Gluten Detection with Portable Visible Light Near-Infrared Reflectance Spectrometer for Celiac Patients

Mert Alperen Beşer, Egecan Yurtcan, Burak Demirtaş
Istanbul Technical University
beser18@itu.edu.tr, yurtcan16@itu.edu.tr, demirtas17@itu.edu.tr

***Abstract*—Celiac disease is one of the long-term auto-immune diseases in which the patients suffer from small intestine damage because of the gluten sensitivity. This small intestine damage can harm patients in different ways such as diarrhea, fatigue, weight loss, bloating and gas, abdominal pain, Nausea and vomiting, and constipation. However, these effects can be reduced with medical support. The hardest part of living with celiac disease is that patients should be aware of consuming gluten-free nutrition. Even if patients use gluten-free products in their homes, they do not have to consume foods at home all the time. Some restaurants have gluten-free menus, however; there is considerable gluten contamination in both industry and food services. Those situations create uncomfortable and life-threatening environment for people who suffering from celiac disease. The suffering of the celiac patients could be over, if the gluten can be detected easily with a smart system. This study proposes a portable IoT smart device that can easily detect gluten in food without chemical use.**

I. INTRODUCTION

In America number of affected people due to celiac disease can reach 3.2 million according to the celiac disease expert Jefferson Adams[1]. Celiac disease was discovered too late and there is no known cure for this disease. In addition, it is very difficult to detect accurately since it is a protein. There are not many devices made for this disease and with the development of IoT technology, new approaches have been brought to this disease. There are a few different types of methods to solve this problem. Some of those methods aren't useful because of their non-portable structure and this can be a big issue when people need to detect gluten in their food when they go out somewhere. This project aims to make a portable smart gluten detection device based on a reflectance spectrometer approach for daily life use. This device will not only be capable of distinguishing gluten-free foods and it will also keep the location and brand information of the food for further investigations.

II. PREVIOUS WORKS

There are several approaches to detecting gluten in the absence of food. PCR (polymerase chain reaction) [2], column chromatography, and matrix-assisted laser desorption/ionization time-of-flight mass spectrometry [3] are some well-known approaches for gluten detection. Therefore, even if the detection sensitivity of gluten is very high in these approaches, they are relatively non-commercial costly processes that are not suitable for daily use. There is a commercial product gluten detection called "Nima" (see Figure 1) which uses antibody-based chemistry [4]. The main deficit of the "Nima" is that patients have to buy a capsule in which one inserts a food for detection, on every test.

Fig. 1. This is a sample of a figure caption.

Buying a capsule for each test is not a sustainable and pro-budget solution. On the other hand, there is the near-infrared (NIR) spectroscopy approach chosen for this work which is simpler and less expensive compared to others [5]. The challenge with this approach that it demands a large training set consisting of gluten and gluten-free foods [6].

III. GENERAL SYSTEM

The general system including:

- Light emitting diode (LED),
- Raspberry-Pi 4 Model B,
- DVD cut-away performs as diffraction grating
- Raspberry NoIR (no IR-filter) camera
- The case of the device.

General system scheme can be seen on Figure 2. First, the user gives the command to the mobile device, after that Raspberrypi takes that command and capture the spectrum picture of reflected food and this image data received by the mobile device. Then image processing algorithm detects whether there is gluten on the food. Designing was separated into two parts as optical design, and image processing part.

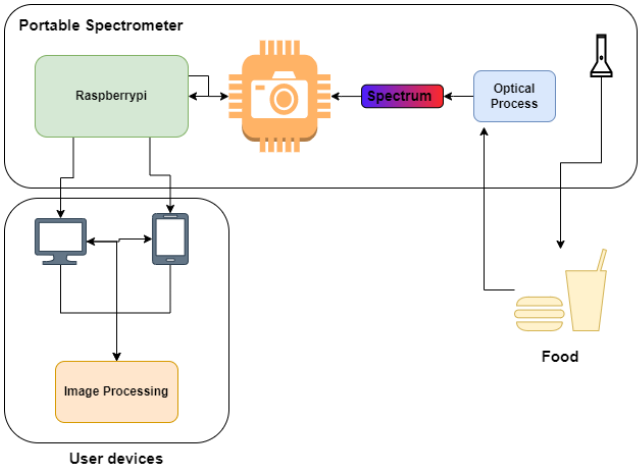


Fig. 2. General system scheme

A. Optical System and Mechanical Design

According to the ISO 20473:2007 Optics and photonics — Spectral bands, Near Infrared range is between 0.78–3 µm wavelengths. The chosen NoIR camera use CMOS sensor [7]. Typical CMOS sensors have capability to capture wavelengths between 400-1000nm [8]. The chosen camera can capture the both visible spectrum and a large part of NIR region.

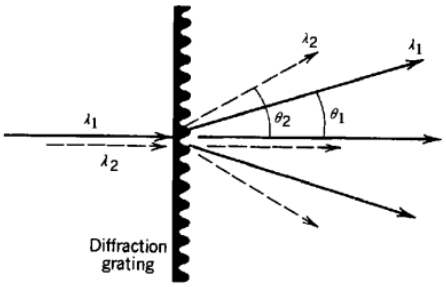


Fig. 3. Structure of diffraction grating [9].

The optical system also use diffraction grating to capture the spectrum of the coming light. DVD's or CD's can act like a diffraction grating because of their structure (see Figure 3-4). For low budget applications using DVD's are reasonable.

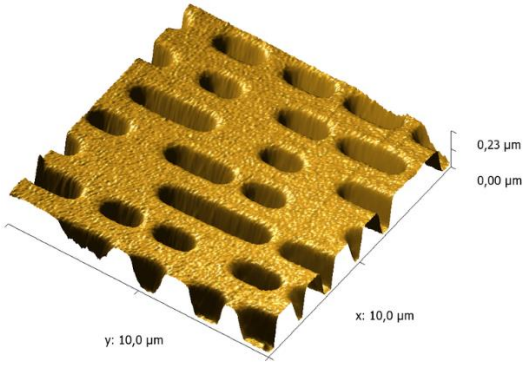


Fig. 4. Structure of the DVD [10]

The whole optical process begins with the LED. The light of LED hits the food at 45° and is reflected on the slit. The purpose of the slit is that avoid too much shining on the camera, in other words, keep the camera from saturated pixel values.

Fig. 4. Front side of spectrometer design

After the slit light comes to the diffraction grating which has 30° angle compare to the base. Then the light diffracted to the camera and camera captures the spectrum.

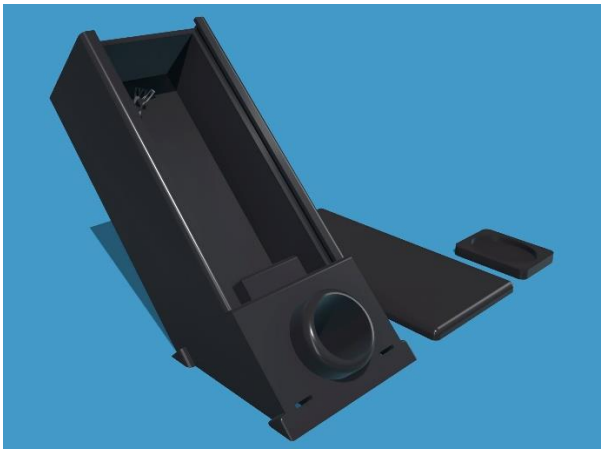


Fig. 5. The design consisting of 3 individual parts. Main body, sampling pot and cover

The camera is placed 30° angle as same the diffraction grating. To make this process possible a 3D print system was designed (see Figures 4-5). In addition, a slit can be seen from the overhead view of the design in Figure 6. The 3D design is printed in black color to avoid over shining.

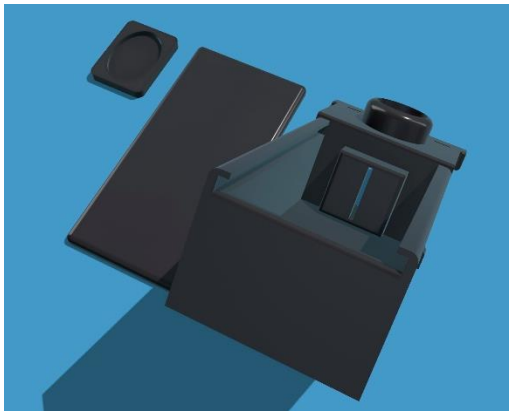


Fig. 6. Overhead view of the design

B. *Image Processing*

The task is detecting gluten absence in the food and investigating whether there are characteristic spectral responses of gluten absence with the designed optical system that was operating in the 400-1000nm wavelength range. To do this, a small dataset was gathered consisting of gluten-free and non-gluten-free classes. Details of the gathered dataset can be seen on Table 1.

TABLE I
DATASET INFORMATION

Product	Number of Sample	Gluten-free
Flour	30	✓
Bread	30	✓
Snack	30	✓
Cookie	30	✓
Mixed Bread-Flour-Snack-Cookie	30	✓
Bread	30	✗
Flour	30	✗
Bread	30	✗
Snack	30	✗
Mixed Flour	30	✗

There are 300 spectra in the dataset. The sample spectrum of the gluten-free flour can be seen in Figure 7. After gathering the dataset, all images reshape as 370-400 array dimensions. Then mean-squared of the two classes got and visualized as RGB (see Figure 8).

The correlation between 2 classes was measured using bivariate correlation method. The bivariate correlation, commonly known as the Pearson product-moment correlation coefficient (PPMCC), or simply the correlation coefficient, is another name for the Pearson correlation coefficient. The bivariate Pearson Correlation gives a group correlation coefficient, abbreviated as r, that assesses the strength and direction of linear correlations among groups of continuous variables. In addition, the population correlation coefficient, abbreviated "rho," or "Pearson correlation," measures the statistical support for a linear relationship between the same pairs of variables in the population [11].

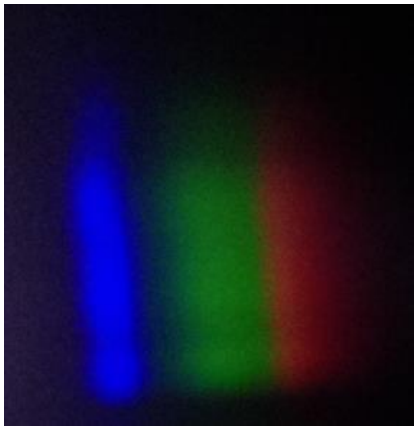


Fig. 7. Sample spectrum of the gluten-free flour

To determine how closely related two sets of data are, correlation coefficient calculations are utilized. The formula (1) provide a result in the range of -1 and 1, where strongly positive relationships are denoted by a 1, strongly negative relationships are denoted by a -1 and zero means there is absolutely no relationship.

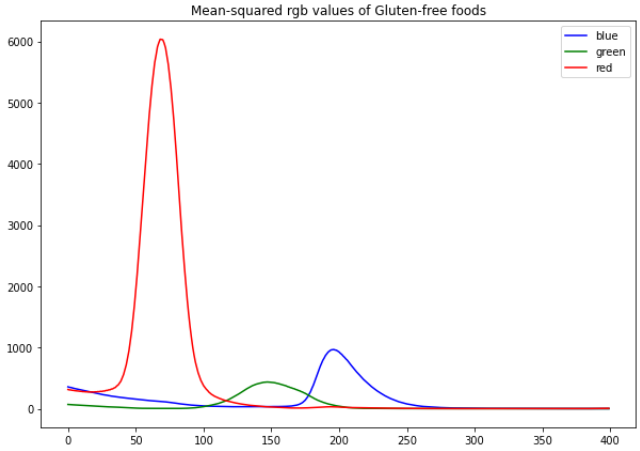


Fig. 8. Mean-squared RGB values of Gluten-free foods

R values between +1 and -1 show that the line of best fit has some variation. Pearson's correlation is a correlation coefficient commonly used in linear regression. [12]

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

After the correlation operation, custom Convolutional Neural Network (CNN) architecture was used for the identification of gluten. Input shapes of all CNN models are 370x400x3 and the output shape is 2 for individual classes. The custom CNN model is represented in Figure 11. The dataset was shuffled and split into training, validation, and tests set with 60-20-20 percent respectively. Adam solver was chosen as the optimizer with a 0.01 learning rate. The loss function is chosen as categorical cross-entropy loss:

$$\text{Loss} = \frac{-1}{\text{output}} \sum_{n=1}^{\text{size}} \left(y_n \log \check{y}_n + (1 - y_n) \cdot \log(1 - \check{y}_n) \right) \quad (2)$$

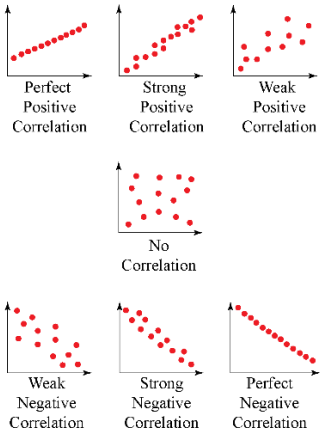


Fig. 9. In this figure, perfectly positive correlation means when one variable changes, other variable changes the exact same rate and way. Strong and weak positive relationships have r values between +1 to 0. The closer r value is to zero, the weaker the correlation and variables are more scattered. When the r value is zero, this tells us that two variables have no correlation between. These two variables are independent from each other. In the range where the r value is negative, the two variables show a similar relationship to the range where it is positive, but in the opposite direction. When one variable decreases, the other one tends to increase.

The 3D printed physical set up that is used for gathering data showed on Figure 10. In addition, for the code implementation, TensorFlow library was used and experiment part of the project.

Fig. 10. 3D printed physical realization of the system

Google Colab was chosen as python environment and all the experiments were performed in this environment which has Nvidia T4 GPU Intel(R) Xeon(R) CPU @ 2.30GHz and 25.46 GB ram.

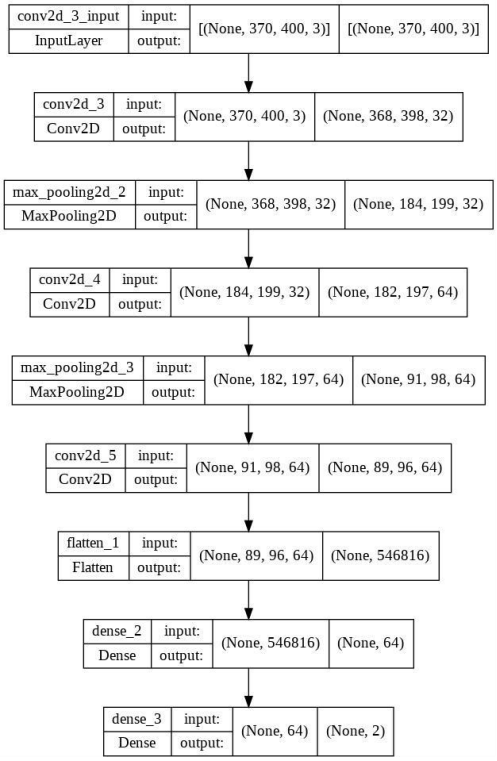


Fig. 11. Custom CNN model

IV. Results and Discussion

CNN model reaches %91 train and %91 validation accuracy after 15 epochs which is desirable since it has no overfitting issue. %86 test accuracy was achieved with the CNN model. The accuracy graph of the training and validation showed in Figure 12.

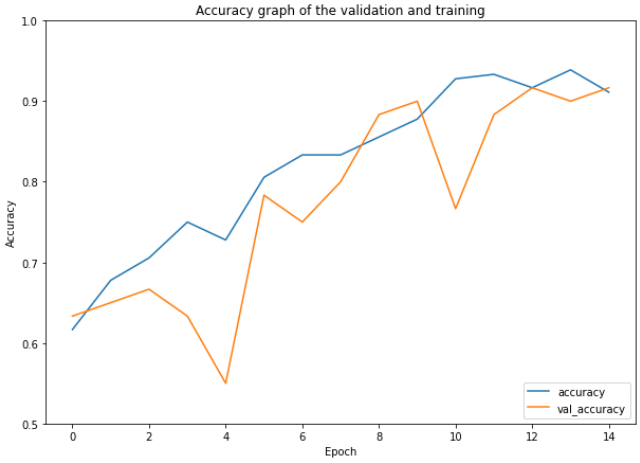


Fig. 12. Accuracy graph of the validation and training

V. Conclusion

For a small dataset, the results were unexpectedly good. However, there should more data since there are immense of food. On the other hand, the thin film diffraction grating can be used since there is a corrosive process to peel the layer of the DVD which affects the accuracy of the results. The dataset should be gathered with more isolated gluten-free foods since

there is a contamination in production lines. Moreover, the prototype that is used can be smaller by decreasing the size of the LED and using a special purposed circuit board instead of Raspberrypi.

REFERENCES

[1] "Celiac Disease Statistics", Celiac.com, 2022. [Online]. Available: https://www.celiac.com/articles.html/celiac-disease-statistics-r1147/. [Accessed: 18- Apr 2022].

[2] M. Allmann, U. Candrian, C. H�felein and J. L�thy, "Polymerase chain reaction (PCR): a possible alternative to immunochemical methods assuring safety and quality of food Detection of wheat contamination in non-wheat food products", Zeitschrift f�r Lebensmittel-Untersuchung und -Forschung, vol. 196, no. 3, pp. 248-251, 1993. Available: 10.1007/bf01202741 [Accessed 18 April 2022].

[3] J. Mejías, X. Lu, C. Osorio, J. Ullman, D. von Wettstein and S. Rustgi, "Analysis of Wheat Prolamins, the Causative Agents of Celiac Sprue, Using Reversed Phase High Performance Liquid Chromatography (RP-HPLC) and Matrix-Assisted Laser Desorption Ionization Time of Flight Mass Spectrometry (MALDI-TOF-MS)", Nutrients, vol. 6, no. 4, pp. 1578-1597, 2014. doi: 10.3390/nu6041578.

[4] Nima Labs, Inc.https://nimasensor.com/. (accessed July 2019).

[5] E. Albanell, B. Miñarro and N. Carrasco, "Detection of low-level gluten content in flour and batter by near infrared reflectance spectroscopy (NIRS)", Journal of Cereal Science, vol. 56, no. 2, pp. 490-495, 2012. doi: 10.1016/j.jcs.2012.06.011

[6] Osorio, Mejías and Rustgi, "Gluten Detection Methods and their Critical Role in Assuring Safe Diets for Celiac Patients", Nutrients, vol. 11, no. 12, p. 2920, 2019. doi: 10.3390/nu11122920.

[7] “Raspberry pi noir camera v2 - farnell.” [Online]. Available: https://www.farnell.com/datasheets/2056180.pdf. [Accessed: 18-Jun-2022].

[8] I. Applications and E. Inc., "Imaging Electronics 101: Understanding Camera Sensors for Machine Vision Applications", Edmundoptics.com, 2022. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/#:~:text=CCD%20and%20CMOS%20sensors%20are,usually%20given%20from%20400%20%2D%201000nm. [Accessed: 18- Jun-2022].

[9] B. Saleh and M. Teich, Fundamentals of photonics. Hoboken: Wiley, 1991, p. 62.

[10] V. Petrov, A. Kryuchyn and I. Gorbov, "High-density optical disks for long-term information storage", SPIE Proceedings, 2011. doi: 10.1117/12.900745.

[11] Kent State University, "SPSS TUTORIALS: PEARSON CORRELATION" kent.edu. https://libguides.library.kent.edu/SPSS/PearsonCorr (accessed May. 24, 2022)

[12]] Stephanie Glen, "Correlation Coefficient: Simple Definition, Formula, Easy Steps" StatisticsHowTo.com. https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/ (accessed June 18, 2022)

Workload Table

Name	Workload
Mert Alperen Beşer	• Optical system design and implementation. • Physical System Design • Data set gathering • CNN model training
Egecan Yurtcan	• Literature Review • Raspberry-pi communication and code implementation • IoT design
Burak Demirtaş	• Literature Review • Data Analysis • Material research

Gluten Detection with Portable Visible Light Near-Infrared Reflectance Spectrometer for Celiac Patients

Colab - Link --> https://colab.research.google.com/drive/1C3MAdgvn70E18F_1WJLF1KU_vVTT5ucW#scrollTo=xCBjRbx-U_nh

Importing required libraries

In [ ]:

```
import cv2
import numpy as np
import os
import shutil
from google.colab.patches import cv2_imshow
```

In [ ]:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
from keras.models import Sequential, load_model
```

Preprocessing the data

In [ ]:

```
dir = "/content/drive/MyDrive/dataset/Glutensiz"
dest_dir = "/content/drive/MyDrive/Glutensiz"
```

In [ ]:

```
counter = 0
for subdir, dirs, files in os.walk(dir):
  for i in files:
    counter+=1
counter
```

Out[ ]:

```
150
```

In [ ]:

```
for item in os.listdir(dir):
  print(item)
```

```
Kurabiye
Atistirmalik
Karisik
Glutenli
Ekmek
Un
```

In [ ]:

```
counter = 1
for subdir, dirs, files in os.walk(dir):

  for file in files:
      #Saving pictures with number names
      new_path = os.path.join(dest_dir,str(counter)+'.jpg')
      full_path = os.path.join(subdir, file)

      #Copying all the images into the new folder
      shutil.copy(full_path, new_path)
      counter = counter + 1

print(counter-1)
```

```
150
```

In [ ]:

```
subdirs, dirs, files = os.walk(dest_dir).__next__()
m = len(files)
m
```

Out[ ]:

```
150
```

In [ ]:

```
gluten_dir = "/content/drive/MyDrive/Glutenli"
No_gluten_dir =  "/content/drive/MyDrive/Glutensiz"
```

In [ ]:

```
img_data = []
for i in range(1,151):

  img = cv2.imread(gluten_dir+"/%s.jpg" %(i))
  cropped_img = img[250:620, 0:400]
```

```
  img_data.append(cropped_img)
```

In [ ]:

```
gluten_data = np.array(img_data)
gluten_data.shape
```

Out[ ]:

```
(150, 370, 400, 3)
```

In [ ]:

```
img_data = []
for i in range(1,151):

  img = cv2.imread(No_gluten_dir+"/%s.jpg" %(i))
  cropped_img = img[250:620, 0:400]
  img_data.append(cropped_img)
```

In [ ]:

```
no_gluten_data = np.array(img_data)
no_gluten_data.shape
```

Out[ ]:

```
(150, 370, 400, 3)
```

In [ ]:

```
labels = np.zeros(150)
labels = np.concatenate((labels, np.ones(150)), axis=0)
labels.shape
```

Out[ ]:

```
(300,)
```

Normalizing the data and concatenate the 2 class

In [ ]:

```
full_data = np.concatenate((no_gluten_data,gluten_data),axis = 0)/255.0 #Normalizing the pixel values between 0 and 1
```

In [ ]:

```
np.save("/content/drive/MyDrive/project_data/X.npy",full_data)
np.save("/content/drive/MyDrive/project_data/Y.npy",labels)
```

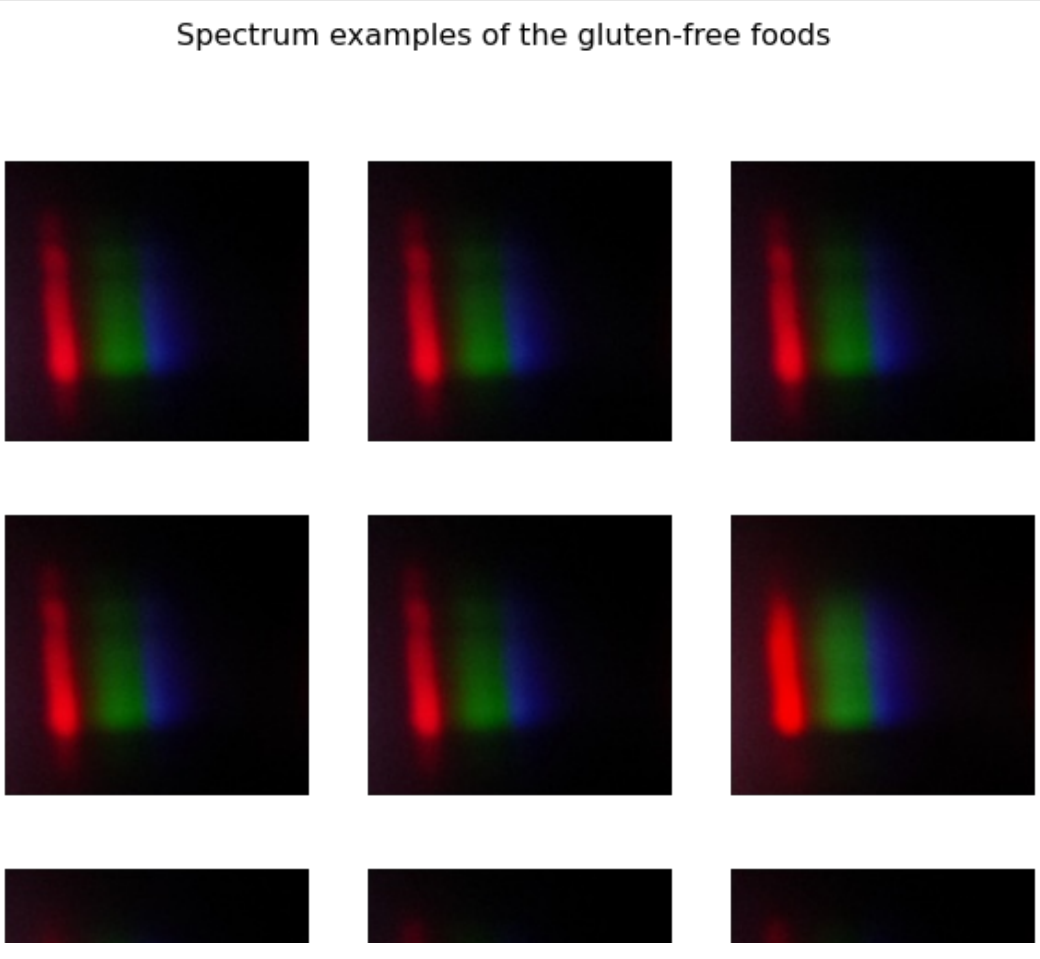
In [ ]:

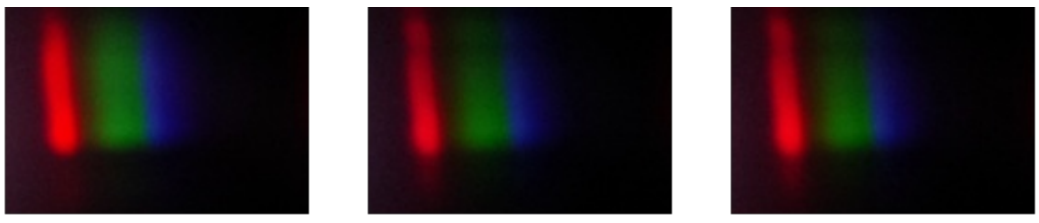
```
X = np.load("/content/drive/MyDrive/project_data/X.npy")
y = np.load("/content/drive/MyDrive/project_data/Y.npy")
```

Visualization of the data and investigation of the correlation

In [ ]:

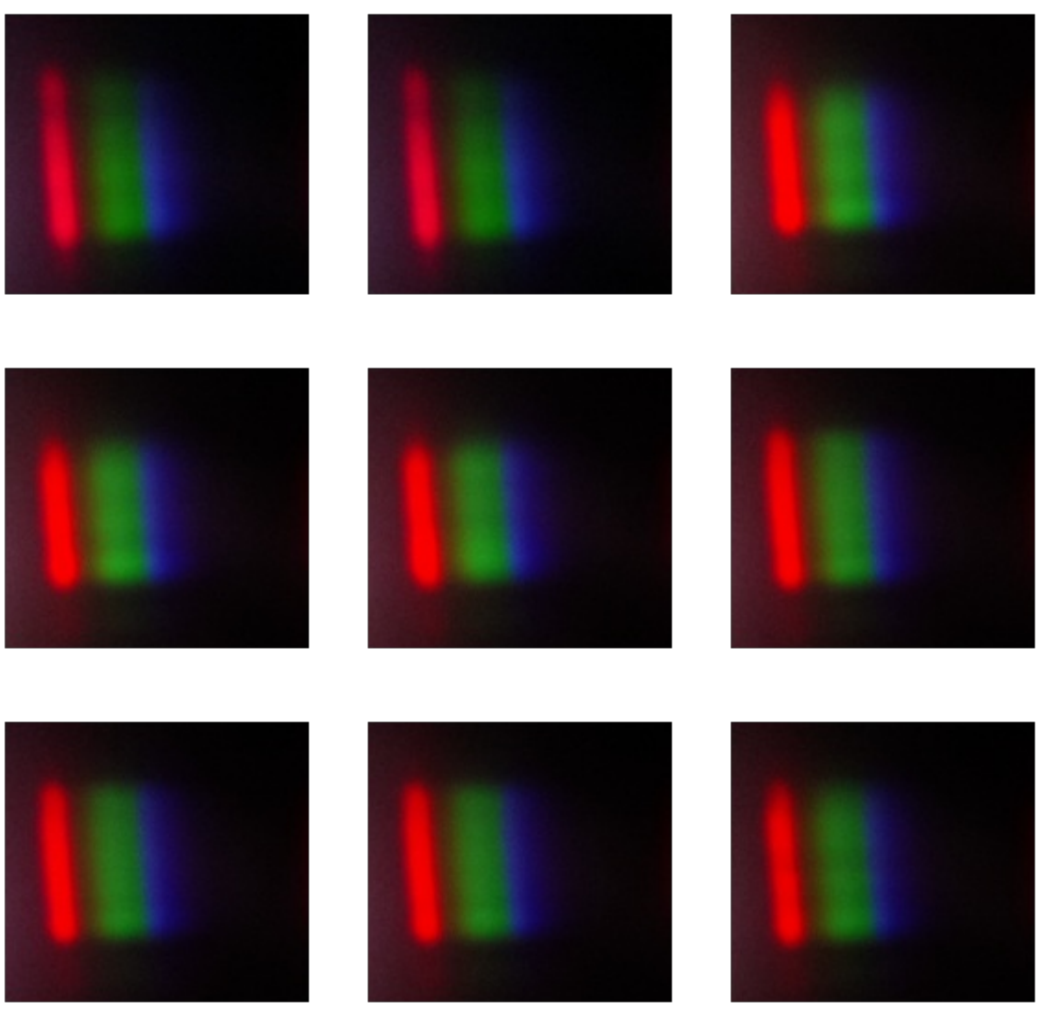
```
plt.figure(figsize=(10, 10))
i = 0

for j in range(91,100):
  img = X[j]*255
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(img.astype("uint8"))
  plt.suptitle('Spectrum examples of the gluten-free foods', fontsize=16)
  plt.axis("off")
  i += 1
```





In [ ]:

```
plt.figure(figsize=(10, 10))
i = 0

for j in range(151,160):
  img = X[j]*255
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(img.astype("uint8"))
  plt.suptitle('Spectrum examples of the non gluten-free foods', fontsize=16)
  plt.axis("off")
  i += 1
```



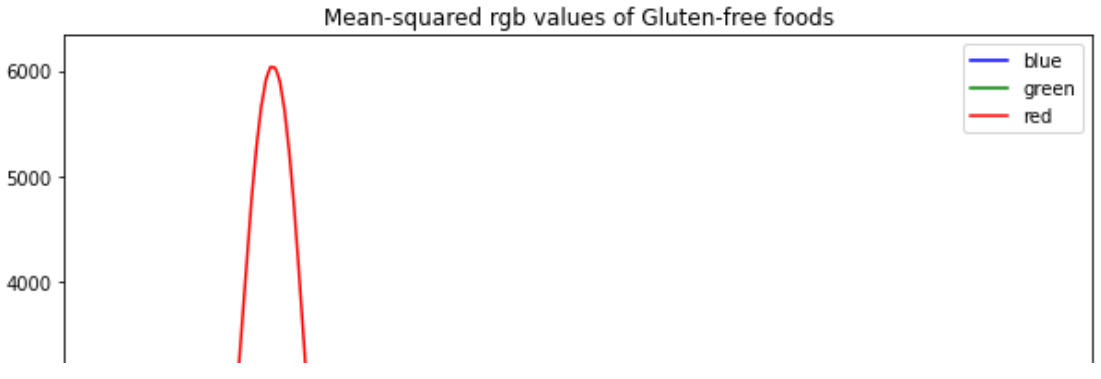
In [ ]:

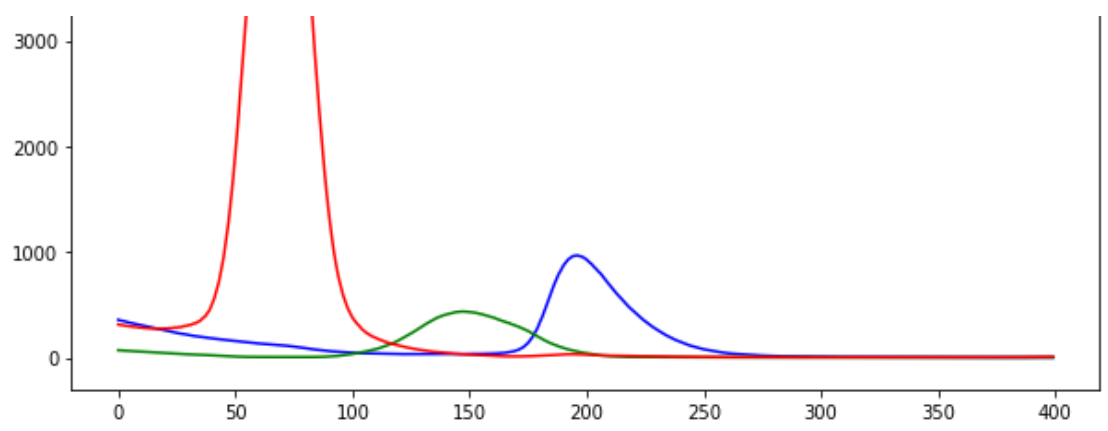
```
img = X[0:150]

red_mean = np.mean(img[:,:,:,0:1],axis = 0)
red_sum = np.sum(red_mean,axis = 0)
red_ms_free = np.square(red_sum)


green_mean = np.mean(img[:,:,:,1:2],axis = 0)
green_sum = np.sum(green_mean,axis = 0)
green_ms_free = np.square(green_sum)


blue_mean = np.mean(img[:,:,:,2:3],axis = 0)
blue_sum = np.sum(blue_mean,axis = 0)
blue_ms_free = np.square(blue_sum)
```

In [ ]:

```
plt.figure(1)
plt.figure(figsize=(10, 7))
plt.plot(blue_ms_free,"b")
plt.plot(green_ms_free,"g")
plt.plot(red_ms_free,"r")
plt.title("Mean-squared rgb values of Gluten-free foods")
plt.legend(['blue','green','red'])
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```





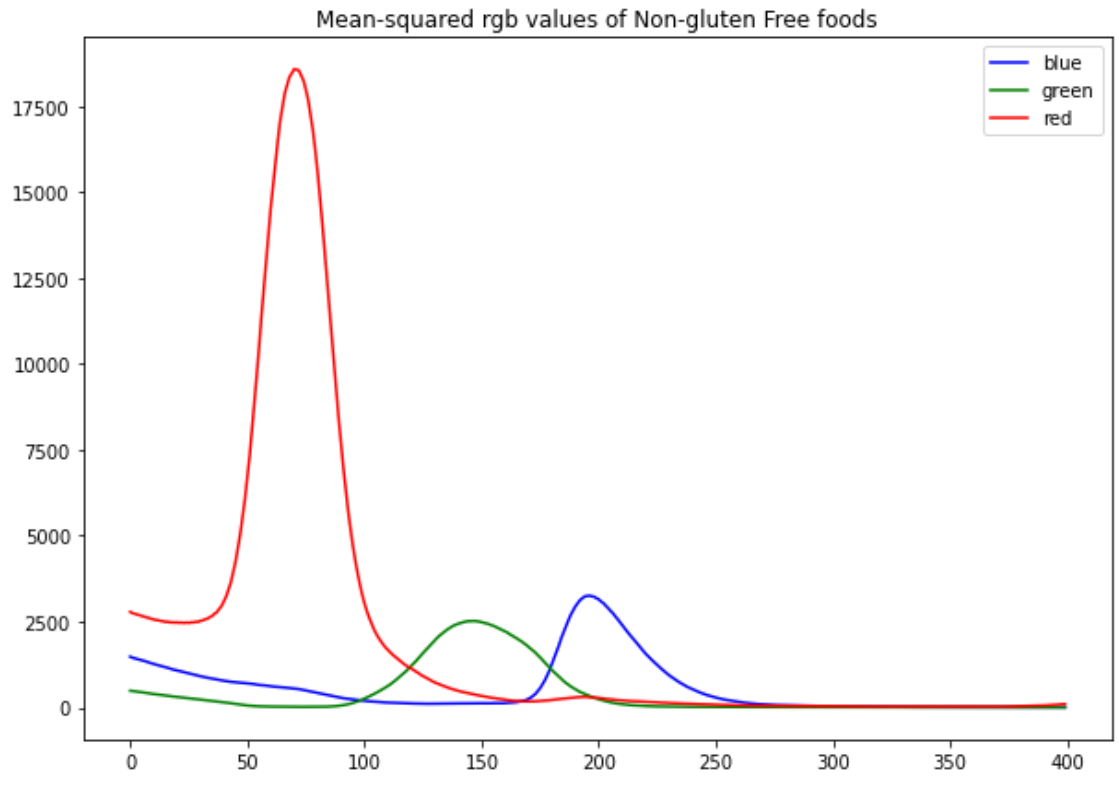
In [ ]:

```
img = X[150:300]

red_mean = np.mean(img[:,:,:,0:1],axis = 0)
red_sum = np.sum(red_mean,axis = 0)
red_ms = np.square(red_sum)


green_mean = np.mean(img[:,:,:,1:2],axis = 0)
green_sum = np.sum(green_mean,axis = 0)
green_ms = np.square(green_sum)


blue_mean = np.mean(img[:,:,:,2:3],axis = 0)
blue_sum = np.sum(blue_mean,axis = 0)
blue_ms = np.square(blue_sum)
```

In [ ]:

```
plt.figure(2)
plt.figure(figsize=(10, 7))
plt.plot(blue_ms,"b")
plt.plot(green_ms,"g")
plt.plot(red_ms,"r")
plt.title("Mean-squared rgb values of Non-gluten Free foods")
plt.legend(['blue','green','red'])
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

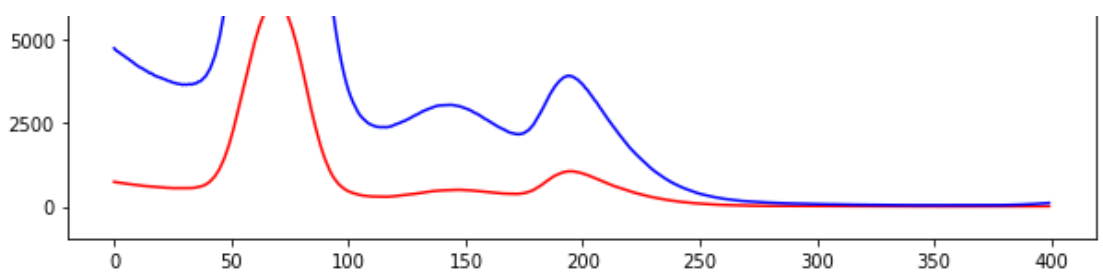


In [ ]:

```
g_free_sum = red_ms_free+green_ms_free+blue_ms_free
g_sum = red_ms+green_ms+blue_ms
plt.figure(figsize=(10, 7))
plt.plot(g_free_sum,'r')
plt.plot(g_sum,'b')
plt.title("Comparison of the Gluten-Free and Non-gluten free food RGB gathered graphs")
plt.legend(['Gluten-free','Non-gluten free'])
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7f851e08c290>
```



Correlation coefficient value between the mean-squared sum of the gluten-free and non-gluten-free data.

In [ ]:

```
corr_ = np.corrcoef(g_free_sum.flatten(),g_sum.flatten())
corr_
```

Out[ ]:

```
array([[1.        , 0.97753465],
       [0.97753465, 1.        ]])
```

Convolutional Neural Network

Train-Test-Validation Split

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size=0.2, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=1) # 0.25 x 0.8 = 0.2
```

In [ ]:

```
full_data.shape
```

Out[ ]:

```
(300, 370, 400, 3)
```

In [ ]:

```
np.save('/content/drive/MyDrive/project_data/X_train.npy', X_train)
np.save('/content/drive/MyDrive/project_data/y_train.npy', y_train)

np.save('/content/drive/MyDrive/project_data/X_val.npy', X_val)
np.save('/content/drive/MyDrive/project_data/y_val.npy', y_val)


np.save('/content/drive/MyDrive/project_data/X_test.npy', X_test)
np.save('/content/drive/MyDrive/project_data/y_test.npy', y_test)
```

In [ ]:

```
X_train = np.load('/content/drive/MyDrive/project_data/X_train.npy')
y_train = np.load('/content/drive/MyDrive/project_data/y_train.npy')

X_val = np.load('/content/drive/MyDrive/project_data/X_val.npy')
y_val = np.load('/content/drive/MyDrive/project_data/y_val.npy')


X_test = np.load('/content/drive/MyDrive/project_data/X_test.npy')
y_test = np.load('/content/drive/MyDrive/project_data/y_test.npy')
```

In [ ]:

```
print("Train set Shape")
print(X_train.shape)
print(y_train.shape)

print("Validation set Shape")
print(X_val.shape)
print(y_val.shape)

print("Test set Shape")
print(X_test.shape)
print(y_test.shape)
```

```
Train set Shape
(180, 370, 400, 3)
(180,)
Validation set Shape
(60, 370, 400, 3)
(60,)
Test set Shape
(60, 370, 400, 3)
(60,)
```

Defining the CNN model

In [ ]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(370, 400, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

In [ ]:

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2))
```

In [ ]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 368, 398, 32)      896

 max_pooling2d_2 (MaxPooling  (None, 184, 199, 32)     0
 2D)

 conv2d_4 (Conv2D)           (None, 182, 197, 64)      18496

 max_pooling2d_3 (MaxPooling  (None, 91, 98, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 89, 96, 64)        36928

 flatten_1 (Flatten)         (None, 546816)            0

 dense_2 (Dense)             (None, 64)                34996288

 dense_3 (Dense)             (None, 2)                 130

=================================================================
Total params: 35,052,738
Trainable params: 35,052,738
Non-trainable params: 0
_________________________________________________________________
```

Training

In [ ]:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=15,
                    validation_data=(X_val, y_val))
```

```
Epoch 1/15
6/6 [==============================] - 2s 211ms/step - loss: 1.3108 - accuracy: 0.6167 - val_loss: 0.6362 - val_accuracy: 0.6333
Epoch 2/15
6/6 [==============================] - 1s 162ms/step - loss: 0.5609 - accuracy: 0.6778 - val_loss: 0.5604 - val_accuracy: 0.6500
Epoch 3/15
6/6 [==============================] - 1s 163ms/step - loss: 0.4902 - accuracy: 0.7056 - val_loss: 0.5282 - val_accuracy: 0.6667
Epoch 4/15
6/6 [==============================] - 1s 164ms/step - loss: 0.4426 - accuracy: 0.7500 - val_loss: 0.4875 - val_accuracy: 0.6333
Epoch 5/15
6/6 [==============================] - 1s 164ms/step - loss: 0.4686 - accuracy: 0.7278 - val_loss: 0.5980 - val_accuracy: 0.5500
Epoch 6/15
6/6 [==============================] - 1s 163ms/step - loss: 0.4016 - accuracy: 0.8056 - val_loss: 0.4979 - val_accuracy: 0.7833
Epoch 7/15
6/6 [==============================] - 1s 170ms/step - loss: 0.3394 - accuracy: 0.8333 - val_loss: 0.4955 - val_accuracy: 0.7500
Epoch 8/15
6/6 [==============================] - 1s 171ms/step - loss: 0.3617 - accuracy: 0.8333 - val_loss: 0.5896 - val_accuracy: 0.8000
Epoch 9/15
6/6 [==============================] - 1s 164ms/step - loss: 0.3170 - accuracy: 0.8556 - val_loss: 0.4110 - val_accuracy: 0.8833
Epoch 10/15
6/6 [==============================] - 1s 166ms/step - loss: 0.2729 - accuracy: 0.8778 - val_loss: 0.3565 - val_accuracy: 0.9000
Epoch 11/15
6/6 [==============================] - 1s 164ms/step - loss: 0.2387 - accuracy: 0.9278 - val_loss: 0.4372 - val_accuracy: 0.7667
Epoch 12/15
6/6 [==============================] - 1s 165ms/step - loss: 0.2212 - accuracy: 0.9333 - val_loss: 0.3819 - val_accuracy: 0.8833
Epoch 13/15
6/6 [==============================] - 1s 165ms/step - loss: 0.1852 - accuracy: 0.9167 - val_loss: 0.3682 - val_accuracy: 0.9167
Epoch 14/15
6/6 [==============================] - 1s 164ms/step - loss: 0.1493 - accuracy: 0.9389 - val_loss: 0.3601 - val_accuracy: 0.9000
Epoch 15/15
6/6 [==============================] - 1s 160ms/step - loss: 0.2477 - accuracy: 0.9111 - val_loss: 0.3406 - val_accuracy: 0.9167
```

In [ ]:

```
model.save("/content/drive/MyDrive/project_data/")
```

In [ ]:

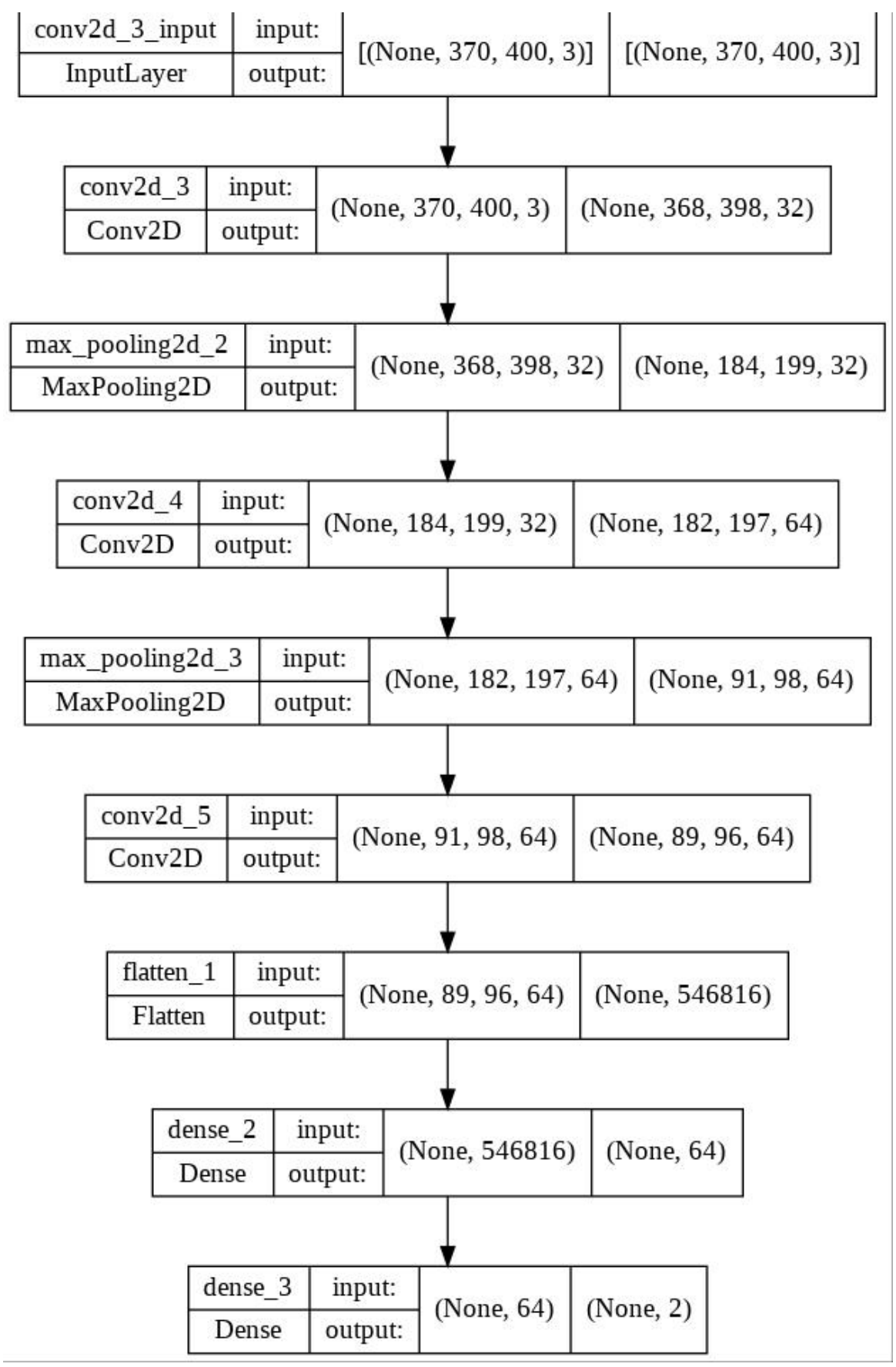
```
model = load_model("/content/drive/MyDrive/project_data/")
```

Model structure

In [ ]:

```
dot_img_file = "/content/drive/MyDrive/project_data"+'/structure.jpg'
tf.keras.utils.plot_model(model, to_file=dot_img_file, show_shapes=True)
```

Out[ ]:



Results

In [ ]:

```
plt.figure(figsize=(10, 7))
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.title("Accuracy graph of the validation and training")
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)
```

```
2/2 - 0s - loss: 0.2523 - accuracy: 0.8667 - 307ms/epoch - 153ms/step
```

In [ ]:

```
print("Test accuracy:" ,test_acc)
```

```
Test accuracy: 0.8666666746139526
```