

Implementation of Citra Technology to Identify the Freshness of Shrimp for Consumption

Aris Prayogo
Yogyakarta University of Technology
Yogyakarta, Indonesia

Enny Itje Sela
Yogyakarta University of Technology
Yogyakarta, Indonesia

ABSTRACT

The increasing market demand for shrimp makes many parties take advantage of this condition by selling shrimp that are not suitable for consumption such as rotten shrimp, diseased shrimp and formalin. To ensure the quality of shrimp received by consumers, it is necessary to test the freshness, so far the tests carried out through microbiological and chemical analysis but in this way it is less effective because it takes longer time, requires a lot of labor, requires a fairly expensive cost, so it affects the production of shrimp. The method used in this research is Convolutional Neural Network (CNN) which is done through classification with a preprocessing stage consisting of rescale, rotation range, horizontal flip, shear range, fill mode, width shift range, height shift range and zoom range. This classification stage produces shrimp freshness output which is divided into 3 categories. System development with the convolutional neural network method gets the best accuracy of 99.39% by using a learning rate of 0.001 and max epoch 100 with the results of the classification of the three classes with the tested Citra is correct..

General Terms

VS Code, Jupyter Notebook, Python, Web Application

Keywords

Convolutional Neural Network (CNN), Shrimp, Freshness

1. INTRODUCTION

Shrimp is one of the main commodities in the fisheries sub-sector that is expected to increase foreign exchange. Market demand tends to increase and the available resources are sufficient to provide enormous opportunities. An efficient and integrated market ensures restoration of stability of prices across geographically separated markets over the long run [1]. In recent years, research on keeping shrimp fresh has become of great interest to aquatic product researchers [2]. Shrimp freshness is the most basic indication in determining the quality of shrimp before consumption because shrimp is a food that quickly decays due to spoilage (perishable food). Therefore, to protect the rights and interests of consumers, shrimp freshness identification technology has received much attention in food research [3].

Modern quality of life and consumer attitudes have increased the interest in food safety, health, food quality, economic issues, and environmental problems [4]. The freshness of shrimp can be seen through changes that occur in the color of the skin. Checking shrimp freshness through microbiological and chemical analysis is ineffective as these methods are time-consuming, labor-intensive and expensive. Detection and identification that is fast, accurate and minimizes false negatives is expected to provide the right treatment [5]. To address the current problem of time-consuming and costly shrimp freshness detection, in order to be able to accurately identify the freshness grade to which shrimp belong, and

provide a fast, accurate and low-cost identification method for the cold chain transportation industry [6]. Convolutional Neural Networks (CNNs) represent a category of deep learning algorithms renowned for their exceptional capacity to extract intricate features from image data [7]. In the process of the development of computer vision technology, image classification is the key to improve the level of technology development, and it also relates to the efficiency of pattern recognition [8]. Therefore, this research focuses on designing an application that is used to identify the freshness of shrimp using the Convolutional Neural Networks method because it has been proven through several previous studies that this method produces good enough accuracy so that the feasibility test for shrimp consumption is well achieved.

2. RESEARCH METHOD

System design is an overview of the flow of the running system. The system design is visualized in Figure 1 below.

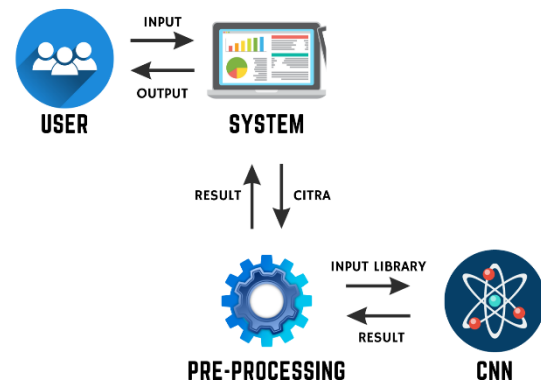


Fig 1: Architecture Model

This system is web-based, users enter image data in the form of images. The image that has been entered will be preprocessed then continued by inputting the Library as a calculation and classification stage using the Convolutional Neural Network model. Data preprocessing is a fundamental stage in deep learning modeling and serves as the cornerstone of reliable data analytics [9]. Preprocessing aims to correct inconsistent data errors to make it easier to understand and process. After the data is classified, it will display the output results in the form of a description of the condition of the shrimp. With these results, users will immediately know the freshness condition of shrimp so that they can avoid high bacterial content in shrimp that are not fresh.

2.1 Data Collection Procedure

The data collection process in this study went through 2 stages, namely observation and literature studies.

2.1.1 Interview

Researchers collected data through direct interviews with relevant parties conducted for two weeks from November 6 to November 20, 2022. Data obtained from shrimp farmers and sellers were taken pictures as research material.

2.1.2 Literature Study

Researchers collect information sourced from scientific journals and books available on the internet. A scientific work in the form of theories in previous studies is needed with the aim of knowing the basic concepts of a problem in the object of research.

2.2 Conceptual Design

The complex product development project is a conversion process from customers' demands to a physical structure with consideration of numerous design constraints [10]. Conceptual design is the process of building information models that do not depend on physical considerations such as application programs, programming languages used, hardware platforms, and others. A good conceptual design will produce a quality system or application that meets user needs. Therefore, the system design consists of algorithms described in the form of flowcharts. Flowcharts provide an alternative approach to teaching complex content, which allows students to organize and summarize information that promotes meaningful learning [11]. The use of flowcharts aims to facilitate understanding of the process, identify problems, and help communication both between developers, users, and other interested parties.

2.2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a popular deep learning algorithm applied to image data due to its high network depth.

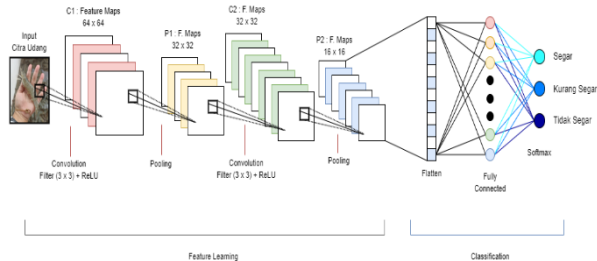


Fig 2: Architecture CNN

Convolutional Neural Network architecture is divided into two main parts, namely Feature Learning and Classification. Feature Learning aims to process an input into features based on the characteristics of the input in the form of numbers in vector form, this layer is divided into two parts namely Convolution Layer and Poling Layer. Classification aims to classify neurons that have been extracted features, classification is divided into three namely Flatten, Fully-connected and Softmax.

2.2.2 Shrimp Identification

In Figure 3 is the data processing process using the CNN method. The process flow will explain in detail the steps to identify shrimp using the CNN method.

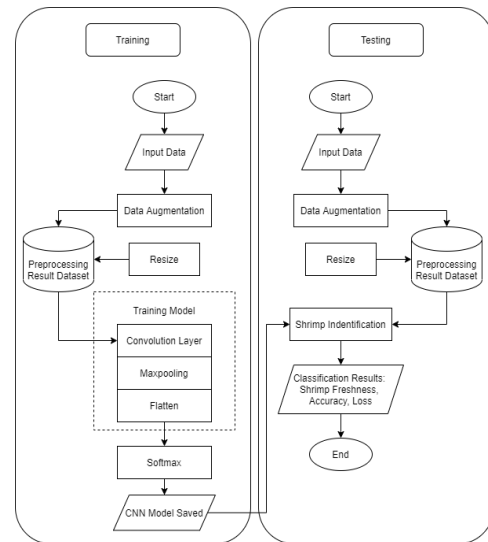


Fig 3: Shrimp Identification

The first stage is the input stage of the image data to be used. Next is the pre-processing stage which functions to process the inputted image data so that it can be used in the next process. Furthermore, the data division stage becomes 3, namely training data, testing data, and validation data. Then the Training data performs a resize operation and is continued with the first CNN convolution operation where the image will be carried out a convolution operation that determines the number of layers and is continued with a maxpooling operation that determines the maximum value of each filter shift. Then flatten which aims to change in matrix form into a vector. Furthermore, the fully connected process at this stage will calculate each neuron that produces an error value using weights that are continuously updated using the backpropagation method. The softmax stage produces the output of class prediction results where the image data will be put into a predetermined class. The Testing process aims to test the data to get accuracy. The first process is to enter the image data to be processed. Next do the Pre-processing stage. Then input the CNN model that has been stored in the Training process where at this stage it will determine the classification of the image class. Finally, display the results and accuracy.

2.2.3 Resize

In Figure 4 is the Resize process, starting with entering the path/file of the image to be resized and the desired new width value. Next, the image is opened using the given path, and the width and height of the image are taken. The height to width ratio is calculated to preserve the aspect of the image. Then, a new height is calculated based on the aspect ratio and the desired new width.

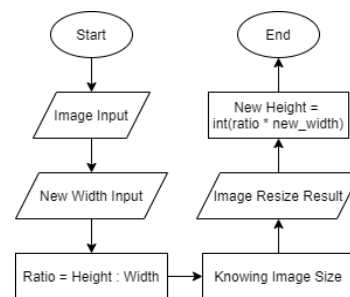


Fig 4: Resize Process

After that, the image is resized according to the new calculated width and height. The resizing process results in a resized image. Finally, the resized image is returned as the output of the function.

2.2.4 Augmentation

In the Augmentation process is a rescale process that aims to change the scale of the image value. First is the rotation range process of 180 followed by a horizontal flip to reflect the image horizontally. Then the shear range serves to tilt the image. Fill mode serves to fill areas that have no value.

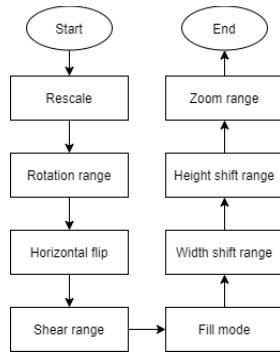


Fig 5: Augmentation Process

Then in the width shift range, the number used is between 0.0 and 1.0. This number determines how much the image is randomly shifted, either to the right or to the left. After that, the height shift range works the same as the width shift range, but the shift is done vertically. Finally, the zoom range serves to enlarge the image.

3. DISCUSSION AND RESULT

3.1 Assumptions

The assumption in this study is that the identification of freshness and survival of vanamei shrimp based on shrimp color images can prevent the spread of disease due to consuming shrimp that are not fresh.

3.2 Hypothesis

Based on the assumptions obtained, this research hypothesizes that "Implementation of the Convolution Neural Network (CNN) method to identify the freshness and feasibility of shrimp can help people to know the level of freshness of shrimp that is suitable for consumption".

3.3 Run an Experiment

After the model design and data collection are complete, the next step is coding. This system uses a Jupyter notebook as the main workspace in building the system. Jupyter Notebook (JN) is an example of an innovative and efficient digital tool that can be used effectively in higher education [12]. Jupyter Notebooks have transformed the communication of data analysis pipelines by facilitating a modular structure that brings together code, markdown text, and interactive visualizations [13]. Then Python was used as the programming language to create the system. Python is an open-source programming language that has been gaining popularity in research and development areas [14]. Python provides a user-friendly and flexible environment for implementing these analytical methods, allowing for more efficient and reliable results [15]. Finally, researchers use Visual Studio Code as a workspace in building web applications.

3.4 Implementations

Implementation aims to apply the results of system analysis and design into an application that can be used by the community to determine the freshness of shrimp.

3.4.1 Implementation of CNN Model

3.4.1.1 Library Configuration

Setup library uses several libraries needed for this research including numpy, pandas, matplotlib for data processing. Tensorflow and Keras libraries for machine learning. And system and openCV libraries for image preprocessing.

```

1 import tensorflow as tf
2 from tensorflow import keras
3 from tensorflow.keras import backend as K
4 from tensorflow.keras.layers import Dense,
5 Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization,
6 Flatten,Input
7
8 from tensorflow.keras.optimizers import Adam, Adamax
9 from tensorflow.keras.metrics import categorical_crossentropy
10 from tensorflow.keras import regularizers
11 from tensorflow.keras.preprocessing.image import
12 ImageDataGenerator
13 from tensorflow.keras.models import Model, load_model,
14 Sequential
15 from keras.callbacks import ModelCheckpoint
16
17 import numpy as np
18 import pandas as pd
19 from sklearn.model_selection import train_test_split
20 import matplotlib.pyplot as plt
21 from matplotlib.pyplot import imshow
22 import os
23 import seaborn as sns
24 sns.set_style('darkgrid')
25 from sklearn.metrics import confusion_matrix,
26 classification_report
27 from IPython.display import display, HTML
28 from PIL import Image
29 import random
30
31
32
33
34



```

Fig 6: Library Configuration

3.4.1.2 Data Preparation

This research uses data in the form of images taken directly from vanamei shrimp farmers and vanamei shrimp sellers. The dataset in this study is an image or image with a total of 413, the dataset is categorized into 3 groups, namely fresh shrimp with a total of 279, less fresh shrimp 59 and not fresh shrimp 75.

Table 1. Sample Shrimp Data

| No | Citra | Description |
|----|--|-------------|
| 1 |  | Fresh |
| 2 |  | Less Fresh |



The second process is the dataset load process where a variable with the name dataset is used to access the material folder. Based on Figure 7 Load dataset, through the folder all files with jpeg, jpg and png types will be collected in the form of a list. The filepath list is created from the paths where the image files are stored. The list is then merged together in the filepaths_ori variable, converted into a dataframe along with the list of labels, and stored in the df variable.

```

1 sdir = r'C:/Users/acer/Documents/1 Proyek
2 Informatika/dataset/bahan'
3
4
5 filepaths = []
6 labels = []
7 classlist = os.listdir(sdir)
8
9
10 for class in classlist:
11     classpath = os.path.join(sdir, class)
12
13
14     if os.path.isdir(classpath):
15         flist = os.listdir(classpath)
16
17         for f in flist:
18             fpath = os.path.join(classpath, f)
19             filepaths.append(fpath)
20             labels.append(class)
21
22
23 df = pd.DataFrame({'filepaths': filepaths, 'labels': labels})
24
25
26 print(df.head())
27
28 print(df['labels'].value_counts())
29

```

Fig 7: Calling Data

3.4.1.3 Data Preprocessing

Data Sharing

Separate the shrimp image data into training data with a ratio of 80%, test data 10%, and validation data 10%.

```

1 train_split = 0.8
2 test_split = 0.1
3 valid_split = 0.1
4
5
6 # Mendapatkan jumlah total data
7 total_data = len(df)
8
9
10 # Mengacak indeks data
11 random_indices = list(range(total_data))
12 random.shuffle(random_indices)
13
14
15 # Menghitung jumlah data untuk setiap subset
16 train_size = int(train_split * total_data)
17 test_size = int(test_split * total_data)
18 valid_size = total_data - train_size - test_size
19
20
21 # Memisahkan data menjadi data latih, data uji, dan data
22 validasi
23 train_df = df.iloc[random_indices[:train_size]]
24 test_df = df.iloc[random_indices[train_size:(train_size +
25 test_size)]]
26 valid_df = df.iloc[random_indices[(train_size + test_size):]]
27
28
29
30 print('Panjang train_df:', len(train_df))
31 print('Panjang test_df:', len(test_df))
32 print('Panjang valid_df:', len(valid_df))
33
34

```

Fig 8: Data Sharing

Resize

The resizing process produces a resized image to be returned as the function output.

```

1 # Fungsi resize gambar
2 def resize_image(image_path, new_width):
3     image = Image.open(image_path)
4     width, height = image.size
5     ratio = height / width
6     new_height = int(ratio * new_width)
7     resized_image = image.resize((new_width, new_height))
8     return resized_image
9
10
11
12 # Folder tujuan untuk menyimpan gambar hasil resize
13 # Tidak menggunakan direktori tujuan khusus
14 resized_train_dir = "C:/Users/acer/Documents/1 Proyek
15 Informatika/dataset/g"
16 resized_train_labels_dir = "C:/Users/acer/Documents/1 Proyek
17 Informatika/dataset/l"
18
19
20
21 # Membuat direktori tujuan jika belum ada
22 if not os.path.exists(resized_train_dir):
23     os.makedirs(resized_train_dir)
24 if not os.path.exists(resized_train_labels_dir):
25     os.makedirs(resized_train_labels_dir)
26
27
28 # Melakukan resize pada data training
29 for index, row in train_df.iterrows():
30     image_path = row['filepaths']
31     label = row['labels']
32     resized_image = resize_image(image_path, 300)
33     resized_image_path = os.path.join(resized_train_dir,
34 f"{index}.jpg")
35     resized_image.save(resized_image_path)
36     resized_label_path =
37 os.path.join(resized_train_labels_dir, f"{index}.txt")
38     with open(resized_label_path, 'w') as f:
39         f.write(label)

```

Fig 9: Resize

Augmentation

Augmentation is the process of modifying images such as rotation, flipping, rescaling and others. Augmentation aims to enable the model to recognize different image conditions and improve model performance.

```

1 height = 224
2 width = 224
3 channels = 3
4 batch_size = 64
5
6
7 img_shape = (height, width, channels)
8 img_size = (height, width)
9 length = len(test_df)
10 test_batch_size = sorted([int(length/n) for n in range(1,
11 length+1) if length % n == 0 and length/n <= 80],
12 reverse=True)[0]
13 test_steps = int(length / test_batch_size)
14 print('test batch size:', test_batch_size, 'test steps:',
15 test_steps)
16
17
18
19
20 gen = ImageDataGenerator(
21     rescale=1./255,
22     rotation_range=180,
23     horizontal_flip=True,
24     shear_range=0.3,
25     fill_mode='nearest',
26     width_shift_range=0.2,
27     height_shift_range=0.2,
28     zoom_range=0.1
29 )
30
31
32
33
34 train_gen = gen.flow_from_dataframe(
35     train_df,
36     x_col='filepaths',
37     y_col='labels',

```

```

target_size=img_size,
class_mode='categorical',
color_mode='rgb',
shuffle=True,
batch_size=batch_size
)

valid_gen = gen.flow_from_dataframe(
    valid_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    class_mode='categorical',
    color_mode='rgb',
    shuffle=True,

```

Fig 10: Augmentation

3.4.1.4 Creating CNN Models

Modeled a CNN using a library called keras. Three convolutional layers with a matrix using the relu activation function and three max pooling layers with a 2x2 matrix. Then the flatten layer is used to make the input results from the previous layer into one line. Followed by the hidden layer using relu activation. Finally, the output layer uses softmax activation with three outputs.

```

1 base_model=tf.keras.applications.Xception(include_top=False,
2 weights="imagenet",input_tensor=Input(shape=(224,224,3)))
3 model_name = 'Udang'
4 print("Building model with", base_model)
5 model = tf.keras.Sequential([
6     base_model,
7     tf.keras.layers.Conv2D(filters=32,
8 padding='same', kernel_size=3, activation='relu', strides=1),
9     tf.keras.layers.MaxPool2D(pool_size=2,
10 strides=2),
11     tf.keras.layers.Dropout(rate=0.5),
12
13     tf.keras.layers.Flatten(),
14     tf.keras.layers.Dense(3, activation='softmax')
15 ])
16
17 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.008), loss='categorical_crossentropy',
18 metrics='accuracy')
19
20
21
22
23
24

```

Fig 11: CNN Model

| Layer (type) | Output Shape | Param |
|---------------------------------|--------------------|----------|
| xception (Functional) | (None, 7, 7, 2048) | 20861480 |
| conv2d_9 (Conv2D) | (None, 7, 7, 32) | 589856 |
| max_pooling2d_1 (MaxPooling 2D) | (None, 3, 3, 32) | 0 |
| dropout_1 (Dropout) | (None, 3, 3, 32) | 0 |
| flatten_1 (Flatten) | (None, 288) | 0 |
| dense_1 (Dense) | (None, 3) | 867 |

Total params: 21,452,203
 Trainable params: 590,723
 Non-trainable params: 20,861,480

Fig 12: Result of CNN Model

3.4.1.5 Train the Model

The model was trained using the adam optimizer, with a maximum epoch of 50. The steps per epoch and the validation steps were adjusted according to the batch size and the amount of data.

```

1 epochs =100
2
3
4 history=model.fit(x=train_gen, epochs=epochs,
5 validation_data=valid_gen)
6

```

Fig 13: Train the Model

The following are the results of evaluating the accuracy and loss of the test data obtained.

1/1 [=====] - 8s 8s/step - loss: 0.1319 - accuracy: 0.9756
accuracy on the test set is 97.56 %

3.4.1.6 Visualization of Accuracy and Loss

Visualize the accuracy and loss results obtained after training the model. The library used for visualization is matplotlib. The x-axis is the epoch length y-axis is the sum of accuracy and loss. The visualization results get a learning rate of 0.008.

```

1 def tr_plot(tr_data, start_epoch):
2     #Plot the training and validation data
3     tacc=tr_data.history['accuracy']
4     tloss=tr_data.history['loss']
5     vacc=tr_data.history['val_accuracy']
6     vloss=tr_data.history['val_loss']
7     Epoch_count=len(tacc)+ start_epoch
8     Epochs=[]
9     for i in range (start_epoch ,Epoch_count):
10        Epochs.append(i+1)
11        index_loss=np.argmin(vloss)# this is the epoch with the
12        lowest validation loss
13        val_lowest=vloss[index_loss]
14        index_acc=np.argmax(vacc)
15        acc_highest=vacc[index_acc]
16        plt.style.use('fivethirtyeight')
17        sc_label='best epoch= ' + str(index_loss+1 +start_epoch)
18        vc_label='best epoch= ' + str(index_acc + 1+ start_epoch)
19        fig,axes=plt.subplots(nrows=1, ncols=2, figsize=(20,8))
20        axes[0].plot(Epochs,tloss, 'r', label='Training loss')
21        axes[0].plot(Epochs,vloss, 'g',label='Validation loss')
22        axes[0].scatter(index_loss+1 +start_epoch,val_lowest,
23        s=150, c= 'blue', label=sc_label)
24        axes[0].set_title('Training and Validation Loss')
25        axes[0].set_xlabel('Epochs')
26        axes[0].set_ylabel('Loss')
27        axes[0].legend()
28        axes[1].plot (Epochs,tacc,'r',label= 'Training Accuracy')
29        axes[1].plot (Epochs,vacc,'g',label= 'Validation
30        Accuracy')
31        axes[1].scatter(index_acc+1 +start_epoch,acc_highest,
32        s=150, c= 'blue', label=vc_label)
33        axes[1].set_title('Training and Validation Accuracy')
34        axes[1].set_xlabel('Epochs')
35        axes[1].set_ylabel('Accuracy')
36        axes[1].legend()
37        plt.tight_layout
38        #plt.style.use('fivethirtyeight')
39        plt.show()

```

Fig 14: Accuracy and Loss Value



Fig 15: Result of Accuracy and Loss Value

3.4.2 Implementation of WEB

3.4.2.1 Implementation of User Page

Implementation of the User page interface is the implementation of web pages used by User web visitors that contain information about the Shrimp Freshness Classification System.

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Image Classification</title>
   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.0/css/bootstrap.min.css">
   <style>
   body {
     background-color: #9aa1a7;
   }
   .container-box {
     display: flex;
     flex-direction: column;
     align-items: center;
     justify-content: center;
     height: 80vh;
   }
   .image-container {

```

Fig 16: Implementation of User Page

3.4.2.2 Flask Module

The flask module is used to connect the web with machine learning using the python programming language.

```

1 from flask import Flask, render_template, request
2 from tensorflow.keras.models import load_model
3 from tensorflow.keras.preprocessing import image
4 import numpy as np
5 import os
6 import tempfile
7
8 app = Flask(__name__, static_folder='static')
9 model = load_model('Udang-Udang-100.0.h5')
10 class_names = ['Udang kurang segar', 'Udang segar', 'Udang
tidak segar']
11
12 def preprocess_image(image_path):
13     img = image.load_img(image_path, target_size=(224, 224))
14     img = image.img_to_array(img)
15     img = np.expand_dims(img, axis=0)
16     img = img / 255.0
17     return img
18
19 def predict_image(image_path):
20     img = preprocess_image(image_path)
21     predictions = model.predict(img)
22     predicted_class = np.argmax(predictions[0])
23     class_label = class_names[predicted_class]
24     return class_label
25
26 @app.route('/')
27 def home():
28     return render_template('index.html', prediction=None)
29
30 import os
31 import shutil
32
33 import os
34 import shutil
35
36 # ...
37 @app.route('/predict', methods=['POST'])

```

Fig 17: Flask Module

3.4.2.3 Home Page

After running the model and knowing the accuracy value both on training data and on testing data, the next step is to make predictions using data that has never been trained and tested before. To make predictions, a python library is used, namely Flask for WEB creation. Based on Figure 18 Implementation of the main form before the image is inserted, in the figure the flow is that the image is inserted using the insert button. When the image is inserted, a condition check will be performed whether the image is png, jpg, or jpeg format, otherwise the system cannot accept the image.

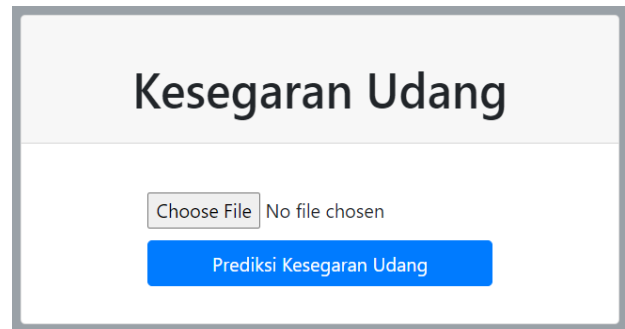


Fig 18: Home Page

If all the conditions have been met, then it can be seen in Figure 19 which is the implementation of the main form after entering the image. All images that appear will be resized so that the size when displayed matches the frame that has been created. Next is to detect the shape of the face by clicking the Shrimp Freshness Prediction button where when the button is clicked it will display the prediction results.

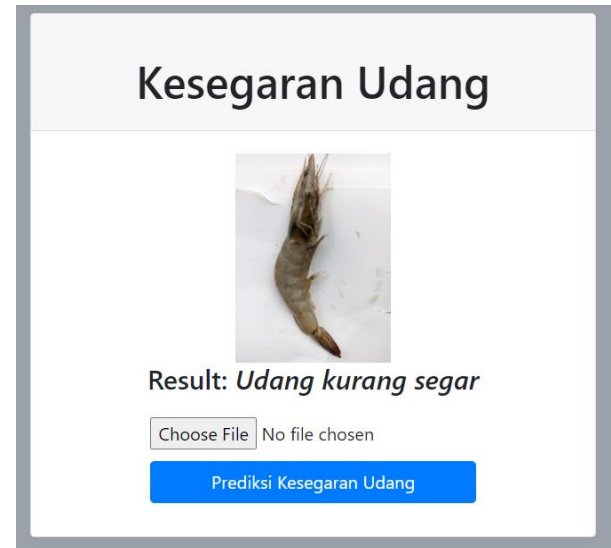


Fig 19: Result after Image Insertion

3.5 Discussion of Result

This stage will test the system results with different parameters. The parameter to be compared is the learning rate. The learning rate tested is divided into 3 namely 0.001, 0.01, and 0.1. The comparison results that will be displayed are accuracy evaluation, image classification, accuracy and loss visualization. The following is a table of comparison results with different learning rates.

Table 2. Comparison Results

| Architecture | Variations | Accuracy | Learning rate |
|--------------|------------|----------|---------------|
| From Scratch | 1 | 88.89 % | 0,001 |
| | 2 | 98.77 % | 0,01 |
| | 3 | 87.50 % | 0,1 |
| Xception | 1 | 99.39 % | 0,001 |
| | 2 | 98.17 % | 0,01 |
| | 3 | 60.49 % | 0,1 |
| NASNetMobile | 1 | 97.53 % | 0,001 |
| | 2 | 96.58 % | 0,01 |
| | 3 | 64.20 % | 0,1 |
| VGG16 | 1 | 91.36 % | 0,001 |
| | 2 | 81.48 % | 0,01 |
| | 3 | 71.60 % | 0,1 |

In Table 2, the comparison results get the best accuracy for learning rate 0.001 with correct classification results for all three classes. For learning rate 0.01 and 0.01 also get good accuracy but, for the less fresh shrimp class is classified as not fresh shrimp. The difference is in the visualization of accuracy and loss.

4. CONCLUSION

Based on the results of the research, a classification system of shrimp freshness level has been made with three classes, namely fresh shrimp, less fresh shrimp, and not fresh shrimp. The freshness level of shrimp in this study is characterized by color changes. Fresh shrimp is bright transparent in color while less fresh shrimp is pale white and there are red and black spots on the head of the shrimp and finally not fresh shrimp is reddish in color.

System design using convolutional neural network model Xception method Xception in this second variation gets the best accuracy of 99.39% by using a learning rate of 0.001 and max epoch 100 with the results of the classification of the three classes with the tested image is correct.

5. REFERENCES

[1] N. D. Singh, M. Krishnan, N. Sivaramane, R. V. and V. R. Kiresur, "Market integration and price transmission in Indian shrimp exports," *Aquaculture*, vol. 561, 2022.

[2] F. Zhan, Z. Li, D. Pan, S. Benjakul, X. Li and B. Zhang, "Investigating the migration hypothesis: Effects of trypsin-like protease on the quality of muscle proteins of red shrimp (*Solenocera crassicornis*) during cold storage," *Food Chemistry: X*, vol. 20, 2023.

[3] Y. Zhou, L. Jiao, J. Wu, Y. Zhang, Q. Zhu and D. Dong, "Non-destructive and in-situ detection of shrimp freshness

using mid-infrared fiber-optic evanescent wave spectroscopy," *Food Chemistry*, vol. 422, 2023.

[4] S. B. Hashim and et. al, "Enhancement of a hybrid colorimetric film incorporating *Origanum compactum* essential oil as antibacterial and monitor chicken breast and shrimp freshness," *Food Chemistry*, vol. 432, 2024.

[5] E. I. Sela and A. Harjoko, "Deteksi Dan Identifikasi Ukuran Obyek Abnormal (Studi Kasus : Citra Otak Manusia)," *Seminar Nasional Informatika (SEMNASIF)*, vol. 1, no. 1, 2011.

[6] K. Wang, C. Zhang, R. Wang and X. Ding, "Quality non-destructive diagnosis of red shrimp based on image processing," *Journal of Food Engineering*, vol. 357, 2023.

[7] A. Jahedsaravani, M. Massinaei and M. Zarie, "Measurement of bubble size and froth velocity using convolutional neural networks," *Minerals Engineering*, vol. 204, 2023.

[8] W. Chen and M. Li, "Standardized motion detection and real time heart rate monitoring of aerobics training based on convolution neural network," *Preventive Medicine*, vol. 174, 2023.

[9] E. Oluwasakin, "Minimization of high computational cost in data preprocessing and modeling using MPI4Py," *Machine Learning with Applications*, vol. 13, 2023.

[10] J. Mao, Y. Zhu, M. Chen, G. Chen, C. Yan and D. Liu, "A contradiction solving method for complex product conceptual design based on deep learning and technological evolution patterns," *Advanced Engineering Informatics*, vol. 55, 2023.

[11] A. E. Zimmermann, E. E. King and D. D. Bode, "Effectiveness and Utility of Flowcharts on Learning in a Classroom Setting: A Mixed-Methods Study," *American Journal of Pharmaceutical Education*, 2023.

[12] J. Bascunana, S. Leon, M. Gonzales-Miquel, E. J. Gonzalez and J. Ramirez, "Impact of Jupyter Notebook as a tool to enhance the learning process in chemical engineering modules," *Education for Chemical Engineers*, vol. 44, pp. 155-163, 2023.

[13] D. J. Clarke, "Apyters: Turning Jupyter Notebooks into data-driven web apps," *Patterns*, vol. 2, no. 3, 2021.

[14] A. J. Cerveira, "Automating behavioral analysis in neuroscience: Development of an open-source python software for more consistent and reliable results," *Journal of Neuroscience Methods*, vol. 398, 2023.

[15] P. Jalili, A. Shateri, A. M. Ganji, B. Jalili and D. D. Ganji, "Analytical analyzing mixed convection flow of nanofluid in a vertical channel using python approach," *Result in Physics*, vol. 52, 2023.