# SVMs for Medical Diagnosis from Chest X-ray Images

Sean Lesch, Wenjing Wu, Hoi Dinh, Shalika Arora, Colleen Xu
CS445/545
Final Project Report

## 1.0 INTRODUCTION

This project investigated the effects of different feature extraction methods on the performance of a Support-Vector-Machine (SVM) classifier. The SVM classifier was trained to classify chest X-ray as either “normal” or “pneumonia-like”. The feature extraction methods used were Haralick features from grey-level co-occurrence matrices (GLCM), histograms of oriented gradients (HOG), and local binary pattern (LBP) histograms. For these feature extraction methods, we used 5-fold cross-validation on the training set to find the best hyperparameters (best mean score over the folds) for the SVM model. After running the trained models on the test data, we calculated the confusion matrix, accuracy, precision, and recall as measures of performance. This report will describe our workflow, the details of the feature extraction methods, and our team’s growth during the project. The feature extraction method HOG was found to have the best performance (highest accuracy, precision, and recall). However, all SVM models struggled to classify “normal” x-rays correctly as “normal”.

The work in this project was inspired by previous studies of SVM classifiers and chest X-rays, which showed that the classification performance of trained models is higher than traditional visual inspections [1]. Reports also indicated a difference in classification performance using different feature extraction methods on X-ray images [2].

## 2.0 METHODOLOGY

A chest X-ray image dataset was retrieved from Kaggle [3]. The data is from pediatric patients (one to five years old) from Guangzhou Women and Children’s Medical Center, Guangzhou, China. The dataset contains 5,863 X-ray images, already split into training, validation, and testing directories. We used the training and testing directories for our project. The two classes were “normal” (0) and “pneumonia-like” (1).

The programming language used was Python version 3.6. The workflow for this project can be seen in Figure 1 (see next page).

### 2.1 Image Preprocessing

Images from the dataset are different-sized JPEG files that use the file structure hierarchy for labeling. They are generally high resolution, but they have different aspect ratios.

First, we resized all images to be 250 x 250 pixels while preserving the aspect ratio of the original images. This was done by first scaling the photo to make its maximum dimension 250 pixels, then filling “empty space” in the other dimension with black pixels. Figure 2 (center, next page) shows an example of this preprocessing.

Then, image contrast was increased using adaptive histogram equalization. It improves local contrast by computing the histogram of pixel values over a section of the image and then spreading those values across the entire range. We hoped that this would make the “cloudiness” in the lungs (a sign of pneumonia) more obvious and would improve feature extraction. Figure 2 (right, next page) shows an example of this preprocessing.

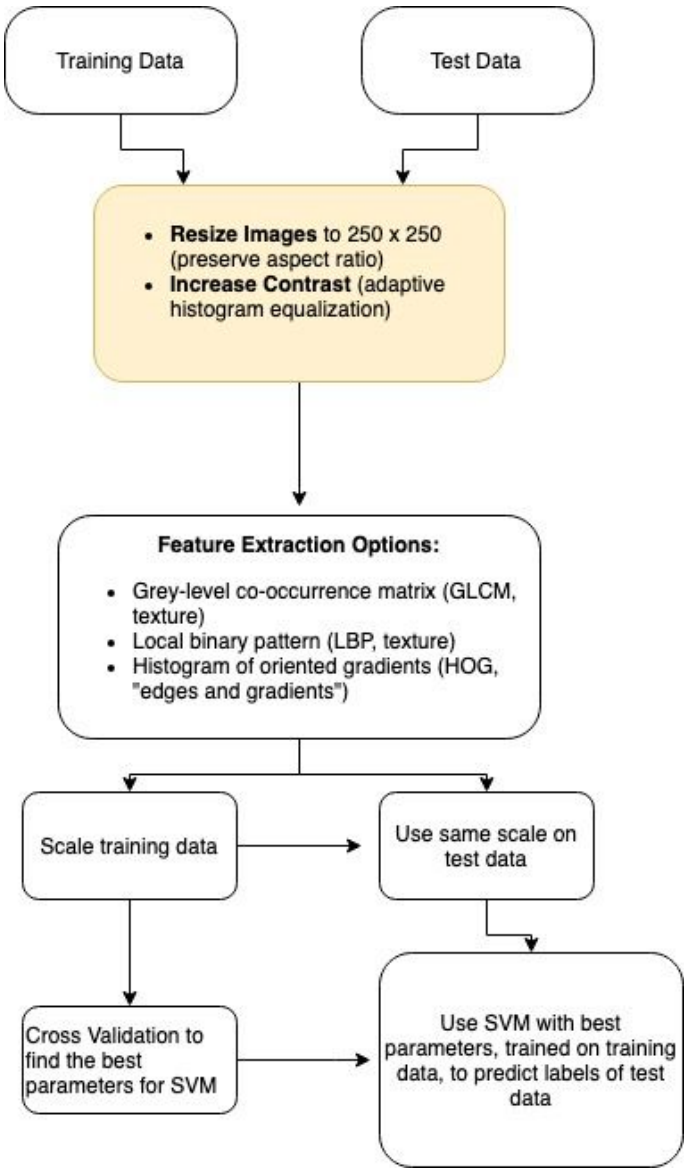



Figure 1: Workflow diagram for the project.

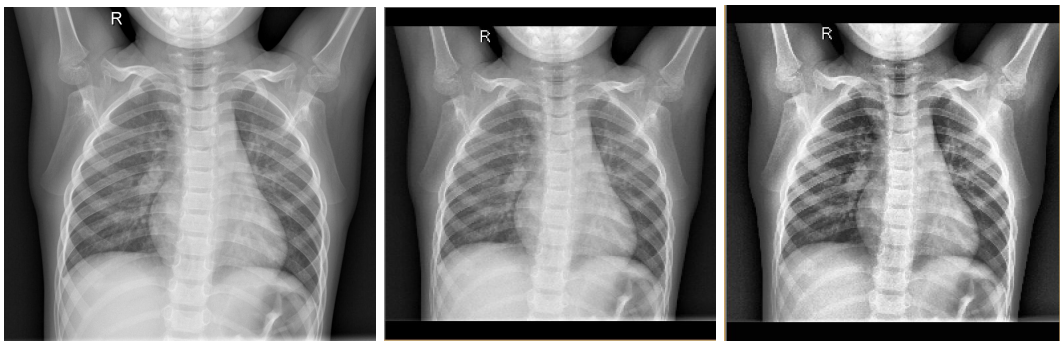


Figure 2: X-ray preprocessing on a X-ray of the “normal” class. Original (left), resized image (center), resized and increased contrast (right).

Finally, downsampling was used to make the number of images in each class even. In the original training set, there are 1341 normal x-rays and 3875 x-rays with signs of pneumonia. If this entire set was used to train the SVM, the model may have poor performance on the minority class ("normal" class) [4]. To combat this, we took a random subset of the pneumonia images (1341 out of 3875) for training. For the GLCM and HOG implementations, we did this subsetting after the resizing step; for the LBP implementation, we did the subsetting after the feature extraction steps described below.

### 2.2 Feature Extraction

The pixel values of even the resized images (250 x 250 = 62500) would be too many features for an SVM. We therefore need feature extraction algorithms to find potentially useful features for classification.

This project compared three different feature extraction methods: GLCM (Haralick features), LBP, and HOG.

#### 2.2.1 Gray Level Co-occurrence Matrix (GLCM)

A gray level co-occurrence matrix describes the patterns of gray-level repetition in an image. It was first introduced by Haralick et al. in 1973 [6] and is used for gathering information from the underlying image structure regarding **texture**. It is constructed by determining the distance and orientation of a target pixel from a reference pixel. The result of computing a GLCM is a four-dimensional matrix, *P(i, j, d, theta)*. Indexes *i* and *j* in the resulting matrix correspond to pixel intensity value pairs. The distance *d* is how far *j* occurs from *i*. The angle *theta* is the angle from *i* to *j*. Thus each value in this matrix "is the number of times that gray-level *j* occurs at a distance *d* and at an angle *theta* from gray-level *i*" [7].

The GLCM is only applicable to grayscale images. The size of the resulting matrix is dependent on the scale of pixel values (ex: 8 bit, 16-bit) and the hyperparameters used. The distances and angles between the reference pixel and neighbor pixel can be changed. For this project we used *d* = [1, 2, 3] and *theta*=[0°, 90°, 180°, 270°]. Values in the GLCM were made symmetric and normed (parameters of the GLCM function call were set to True).

Once a GLCM has been determined, relevant texture features can be extracted from it. Our project computed the contrast, correlation, homogeneity, energy, angular second moment and dissimilarity of each image's GLCM, resulting in 72 values (one for each property, angle, and distance combination). The mean across the angles was computed for each property-distance combination, resulting in our final feature vector. Each image's final feature vector had 18 elements (6 properties x 3 distances).

#### 2.2.2 Local Binary Pattern

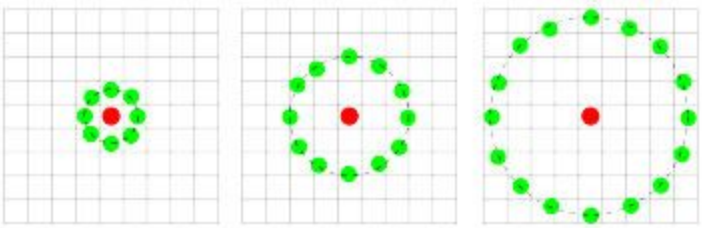
The local binary pattern (LBP) features, which are texture descriptors, gained popularity in computer vision starting in 2002 and are widely used for texture matching. LBP is a local representation of texture that compares each pixel with its surrounding neighborhood of pixels. LBP only works on grayscale images.

There are three steps in the algorithm:

1. **(optional) Convert images to grayscale.** Since our source images are gray-scale X-ray images, we don't have to worry about this part.
2. **Calculate the LBP mask for each pixel.**

LBP selects a circular neighborhood of radius *r* with *p* number of pixels surrounding the center pixel. If the intensity of the center pixel is greater-than-or-equal to its neighbor, then the value is set to 1; otherwise, it is set to 0. Based on these comparisons, we get a *p*-length binary array.

The array is then classified as one of the uniform patterns or as non-uniform. First, the number of 0-1 and 1-0 transitions in the array are calculated. If the number of transitions is less than 3, the array's pattern is considered "uniform". There are $p$+1 possible uniform patterns corresponding to $p$+1 uniformity values. Otherwise, the pattern is considered "non-uniform" and assigned the "non-uniform" value. [5; image below is also from reference 5]



**3. Calculate the LBP Histogram and normalize**

Calculate a histogram describing the image based on the uniformity values for each pixel, found in the previous step. The number of bins will be $p$+2 (number of uniform patterns+1 for the non-uniform patterns)

**We used a radius of 3 (24 neighboring points) to end up with 26 features per image.**

### 2.2.3 Histogram of Oriented Gradients

A Histogram of Oriented Gradients (HOG) is commonly used in object and edge detection. It was popularized by Dalal and Triggs in 2005, who used its features to train an SVM to detect pedestrians in static images [8]. HOG involves the calculation of the magnitudes of the pixel intensity vectors from various angles around the center of an image's cell. The resulting histogram of angle vectors is immune to image deformations. These histograms are then concatenated to give a descriptor of the full image.

The first step in HOG is to compute the gradients of each window/cell. Cells must have a fixed aspect ratio [12], which is guaranteed for our project by using square, **50x50-pixel cells**. Gradients are computed by filtering the intensity data of the image with horizontal and vertical kernels. Next, the cell's histogram is created. This is done by counting the number of gradient vectors in each angular bin [12]. An angular bin is a set of gradient directions evenly spaced from 0-180°. To see a visual example of this process, see Figure 3 (next page). To deal with illumination and contrast differences, gradient strengths are then normalized over larger blocks [8]. Our project normalized across a **block size of 3x3 cells**. The default L2-hys block normalization method was used. As blocks are normalized they are concatenated into a final feature vector. **For our project, each image had 729 features.**

### 2.3 Shuffling Data and Scaling Features

The training data started out ordered: there were 1341 "normal" observations followed by 1341 "pneumonia" observations. Although SVM is less sensitive than other methods to the order of the input data, we shuffled the order of training data.

The SVM algorithm is sensitive to the scale of the input features because the optimal SVM is found by minimizing the decision vector $w$ [9]. Additionally, scaled features are an assumption of the RBF kernel of SVM [10]. We therefore need to scale the features of the training data so that each feature has a mean of 0 and a standard deviation of 1 over all the observations. This was done using a Scaler object in the preprocessing library of scikit-learn. The scaled training data was then fed to the SVM for cross-validation.

The test data was scaled using the **same** scaling as the training data (using the training data's means and standard deviations). In general, test data underwent the same preprocessing procedures as the training data.

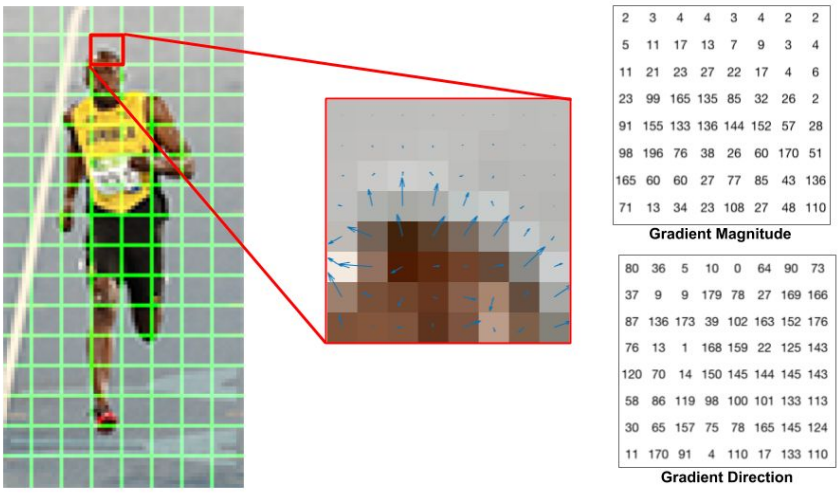



Figure 3: Results of computing the gradients of a single cell in HOG feature extraction [11].

### 2.4 Classification and Performance Evaluation

We used a **soft-margin** SVM classifier to predict the label of the test X-ray images using the features extracted with each method. SVMs are supervised, binary classifiers that find an optimal decision hyperplane by maximizing the margin between the points closest to the hyperplane (known as support vectors). A trained SVM classifies a new datapoint by analyzing which side of the hyperplane the point lies on.

The hyperparameters for SVM models include the choice of kernel, choice of C (large C makes the cost of misclassification high), and choice of gamma for non-linear kernels. For the LBP implementation, we used grid search 5-fold cross-validation to find the best parameter values among 32 options: linear or rbf kernel (non-linear), 4 C values (0.1, 1, 10, 100), and 4 gamma values ('scale', 0.01, 0.001, 0.0001) [for definition of scale, see sci-kit learn's SVC function documentation]. For the other two feature extraction implementations (GLCM and HOG), we did cross validation with 16 options due to time constraints ('rbf' kernel only, same possible values for C and gamma).

All three final models used the 'rbf' kernel (non-linear) to transform the input into a higher dimension. The model with GLCM features used C=100, gamma='scale'. The model with HOG features used C=10, gamma=0.001. The model with LBP features used C=10, gamma=0.01.

The classifiers' performance was evaluated using confusion matrices and accuracy, precision, and recall calculations.

## 3.0 RESULTS

### Model with GLCM Features

Confusion Matrix:

| | 0 (True Normal) | 1 (True Pneumonia) |
|---|---|---|
| 0 (False Normal) | 118 | 32 |
| 1 (False Pneumonia) | 116 | 358 |

Accuracy : 76.28 %
Precision : 75.53 %
Recall : 91.80 %

**Model with HOG Features**

Confusion Matrix:

| | 0 (True Normal) | 1 (True Pneumonia) |
|---|---|---|
| 0 (False Normal) | 134 | 6 |
| 1 (False Pneumonia) | 100 | 384 |

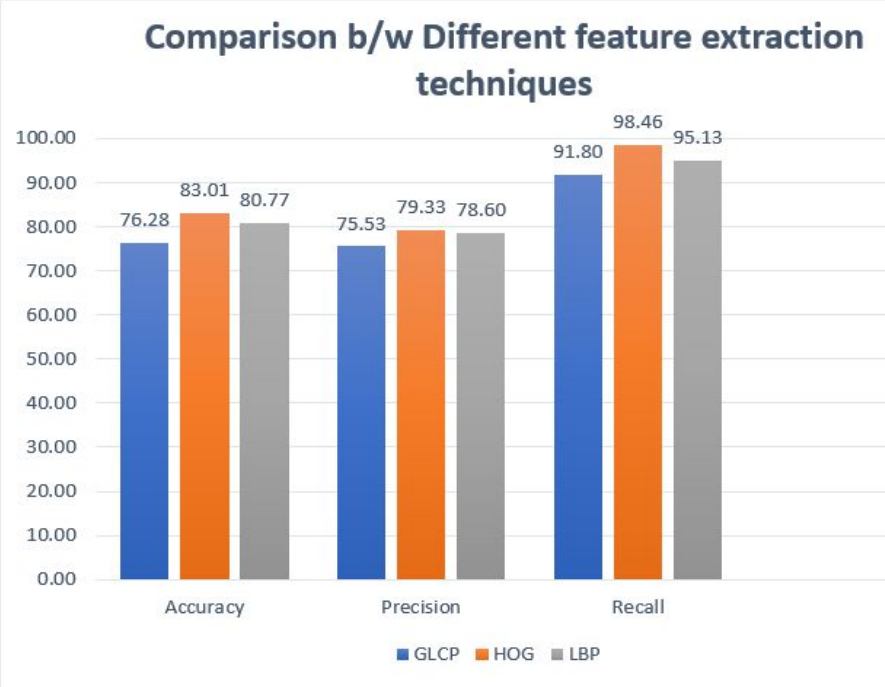
Accuracy : 83.01 %
Precision : 79.33 %
Recall : 98.46 %

**Model with LBP Features**

Confusion Matrix:

| | 0 (True Normal) | 1 (True Pneumonia) |
|---|---|---|
| 0 (False Normal) | 133 | 19 |
| 1 (False Pneumonia) | 101 | 371 |

Accuracy : 80.77 %
Precision : 78.60 %
Recall : 95.13%




## 4.0 CONCLUSIONS

The model using HOG features had the best performance (highest accuracy, precision, and recall). All models did well (>90% recall) on classifying the true “pneumonia-like” test samples as “pneumonia-like”. However, all models struggled with classifying the true normal test samples as normal. This classification of true normal test samples as normal was difficult for us to do as well

(looking by eye). Some images labeled “normal” had some cloudiness in the lung area, which we and possibly the SVM confused for a sign of pneumonia.

The feature extraction method HOG differed from the other two feature extraction methods in that it did not extract texture information. Instead, HOG extracted gradient information, which helps in object and edge detection. It may be that the gradient information HOG extracted were more useful predictors than the texture information extracted by the other feature extraction methods.

A potential way for us to improve the performance of our SVM models would be to combine the features from our multiple feature extraction methods. The literature we found used combinations of features from various feature extraction methods [1, 2]. Combinations of features from various feature extraction methods allows the SVM to use different properties of the original image (texture, gradient, edge information) as predictors.

**Citations**


[1] https://www.hindawi.com/journals/jhe/2017/4620732/
[2] http://ejum.fsktm.um.edu.my/article/1343.pdf
[3] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
[4] https://elitedatascience.com/imbalanced-classes
[5] https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/
[6] http://haralick.org/journals/TexturalFeatures.pdf
[7] https://scikit-image.org/docs/0.7.0/api/skimage.feature.texture.html
[8] https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
[9]https://towardsdatascience.com/effect-of-feature-standardization-on-linear-support-vector-machines-13213765b812
[10]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[11]https://www.learnopencv.com/histogram-of-oriented-gradients/
[12]https://www.learnopencv.com/histogram-of-oriented-gradients/