Project Name: Defect Detection

Introduction: Computer vision applications are very important in solving many problems. Some of the problems it could solve include object detection, recognition (Clustering and classification), improve robotics development, ease feature extraction/selection problems and so on. In this project, we are going to go through how computer vision algorithms and artificial neural networks could help us detect the defect in images. Particularly, we are going to employ and compare deep neural networks and bag of visual words (BOVW). I will show the steps that are taken in both approaches and demonstrate the results afterwards. Due to small data, I have discovered that BOVW performed well on this specific data and obtained very high accuracy without overfitting to the data.

The project is ordered as follows. Section 2 focuses on the analysis of the provided data. Section 3 discusses the two approaches. Section 4 demonstrates the results. And finally, section 5 concludes and reviews further approaches that could improve the accuracy of Convolutional Neural Networks (CNN).

Data: In this project, a labeled image dataset is provided. According to the structure of the data, fault images are given positive (pos) labels and non-defect images are given negative (neg) labels. "This was the format that the data was sent". On the CNN approach, I worked on the provided structure. But on the BOVW approach, I had to separate the neg and pos images in the Test folder (see fig 1). Apart from this, I analyzed the data and discovered two things: I) The dataset is small (575 train and 575 test) and in this respect, CNN has limitations in providing good results. II) the second discovery is that the dataset is imbalanced (see fig 2). Imbalanced data means when one class is overly represented than other classes – in our case (pos) class -. For example, in our case, we have a 2-class (binary) classification problem with 575 train images. A total of 492 instances are labeled with (pos) class and the remaining 83 instances are labeled with (neg) class. This brings a situation where some models give false results by overfitting to the overly represented class. **However, in most cases, especially in industries producing tons of products it is difficult to obtain a balanced and enough data. Therefore, providing a model that could easily detect defects in their systems are matters of paramount importance. In this approach/project I demonstrate how my models could easily overcome this issue and provide a better result even if the dataset is small and skewed.**

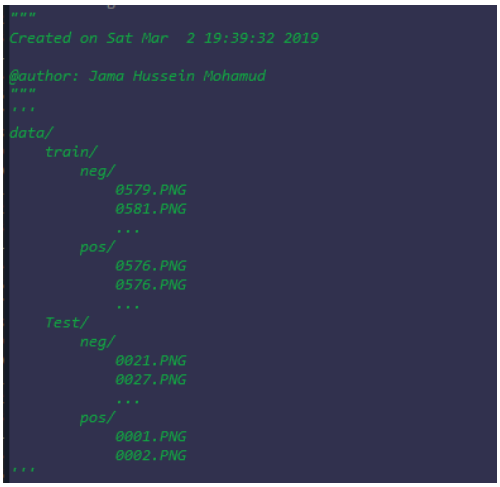

Figure 1. structure of the data (only for BOW), but for CNN approach the format is preserved as it was received.

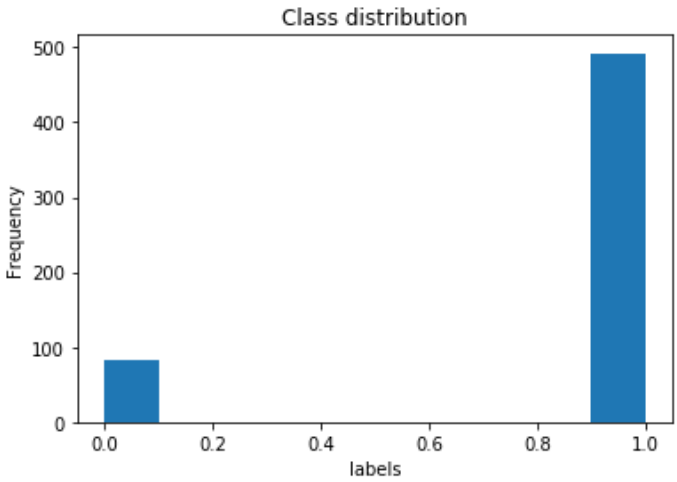

Figure 2. Histogram showing disproportion of the classes. 0 (neg) and 1 is (pos)

Approach 1: Convolutional Neural Networks (CNN)

CNN is one of the best and promising deep neural network algorithms. It is mostly applied to visual imagery. It has totally reshaped the conventional computer vision approaches to image recognition and object detection. In this workflow, I constructed a CNN approach based on Keras and TensorFlow libraries to extract features that indicate the defects in the images. Although deep learning models such as CNN are designed to learn from massive data, the smallness of the train data made it tough the algorithm to perform well on test data. Therefore, to overcome this issue and benefit from the beauty of deep neural network, I tried the concept of "transfer learning".

Transfer learning is a technique where a model trained on a specific task is reused on another task. Models such as VGG16, VGG19, and RESNET are trained on thousands or millions of images (ImageNet datasets). Therefore, by training the model on these tons of images, the weights in the different layers of the classifier are well tuned to identify all sorts of useful features. some of the features learned by these pre-trained models are shapes, edges, and different intensities of white and black pixels. These extracted indicators/features are beneficial and can be exploited to our task (defect detection). The approach is implemented as follow:

- Select source model (in my case I chose VGG19)
- Reuse the model (I froze the layers that I did not want to train – in my case the first 15 layers)
- Finally tuned the model to fit my own task.

In the implementation of CNN, I divided the training data into two portions. One portion to train the model and another validation portion to tune the parameters of the model. And the test data was kept untouched.

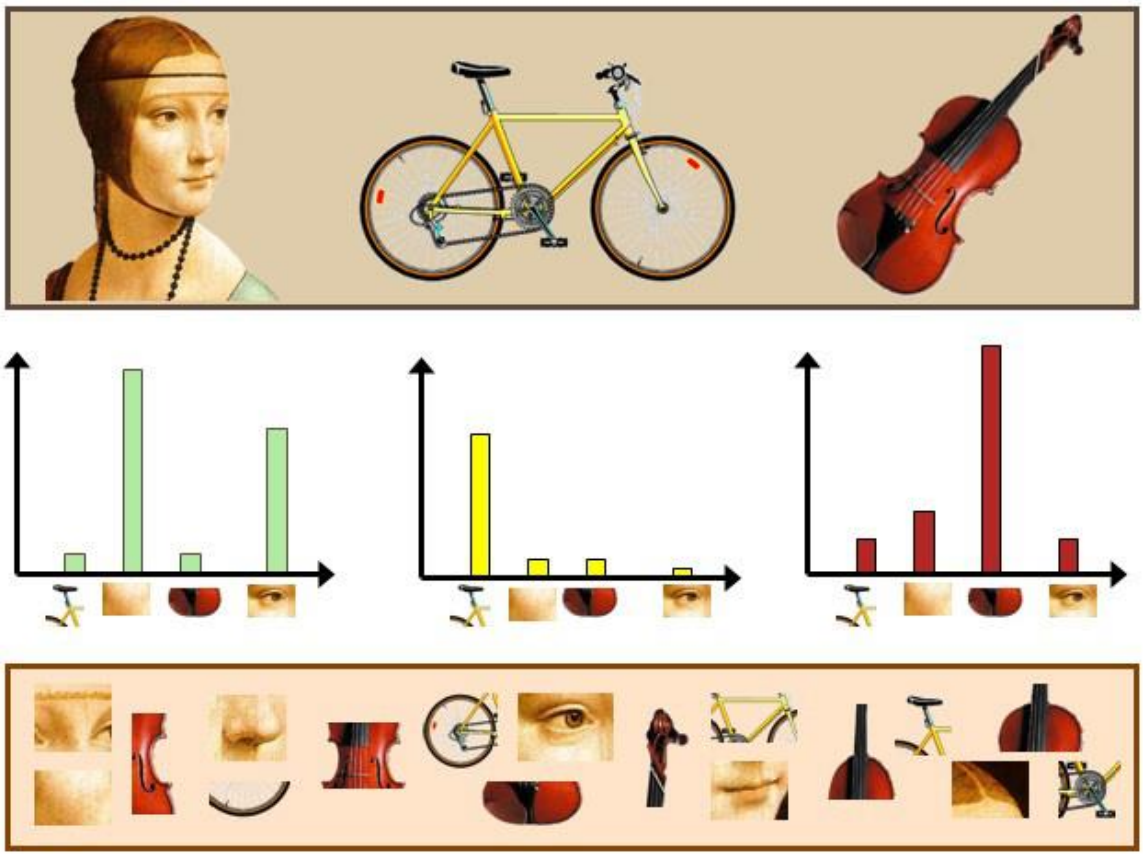

Figure 3 Bag of Words model (Image taken from Vaibhav dedhia)

Approach 2: Bag of Visual Words

Bag of Visual Words (BOVW) is a widely used method for image classification. It's adapted from NLP's bag of words (BOW). In BOVW, the images are represented as a set of features consisting of key points and descriptors. The method generally follows 3 simple steps: Feature detection, feature description, and codebook generation. Detecting features and extracting descriptors in an image is done by using SIFT or SURF feature extraction algorithms. Then clusters are made from the descriptors and the center of each cluster is used as the visual dictionary's vocabularies. Finally, frequency histogram is computed from the frequency of vocabularies in the image.

In short, in this approach I performed the following tasks:

- Feature Extraction: the first step to build a BOVW is to perform feature extraction by extracting descriptors from each image in our data. This can be done by utilizing feature extractor algorithms such as SIFT and SURF. From the mathematical point of view, SIFT or SURF will provide us multiple feature vectors from each image (see fig 4).
- Codebook construction: After feature vectors are extracted from each image in our data, we will construct our vocabulary of possible visual words. This can easily be achieved by using clustering algorithms (e.g. k-means

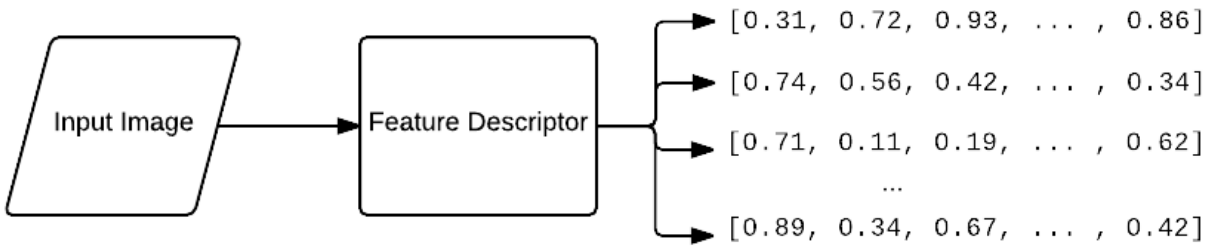

Figure 4 extracting multiple feature vectors per image (Image taken from Gurus blog post)

or DBSCAN). These clustering models will cluster features we extracted in the first step. Then, the resulting cluster center i.e. centroids are treated as the dictionary of visual words.

- Vector Quantization: in this stage, we will take a set of nearest neighbor labels and construct a histogram of its length equal to the number of clusters generated from K-means. And the I'th value of the histogram is the frequency of the I'th visual word.
- Finally, as we have a bag of visual words representation of our images, we can flawlessly apply to traditional machine learning models. In my case I used SVM, but a code for KNN is also provided, you only need to comment either the section of SVM or KNN if you intend to train again.

Results: Due to small dataset and disproportion of the distribution of the classes, we evaluated our models with different metrics like accuracy, confusion matrix, and train, validation and test accuracies obtained from CNN in Keras. I can argue that both models performed well on the provided dataset. In such situations (i.e. small and unbalanced data) I advise the deployment of BOVW instead of CNN. As it not highly recommended (due to overfitting) the usage of the deep neural network on small datasets.

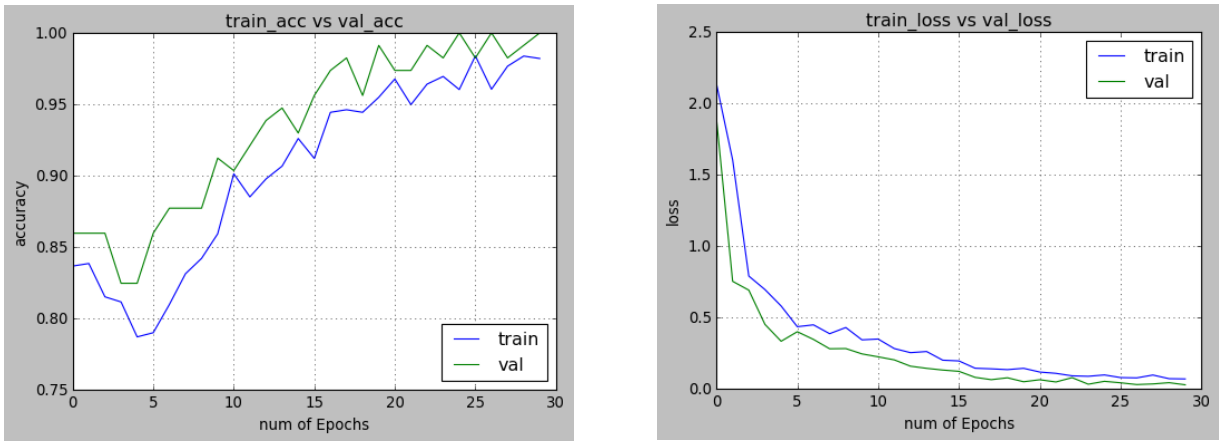

Figure 5 train-validation accuracy and train-validation loss of CNN

In Figure 5, we see that how good CNN performs on train and validation data. This is due to overfitting if you try to predict on unseen data (Test set) it will fail to provide good results. Because of imbalanced data, the model will predict all test images to the majority class – our case (pos) class. This is the reason that inspired me to use BOVW. Fortunately, as soon as I tried to use this method, I got an astonishing performance (see Table 1). Since our data is imbalanced, we cannot be dependent only on train-test accuracy. But we have to calculate the confusion matrix as well which will give us results that are representative of how our model will perform on unseen data (See fig. 6).

From the confusion matrix we can conclude three things: I) the classifier made a total of 575 predictions, II) out of those 575, the classifier predicted "**pos**" 496 times and "**neg"** 79 times. III) In reality, 508 images are **pos** and 67 images are **neg**. this is a good performance as the model wrongly predicted only 36 images without too much hyperparameter tuning. The test accuracy of BOVW can be improved if further parameter tuning is done. We can also try using other models instead of SVM and see how it performs.

Table 1 SVM results

Classifier	Train Accuracy	Test Accuracy
SVM classifier	1.0	0.937

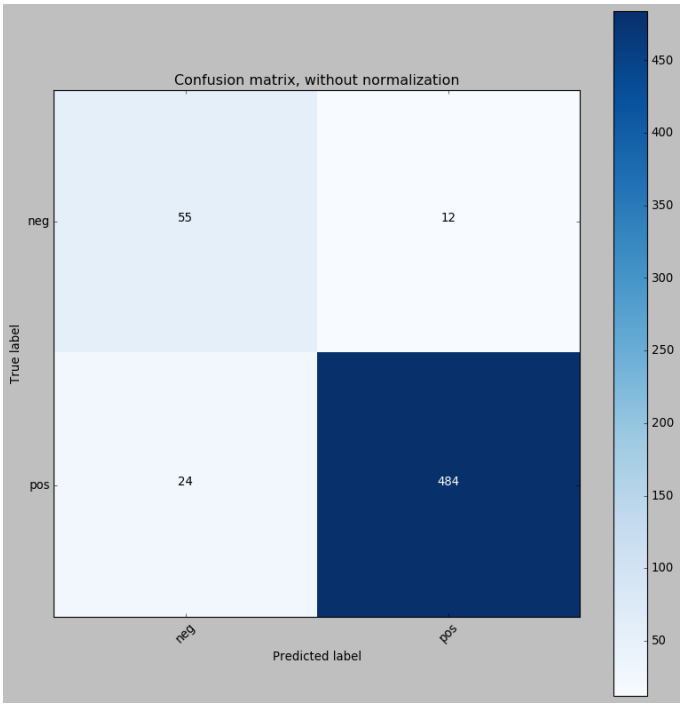

Figure 6. BOVW confusion matrix

Note: "According to the structure in which the data was provided, the **neg** folder consists of fault/defect images (according to my understanding), and the **pos** folder consists of non-defect images. Whatever the case, this will not change the main concept of solving the problem."

Conclusion: In this project, I showed how machine learning algorithms could help us solve the task of "Defect Detection". In my workflow, I trained traditional machine learning models (particularly SVM) and convolutional neural networks. From the results, we saw that BOVW provides a promising result on this specific data. Apart from this, various other methods can be performed to improve the accuracy of CNN. Some of the further steps we can take can be;

- Perform more aggressive data augmentation
- Use more aggressive dropout
- Use L1 or L2 regularization
- Try freezing different layers of the VGG19 architecture
- Increase the dataset if possible

But, nonetheless, if any of these is not possible, we have to use BOVW as an alternative to CNN that will do almost all the work we need. If you would like to use these models to predict to new images you can easily use the provided saved model. Also, the weights of the model are preserved if further hyper-parameter tuning is needed.

Jama Hussein Mohamud
MSc Electrical and Electronic Engineering
Anadolu University
Jamahusseinmohamud@anadolu.edu.tr
+905522498411