Industrial Quality Control Of Packages

Domitila Desai
Karthick Narayanen Murugan
Mouriya Srinivasan

Introduction:

- Packaging machines are used by several industrial companies to wrap their products. These machines are normally fully automated, but occasionally, incorrect packages are generated, for example, because little changes in package position leads them to be dented or bent.
- As a result, across the manufacturing line, there are in-line quality control inspection sites where various quality steps are done to guarantee that incorrect products are removed from the process before they are shipped out.

Project Objective

- The Objective of this project is to create a model of a pharmaceutical manufacturing plant.
- The machine utilized occasionally generates dented and damaged packages.
- The packs are transported on a conveyor belt along an inspection line, which is equipped with two cameras (top and side).
- The objective is to develop a machine learning model that can distinguish between damaged and undamaged parcels.

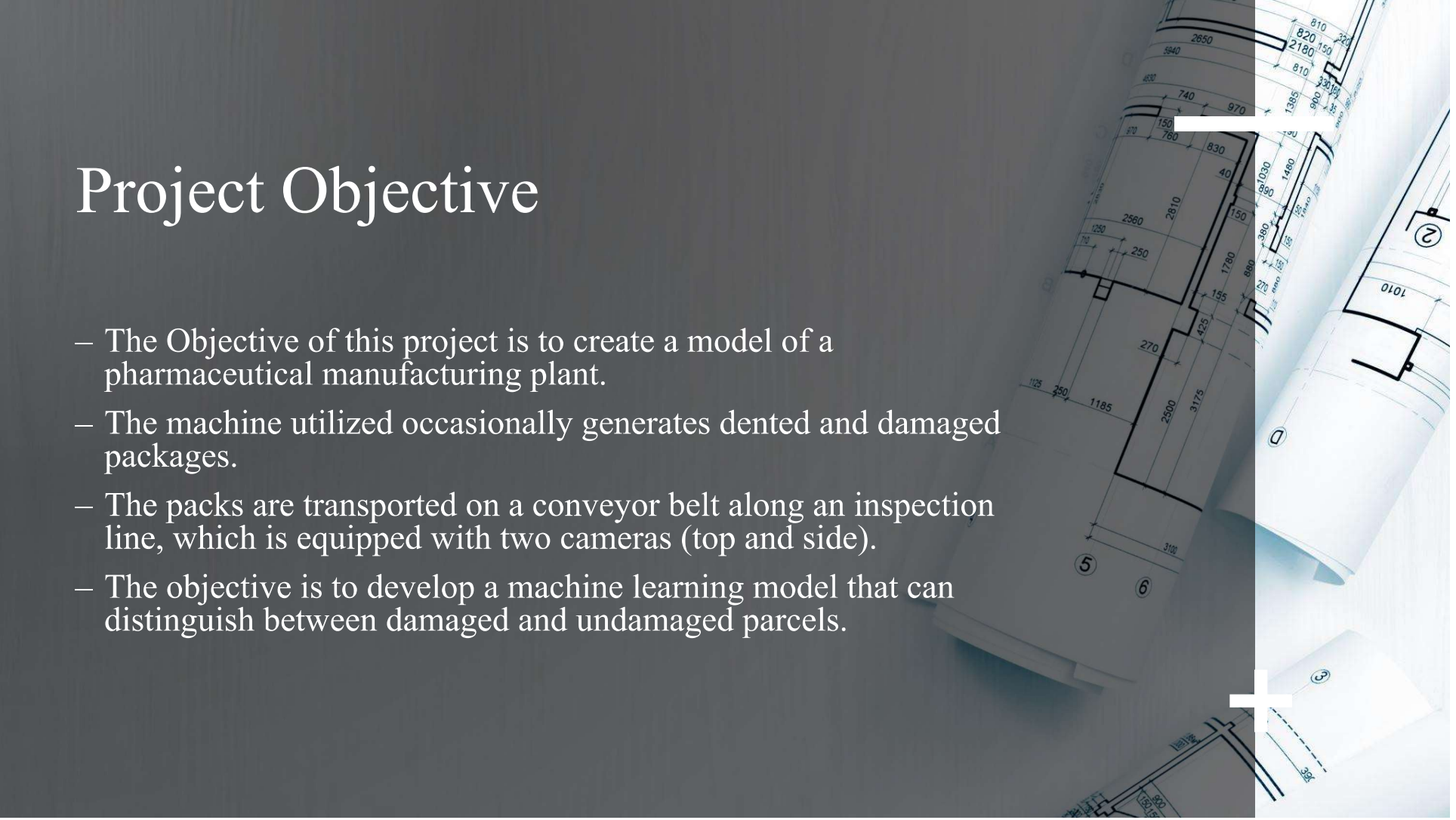


Image is classified as either intact or damaged

- Damaged
  -> side : 100
  -> top : 100

- Intact
  -> side : 100
  -> top : 100

Preparation of the Data

→ We are first defining all the functions that we are going to use

find_largest_contour-This function finds all the contours in an image and return the largest contour area.

Show-A simple function to visualize OpenCV images on screen.

apply_new_background-This function applies a new background to the extracted foreground image.

We blur the image to smooth out the edges a bit and also reduces a bit of noise then convert the image to grayscale.

Apply thresholding to convert the image to binary format after this operation all the pixels below 200 value will be 0 and all the pixels above 200 will be 255.

→ Now we find the largest contour area in the image:

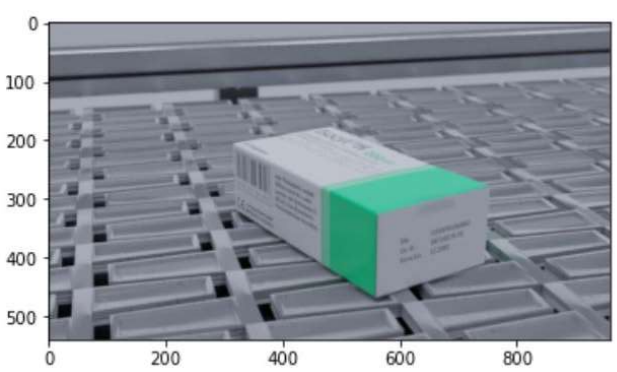
We create a black `mask` the same size as the original grayscale image, fill the mask with the shape of the largest contour all the pixels inside that area will be white.

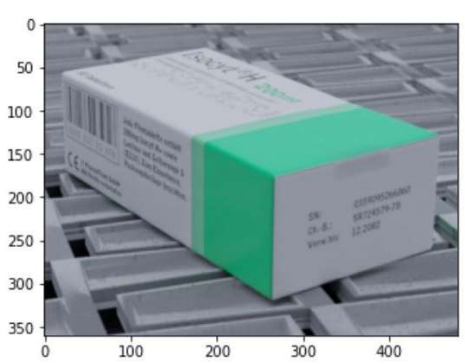
Create a mask for obvious and probable foreground pixels all the obvious foreground pixels will be white and all the probable foreground pixels will be black.

We then create `new_mask` from `mask` but with 3 dimensions instead of 2 then we apply Gaussian blurring to smoothen out the edges

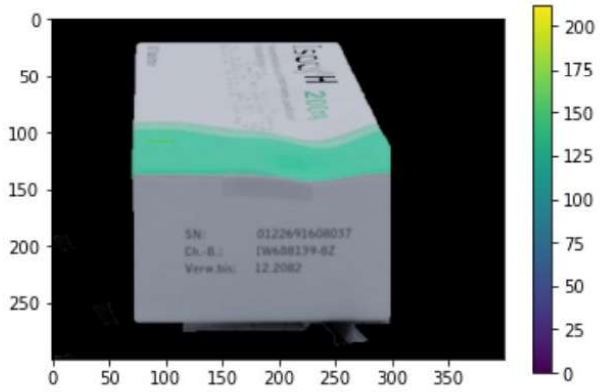
→ For the original,intact,side image in the dataset we are using the function and hyperparameters cv2.cvtColor(img, cv2.COLOR_BGR2RGB) to convert the image from BGR to RGB.

We then crop the image and focus on the package specifically

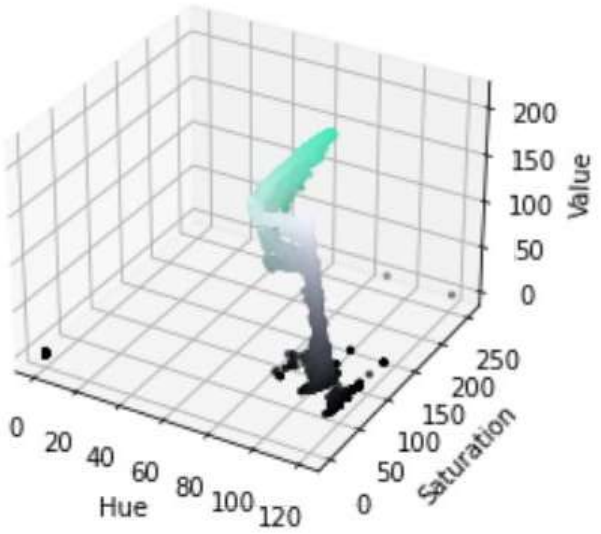




Then we use.GrabCut to accurately segment the foreground of an image from the background.
Foreground extract is any technique which allows an image's foreground to be extracted for further processing like object recognition, tracking etc.
In this algorithm, the region is drawn in accordance with the foreground, a rectangle is drawn over it. This is the rectangle that encases our main object.
Everything outside the ROI(region of interest) is considered as background and turned black. The elements inside the ROI is still unknown.
Then Gaussian Mixture Model(GMM) is used for modeling the foreground and the background and it creates labels for the unknown pixels and each pixel is clustered in terms of color statistics.
A graph is generated from this pixel distribution where the pixels are considered as nodes and two additional nodes are added that is the Source node and Sink node. All the foreground pixels are connected to the Source node and every Background pixel is connected to the Sink node.
The pixels that are connected to the Source node is labeled as foreground and those pixels which are connected to the Sink node is labeled as background.



HSV stands for Hue, Saturation, and Value (or brightness),
In HSV space, the package's green white and black are much more localized and visually separable. This is the key point that can be leveraged for segmentation.



→ Picking Out a Range

We threshold the package based on a simple range of green and black that can be chosen from a color picking app online such as this RGB to HSV tool.

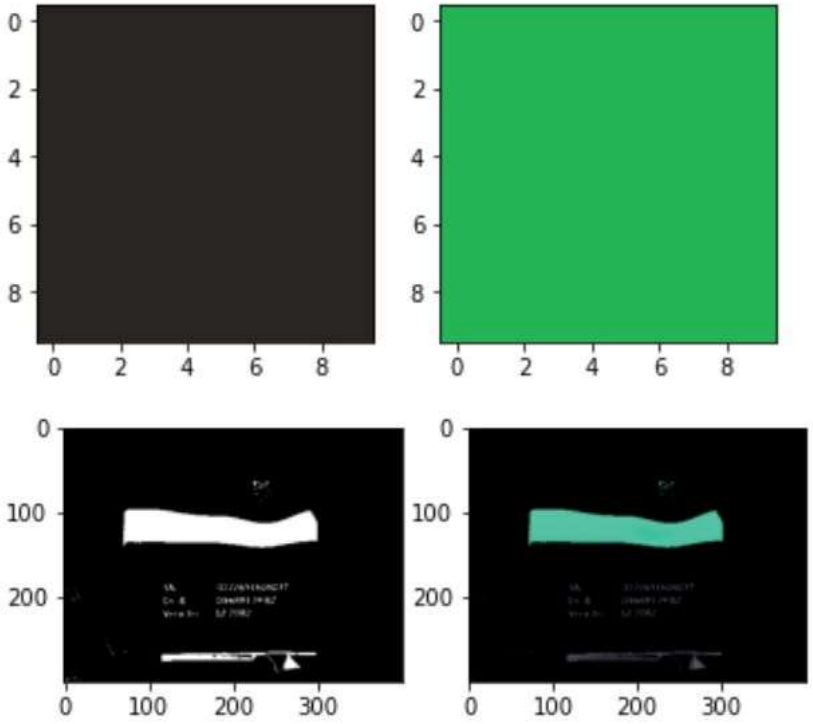
To impose the mask on top of the original image, we use cv2.bitwise_and(), which keeps every pixel in the given image if the corresponding value in the mask is 1

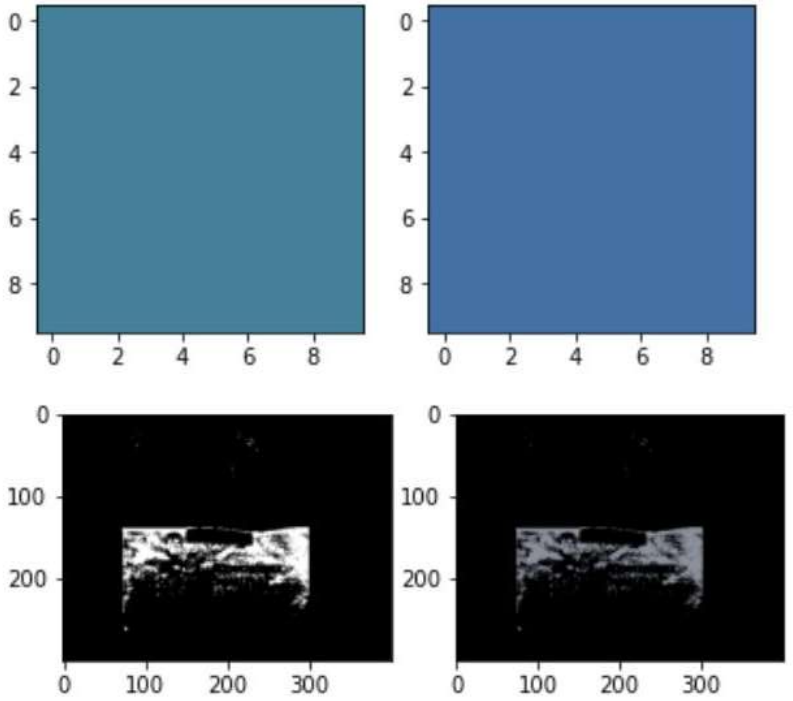
result = cv2.bitwise_and(img, img, mask=mask)

Now we do plt.imshow(mask, cmap="gray") and plt.imshow(result) to see what that did exactly

Now we view both the mask and the original image with the mask on top

→ Now we it similarly for the whitish grey color on the side of the package





→ Data Augmentation and PCA

In order to make the most of our few training examples, we will "augment" them via a number of random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better

datagen = ImageDataGenerator(rotation_range=40, width_shift_range=0.05, height_shift_range=0.05, shear_range=0.01, zoom_range=0.2, brightness_range=[0.2,1.0], horizontal_flip=True, fill_mode='nearest')

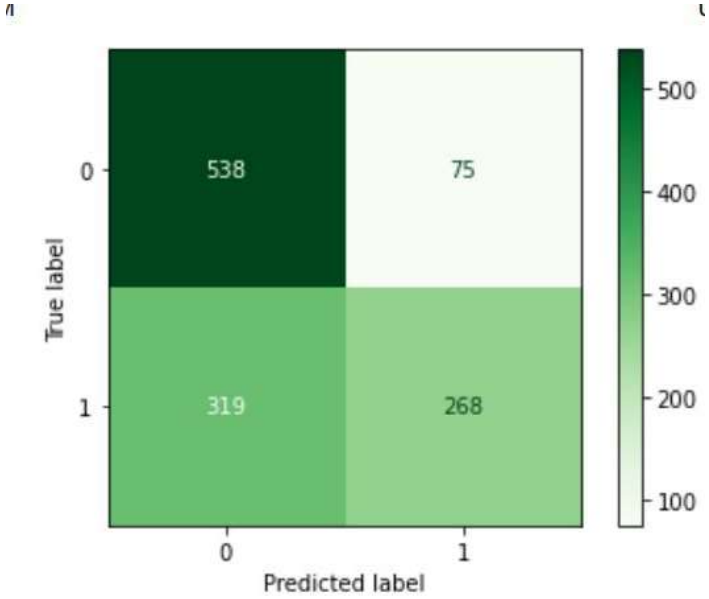
We then save this into a separate directory called aug_dir

Label encoding the target images in the aug data directory

Performing PCA -principal component analysis a dimensional reduction technique , here we are compressing the image data into smaller one which contains most of the information to build an efficient model.

Then we use the PCA transformed data for test train split - 80% train and 20% test.

KNN Model



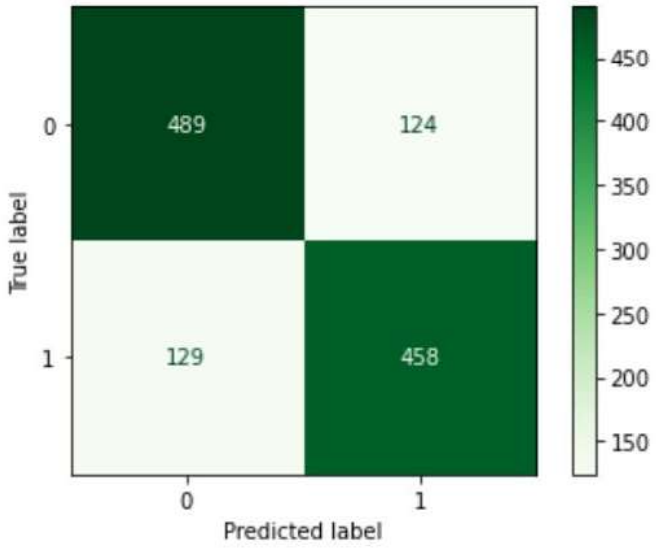
KNN model (k=2)
Accuracy of true class prediction is 67%

The **K-nearest neighbors** (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

SVM Model

SVM or Support Vector Machine is **a linear model for classification and regression problems**. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example.
For large values of C, the optimization will choose a smaller-margin hyperplane.
Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane.



SVM model (c =100)
Accuracy of true class prediction is 78.9%

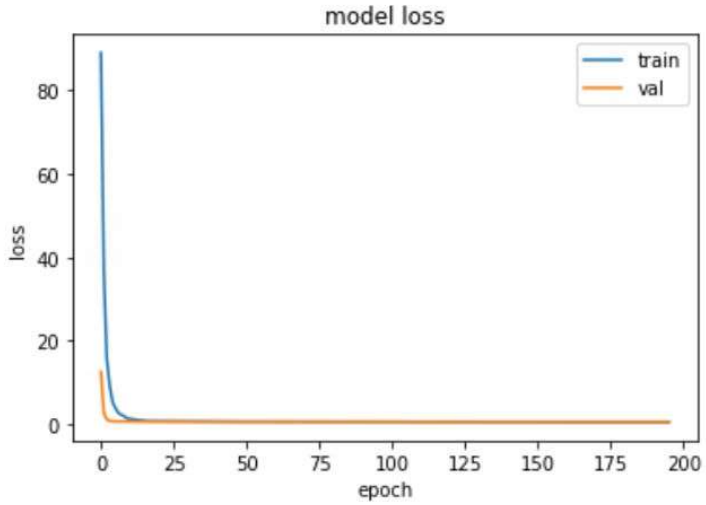
Neural Network Model:

We are using a Sequential model where each layer has exactly one input tensor and one output tensor

While building the model we are taking n_epochs=200.

validation set 20%

Results obtained are: **Accuracy on training data: 66.33% ,Accuracy on test data: 59.99%,Accuracy on whole data: 65%**



Conv Neural Network model

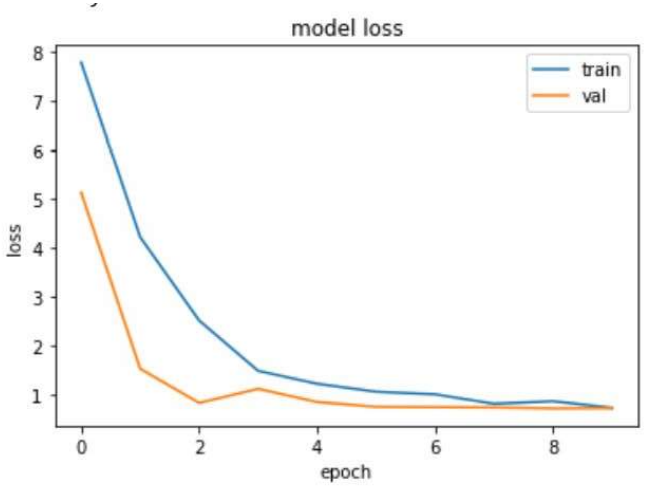
Systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map.

We are using max pooling which is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map with n_epochs = 10

Results obtained are : **Accuracy on training data: 51.87%,Accuracy on validation data: 43.75%,**

Accuracy on test data: 52.49% Accuracy on whole data: 51.87%



Result

Comparing the true prediction accuracies of all the models we can say that SVM model gives us the highest accuracy of 78.9% for correctly predicted images of damaged and intact.

Thank You