# COMP9417 Project Report

**Project name:** Machine Learning vs. Cancer

**Group name**: Team P

**Group member:**


| | |
|---|---|
| Hao Fu | Student ID: z5253457 |
| Leyang Li | Student ID: z5285799 |
| Yingchang Song | Student ID: z5324789 |
| Yanxuan Zhao | Student ID: z5297405 |
| Weihui Zhuang | Student ID: z5175856 |


## 1. Introduction

With The times progress and the development of economics, people's health consciousness is increasing day by day, at the same time with the development of AI technology, AI medical diagnosis deservedly become the tuyere of this industry.

For this project, our task is to utilize machine learning to build some good models so that we can efficiently classify some histologic images which are usually analyzed by a pathologist. A small set of high-resolution images that have already been classified are provided and they are put into different folders: "X_train.npy", "X_test.npy" and "y_train.npy". All of these images are assigned to one of four possible classes, class 0-3. Class 0 indicates there is no tumor and class 1-3 represent different types of tumors. Our model will expertly classify these different types of images and give a good result.

For the special conditions of this task, such as large dataset and quite big size of images, we will first do some processing on the dataset and then implement some sort of feature extraction. Also, three models will be implemented for this task and all of these contents will be fully explained in the following paragraphs.

## 2. Exploratory Data Analysis

In this part, we will process and analyze our datasets to explore different features and find the differences between these features. Also, we will show the data graphically in a way that with a wealth of information.

### 2.1 Training Dataset Analysis

There are three datasets provided in this task, two for our training and one for testing. Firstly, we do some fundamental processing on the training set ('X_training.npy' and 'y_training.npy').

The plot below shows some features of X_train dataset. Based on the basic theory of data analysis, we compute the minimum/maximum value of X_train and the mean standard deviation value.

```
X_shape = (858, 1024, 1024, 3)
X_min = 0.0
X_mean = 0.760689081046465
X_std = 0.2154392658825518
X_max = 1.0
```

The plot below shows some features of y_train dataset. Same as the X training set, we compute the minimum/maximum value of y_train and the mean standard deviation value.

```
y_shape = (858,)
y_min = 0.0
y_mean = 0.8321678321678322
y_std = 0.915950452029815
y_max = 3.0
```

### 2.2 The distribution of the classes

In this task, there are 4 possible classes and each of them indicates different types of tumors so here we do some processing on the classes and the plot below shows the basic information about the classes.

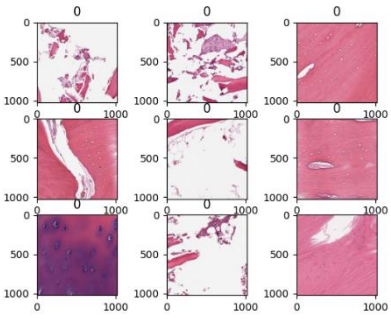
```
Total number of images: 858
Number of Type0 Images: 407
Number of Type1 Images: 225
Number of Type2 Images: 189
Number of Type3 Images: 37
Image shape (Width, Height, Channels): (1024, 1024, 3)
```

It can be found that each category is evenly distributed, and each image has dimension $1024 \times 1024 \times 3$. 3 here represents three color channels (red, green, blue) and based on the size of the image we can compute that there are $1024 \times 1024 = 1048576$ pixels in each image.

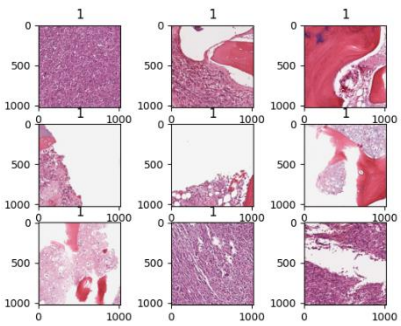
### 2.3 Graphically represent the data

For this section, we will take a bunch of samples and represent the data graphically to make our data expressions more specific and more intuitive. The total number of the samples is 858 and the distribution of each class is shown in last section. Here 9 samples are shown for each class.

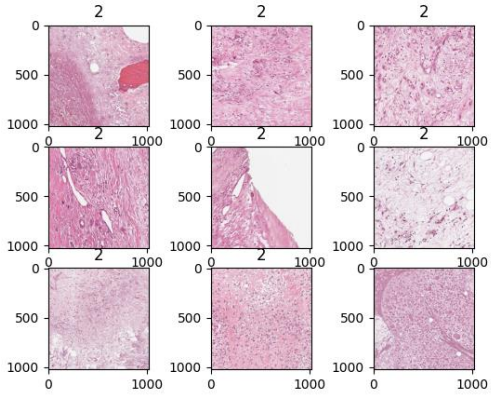
**Class 0**:



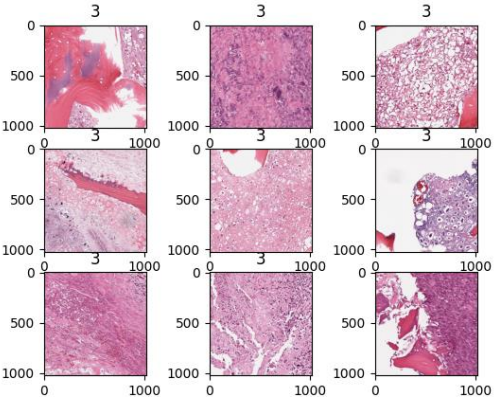
**Class 1:**



**Class 2:**

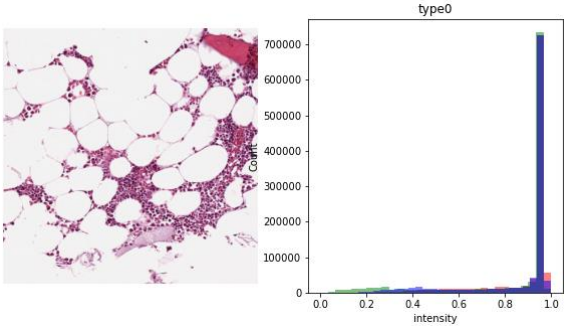


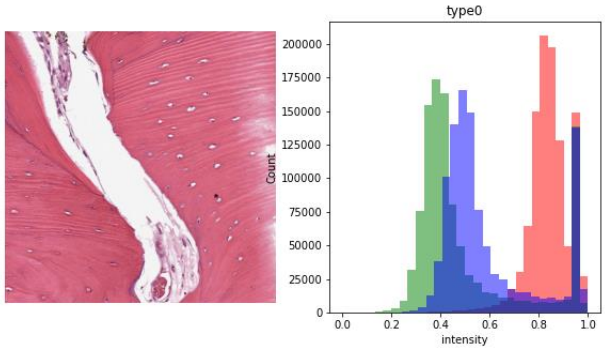
**Class 3:**



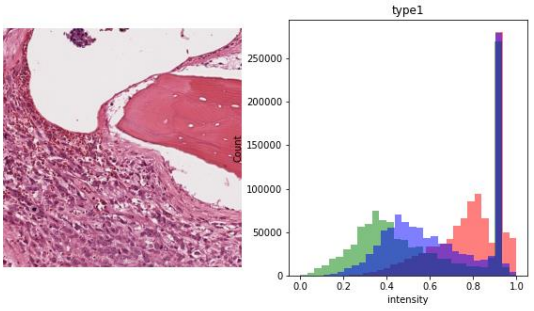
At the same time with visualization, we draw its three-channel values. The x-axis represents the pixel intensity, and the t-axis represents the pixel count number. As the same as before, we sperate them from different class and take some typical examples out.

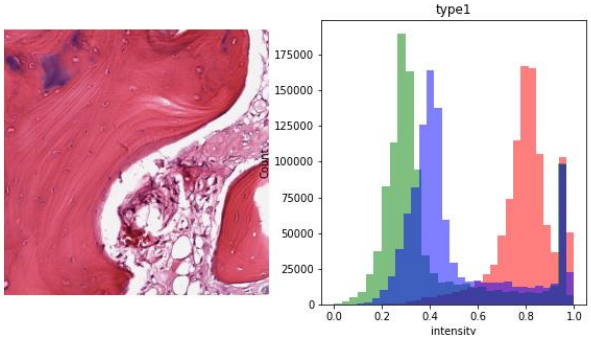
**Class 0**:



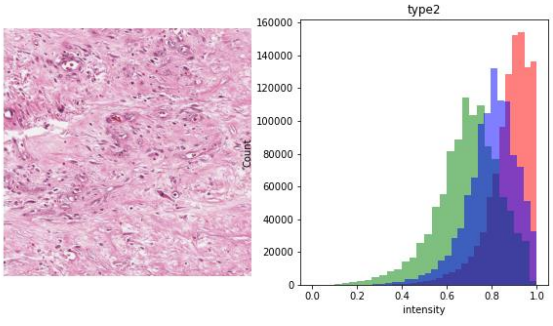



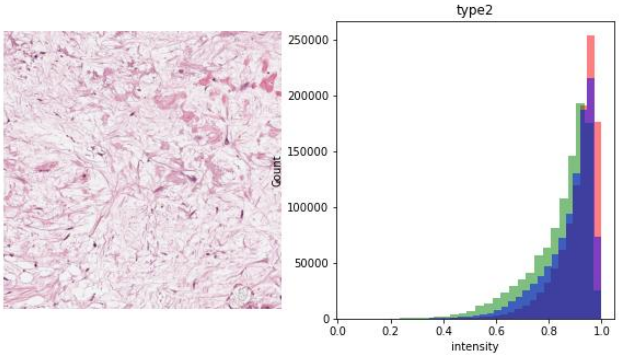

**Class 1:**



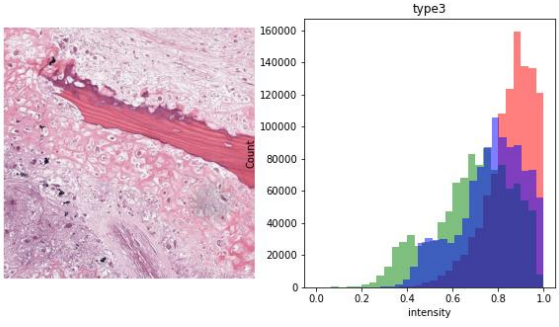



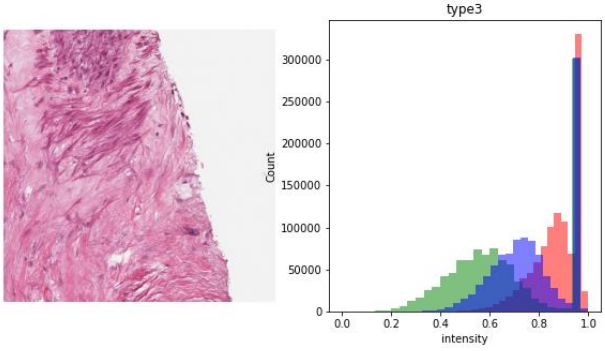

**Class 2:**




**Class 3:**



### 2.4 Feature Analysis

#### 2.4.1 Scaling

For some of the machine learning algorithms, we need to do normalization to make the preprocessed data is restricted to a certain range, so as to eliminate the adverse effects caused by singular sample data. In our original datasets, the pixel range is already 0 to 1 so we do not need to do extra processing such as normalization.

#### 2.4.2 Feature Observation Results

After our discussion, it can be found that not all the pixel color features are related to different class. For example, we find that the proportion of red pixels is higher in all the four types, which means that it may have a weak correlation with the results. However, the proportion of green pixels in various types is uneven so the correlation between green pixels and results should be given priority. On this point, we may apply some algorithm to process it. At the same time, the frequency distribution is another point we will consider since it is easy to show the difference between different classes. Therefore, the histogram of pixels of a single color in each graph should also be taken into account to observe its correlation with the results.

## 3. Methodology

### 3.1 Model1- Gaussian Naïve Bayes

#### 3.1.1 Model Select and Introduction

Bayes theorem is applied in this model. Naïve Bayes model assumes the probability for all features when a hypothesis is independent [1], which is:

$$v_{NB} = \arg max_{v_l \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i|v_j)$$

The final decision is that the hypothesis can provide maximum probability for the equation. Following this theorem, we choose Gaussian Naïve Bayes as our first model.

#### 3.1.2 Feature Analysis

This model assumes that the value of all the features follow the Gaussian distribution (normal distribution). In this way, the likelihood of the feature is assumed to be Gaussian which is:

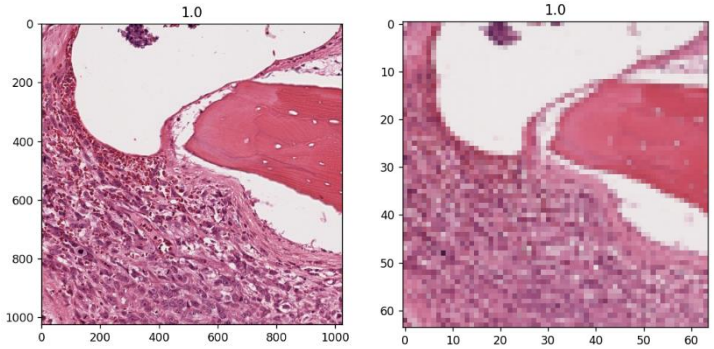
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

#### 3.1.3 Pre-processing

In this section, we choose $sklearn.naive\_bayes.GaussianNB$ to fit the train set. Before this step we need to split the datasets into $X\_train$, $X\_test$, $y\_train$, $y\_test$ by using $sklearn.model\_selection.train\_test\_split$.

The parameter we choose here is: $test\_size = 0.7$ and $random\_state = 42$ .

For each image, there are $1024 \times 1024 \times 3$ pixels which is too large to fit the model. Therefore, we need to simply the dataset. Firstly, without using all the data, we split the images into $64 \times 64$. Each part we use the mean of RGB features in this region. Show this process graphically, the second image is the one after processing:



<before> <after>

Now the dimension of our data is still $64 \times 64 \times 3$. Then we try to collapse the three RGB channels into one. The equation is:

$$p = 0.3 \times r + 0.3 \times g + 0.3 \times b$$

To fit the distribution better, we round all the data and only the first decimal is left. Finally, we will get simpler $64 \times 64$ images than before.

### 3.1.4 Evaluation

Here we use accuracy to evaluate our model. The equation is:

$$accuracy = \frac{n_{true\ positive} + n_{true\ negative}}{n_{sample}}$$

$$F_1 = 2\frac{precision \times recall}{precision + recall}$$

After training, the accuracy of Gaussian Bayes model we got is:

```
accuracy_score: 0.5920745920745921
f1_score: 0.5978393886519331
```

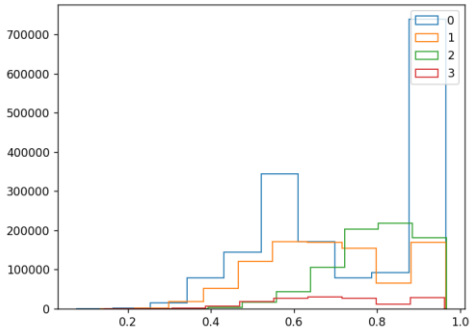
The prediction of $X\_test$ is:

```
[1. 2. 1. 1. 2. 0. 1. 0. 0. 0. 1. 0. 1. 2. 1. 1. 0. 0. 2. 0. 0. 2. 2. 2.
 2. 0. 0. 1. 0. 0. 0. 1. 2. 2. 1. 2. 1. 2. 1. 0. 2. 0. 2. 0. 2. 1. 0. 0.
 0. 0. 1. 1. 2. 1. 0. 0. 1. 2. 2. 0. 1. 0. 1. 3. 2. 3. 3. 2. 0. 0. 1. 2.
 1. 0. 0. 1. 0. 0. 0. 2. 2. 2. 2. 2. 1. 3. 2. 3. 0. 2. 1. 2. 2. 2. 1. 2. 2.
 0. 1. 0. 0. 1. 2. 1. 3. 0. 0. 2. 0. 0. 3. 2. 1. 2. 2. 0. 2. 2. 0. 2. 0.
 0. 2. 2. 0. 1. 0. 0. 2. 1. 0. 0. 3. 1. 2. 1. 0. 0. 3. 2. 0. 0. 0. 1. 0.
 0. 0. 2. 3. 0. 0. 2. 2. 2. 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. 2. 1. 0. 0. 3.
 3. 2. 3. 1. 0. 0. 2. 0. 1. 0. 0. 2. 1. 3. 0. 2. 2. 2. 0. 1. 0. 2. 0. 1.
 0. 0. 2. 1. 1. 2. 0. 1. 2. 1. 2. 2. 0. 2. 2. 1. 2. 2. 1. 2. 3. 1. 2. 1.
 1. 2. 1. 2. 1. 0. 0. 0. 3. 2. 0. 0. 2. 0. 0. 1. 0. 1. 2. 0. 0. 0. 1. 0.
 2. 0. 0. 0. 3. 1. 0. 2. 1. 0. 1. 1. 2. 3. 2. 0. 1. 2. 2. 0. 1. 0. 1. 3.
 3. 1. 1. 1. 1. 0. 1. 2. 0. 2. 1. 0. 1. 2. 0. 1. 2. 0. 1. 2. 2. 0. 1.]
```

This result is stored as $'y\_test\_of\_GaussianNB.npy'$ file.

### 3.1.5 Justification of the method choices

According to this result we got, the accuracy is not particularly good. We think the reason may be that Gaussian Naïve Bayes model assume all the features distribution as Gaussian distribution, but after drawing the histogram of the four classes, it is clear that class 0 and class 1 do not follow the Gaussian distribution.

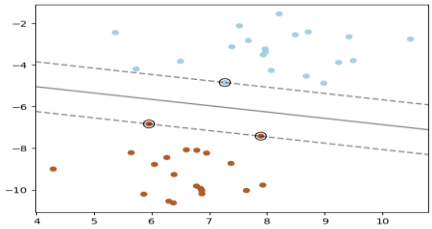


The reason is that there are too many feature's values are 0.9, which there are too much white color region in the pictures. However, if we remove those features which are in high values, for example, the first 80% features, the accuracy will decrease to a very low level. Therefore, we need to find other good models for this project, and we will show them in the following sections.

## 3.2 Model2-Support Vector Machine Classification

### 3.2.1 Model Select and Introduction

This model is applying a linear classification to classify the images into four classes for this task [2]. The final decision of function of SVM is only determined by a few support vectors, and the complexity of calculation depends on the number of support vectors rather than the dimension of sample space, which avoids 'dimension disaster' in a sense. The basic SVM algorithm is like this:



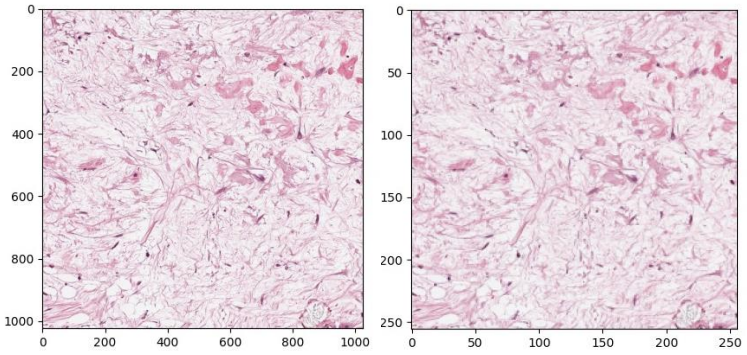
The main expression for this algorithm is:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max(0, |y_i - (w^T \phi(x_i) + b)| - \varepsilon),$$

### 3.2.2 Pre-processing

Same thing we did before, we split the datasets into $X\_train,\ X\_test,\ y\_train,\ y\_test$ by using $sklearn.model\_selection.train\_test\_split$.

The thing different here is we reduce the dimension to 256× 256× 3 to better fit the SVM model



<before> <after>

Then we try to collapse the three RGB channels into one. The equation is:

$$p = 0.1 \times r + 0.6 \times g + 0.3 \times b$$

The parameter we choose here is: $test\_size = 0.2$ and $random\_state = 9$ .

### 3.2.3 Evaluation

Not like normal SVM algorithm (It is a dichotomy strategy). Here we use 'ovr' method to classify the four classes by using four different ways. 0 and not 0, 1 and not 1, 2 and not 2, 3 and not 3

Same things as before we use accuracy to evaluate our model. The equation is:

$$accuracy = \frac{n_{true\ positive} + n_{true\ negative}}{n_{sample}}$$

$$F_1 = 2\frac{precision \times recall}{precision + recall}$$

The accuracy of SVM model is:

```
accuracy score:0.6627906976744186
```

The prediction of $X\_test$ is:

```
[1. 0. 0. 2. 1. 1. 1. 0. 1. 0. 2. 1. 1. 2. 0. 1. 0. 0. 2. 1. 1. 2. 2. 2.
 0. 0. 0. 0. 0. 0. 0. 0. 2. 2. 1. 2. 1. 2. 1. 0. 2. 0. 2. 0. 2. 1. 0. 0.
 2. 0. 1. 0. 2. 1. 0. 0. 2. 2. 2. 0. 2. 0. 0. 0. 2. 0. 1. 2. 1. 0. 1. 2.
 1. 0. 0. 1. 0. 0. 0. 2. 2. 2. 2. 1. 0. 2. 0. 0. 2. 0. 2. 2. 0. 0. 2. 0.
 0. 1. 0. 0. 1. 2. 1. 2. 0. 0. 2. 0. 0. 1. 2. 0. 2. 0. 0. 2. 0. 0. 1. 2.
 0. 2. 2. 0. 0. 0. 0. 2. 0. 0. 0. 0. 2. 0. 1. 0. 0. 0. 2. 2. 0. 0. 0. 2. 0.
 0. 0. 2. 0. 0. 0. 2. 2. 2. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 0.
 0. 0. 1. 1. 1. 0. 2. 0. 0. 0. 1. 2. 0. 1. 0. 2. 2. 2. 0. 0. 0. 0. 0. 0.
 0. 0. 2. 1. 1. 2. 1. 1. 2. 1. 1. 2. 0. 2. 2. 1. 2. 2. 1. 2. 0. 1. 2. 0.
 1. 0. 1. 2. 0. 0. 0. 0. 2. 2. 0. 0. 2. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0.
 2. 1. 0. 1. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 2. 0. 1. 2. 2. 0. 1. 0. 1. 0.
 0. 1. 1. 1. 1. 0. 0. 2. 0. 2. 0. 1. 1. 1. 1. 1. 2. 0. 0. 2. 2. 0. 0.]
```

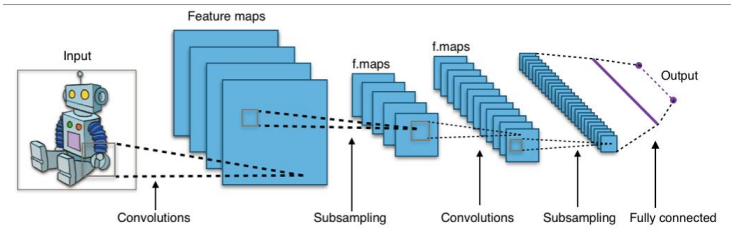
### 3.2.4 Justification of the method choices

According to the accuracy we got, we found that the results were not particularly good. By checking the prediction of $X\_test$ we find that there is no prediction for class 3. One reason may be that the training set of class 3 is not enough, and our algorithm is not that perfect so it can not classify class 3, which leads to low accuracy for the final results. After discussing, we try to do another model to get a better result than the other two, which is CNN.

## 3.3 Model3-Convolutional Neural Networks

### 3.3.1 Model Select and Introduction

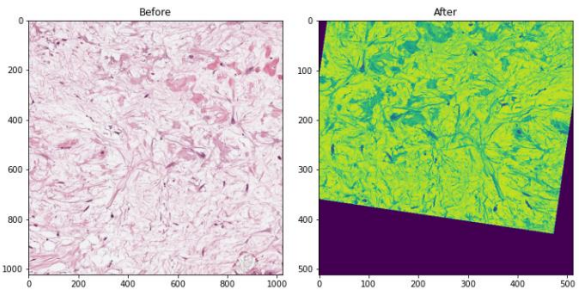
Convolutional Neural Networks (CNN) is a feedforward Network, and its artificial neurons can respond to a part of the surrounding units within the coverage area, which has dominated the computer vision field for a long time. The core of this is convolution and pooling. [3] It is a sub field of deep learning which is mostly used for analysis of visual imagery. This Neural Network uses the already supplied dataset to it for training purposes and predicts the possible future labels to be assigned. [4]




### 3.3.2 Pre-processing

Due to the high pixel of the training datasets and the large capacity, we need to do some transform pre-processing on images to effectively extract the features and strengthen generalization capacity. At the beginning, we find that

the input image was very slow due to too many pixels, so we do transform to reduce the dimension to 512× 512× 3. Then we try to collapse the three RGB channels with different weights into one.



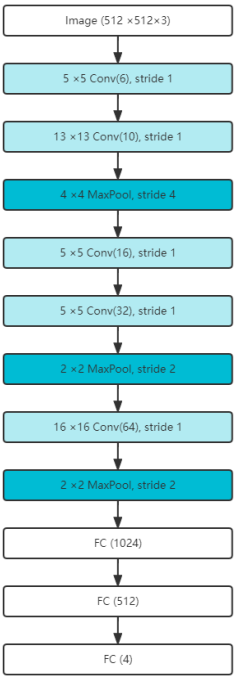

### 3.3.3 Model Structure

AlexNet is widely used in image classification. Although the constructure of AlexNet is quite simple, which only contains eight layers, it has superior performance and speed. The first five are convolutional layers, some of them followed by max-pooling layers, and the last three are fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid. [5]

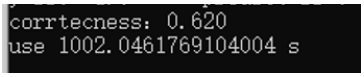
The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. We use Adam optimizer to optimize the CNN.

### 3.3.4 Structure Adjustment

At the beginning, we built a structure like this:



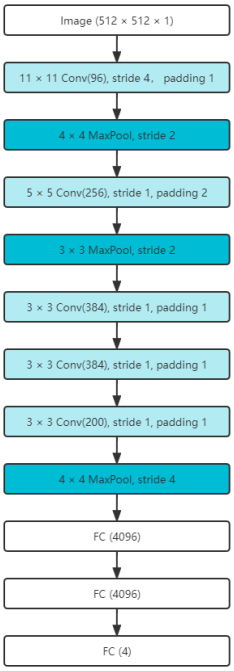

After training, the result is not good as we expected, and the accuracy is very low, only 62%.



Then we discussed the overall structure and analysis the details we found some problems:

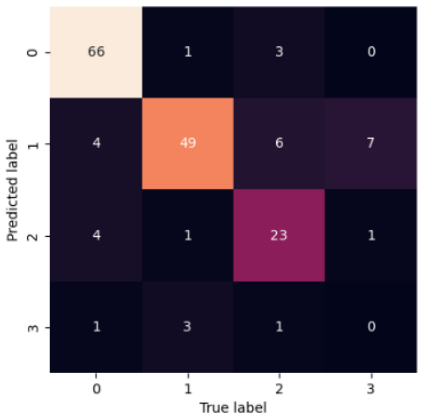
- We used a large kernel and the kernel size does not go from large to small as the convolutional layer goes deeper.
- The depth of our filter is not enough.
- We only used ‘maxpool’ to reduce the number of parameters but ignored ‘stride’ and ‘padding’.

After this process and we got another CNN structure with some improvements.




### 3.3.5 Feature and Hyper-parameter Tuning

Keep training, although the accuracy is increasing, it is still not in a good range. According to the confusion matrix, the recognition rate of model to type 0, type 1 and type 2 is higher than before. But due to the number of type 3 is very low, only 37/858 and it is very scattered, the model cannot classify class 3 which is our main problem now.




Confusion matrix (poor performance)

Therefore, when the model misjudged Type3 as another type, we artificially multiplied a coefficient ‘enhance_rate’ before the calculated cross entropy loss. This forces our model to get bigger results when calculating gradients. This allows the Optimizer to update

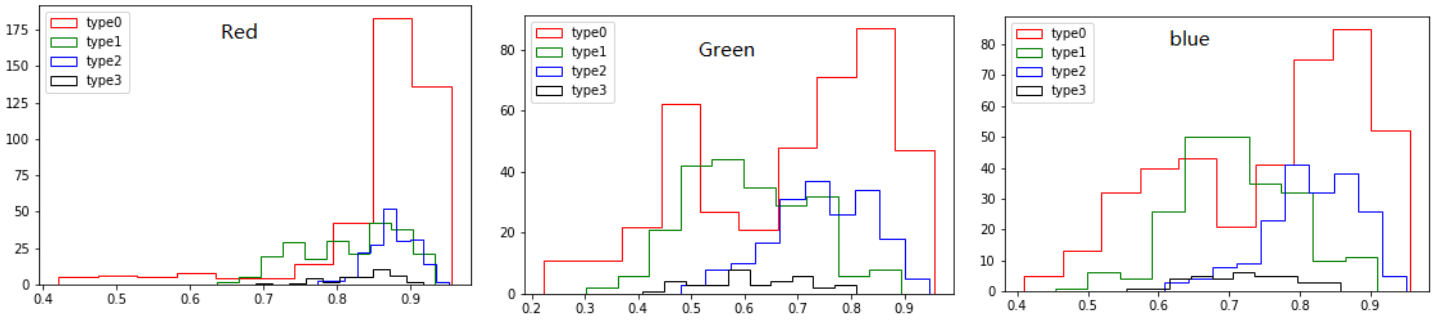
parameters faster based on this gradient. In other words, the model has a higher learning rate when type3 is misjudged

- **Preprocessing Parameter**

Also, we adjusted the preprocessing parameters to better fit the model and get a good result. By analyzing the mean intensity distribution of the three channels in the picture, it is found that the distribution difference of the four categories in the green channel is the most obvious. In order to have a better classification, we multiple the values of the three channels by their perspective weights and then add them up to transform each *x_train* to 512× 512× 1. In this way, the feature extraction can be more efficient, and the parameters entering the network can be further reduced to improve the training speed.

- **Color Weight Parameter**

For Color weight parameters, we set the weight of **red** channel to 0.05, which can reduce its impact on the results. On the two distinct channels, **green** and **blue**, we try to set them like this: $(weight\_green + weight\_blue) = 0.95$.




After trying the average allocation, the allocation focusing on green and the allocation focusing on blue, the allocation focusing on green was selected out according to the accuracy rate and F1 score. Finally, the weight allocation of the three channels is:

$$weight_{red}: 0.05, weight_{green} = 0.75, weight_{blue} = 0.2$$

- **Learning Rate**

At the beginning, we try mixing the two optimizers. Use Adam first and in the final stage, SGD fine tuning was used to end the work, but the effect was not good. After discussion, we decide to use two Adam optimizers but with different learning rate. When the label is type 3, we use the one with larger learning rate to better classify it, but the result is not good.

Finally, we artificially multiplied a coefficient 'enhance_rate' before the calculated cross entropy loss to accelerate training process.The Learning rate started with the magnitude of $10^{-3}$, and repeatedly lowered the magnitude of $10^{-1}$. It was found that the magnitude of $10^{-5}$ was more appropriate. Finally, $10^{-5}$ was selected as the Learning rate.

### 3.3.6 Evaluation

After training, we use correctness and f1 score to evaluate the performance of the CNN model.

The correctness is calculated simply by

$$Correctness = \frac{correct\ predictions}{test\ samples}$$

As the sample number is imbalance, weighted f1 score is considered as well as macro f1 score.

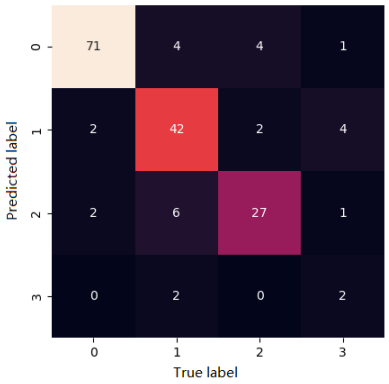
For the macro f1 score, we calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

For the weighted f1 score, we calculate metrics for each label, and find their average weighted by support (the number of true instances for each label).

The accuracy and f1 score of CNN model we got is (last 20% labels of the training set):

```
corrtecness: 0.835
f1(weighted) score is :  0.8283400921054579
f1(macro) score is :  0.7099408422339699
```

The confusion matrix of $X\_test$ is:




## 4. Results

For the first two model (Gaussian Naïve Bayes and Support Vector Machine), we use accuracy to evaluate the result. For the last but the most appreciate model-CNN, we use correctness to evaluate the result. Both of the metrics are shown below:

$$accuracy = \frac{n_{true\ positive} + n_{true\ negative}}{n_{sample}}$$

$$Correctness = \frac{correct\ predictions}{test\ samples}$$

The results of three model are shown below:

| Model | Naïve Bayes | SVM | CNN |
|---|---|---|---|
| Accuracy/Correctness | 0.5921 | 0.6628 | 0.8354 |
| F1(weighted) Score | 0.5988 | 0.6574 | 0.8283 |
| F1(macro) Score | 0.5978 | 0.6436 | 0.7099 |

After our discussion and the training process, it is clear that compared to the other two models, CNN has a better performance. One reason is that for CNN model, we do adequate preprocessing and more adjustments on various hyper-parameters, for example, we artificially multiplied a coefficient 'enhance_rate' before calculating the cross-entropy loss to accelerate training process. Also, constantly adjusted learning rate has a important influence on the final result. In the end, we choose CNN as our final model.

## 5. Discussion

As we can see from the result table, the accuracy of the first two models is not particularly good. For Naïve Bayes model, we choose the distribution as our feature, and it has some limitations, for example, type 0 and type 1 does not follow the Gaussian distribution according to the distribution histogram. For SVM model, it can hardly classify type 3 and our pre-processing is not good enough, so it leads to a low accuracy.

However, after a lot of research and explorations, and a lot of changes and improvements, the result of this model is in a good range as our expected. However, this result is not perfect. Since the pooling layer will lose a large number of valuable information and ignore the association between the part and the whole so it may influence the final result. Fortunately, if the training sets is larger enough, it is going to offset this influence which is our future improvement. In addition, this algorithm can easily make the training result converge to the local minimum rather than the global minimum which is also an important point that we cannot ignore.

## 6. Conclusion

Overall, for this project, to efficiently classify different types of histologic images, CNN is a relatively well performing model. For the other two models, if we have more time, although the algorithm may not perfectly match our task, we could do better on pre-processing and feature extraction. In addition, if we have more time in the future, we may try other good machine learning algorithm, for example, GoogLeNet algorithm. It has a new deep CNN architecture –inception, without full connection layer which can save computation and reduce many parameters. The number of parameters is 1/12 of AlexNet.

## 7. Reference

[1] Jérémie du, J., 2022. 1.9. Naive Bayes. [online] scikit-learn. Available at: https://scikit-learn.org/stable/modules/naive_bayes.html .
[2] Bossche, J., 2022. sklearn.svm.SVC. [online] scikit-learn. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html .
[3] Yamashita, R., Nishio, M., Do, R. and Togashi, K., 2018. Convolutional neural networks: an overview and application in radiology. Insights into Imaging, 9(4), pp.611-629.
[4] Krizhevsky, A., 2022. [online] Proceedings.neurips.cc. Available at: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
[5] Arxiv.org. 2022. [online] Available at: <https://arxiv.org/pdf/1803.08375.pdf>.