Capstone Project

Chip Happens: A Machine Learning Approach for Detecting Potato Chip Defects

Nick Hay

16.02.2024

Institute of Data

Table of Contents

Problem Statement

This project is investigating the opportunity to use image recognition and deep learning techniques to accurately classify potato chips as either defective or non-defective.

The current situation will depend on the manufacturer, so small, less sophisticated operations will carry out manual inspections using labour. Larger corporations that have the financial means may have already implemented automated systems to assist with quality control.

This problem has been researched previously in one such article ("Chapter 22 - Quality Evaluation and Control of Potato Chips") [0], however, this article is looking at how neural networks can be trained to classify chips according to colour and therefore predict acrylamide levels which is highly related to the colour of potato chips and is a known carcinogen in rats. This project is researching if image recognition can be used to classify potato chips after deep frying as defective if they have black, or dark brown or green colouration. This could then be applied the manufacturing operation where the detective chips are removed from the production line using compressed air (Watson, Michael. "Machine Learning & High Quality Potato Chips)[1]

Industry / Domain

The industry is Fast Moving Consumer Goods (FMGC) or food manufacturing, specifically potato chips manufactured by companies such as Smiths in Australia, Bluebird, Heartland, Proper and ETA in New Zealand or PepsiCo (Lays) which is where the images came from for this machine learning project.

The value-chain is growing potatoes, harvesting them, processing the potatoes into chips and a manufacturing facility, packaging the chips for distribution to supermarkets, corner shops (dairies in New Zealand) and finally consumption by the customer. One of the stages where machine learning using image recognition can be applied is during the manufacturing process. Once the potatoes are cleaned, peeled, sliced and deep fried it is here where image recognition can be used to check for any defects, such as deep browning (undesired flavours), broken chips or green colouration. The green part of a part comes from either Chlorophyll or Solanine [2].
Chlorophyll is the pigment responsible for giving plants their green colour and occurs when parts of the potato are exposed to sunlight before or during storage. Solanine on the other hand is a natural toxin found in potato plants and is produced as a defense against insects and animals. Solanine has a bitter taste and is typically found near the skin of the potato that is removed during chip processing.

Key concepts of the potato chip industry are type (fried vs baked), flavour innovation, distribution channels, sustainability (packaging), quality control (implementing checks) health & wellness (healthier options).

In New Zealand, 23% of the $1.1 billion industry can be attributed to chips [3].

If a successful machine learning model can be built to identify defects in potato chips on the production line then this model could be adapted and used elsewhere in other industries, such as identifying belly rot on cucumbers, Blight on potato leaves and classifying poisons mushrooms.

Stakeholders

The stakeholders for this project is company wide being the factory workers, the quality control team, the sales team, management and the owners of the company. The reason being is that by using machine learning to monitor and remove defective potato chips from the production line, is that it saves employing factory workers to grade the potato chips allowing them to be re-deployed elsewhere or not employing staff at all (saving in wages). It helps the quality team achieve their KPI's and they will receive fewer customer complaints. Having a higher quality product produced helps the sales team promote a point of difference and drive sales. Automated systems offer enhanced efficiency by enabling continuous operation and quicker decision-making compared to human inspectors. This has the potential to boost the throughput of the production line, thereby pleasing the owners. It is also the potato farmers who want to ensure they are growing high-quality potatoes that meet the manufacturer's standards. And finally the consumer who expects a high quality product every time they delve into a delicious bag of chips.

It can be challenging to put an exact figure on the expected return on investment (ROI) or cost savings for implementing an image recognition system for quality control in the potato chips industry without knowing the current operating costs however larger scale manufacturers may see a higher ROI due to the volume of throughput.

Business Question

The business question that needs to be answered is "Could image recognition assist with quality control, enhance efficiency, reduce waste, and ultimately increase profitability in the potato chip industry?"

The business value of answering this question is the potential for cost savings and efficiency by achieving the following:

- Lower labour costs: reduce the need for manual inspection

- Improved efficiency: automation can make decisions faster than humans

The implications of false positives (non-defective chips identified as defective) and false negatives (defective chips not identified) are as follows:

- False Positives: These would lead to non-defective chips being discarded, reducing the yield of the production process and potentially increasing costs. However, false positives would not impact the quality of the chips that reach the market.
- False Negatives: These would allow defective chips to pass through the quality control process and reach the market, which could harm the manufacturer's reputation and lead to lost sales.

Data Question

Can image recognition and deep learning be used to accurately classify potato chips as either defective or non-defective using image recognition? To answer this question, the following will be required.

- Images of both defective and non-defective potato chips.
- The quantity of images (the more the better).
- Labelled data, it would be helpful if images were labelled as defective or non-defective otherwise the images will need to be manually labelled.
- Validation data: Having a separate set of labelled images to validate the performance of the model. This can be by way of splitting the training data (80/20) or using augmented images.

Data

The images for this capstone project were sourced from Kaggle from a dataset called "PepsiCo Lab Potato Chips Quality Control"[4]. The size of the images ranged from 606kb to 1.52MB.

The images were loaded into Jupyter Notebook using ImageDataGenerator (Keras library in TensorFlow, used for deep learning) and flow_from_directory. ImageDataGenerator is used to generate batches of tensor image data with real-time data augmentation [5]. Experimentation with cv2 was also used as this reads images into an array, where an X train, y train split could be performed. Given there are 192 Test images (20%) and 769 Train images (80%) in the folders, ImageDataGenerator was chosen.

Data Science Process

I. Data Analysis

1a. Visualise Sample Images:

The dataset consists of two directories labelled as Test & Train. Within each of these folders there are two further folders called Defective and Not Defective.

- Test, Defective - 92 images
- Test, Not Defective - 100 images
- Train, Defective - 369 images
- Train, Non Defective - 400 images

Figure 1 shows a sample representation of the dataset. It appears to mean that the potato chips have been coloured in with black ink.

Figure 1 shows a sample of defective and Not Defective Images from the dataset

1b. Class Distribution:

Figure 2 shows the class distribution for the Training images and Figure 3 shows the distribution for the Test images.

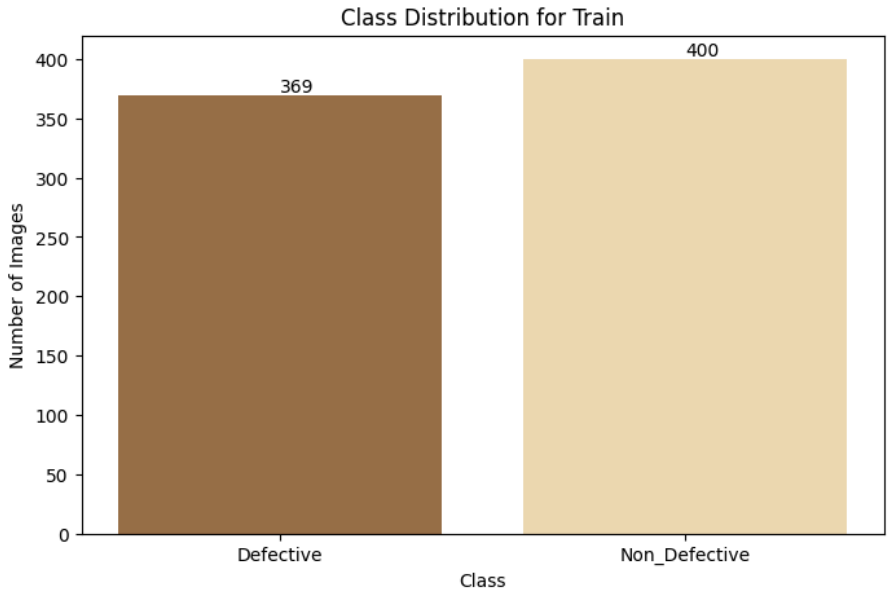


Figure 2 Class Distribution for the Training set

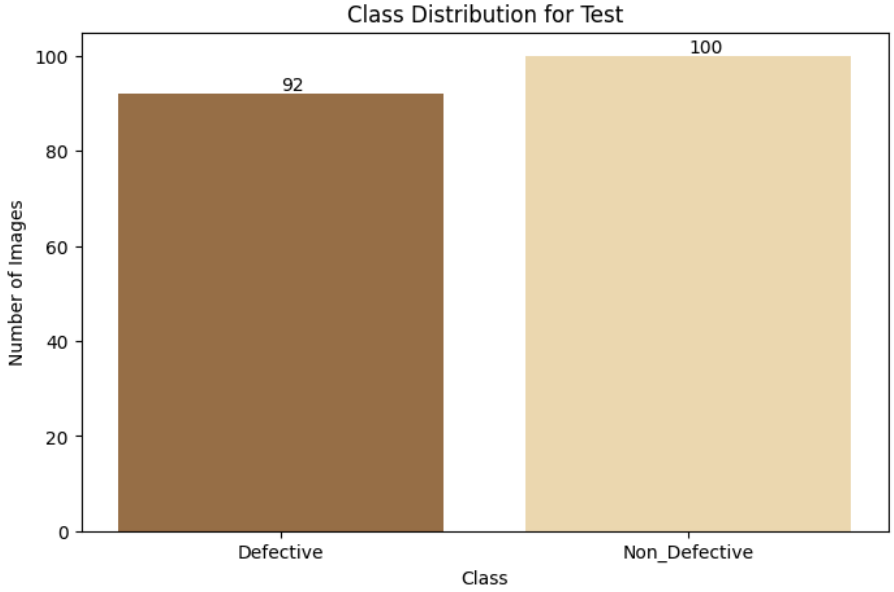


Figure 3 Class Distribution for the Test set

1c. Image Statistics:

Figure 4 shows the widths and heights of the Original Train images and the resized Train images. For the Exploratory data notebook, the images were resized to 150 x 150.

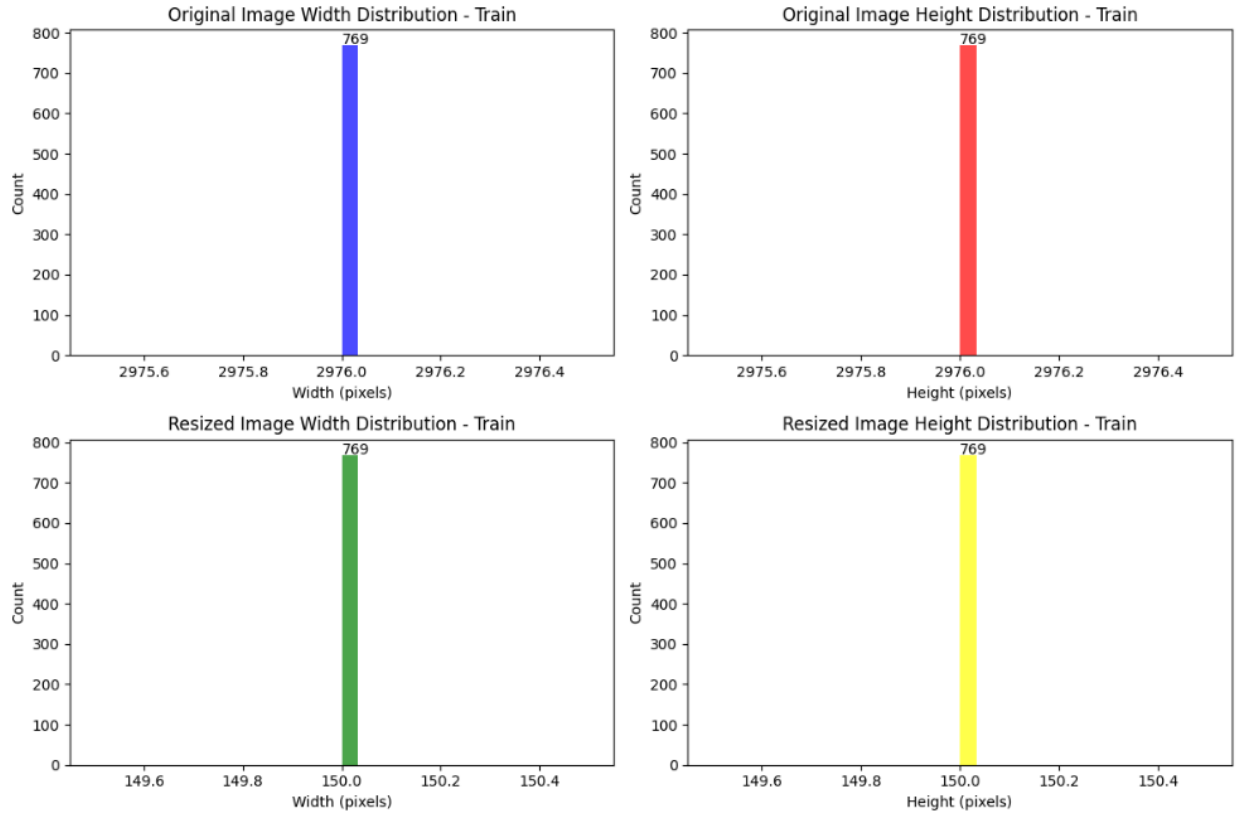


Figure 4 Width & Height Distribution of Train Images

1d. Augmentation:

Figure 5 & 6 shows how the images look once augmentation has been applied. Augmentation is where rotation, width & height shift, zoom, and flip can be applied. Augmentation can help when there is overfitting. Augmentation was only performed on the exploratory analysis.

```
# Example: Data augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

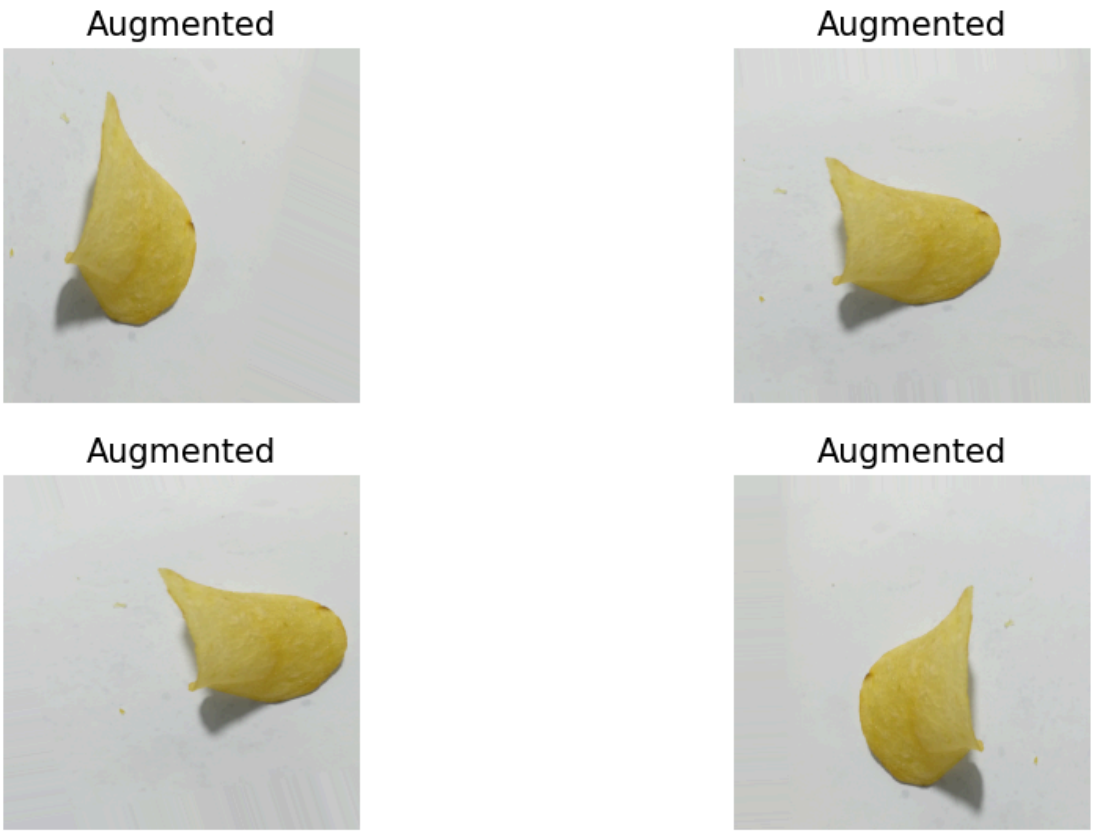


Figure 5 Example of Augmented images

```
Found 769 images belonging to 2 classes.
```

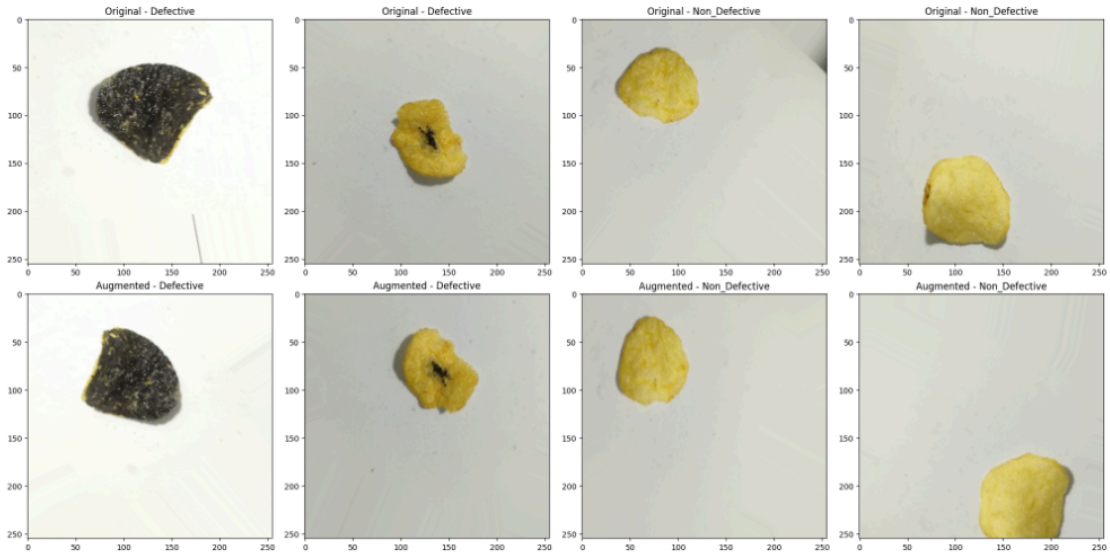


Figure 6 Example of Train Augmented images

1e. Red, Green, Blue (RGB) Distribution of the images:

Figure 7 shows the RGB distribution of the original, non-defective images.

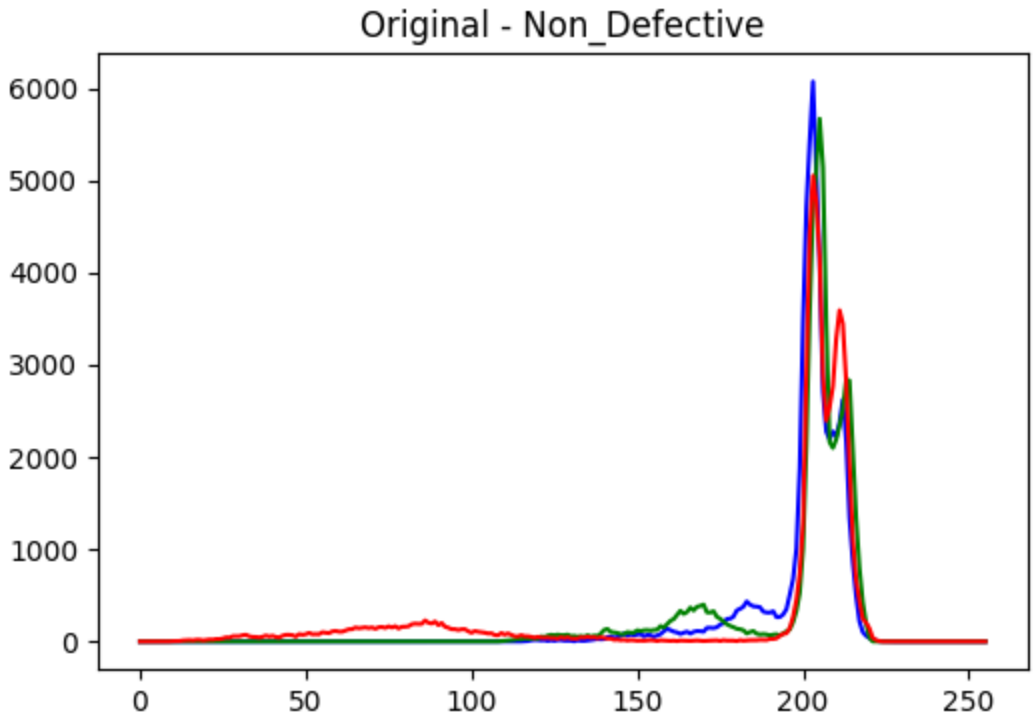


Figure 7 RGB Distribution of the Original, Non-Defective Images.

Figure 8 belows shows the Red, Green and Blue channels split out.

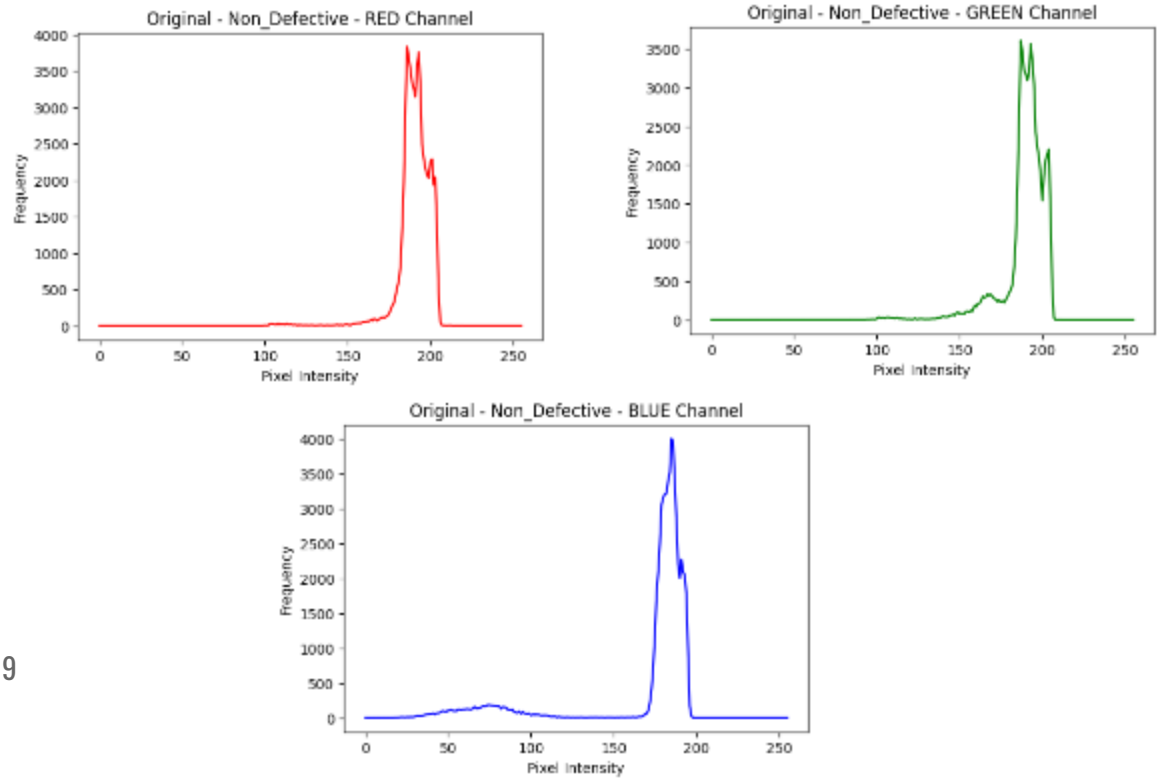


Figure 9 shows the RGB of the Augmented images:

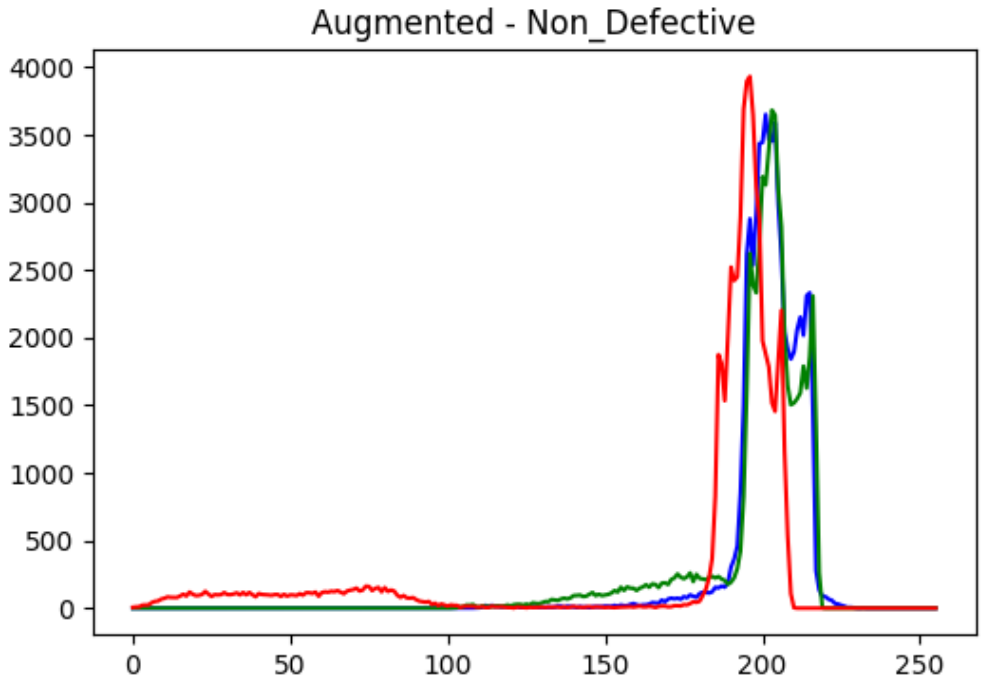


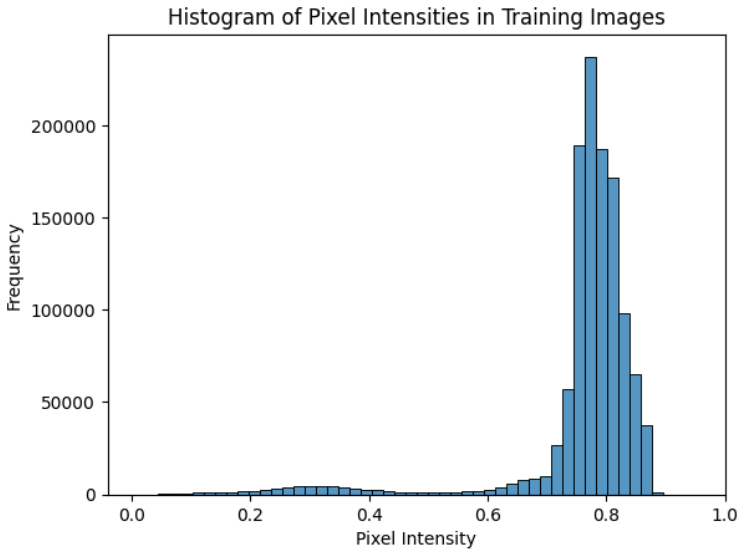
Figure 9 RGB Distribution of the Augmented, Non-Defective Images.

1f. Pixel Value Distribution:

Figure 10 is a Histogram of Pixel values which can be useful to understand the range and distribution of pixel values. The x-axis represents the pixel intensity, which ranges from 0 to 1. Pixel intensity refers to the value of the pixel in an image, where 0 represents black and 1 represents white. The y-axis represents the frequency of each intensity. Frequency here means the number of times a particular pixel intensity appears in the training images.

Around the 0.8 pixel intensity mark, there is a high number of pixels in the training images.

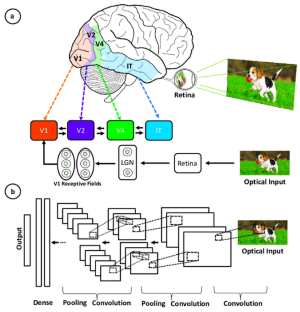
Fig 10 - Histogram of Pixel Intensities:



II. Modelling

2a. Convolutional Neural Network (CNN) Explained [6]

A Convolutional Neural Network is a specialised type of deep learning algorithm mainly designed for tasks that necessitate object recognition, including image classification, detection, and segmentation. Inspiration came from the layered architecture of the human visual cortex.



Key Components of a CNN, made up of four parts:

- Convolutional layers
- Activation function - Rectified Linear Unit (ReLU for short)
- Pooling layers
- Fully connected layers

Convolution layer:

Several filters of equal size are applied, and each filter is used to recognize a specific pattern from the image, such as the curving of the digits, the edges, the whole shape of the digits, and more.

Activation function:

A ReLU activation function is applied after each convolution operation. This function helps the network learn non-linear relationships between the features in the image.

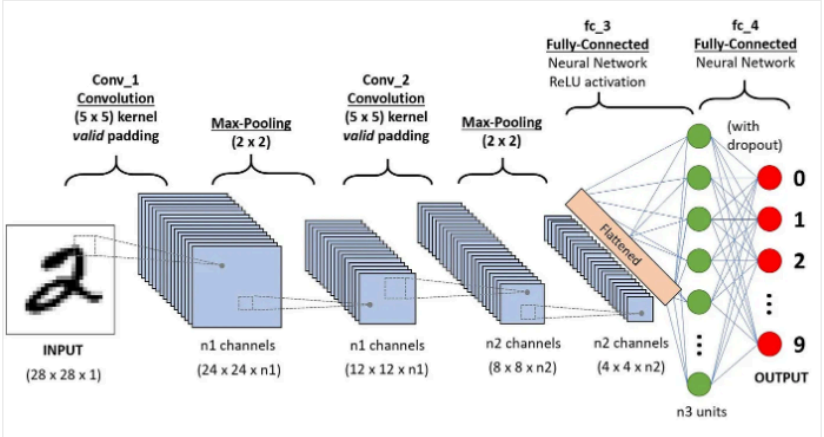
Pooling layer

The goal of the pooling layer is to pull the most significant features from the convoluted matrix. The most common aggregation functions that can be applied are:

- Max pooling, which is the maximum value of the feature map
- Sum pooling corresponds to the sum of all the values of the feature map
- Average pooling is the average of all the values.

Fully connected layers:

Finally, a softmax prediction layer is used to generate probability values for each of the possible output labels, and the final label predicted is the one with the highest probability score.



2b. Building a Convolutional Neural Network (CNN):

Building a Convolutional Neural Network (CNN) involves several steps, including choosing the architecture, setting hyperparameters, selecting optimization algorithms, and deciding on loss functions. Here are the steps: [7].

Model Architecture:

Sequential Model: A Linear stack of layers, where each layer has exactly one input tensor and one output tensor.

Hyperparameters:

Image size = (150 x 150),(224 x 224),(255 x 255)

Batch size = 9, 25, 32

Epochs = 10, 20, 24, 30, 100

Colour Representation = RGB and Greyscale

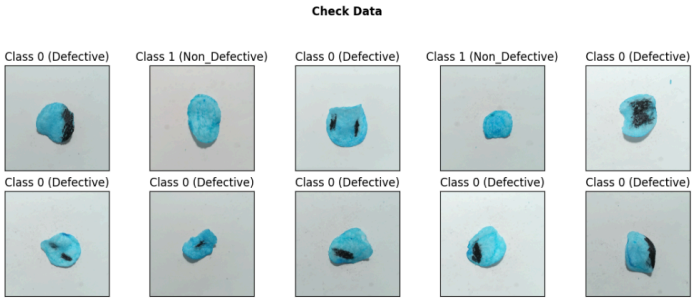
Validation Split = 20% of Training data

Number of convolutional layers

Pooling size

Learning rate = The rate at which the model updates its weights during training

Dropout rate = The fraction of input units to drop during training.



Optimization Algorithm:

Adam: An adaptive learning rate optimization algorithm that combines ideas from RMSProp and Momentum.

SGD (Stochastic Gradient Descent): A basic optimization algorithm that updates weights in the direction of the negative gradient of the loss function.

RMSProp: Another adaptive learning rate optimization algorithm that divides the learning rate by an exponentially decaying average of squared gradients.

Architecture:

VGG19: consists of 19 layers (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer)

ResNet-50: This is a variant of the ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer.

Loss Function:

Binary Crossentropy: Used for binary classification tasks where each example belongs to one of two classes.

Categorical Crossentropy: Used for multiclass classification tasks where each example belongs to one of multiple classes.

Callbacks:

Early Stopping = Patience (3) on Validation Loss.

Compile the Model

After specifying the architecture, hyperparameters, optimizer, and loss function, compile the model using the compile() function in Keras or TensorFlow.

Train the model:

Train the model using the fit() function, passing in the training data, validation data, batch size, and number of epochs.
Validation data should be separate from both the training and test data. In this project, a 20% split of training data was used.

Evaluate the model:

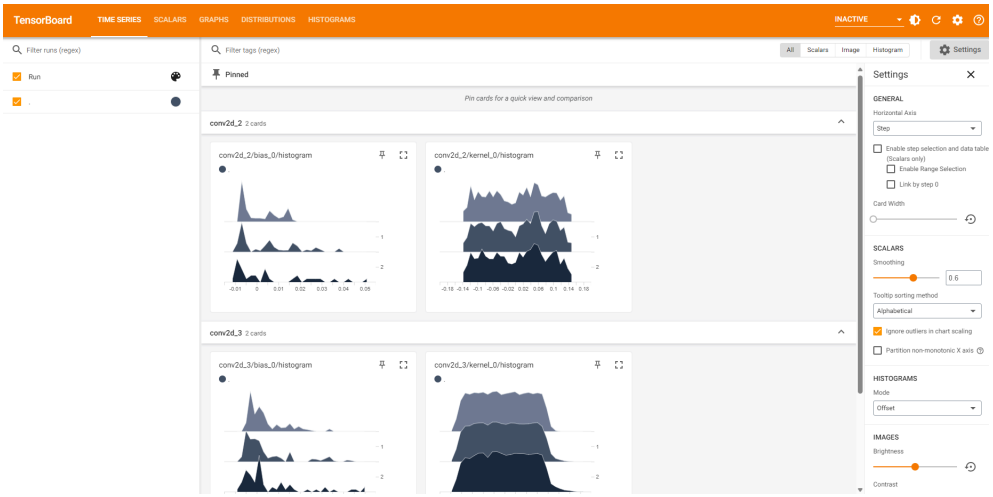
Evaluate the trained model on the test set to assess its performance on unseen data.

Make predictions:

Make predictions using the test set or ideally new test data. ResNet50, VGG19

TensorBoard Integration.

TensorBoard is a visualization tool provided by TensorFlow for monitoring and analyzing the training process of deep learning models. It allows you to track metrics such as loss and accuracy, visualize the model graph, and inspect histograms of weights and biases, among other features.



2c. The following tables show a summary of the models that were evaluated:

Model	First attempt - Simple Model 2
Size	255 x 255
Random Seed	40
Batch Size	32
Hyperparameters	Conv2D(64), MaxPooling, Conv2D(32), Maxpool, Flatten, Dense(32)
Activation	ReLu, Sigmoid
Optimiser / Loss	Adam / binary_crossentropy
Epochs / Run Time	Epoch = 10 / 13 mins

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 253, 253, 64)      1792

 max_pooling2d (MaxPooling2  (None, 126, 126, 64)      0
 D)

 conv2d_1 (Conv2D)           (None, 124, 124, 32)      18464

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)

 flatten (Flatten)           (None, 123008)            0

 dense (Dense)               (None, 32)                3936288

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 3956577 (15.09 MB)
Trainable params: 3956577 (15.09 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Model	VGG19 - Model 1
Size	224x 224
Random Seed	40
Batch Size	32
Base model	VGG19 (weights='imagenet', include_top=False)
Hyperparameters	Flatten, Dense(512)
Activation	ReLu, Sigmoid
Optimiser / Loss	Rmsprop / binary_crossentropy
Epochs / Run Time	Epoch = 24 / 53 mins

```
print(model.summary())
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 flatten_5 (Flatten)         (None, 25088)             0

 dense_20 (Dense)            (None, 512)               12845568

 dense_21 (Dense)            (None, 1)                 513

=================================================================
Total params: 32870465 (125.39 MB)
Trainable params: 32870465 (125.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Model	ResNet50 - Model 1
Size	255 x 255
Random Seed	40
Batch Size	32
Base model	ResNet50 (weights='imagenet', include_top=False)
Hyperparameters	GlobalAveragePooling2D, Dropout(0.5)
Activation	Sigmoid
Optimiser / Loss	Rmsprop / binary_crossentropy
Epochs / Run Time	Epoch = 20 / 13 mins

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_1  (None, 2048)              0
  (GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense_1 (Dense)             (None, 1)                 2049

=================================================================
Total params: 23589761 (89.99 MB)
Trainable params: 23536641 (89.79 MB)
Non-trainable params: 53120 (207.50 KB)
_________________________________________________________________
None
```

Model	Augmentation using ImageDataGenerator - Model 2
Size	224 x 224
Random Seed	40
Batch Size	32
Hyperparameters	Conv2D(64), MaxPooling, Conv2D(32), Maxpool, Flatten, Dense(32)
Activation	ReLu, Sigmoid
Optimiser / Loss	Adam / binary_crossentropy
Epochs / Run Time	Epoch = 20 / 53 mins

```
# Create a data generator for the training set
aug_data = ImageDataGenerator(
    rescale=1./255,         # normalize pixel values
    rotation_range=30,      # randomly rotate images in the range (degrees, 0 to 180)
    width_shift_range=0.2, # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.2,# randomly shift images vertically (fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=True)    # randomly flip images
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 222, 222, 64)      1792

 max_pooling2d_2 (MaxPoolin  (None, 111, 111, 64)      0
 g2D)

 conv2d_3 (Conv2D)           (None, 109, 109, 32)      18464

 max_pooling2d_3 (MaxPoolin  (None, 54, 54, 32)        0
 g2D)

 flatten_1 (Flatten)         (None, 93312)             0

 dense_2 (Dense)             (None, 32)                2986016

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 3006305 (11.47 MB)
Trainable params: 3006305 (11.47 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

The models took anywhere from 13 minutes to almost 4 hours to complete for some of the more complex but less successful attempts.

Gridserach turning was attempted in Google Colab but resulted in a data leakage warning.

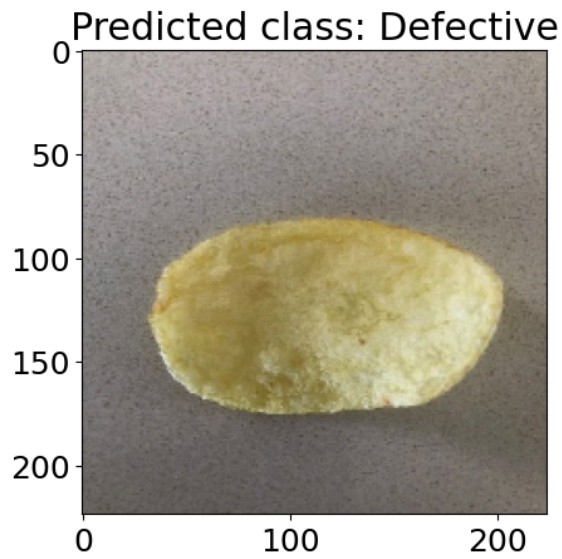
III. Outcomes

3a. Below is a summary of the results:

Model	Training Accuracy	Validation Accuracy	Evaluation Accuracy
First attempt - Simple Model 2	100%	99%	98.44%
VGG19 - Model 1	100%	99%	98.44%
ResNet50 - Model 1	100%	52%	52.08%
Augmentation - Model 2	100%	100%	98.96%

Predictions on unseen test images - VGG19 - Model 1 Incorrect

```
1/1 [==============================] - 0s 214ms/step
Probability of positive class: 0.0
Predicted class: Defective
```

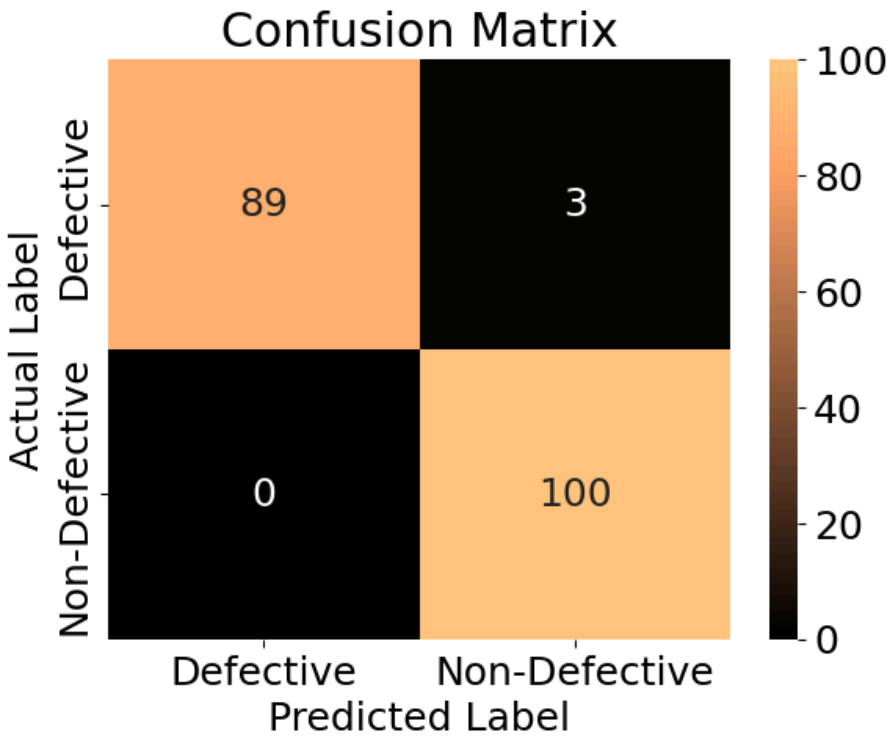


VGG19 - Model 1

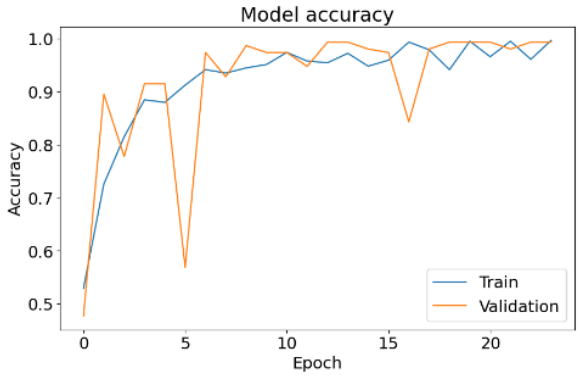
Classification Report:

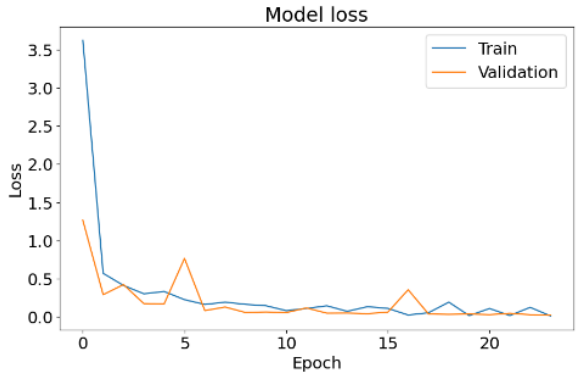
```
Classification Report:
              precision    recall  f1-score   support
0                1.0000    0.9674    0.9834   92.0000
1                0.9709    1.0000    0.9852  100.0000
accuracy         0.9844    0.9844    0.9844    0.9844
macro avg        0.9854    0.9837    0.9843  192.0000
weighted avg     0.9848    0.9844    0.9844  192.0000
```

Confusion Matrix:



Visualisation of the cost (fig 11)





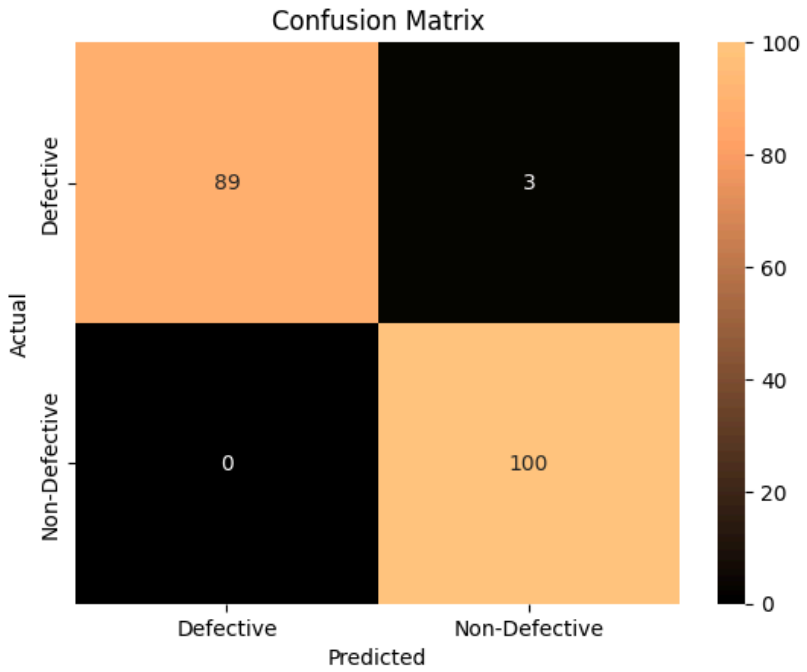
First Attempt - Simple Model 2

Classification Report:

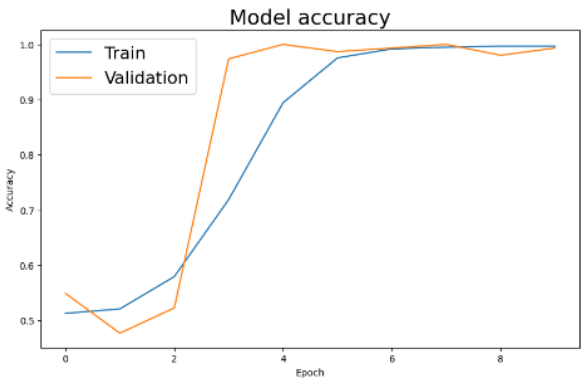
	precision	recall	f1-score	support
0	1.00	0.97	0.98	92
1	0.97	1.00	0.99	100
accuracy			0.98	192
macro avg	0.99	0.98	0.98	192
weighted avg	0.98	0.98	0.98	192

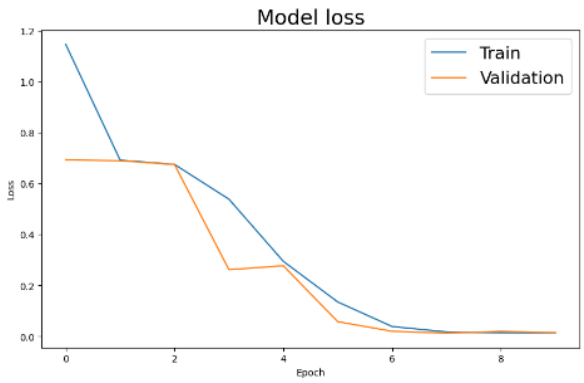
```
Precision: 0.970873786407767
F-score: 0.9852216748768473
```

Confusion Matrix = same result as VGG19



Visualisation of the cost (fig 12):





Make predictions - First Attempt - Simple Model 2

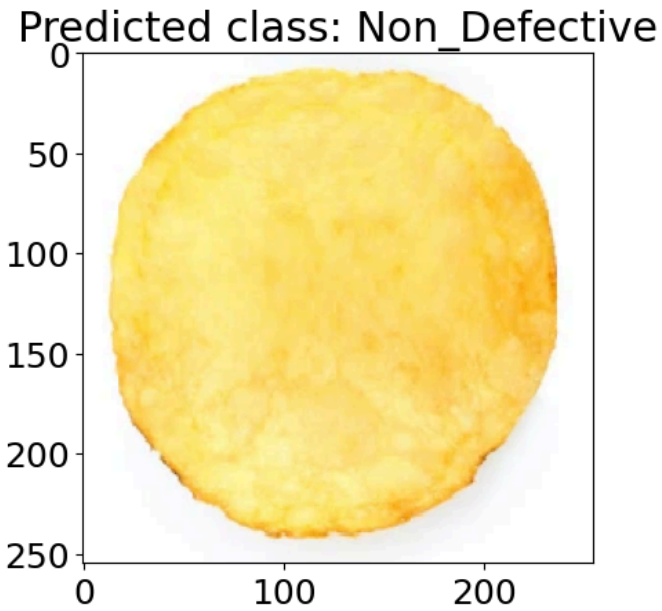
Fig 13 - predictions

```
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
Predicted: Defective, Actual: Defective
```

Predictions on unseen test images - First Attempt - Model 2 Correct

Fig 14 - predictions

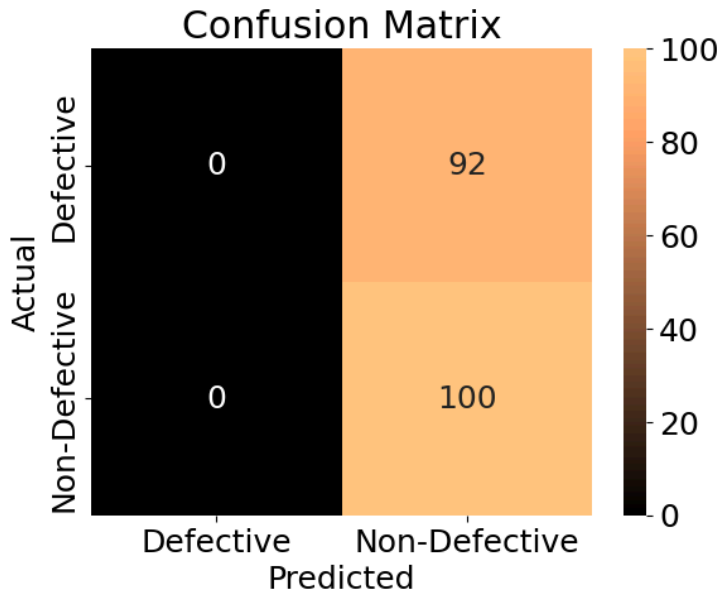
```
1/1 [==============================] - 0s 39ms/step
Probability of positive class: 1.0
Predicted class: Non_Defective
```



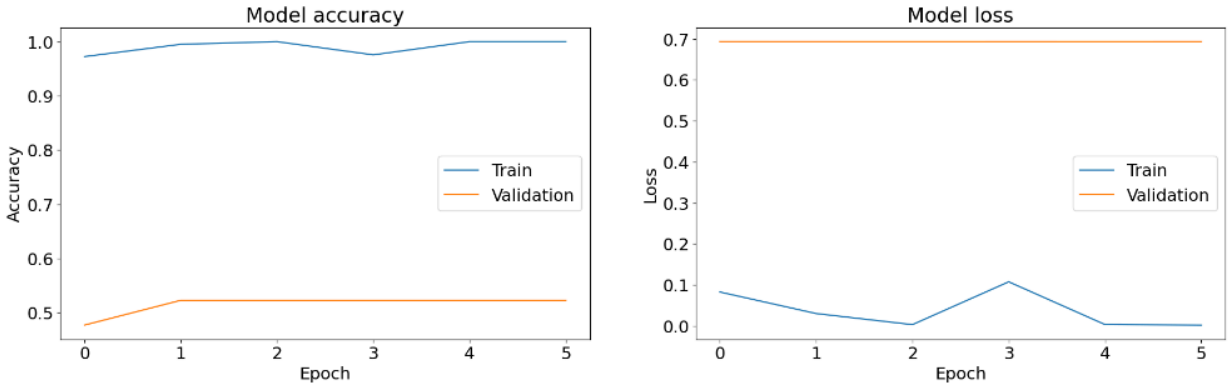
The reason why ResNet50 and Augment Models were not chosen was because of their confusion matrix and they predict non-defective chips defective on unseen test data. Here are the results of their Confusion Matrix and Loss Visualisation and Predictions.

ResNet50 - Model 1

Confusion Matrix



Visualisation of the cost (fig 15):

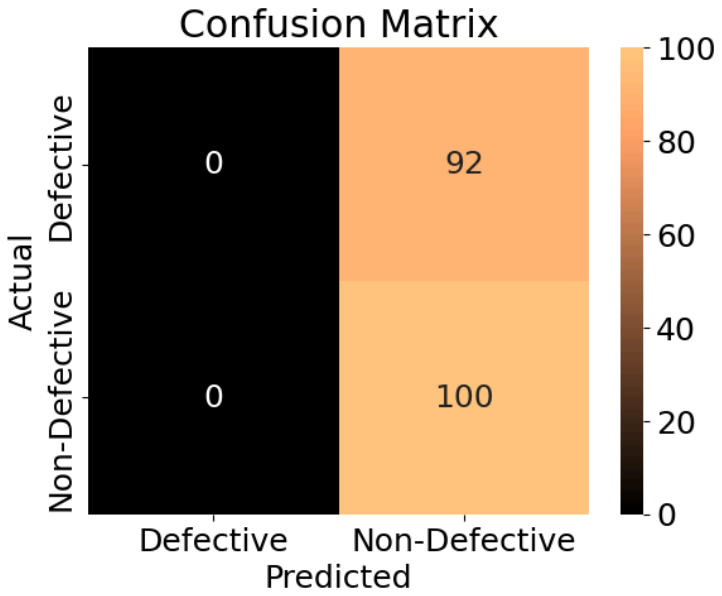


Predictions (fig16):

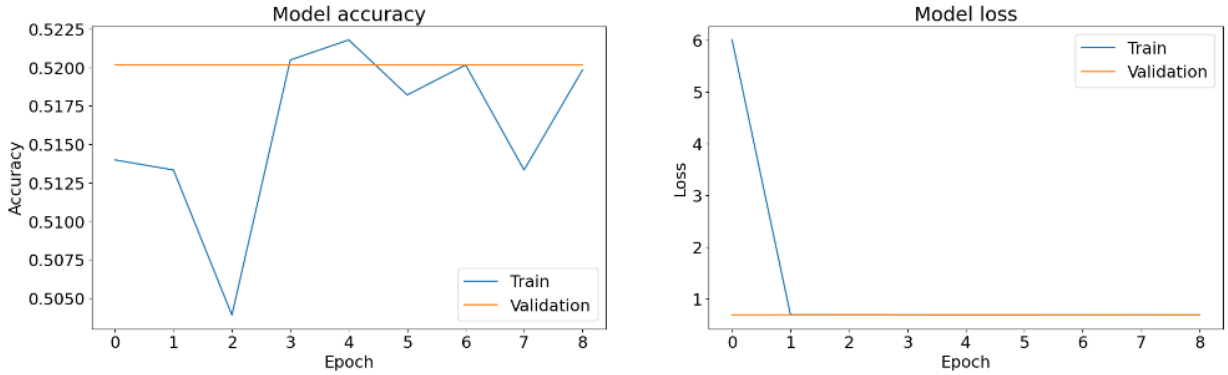
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective

Augmented Model using ImageDataGenerator - Model 2

Confusion Matrix



Visualisation of the cost (fig 17):



Predictions (fig 18):

```
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
Predicted: **Non_Defective** (incorrect), Actual: Defective
```

The model that was selected for the Streamlit application was VGG19 & First Model

Reson for the poor performance on unseen data (New Test Images)

The reason some of the models performed poorly when predicting defective or non-defective on new test data could be because the models are trained for too many epochs or not enough or the models are too complex or there simply isn't enough images for the model to learn. Figure 19 below shows strategies that can be used to mitigate overfitting.

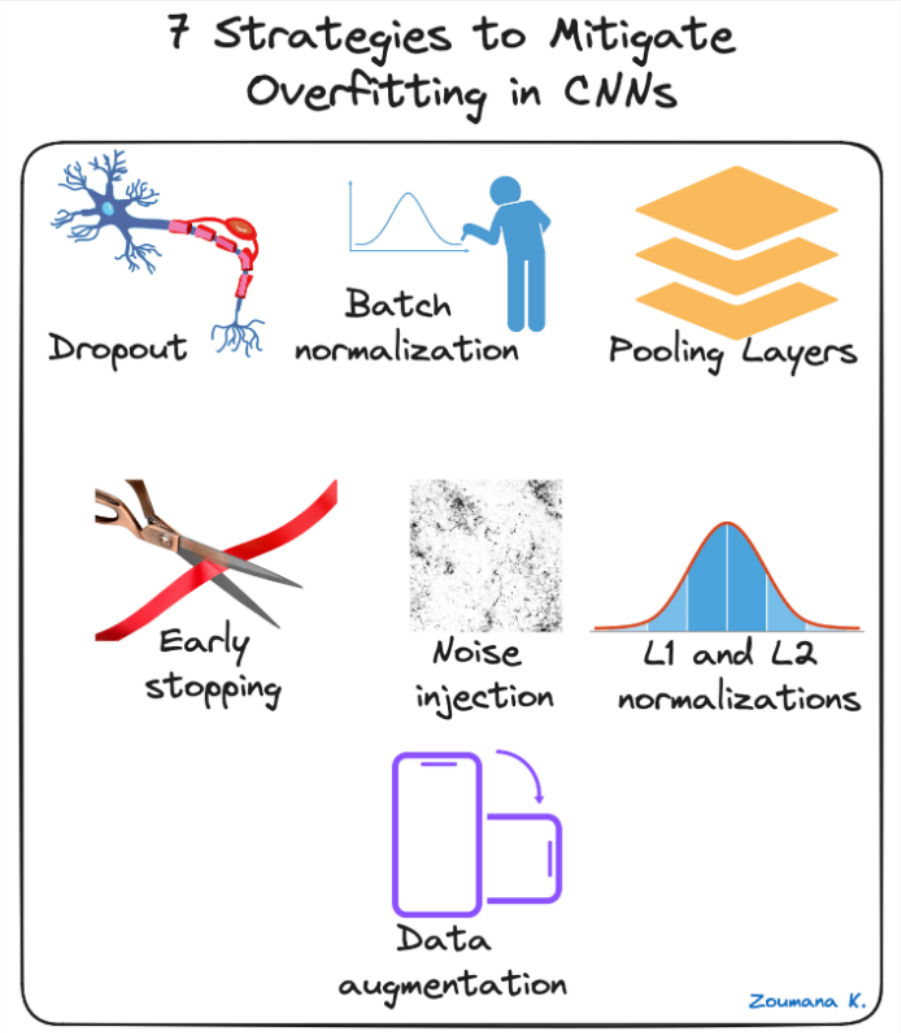
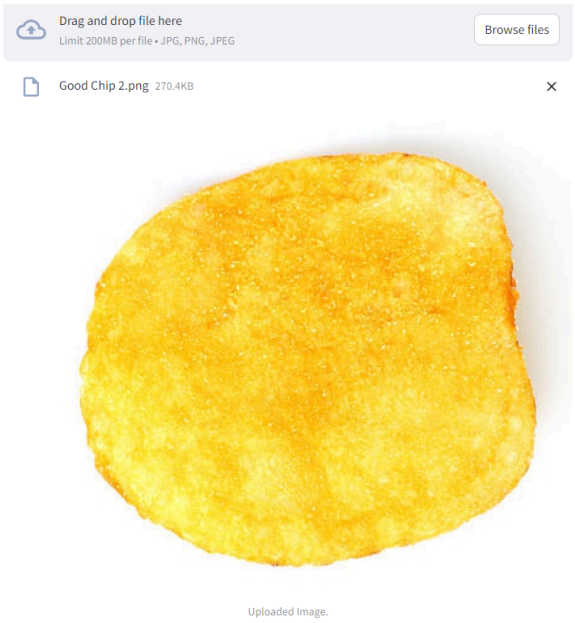


Fig 19 Strategies to Mitigate Overfitting in CNNs [6]

IV. Implementation

The next step would be to run factory trials using real-time images. The scale up process would involve optimising the model to account for speed (blurry images), adding new hardware (compressed air to blow defective chips off the production line), and testing the software. The application showcased using Streamlit was a basic example to show how the quality team might verify the accuracy of the model. This could be further improved by inputting live images of the potato chips moving along a production line at speed to create a feedback loop to monitor the performance of the model.



Classifying Image..... Predicted class: Not Defective

Data Answer

The modelling showed that machine learning can be used to predict whether a potato chip is defective or not defective. The best model predicted 89 out of 92 test images as defective and correctly predicted 100 out of 100 non-defective images. When the model was tested against unseen images to make predictions, it often classified non-defective chips as defective. Training the models with a lot more images that have different features may help to improve the predictions as well as looking into the complexity of the model's architecture.

Business Answer

Image recognition can assist with quality control, enhance efficiency, reduce waste, and ultimately increase profitability in the potato chip manufacturing industry and other industries.

Response to Stakeholders

It would be our recommendation to proceed to the next step by investigating the feasibility of scaling the image recognition model up using actual defective images and proceeding with production trials. This model would be an additional tool for ensuring quality, reducing the manual workload, and increasing the efficiency of the production line.

End-to-End Solution

The end-to end solution for implementing an image recognition model on a larger scale would require gathering many many examples of actual defective images that hadn't been manipulated with black shading, preparing these with pre-processing, training the model and evaluating the performance of the model then further optimizing the model and make adjustments before deploying into the production environment. The model would require continuous monitoring and updating as required.

References

[0] Chapter 22 - Quality Evaluation and Control of Potato Chips." *Computer Vision Technology for Food Quality Evaluation*, edited by Da-Wen Sun, Second Edition ed., Elsevier Science, 2016, pp. 591-613. *Quality Evaluation and Control of Potato Chips*, https://www.sciencedirect.com/science/article/abs/pii/B9780128022320000220?via%3Dihub

[1] Watson, Michael. "Machine Learning & High Quality Potato Chips | by Opex Analytics | The Opex Analytics Blog." *Medium*, 16 December 2015, https://medium.com/opex-analytics/machine-learning-high-quality-potato-chips-45d25d671be2

[2] Richter, Amy, and Kathy W. Warwick. "Green Potatoes: Harmless or Poisonous?" *Healthline*, 29 June 2023, https://www.healthline.com/nutrition/green-potatoes#cause

[3] "Crisp." *Potatoes New Zealand*, 11 August 2022, https://potatoesnz.co.nz/production/processing-sector/crisp/

[4] Navid, Usama. "PepsiCo Lab Potato Chips Quality Control." *Kaggle*, https://www.kaggle.com/datasets/concaption/pepsico-lab-potato-quality-control

[5] "Comparing Data loaded using ImageDataGenerator() and cv2.imread()." Stack Overflow, 18 April 2020, https://stackoverflow.com/questions/61287418/comparing-data-loaded-using-imagedatagenerator-and-cv2-imread

[6] Keita, Zoumana. "An Introduction to Convolutional Neural Networks: A Comprehensive Guide to CNNs in Deep Learning." *DataCamp*, https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns.

[7] Chat GPT: https://chat.openai.com/