BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Penyajian Data

Pada bab ini akan dijelaskan hasil penelitian tentang klasifikasi ikan tongkol berformalin menggunakan metode K-*Nearest Neighbor* (K-NN) berdasarkan warna dan tekstur dengan menerapkan.Tahapan penelitian yang telah dijelaskan pada bab sebelumnya, dan akan diimplementasikan pada bab ini. Antara lain meliputi hasil dari pengumpulan data set, hasil preprocessing, hasil implementasi ekstraksi warna daging, hasil implementasi tekstur implementasi K-*Nearest Neighbor* (K-NN) dan hasil dari uji coba metode tersebut.

4.1.1 Pengumpulan Data

Pada penelitian ini data gambar yang diambil yaitu data dari ikan tongkol yang masih belum diformalin dan yang sudah di campuri oleh formmalin. Data training terdiri dari1000 gambar, 500 gambar ikan tongkol yang tidak berformalin, dan 500 gambar ikan tongkol berformalin. Dari 1000 data gambar tersebut, dibagi menjadi data training dan data testing. 800 untuk data testing dan 200 untuk data training. Data hasil pengambilan gambar sebagai contoh data pelatihan ditunjukkan pada **Gambar 4.1.**

a. Tidak berformalin b. Berformalin

Gambar 4.1. Contoh Data Penelitian

4.1.2 *Prepocesing*

Pada tahap *preprocessing* data gambar ikan tongkol yang sudah di ambil, di *crooping* terlebih dahulu di ambil bagian dagingnya saja, dikarenakan yang akan diteliti hanya bagian daging ikan tongkol. Gambar yang telah di *cropping* akan di *grayscale*, kemudian hasil dari *grayscale* di resize pada sistem. Dan ukuran semua gambar disamakan menjadi 700x700, agar nantinya sistem dapat dideteksi ketika diproses pada sistem *GLCM (Gray Level Co-Occurrence Matrix)*.berikut adalah langkah-labngkah *prepocesing*.

a. *Cropping*

Cropping adalah penghapusan bagian sudut dari suatu gambar untuk memotong/mengambil/mengeluarkan sebagian isi dari gambar guna memperoleh hasil yang diinginkan. Microsoft Paint adalah program bawaan system operasi windows dan digunakan sebagai pembantu proses Cropping.Pada gambar 4.2 akan di tunjukkan data gambar hasil dari proses cropping. Pada gambar (a) di tunjukkan data gambar atau citra asli, dan pada gambar (b) yaitu data gambar hasil cropping.

(a) Data gambar asli (b) Hasil *Cropping*

Gambar 4.2. Hasil Cropping

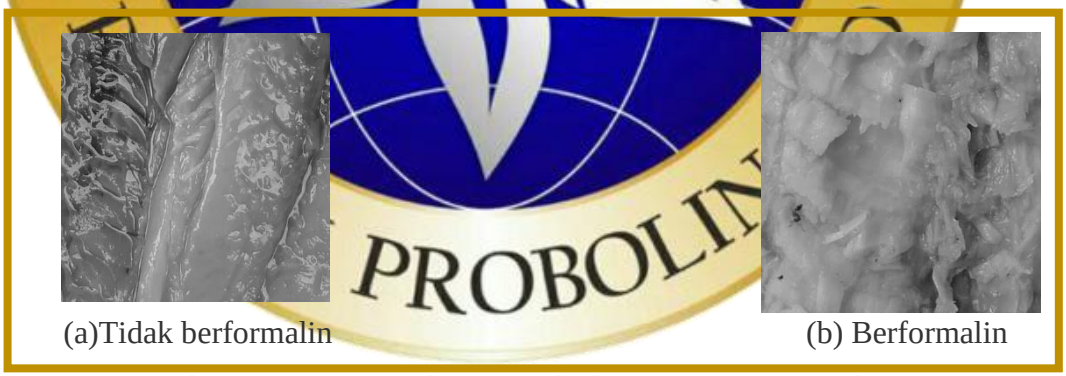
b. *Grayscale*

Proses *Grayscale* yaitu mengubah citra warna asli gambar menjadi warna hitam, putih dan abu-abu. Diubah mengunakan program *python* dan format *save* nya jpg. Berikut segmen program untuk melakukan proses *grayscale*.

Segmen Program 4.1. Proses *Grayscale*

```
import os
import cv2
# Main
root_folder = '/content/drive/My Drive/ikanku/training'
folders = [os.path.join(root_folder, x) for x in ('/content/drive/My
Drive/ikanku/training')]
all_images = [img for folder in folders for img in (folder)]
tgt_base_path = "/content/drive/MyDrive/ikan_abu/training"
for cur_path in os.listdir(root_folder):
    src_sub_path = os.path.join(root_folder, cur_path)
    tgt_sub_path = os.path.join(tgt_base_path, cur_path)
    if not os.path.isdir(tgt_sub_path):
        os.mkdir(tgt_sub_path)
    jdx = 0
    for filename in os.listdir(src_sub_path):
          filepath = os.path.join(src_sub_path, filename)
          img = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
          target_path = os.path.join(tgt_sub_path, "%s-
%03d.jpg" % (cur_path, jdx+1))
          print(target_path)
          cv2.imwrite(target_path, img)
          jdx += 1
```

Pada gambar 4.3 ditunjukkan gambar hasil grayscale dari masing-masing daging ikan.



(a)Tidak berformalin (b) Berformalin

Gambar 4.4. Data Hasil *Grayscale* dari masing-masing Ikan

c. *Resize*

Proses *resize* adalah proses mengubah ukuran pada suatu gambar, disini yang akan dilakukan *resize* yaitu hasil dari gambar *grayscale*, berikut adalah segmen program untuk *resize*. Gambar di*resize* dari 4160*3120 pixel dan sudah melalui cropping dengan program Microsoft Paint menjadi 800x 800 pixel akan diresize menjadi 700x700 pixel. *Script Python* dapat dilihat pada segemen program 4.2.

Segmen Program 4.2. Proses *Resize*

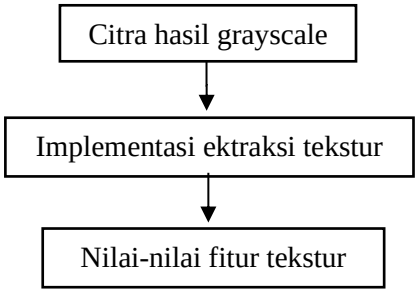
```
repo_url = 'https://github.com/GotG/object_detection_demo_f
           low'
import os
%cd /content
repo_dir_path = os.path.abspath(os.path.join('.', os.path.b
                asename(repo_url)))
!git clone {repo_url}
%cd {repo_dir_path}
!git pull
#mount your google drive.
#it will be visible in the file navigator on the left of th
is notebook
#there should be a folder in your drive with your data
from google.colab import drive
drive.mount('/content/gdrive')
!python resize_images.py --raw-
dir '/content/drive/MyDrive/ikan_abu/training/01' --save-
dir '/content/drive/MyDrive/resize/training/01' --ext jpg -
-target-size "(700,700)"

!python resize_images.py --raw-
dir '/content/drive/MyDrive/ikan_abu/training/02' --save-
dir '/content/drive/MyDrive/resize/training/02' --ext jpg -
-target-size "(700,700)"
```

4.2 Hasil Implementasi

4.2.1 Implementasi Ekstraksi Tekstur

Citra hasil *greyscale* akan dilanjutkan pada tahap ekstraksi fitur tekstur. Tujuan dari ekstraksi fitur tekstur adalah untuk mendapatkan nilai dari objek yang akan diteliti. Adapun ekstraksi yang dilakukan pada penelitian ini menggunakan ekstraksi fitur GLCM *(Gray Level Co-Occurrence Matrix)*yang dipakai antara lain nilai *dari energy, contras, homogeneity, dan correlation*. Tahapan ekstraksi fitur tekstur dengan menggunakan GLCM *(Gray Level Co-Occurrence Matrix)*dapat dilihat pada gambar 4.5.



Gambar 4. 5 Tahapan Ekstraksi Fitur Tekstur

Adapun cara untuk mendapatkan nilai tersebut dengan melihat dari beberapa persamaan yang telah diterapkan pada proses pengkodean (*coding*) yang telah dimengerti oleh mesin. Dibawah ini merupakan implementasi ekstraksi fitur tekstur *energy, contras, homogeneity, correlation 0, energy, contras, homogeneity, correlation 45, energy, contras, homogeneity, correlation 90, energy, contras, homogeneity, correlation 135, dan energy, contras, homogeneity, correlation mean*. Berikut *script python* dari ekstraksi fitur tekstur ditunjukkan pada segmen program 4.3, 4.4, dan 4.5 beserta hasil ekstraksi fitur tekstur yang dapat dilihat pada Tabel 4.1.

Segmen Program 4.3. Fitur GLCM

```
import math
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as nm

class Glcm:
    def __init__(self):
        self.result =[]
    def forOder(self,a, b):
        size = 0
        for i in a[0]:
            size=size+1
        for i in range(len(a)):
                for j in range(len(a[i])-1):
                    p=a[i][j]
                    q=a[i][j+1]



                    b[p][q]=b[p][q]+1
        matriksIterasi1=list(map(list,zip(*b)))
       for i in range(len(b)):
```

Segmen Program Lanjutan 4.3. Fitur GLCM

```
                for j in range(len(b)):
                    b[i][j]=b[i][j]+matriksIterasi1[i][j]
#          print("Grayscale Dependenc Mariks 0 derajat ")
#          for i in b:
#              print(i)
        count=0
        for i in b:
            for j in i
                count=count+j
#          print(b)
        return self.normalisasi(b, count)
```

Pada Segmen Program 4.4 berisi *script python* untuk menyiapkan kelas data yang akan disimpan ke file csv dari hasil ekstraksi glcm. Dan fitur-fitur yang digunakan adalah fitur *energy*, *contrast*, *homogeneity*, dan *correlation*.

Segmen Program 4.4 Kelas data hasil output dari GLCM

```
from PIL import Image
import os
from io import BytesIO
import csv
import numpy as nm

class Data:
    def __init__(self, path):
        imageObject  = Image.open(path)
        cropped     = imageObject.crop((15,15,85,85))
        self.citra = nm.array(cropped)
        self.citra = self.citra.transpose(2,0,1).reshape(3,-1)
        self.matriksIterasi = [[0 for i in range(256)]
       for j in range(256)]

    def insertoCsv(self,data):
        row = data;
        with open('datasetglcm.csv', 'a') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        print("succes")
        csvFile.close()
```

Segmen Program 4.5. Proses input Data ikan

```
import glob
x =0
for filename in glob.glob('/content/drive/MyDrive/resize/training/01/
*.jpg'):
    if x <= 500 :
        data = Data(filename)
        glcm = Glcm()
        feture = [filename, "G1"]
        data0 = glcm.forOder(data.citra, data.matriksIterasi)
        countfeture(data0, glcm)
        data45 = glcm.forOder45(data.citra, data.matriksIterasi)
        countfeture(data45, glcm)
        data90 = glcm.forOder90(data.citra, data.matriksIterasi)
        countfeture(data90, glcm)
        data135 = glcm.forOder135(data.citra, data.matriksIterasi)
        countfeture(data135, glcm)
#         count mean of all feature
        for i in range(len(data0)-1):
            for j in range(len(data0[i])-1):
                data0[i][j]=(data0[i][j]+data45[i][j]+data90[i][j]+da
ta135[i][j])/4
        countfeture(data0, glcm)
        data.insertoCsv(feture)
        print(x)
        x=x+1
    else:
         break
```

Pada segmen program 4.5 adalah *script python* untuk menginputkan dataset jenis ikan yang telah di ekstraksi ke glcm, penginputan data dilakukan sebanyak 2 kali, karena ada 2 jenis ikan. *Filename*G1 untuk ikan yang tidak berformalin, dan G2 untuk ikan berformalin. Pada table 4.1 akan di tunjukkan beberapa contoh nilai hasil dari ekstasi tekstur.

Table 4.1. Hasil Ektraksi Tekstur

Fitur	Data 1	Data 2	Data 3	Data4	DST
energy0	0.073409927	0.042811424	0.070666174	0.032158003	…
contras0	0.391709203	0.269466648	0.510123342	0.227300829	…
homogenity0	18.312921	31.24066136	59.13329251	120.061237	…
correlation0	0.93020098	0.919328459	0.877306064	0.873715101	…
energy45	0.073409927	0.042811424	0.070666174	0.032158003	…
contras45	0.391709203	0.269466648	0.510123342	0.227300829	….
homogenity45	18.312921	31.24066136	59.13329251	120.061237	…
correlation45	0.93020098	0.919328459	0.877306064	0.873715101	…
energy90	0.193494536	0.153305489	0.163059436	0.125212823	…
contras90	0.999937936	0.999925463	0.999950018	0.999921161	…
homogenity90	0.001868475	0.003187497	0.006033394	0.012249897	…
correlation90	0.999992878	0.999991768	0.999987482	0.999987116	…
energy135	0.07341467	0.042813842	0.070671131	0.032159669	…
contras135	0.391771273	0.269541192	0.51017333	0.227379675	…
homogenity135	18.31105234	31.23747354	59.1272585	120.0489858	…
correlation135	0.930208101	0.919336689	0.877318584	0.873727987	…
Label	Tidak Berformalin	Berformalin	Tidak Berformalin	Berformalin	…

4.2.2 Implementasi Ekstraksi Warna

Citra asli akan dilanjutkan pada tahap ekstraksi fitur warna. Tujuan dari ekstraksi fitur warna adalah untuk mendapatkan nilai dari objek yang akan diteliti. Adapun ekstraksi yang dilakukan pada penelitian ini menggunakan ekstraksi fitur warna RGB dengan menggunakan perhitungan R,G, dan B. Tahapan ekstraksi fitur warna dapat dilihat pada gambar 4.6.

Gambar 4.6. Tahapan ekstraksi Fitur warna

Cara untuk mendapatkan nilai tersebut dengan melihat dari beberapa persamaan yang telah diterapkan pada proses pengkodean (*coding*) yang telah dimengerti oleh mesin. Dibawah ini merupakan implementasi ekstraksi warna R,

G dan B. Berikut *script python* dari ekstraksi fitur warna ditunjukkan pada segmen program 4.6, beserta hasil ekstraksi fitur warna yang dapat dilihat pada Tabel 4.4.

Segmen Program 4.6. Ektraksi Warna

```
import cv2 as cv
import numpy as np
import os
from imageio import imread

root_folder = '/content/drive/MyDrive/ikanku/training'
folders = [os.path.join(root_folder, x) for x in ('/content/drive/My
Drive/ikanku/training/01, /content/drive/My Drive/ikanku/training/02'
)]
all_images = [img for folder in folders for img in (folder)]

for cur_path in os.listdir(root_folder):
    src_sub_path = os.path.join(root_folder, cur_path)
    jdx = 0
    for filename in os.listdir(src_sub_path):
        filepath = os.path.join(src_sub_path, filename)
        image = cv.imread(filepath, cv.COLOR_RGB2BGR)
        rows, cols, _ = image.shape

        color_B = 0
        color_G = 0
        color_R = 0

        for i in range(rows):
            for j in range(cols):
                k = image[i,j]
                if k[0] > k[1] and k[0] > k[2]:
                    color_B = color_B + 1
                    continue
                if k[1] > k[0] and k[1] > k[2]:
                    color_G = color_G + 1
                    continue
                if k[2] > k[0] and k[2] > k[1]:
                    color_R = color_R + 1


        pix_total = rows * cols
        print(filepath)
        print('Blue:', color_B/pix_total, 'Green:', color_G/pix_total
, 'Red:',  color_R/pix_total)
        print("    ")
```

Beberapa contoh nilai hasil ektraksi warna dari beberapa data bisa dilihat pada table 4.2.

Table 4.2. Hasil Ekstraksi Warna

Fitur	Data 1	Data 2	Data 3	Data4	DST
R	0.815189063	0.948579688	0.696967188	0.938542188	…
G	1.25E-05	0.037976563	0	0.0423875	…
B	0.1765125	0.000435938	0.2961375	0.000290625	…
Label	Tidak Berformalin	Berformalin	Tidak Berformalin	Berformalin	…

4.2.3 Klasifikasi Menggunakan Metode K-Nearest Neighbor (KNN)

Pada bagian ini implementasi metode Klasifikasi Menggunakan Metode K-*Nearest Neighbor* (KNN)dengan membuat fungsi dan script di sublime dengan format python. Dari data hasil nilai ekstraksi tekstur GLCM dan ekstraksi warna yang sudah di convert ke file csv. Proses klasifikasi daging ikan berformalin dengan jumlah keseluruhan 1000 data citra yang sudah di split di segmen program metode K-*Nearest Neighbor* (KNN), 20% dari data tersebut dijadikan data testing. Pada tahap ini akan dijelaskan setiap tahapannya dan hasil dari implementasi K-*Nearest Neighbor* (KNN)serta dengan potongan-potongan coding menggunakan python :

1. Membaca dan memanggil file csv

   Membaca file csv adalah membaca file csv yg berisi hasil data nilaidari ekstraksi tekstur dan warna.*Script python*membaca atau memanggil file csv ditunjukkan pada segmen program 4.7.

   Segmen Program 4.7. Membaca dan memanggil File Csv

```
import numpy as np  #array
import matplotlib.pyplot as plt #plot diagram
import pandas as pd  #membaca file csv
import numpy as np  #array
import matplotlib.pyplot as plt #plot diagram
import pandas as pd  #membaca file csv

# Set and input dataset into KNN
dataset = pd.read_csv("/content/abcd500.csv")
y = dataset['Class']
x = dataset.drop('Class',axis = 1)
x = x.drop('Citra', axis = 1)
i =len(x.columns)
x = x.drop(x.columns[i-1], axis = 1)
y.replace(('G1','G2',),(0,1), inplace=True)#set class
```

2. Split data *testing* dan data *training*

   Dari data gambar asli data mentah 1000 data, data akan di *split* dengan data *training* 80% dan data *testing* 20%.*Script python*untuk *split* data *training* dan data *testing* ditunjukkan pada segmen program 4.8.

   Segmen Program 4.8. Split Data *Training* dan Data *Testing*

```
# split training data and testing data
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=5)
```

3. Menentukan nilai akurasi

   Pada tahap ini menunjukkan hasil akurasi dari yang sudah di split.*Script python*pada sgmen program 4.9 akan menunjukan nilai akurasi dari 20% data *Testing*.

   Segmen Program 4.9. Menampilkan Nilai Akurasi

```
# show the score or akurasi
from sklearn.neighbors import KNeighborsClassifier
#library KNN
clf = KNeighborsClassifier(n_neighbors=1)
clf.fit(X_train, y_train)#training
print(clf.score(X_test, y_test))#testingnya
```

4.3 Evaluasi *Dataset*

Pada tahap ini akan dibahas hasil uji coba menggunakan metodeK-*Nearest Neighbor* (KNN). Uji coba menggunakan 200 data gambar ikan, Dari data yang telah diuji cobakan maka mendapatkan kesimpulan ada 120 data yang benar dan 80 data yang salah danmendapatkan nilai akurai tertinggi 60% pada K=1. Gambar 4.4 menunjukkan hasil akurasi dari perhitungan data benar dan data salah.

Tabel 4.4. Hasil Jumlah Data

Jumlah Data	Data Benar	Data Salah	Nilai Akurasi
200	120	80	60%

$$\text{Akurasi } \frac{120}{200} \times 100$$

Untuk mengetahui penilaian pada folder berapa data yang terdeteksi salah yaitu Pada outputrun program akan terdeteksi nama folder *testing*. Contohnya, hasil akan terdapat nilai *array* [0 ; 0] dan [1 ; 1] yang menandakan jumlah data benar (*incorrect*)yaitu sama dengan nama folder *testing*. Sedangkan jika data salah (*correct*) maka hasil *array* tidak sama dengan nama folder. Contohnya, hasil *run array* bernilai [0 ; 1] atau [1 ; 0]. Pada gambar 4.7 akan ditunjukkan contoh hasil *running* dari data yg benar (*correct*) dan data salah (*incorrect*).

Gambar 4.7. Hasil *Running Correct*dan *Incorrect*

Script python dari hasil *running* testing di atas ditunjukkan pada segmen program 4.8.

Segmen Program 4.10. Prediksi Data *Testing*

```
# predict the testing data
r_test=[]
for i in y_test:
    r_test.append(i)
r_test = np.array(r_test)
print("Class Awal", end= " ; ")
print("Class Prediksi KNN")
for j in range(len(r_test)-1):
    print(r_test[j], end = " ; ")
    print(y_pred[j], end = " => ")
    if(r_test[j]==y_pred[j]):
        print("correct")
    else:
        print("incorrect")
```

Pada table 4.5 akan diberikan contoh 29 data prediksi data*testing* yang benar (*correct*) dan salah (*incorrect*).

Tabel 4.6. Hasil Prediksi Data Testing

No.	Class	Prediksi		Label
1	544	1	Incorrect	Formalin
2	515	1	Incorrect	Formalin
3	193	0	Incorrect	Tidak berformalin
4	11	0	Incorrect	Tidak berformalin
5	279	0	Correct	Tidak berformalin
6	653	1	Incorrect	Formalin
7	643	1	Correct	Formalin
8	763	1	Correct	Formalin
9	198	0	Correct	Tidak berformalin
10	721	1	Correct	Formalin
11	236	0	Correct	Tidak berformalin
12	500	1	Incorrect	Formalin
13	567	1	Incorrect	Formalin
14	979	1	Correct	Formalin
15	969	1	Correct	Formalin
16	906	1	Correct	Formalin
17	60	0	Correct	Tidak berformalin
18	353	0	Incorrect	Tidak berformalin
19	929	1	Incorrect	Formalin
20	336	0	Incorrect	Tidak berformalin
21	817	1	Correct	Formalin
22	283	0	Correct	Tidak berformalin
23	948	1	Correct	Formalin
24	595	1	Incorrect	Formalin
25	737	1	Correct	Formalin
26	947	1	Correct	Formalin
27	3	0	Incorrect	Tidak berformalin
28	756	1	Correct	Formalin
29	573	1	Correct	Formalin

Semakin kecil nilai *error* nya maka akurasinya semakin tinggi dan semakin besar nilai *error* nya maka semakin kecil tingkat akurasi k *value* nya. Pada table 4.7 akan ditunjukkan nilai akurasi dan *eror*akurasi pada k=1 smpai k=10.

Tabel 4.7. Niai Hasil Akurasi

no.	Error	Akurasi
1	0.4,	0.6
2	0.46,	0.54
3	0.45,	0.55
4	0.465,	0.535
5	0.455,	0.545
6	0.445,	0.555
7	0.44,	0.56
8	0.445,	0.555
9	0.43,	0.57
10	0.44,	0.56

Hasil dari penelitian klasifikasi ikan berformalin berdasarkan warna dan tekstur dengan metode K-*Nearest Neighbor* akurasi yang didapatkan dari k=1 sampai k=7yaitu k=1 yaitu 60% sedangkan k=2 54%, k=3 55%, k=4 53%, k=5 54%, k=6 55%, dan k=7 6%. Hasil akurasi proses klasifikasi yang tertinggi dari proses klasifikasi K-NN antara nilai k=1 sampai k=7 terletak pada nilai k=1 dengan nilai akurasi 60%.