles of the classifiers at different levels. For the classifiers at the first two levels, because the purpose is to select the tiles containing significant features from each sample, the stability and accuracy of the training set at the sample level are more important. For the final results, the performance on the independent test set can better reflect the generalization of the model and the validity of the classification, so the test results are more critical.

Figure 21 shows a basic structure of our ensemble strategies which are described in the previous section. All the three strategies involve a three-level structure. Each level contains some different binary classifiers and three-class classifiers, respectively. The following shows the performance of classifiers trained by different strategies.

6.2.1 Results of The First Strategy

Table 4 shows the classification results of the classifiers at the first level and the second level from the first strategy described above. The ROC curves in Appendix A, Figure 25, show the performance of these classifiers in the validation set.

By analyzing the misclassification in the first layer, there are two types of misclassification at sample level that are noteworthy.

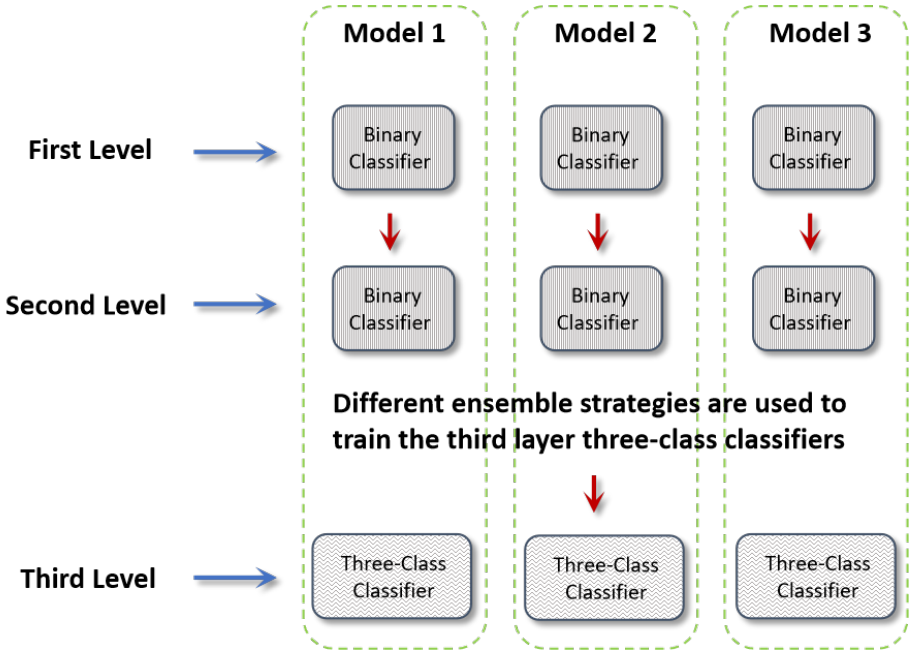


Figure 21: A basic structure of our three ensemble strategies. Based on different strategies, the number of classifiers in the third level is different.

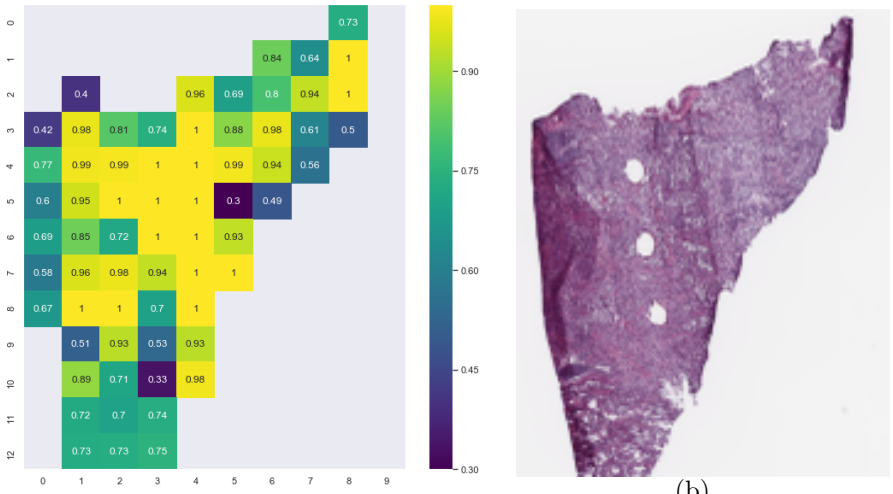
Table 4: Comparison of stability between the models at first level and second level.

Model	Level	Validation Accuracy at Sample Level
Model 1	1	95.8%(115/120)
Model 2	1	92.5%(111/120)
Model 3	1	90.8%(109/120)
Model 1	2	95.0%(114/120)
Model 2	2	94.2%(113/120)
Model 3	2	95.0%(114/120)

The first type of misclassification usually occurs at the edge of the sample, but in this case, the classification of the whole-slide images will not be affected, see Figure 22.

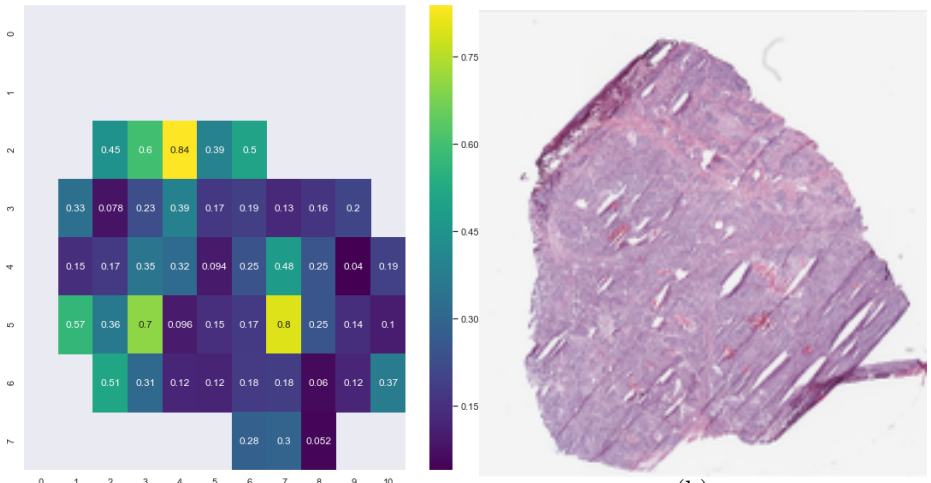
The second type of misclassification indicates that almost all of the tiles from one sample are misclassified, as show in Figure 23. This not only reduces the classification accuracy at the sample level, but also makes it difficult to extract the tiles with main cancer features from these misclassified samples.

After training the binary model at the second level with the new dataset, the accuracy at sample level of the model has more stable performance. The enhanced stability of the classifier helps to better screen the tiles. That is to say, the second level classifiers can selected more tiles with important cancer features from the different samples, which makes the dataset of the three-class classifier more reliable and informative.



(a) (b)

Figure 22: The first typical misclassification. Figure (a) shows a heatmap of a sample, yellow indicates that the classification is completely correct. Figure (b) is the corresponding whole-slide image.



(a) (b)

Figure 23: The second typical misclassification. Figure (a) shows a heatmap of a sample, black indicates that the classification is completely wrong. Figure (b) is the corresponding whole-slide image.

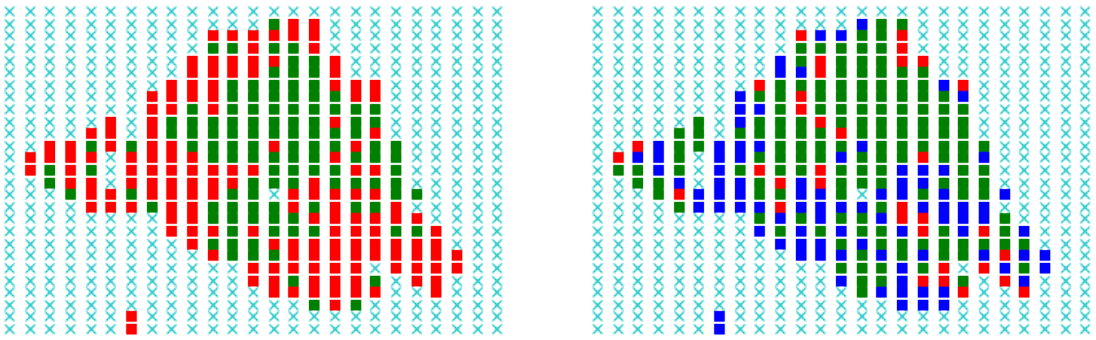
Table 5: Test results for the models at different levels

Model	Level	Test Accuracy at Sample Level	Average Test Accuracy at Tile Level
Model 1	1	80.0%(64/80)	69.9%
Model 2	1	80.0%(64/80)	70.0%
Model 3	1	78.8%(63/80)	71.0%
Model 1	2	81.3%(65/80)	69.7%
Model 2	2	81.3%(65/80)	71.6%
Model 3	2	77.5%(62/80)	70.1%
Model 1	3	85.0%(68/80)	74.4%
Model 2	3	85.0%(68/80)	73.5%
Model 3	3	82.5%(66/80)	73.8%

Table 5 shows the classification results from three independent models with the first strategy. The ROC curves in Appendix A, Figure 26, shows the performance of all the binary classifiers in the test sets.

The principle of calculating the accuracy indicators of a trained three-class classifier is to count the number of correctly classified tiles after excluding tiles that are considered as normal (because there is no tile from a cancer sample that is labeled as normal).

The classification results of the three-class classifiers at the sample level are better than the binary classifiers at the first two levels. The average classification accuracy at the tile level of each sample is also significantly improved. Compared with the model at the first level, the model at the second level got more stable classification results with the whole-slide images. The other indicators at the tile level are roughly equal.



(a) (b)

Figure 24: Comparison of classification results between binary classifier and three-class classifier. Figure (a) is the classification results of tiles from a sample by the binary classifier. Figure (b) is the classification results of tiles from the same sample by the three-class classifier. Red dots represent misclassified tiles, blue dots represent normal tiles, and green dots represent correctly classified tiles.

Visualizing the classification results of the tiles in each sample helps to see the improvement of the three-class classifiers. As shown in Figure 24, the three-class classifier judges most of the misclassified tiles in the binary classification as normal tissues that do not contain cancer features. This result is consistent with our expectation.

6.2.2 Results of The Second Strategy

Table 6: Final results from the first and second strategies with the training samples and test samples.

Model	Level	Validation Accuracy at Sample Level	Average Validation Accuracy at Tile Level
Model 1	3	90.8%(109/120)	83.8%
Model 2	3	92.5%(111/120)	83.9%
Model 3	3	90.0%(108/120)	83.0%
Ensemble	3	89.2%(107/120)	84.8%
Model	Level	Test Accuracy at Sample Level	Average Test Accuracy at Tile Level
Model 1	3	85.0%(68/80)	74.4%
Model 2	3	85.0%(68/80)	73.5%
Model 3	3	82.5%(66/80)	73.8%
Ensemble	3	85.0%(68/80)	74.5%

The results from the second ensemble strategy shows that the final result determined by the average of the three classifiers is very close to the previous single three-class classifier in Table 6. For the original training samples without color perturbation, the classification results at the sample level are even slightly lower than the training result from the single three-class classifier.

We believe that such results are due to the similarity of these three classifiers. Although the randomness generated during the data selection process will make certain differences in the trained classifiers, such classifiers will still make similar judgements at the sample level. Therefore, after combining the classification results of the three classifiers, the results are more likely to produce an average accuracy, not the highest.

6.2.3 Results of The Third Strategy

Table 7: Performance of different strategies with training samples and test samples.

Frame	Validation Accuracy at Sample Level	Average Training Accuracy at Tile Level
strategy 1	90.8%(109/120)	83.6%
strategy 2	89.2%(107/120)	84.8%
strategy 3	94.2%(113/120)	86.1%
Frame	Test Accuracy at Sample Level	Average Test Accuracy at Tile Level
strategy 1	83.8%(67/80)	73.9%
strategy 2	85.0%(68/80)	74.5%
strategy 3	86.3%(69/80)	73.6%

Table 7 shows the final performance of the third strategy on the validation set and the test set, and also compares these results with the first and second strategy. With the third strategy, we got better results on both the original training samples and the independent test samples.

To evaluate the performance of the three-class classifiers and make them easier to compare with the results of the binary classifiers, tiles which are classified as normal by the three-class classifiers were excluded. Finally, the classification accuracy was calculated based on the tiles which are considered to contain the cancer features.

It should be pointed out that when training all the models in this study, the random color perturbation and mirroring operations were performed on the original tiles. But when calculating the four indicators at the sample level and tile level, the original samples (undisturbed tiles) were used.

Comparing the results based on the original training samples and the test samples, it can be deduced that the third strategy is slightly better than the first two strategies. The first and second strategies have different results under different verification criteria.

Table 8: Results from another ensemble method based on the third strategy.

Dataset	Average Accuracy at Tile level with the new ensemble method	Average Accuracy at Tile level with the ensemble method used before
Validation	87.6%	86.1%
Test	75.4%	73.6%

In order to further improve the accuracy at the tile level, another ensemble method is also considered: when calculating the classification accuracy of a sample, we also exclude the tiles which are classified into different classes by the two three-class classifiers. By comparing the results, it is found that adopt this ensemble method can slightly improve the average classification accuracy at tile level based on the third strategy, see Table 8.

7 Outlook

7.1 Conclusion

By comparing the transfer learning and the neural network architecture designed by ourself, the $VGG16$ model which was pre-trained on *imagenet* has an excellent improvement in the classification of lung cancer images. Therefore, we decided to use the single classifiers based on $VGG16$ to design a powerful strategy to achieve better classification results.

Three different ensemble strategies were compared based on the idea of excluding normal tiles from original samples. The third strategy is considered to be the best ensemble learning strategy. The first two strategies have different performances on different indicators.

In general, all of the three ensemble strategies improve the accuracy of cancer classification much more than a single classifier. Especially when the samples are limited, all the three strategies have strong ability to extract and generalize significant features.

7.2 Further work

The results of this study show that combining multiple classifiers to clean the original data and using the method of ensemble learning to classify lung cancer images can greatly improve the classification accuracy. Therefore, if more computational power is available, we can try to use most of the samples (more than half) in the database to train the classification system with these strategies. When more images are used to train neural networks, the classification results should be further improved because the features can be better generalized. Since different training strategies have different performances on

different indicators, all three strategies are worth trying.

Extracting features from the dense layer of the classifiers and using Support Vector Machine (SVM) for final classification can obtain more stable results with the training set (Appendix B) [40]. However, SVM does not perform well on independent test sets. A possible reason is that the limited training samples lead to over-fitting of SVM with certain features that only exist in the training set. Therefore, using SVM is still a worthwhile approach when training with more samples.

A The ROC curve of the binary classifiers from ensemble learning

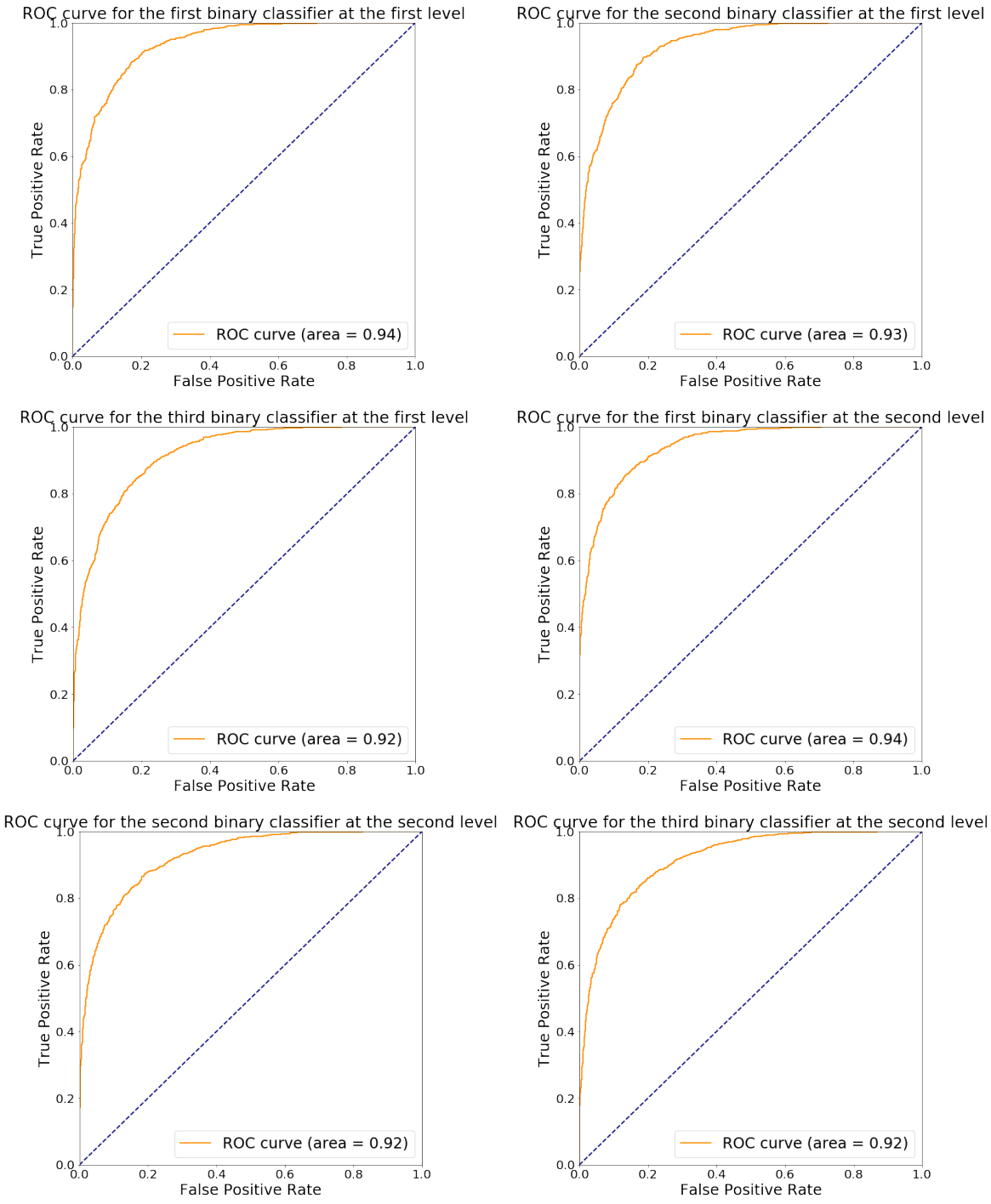


Figure 25: ROC curve of the classifiers at the first and second level with the validation sets.

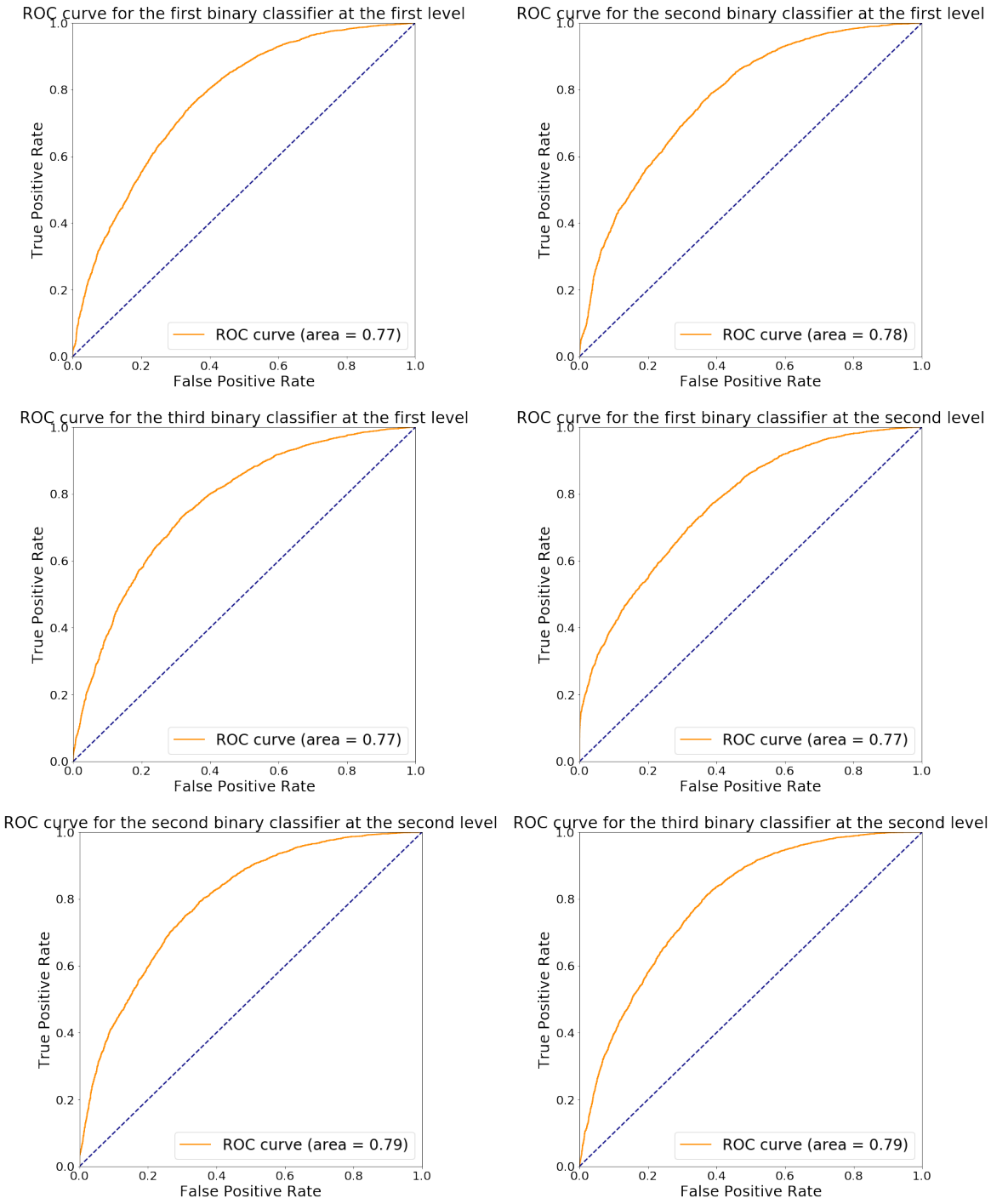


Figure 26: ROC curve of the classifiers at the first and second level with the test sets.

B Classification results with SVM

Table 9: The classification results from the SVM with third strategy at sample level.

Classifier	Validation Accuracy at Sample Level	Test Accuracy at Sample Level
SVM 1	93.3%(112/120)	82.5%(66/80)
SVM 2	95.0%(114/120)	82.5%(66/80)
SVM 1 + SVM 2	94.2%(113/120)	83.6%(67/80)

Table 9 shows the SVM classification results from the third ensemble strategy. Here, we extracted the output features from the flatten layer in the two three-class classifiers and used them as input to train the SVM. The Gaussian kernel was used during training. The results show that SVM achieves high accuracy at sample level with the training set, but just have average performance on the independent test set.

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my thesis supervisor Mattias Ohlsson for his constant participation and helpful advice. Second, I would also like to thank Mattias Aine and Johan Staaf for preparing the data that were used in this project. Mattias Aine, in particular, also provided me with many inspiring ideas about the challenges I encountered in this project. Finally, I would like to thank the department of theoretical physics and the people working in it who gave me the permission to work on the computer with a powerful GPU.

References

[1] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, Volume 2018, Article ID 7068349, 13 pages.

[2] Metin N. Gurcan, Laura Boucheron, Ali Can, Anant Madabhushi, Nasir Rajpoot, and Bulent Yener. Histopathological Image Analysis: A Review. *IEEE Reviews in Biomedical Engineerin*, Volume 2, Page(s): 147 - 171.

[3] Charles S. Dela Cruz, Lynn T. Tanoue, and Richard A. Matthay. Lung Cancer: Epidemiology, Etiology, and Prevention. *Clinics in Chest Medicine*, 2011 Dec;32(4):605-44.

[4] Cancer Statistics Center - American Cancer Society. `https://cancerstatisticscenter.cancer.org/`.

[5] American Cancer Society. What Is Non-Small Cell Lung Cancer. `https://www.cancer.org/cancer/non-small-cell-lung-cancer/about/what-is-non-small-cell-lung-cancer.html`.

[6] Pikor LA, Ramnarine VR, Lam S, Lam WL. Genetic alterations defining NSCLC subtypes and their therapeutic implications. *Lung Cancer*. 2013;82:179–189.

[7] NIH. The Cancer Genome Atlas (TCGA). `https://cancergenome.nih.gov/`.

[8] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E. Dahl, Timo Kohlberger, et al. Detecting Cancer Metastases on Gigapixel Pathology Images. Published 2017 in ArXiv.

[9] Benoit Favre. Deep learning for natural language processing A short primer on deep learning. 03-2018.

[10] Warren S. McCulloch, Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, December 1943, Volume 5, Issue 4, pp 115–133.

[11] Yanbo Huang. Advances in Artificial Neural Networks – Methodological Development and Application. *Algorithms*, 2009, 2(3), 973-1007.

[12] M.Egmont-Petersen, D.de Ridder, H.Handels. Image processing with neural networks—a review. *Pattern Recognition*, Volume 35, Issue 10, October 2002, Pages 2279-2301.

[13] Yoav Freund, Robert E. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, December 1999, Volume 37, Issue 3, pp 277–296.

[14] Yue Liu. Weather impact on road accident severity in Maryland.

[15] Akshay Jain. UC Irvine UC Irvine Electronic Theses and Dissertations Title Deep Learning in Chemoinformatics using Tensor Flow. Published 2017.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. An MIT Press book. https://www.deeplearningbook.org/.

[17] Manish Shrivastava, Manoj Kumar Chinnakotla. Retrieving Semantically Similar Questions in Community Question. Published 2017.

[18] Sumit Saha. A comprehensive guide to Convolutional Neural Networks — the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[19] Huyuan Li. Acceleration of Deep Learning on FPGA. Published 2018.

[20] Yunzhe Xue, Usman Roshan. Image classification and retrieval with random depthwise signed convolutional neural networks. arXiv:1806.05789v2, cs.CV, 9 Oct 2018.

[21] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning: Adaptive Computation and Machine Learning series. The MIT Press, 2016.

[22] Machine Learning Cheatsheet. *https : //ml − cheatsheet.readthedocs.io/en/latest/loss*$_f$*unctions.html.*

[23] Sebastian Ruder. An overview of gradient descent optimization algorithms. Published 2016 in ArXiv.

[24] Diederik P. Kingma, and Jimmy Ba. Adam: A Method for Stochastic Optimization. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[25] Sergey Ioffe, and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:448-456, 2015.

[26] Konstantin. The Mystery of Early Stopping. http://fouryears.eu/tags/data-analysis/.

[27] Lutz Prechelt. Early Stopping — But When. *Neural Networks: Tricks of the Trade*, pp 53-67.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, Volume 15 Issue 1, January 2014, Pages 1929-1958.

[29] Dropout and the Deep Complexity of Neural Networks. https://rcoh.me/posts/dropout-deep-complexity/.

[30] Yudong Zhang, Lenan Wu. Classification of Fruits Using Computer Vision and a Multiclass Support Vector Machine. Published 2012 in Sensors. DOI:10.3390/s120912489.

[31] Payam, RefaeilzadehLei, and TangHuan Liu. Cross-Validation. *Encyclopedia of Database Systems*, pp.532-538.

[32] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, Volume 27, Issue 8, June 2006, Pages 861-874.

[33] Andrew P.Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, Volume 30, Issue 7, July 1997, Pages 1145-1159.

[34] Scikit Learn. Receiver Operating Characteristic (ROC).

[35] Sinno Jialin Pan, and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, Volume 22 Issue 10, October 2010, Pages 1345-1359.

[36] Keras Documentation. Applications. `https://keras.io/applications/`.

[37] Kasthurirangan Gopalakrishnan. Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, 157:322-330, September 2017.

[38] Thomas G. Dietterich. Ensemble Methods in Machine Learning. *International Workshop on Multiple Classifier Systems*, MCS 2000: Multiple Classifier Systems pp 1-15.

[39] R. Maclin, and D. Opitz. Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, Volume 11 Issue 1, July 1999, Pages 169-198.

[40] Corinna Cortes, and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3): 273–297.