4. IMPLEMENTASI SISTEM

Bab ini mencakup implementasi sistem yang dijelaskan pada bab sebelumnya. Penerapan sistem mencakup instalasi, *import library* dan *dependency* yang digunakan dalam penelitian ini. Proses pertama adalah memuat *dataset* pelatihan dan pengujian ke dalam memori. Setelah data berhasil dimuat ke dalam memori, langkah selanjutnya adalah memulai proses pelatihan pada model *CNN*. Kemudian melanjutkan proses pengujian pada kumpulan data pengujian untuk mengetahui keakuratan model yang dilatih.

4.1 Instalasi, *Import Library* dan *Dependency*

Dalam Sub bab membahas mengenai *library* dan *dependency* yang digunakan dalam penelitian ini.

4.1.1 Instalasi *Visual Studio Code*

Langkah pertama yang dilakukan untuk memulai implementasi sistem adalah instalasi *Visual Studio Code*. Berikut cara untuk melakukan instalasi *Visual Studio Code* :

1. Membuka situs resmi di https://code.visualstudio.com/
2. Klik pada bagian *Download* for *Windows* di halaman utama
3. Setelah *file* telah diunduh, jalankan *file* tersebut dan ikuti petunjuk pada instalasi *Visual Studio Code*
4. Buka *Visual Studio Code* dan install *extension python* yang ada di bagian *extensions*
5. Buat *folder* baru sebagai tempat penyimpanan *file python* yang akan dibuat
6. Buat *file* baru dan beri nama *file* dengan format akhiran .py supaya *file* terbaca dalam format *python*

4.1.2 Instalasi *Library*

Library yang digunakan dalam penelitian ini adalah *Streamlit* untuk membuat aplikasi web, *Tensorflow* merupakan *library framework machine learning open source* yang digunakan untuk membuat model *deep learning* dan *Scikit-Learn* untuk memvisualisasikan hasil klasifikasi berupa akurasi. Instalasi *library* dilakukan menggunakan perintah *pip* yang dijalankan di terminal *Visual Studio Code*.

4.2 Hubungan Desain dan Implementasi

Pada sub bab ini bertujuan untuk menjelaskan hubungan antara desain dengan implementasi.

Tabel 4.1

Hubungan Desain dan Implementasi

Desain	Implementasi
Resize Gambar	Proses *resizing* gambar menjadi 224 x 224 *pixel*.
Penghilangan *Noise*	Proses menghilangkan *noise* menggunakan *filter median*.
Segmentasi Gambar	Proses segmentasi gambar untuk memisahkan objek gambar ikan dari latar belakang. Metode yang digunakan adalah *thresholding*.
Normalisasi	Proses normalisasi dengan nilai *pixel* gambar ke rentang [0, 1] untuk mempersiapkan data pelatihan.
Augmentasi Data	Proses augmentasi data dengan melakukan rotasi, pergeseran, zoom dan pembalikan gambar untuk meningkatkan variasi data.
Arsitektur Model *CNN*	Menggunakan arsitektur *CNN* dengan 3 lapisan konvolusi, 2 lapisan *pooling* dan 2 lapisan *fully connected*.

4.3 *Preprocessing*

Preprocessing yang dilakukan adalah dengan melakukan perubahan seperti *resize*, penghilangan *noise*, segmentasi data, normalisasi dan augmentasi data.

Segmen Program 4.1 *Preprocessing*

```
# Fungsi untuk melakukan preprocessing gambar
def preprocess_image(image):
    # Resize gambar menjadi ukuran yang diinginkan (misalnya: 224x224 pixel)
    resized_image = cv2.resize(image, (224, 224))

    # Penghilangan noise menggunakan filter median
    denoised_image = cv2.medianBlur(resized_image, 3)

    # Segmentasi gambar menggunakan thresholding
    gray_image = cv2.cvtColor(denoised_image, cv2.COLOR_BGR2GRAY)
    _, segmented_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

    # Konversi gambar menjadi format RGB dengan menambahkan channel warna
    rgb_image = cv2.cvtColor(segmented_image, cv2.COLOR_GRAY2RGB)

    return rgb_image

# Inisialisasi ImageDataGenerator untuk preprocessing dan augmentasi data pada data latih
train_datagen = ImageDataGenerator(
    rescale=1./255,                # Normalisasi nilai piksel
    rotation_range=40,             # Rentang rotasi gambar (0-40 derajat)
     width_shift_range=0.2,              # Rentang pergeseran horizontal gambar (0-20% lebar
gambar)
    height_shift_range=0.2,        # Rentang pergeseran vertikal gambar (0-20% tinggi gambar)
    shear_range=0.2,               # Rentang shear transformasi gambar (0-20% shear)
    zoom_range=0.2,                # Rentang zoom gambar (0-20% zoom)
    horizontal_flip=True,          # Melakukan flipping horizontal secara acak
    fill_mode='nearest',           # Strategi pengisian piksel yang hilang
    preprocessing_function=preprocess_image  # Fungsi preprocessing gambar
)
```

```
# Inisialisasi ImageDataGenerator untuk preprocessing pada data validasi dan data uji
valid_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# Anda dapat menggunakan objek train_datagen, valid_datagen, dan test_datagen untuk memuat dan mengolah data
# dengan metode flow_from_directory atau flow_from_dataframe sesuai dengan struktur data Anda.
```

4.4 Implementasi Model *CNN*

Pada sub bab ini dibahas mengenai proses implementasi model *CNN*.

Segmen Program 4.2 Model *CNN*

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Inisialisasi model Sequential
model = Sequential()

# Tambahkan layer konvolusi pertama
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))

# Tambahkan layer pooling pertama
model.add(MaxPooling2D((2, 2)))

# Tambahkan layer konvolusi kedua
model.add(Conv2D(64, (3, 3), activation='relu'))

# Tambahkan layer pooling kedua
model.add(MaxPooling2D((2, 2)))

# Tambahkan layer konvolusi ketiga
```

```
model.add(Conv2D(128, (3, 3), activation='relu'))

# Tambahkan layer pooling ketiga
model.add(MaxPooling2D((2, 2)))

# Flatten layer untuk mengubah matriks 3D menjadi vektor 1D
model.add(Flatten())

# Tambahkan fully connected layer pertama
model.add(Dense(128, activation='relu'))

# Tambahkan fully connected layer kedua (output layer)
model.add(Dense(31, activation='softmax'))  # Jumlah kelas = 31 (jumlah spesies ikan)

# Compile model dengan optimizer dan loss function yang sesuai
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Tampilkan ringkasan arsitektur model
model.summary()
```

Setelah Model *CNN* didefinisikan, langkah selanjutnya adalah mengkompilasi model *CNN*.

Segmen Program 4.3 Kompilasi Model *CNN*

```
# Kompilasi model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',  # Loss function untuk klasifikasi kategori
              metrics=['accuracy'])
```

4.5 Implementasi Proses *Training*

Pada sub bab ini dijelaskan mengenai proses *training* menggunakan model *CNN* yang sudah di *load* pada sub bab sebelumnya, Fungsi *training* digunakan untuk melakukan *training* pada model sebanyak 10 *epoch*.

Segmen Program 4.4 Proses *Training*

```
# Train model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // 32,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // 32
)
```

4.6 Implementasi Proses *Testing*

Pada sub bab ini menjelaskan mengenai proses *testing* atau prediksi terhadap data baru. Proses diawali dengan melakukan prediksi pada *testing dataset* dengan menggunakan model yang sudah di *training*. Hasil *output* dari pengujian ini berupa list hasil prediksi spesies ikan dari model yang sudah melalui proses *training*.

Segmen Program 4.5 Hasil Prediksi

```
class_name=['Bangus', 'Big Head Carp', 'Black Spotted Barb', 'Catfish', 'Climbing Perch', 'Fourfinger Threadfin',
        'Freshwater Eel', 'Glass Perchlet', 'Goby', 'Gold Fish', 'Gourami', 'Grass Carp', 'Green Spotted Puffer',
            'Indian Carp', 'Indo-Pacific Tarpon', 'Jaguar Gapote', 'Janitor Fish', 'Knifefish', 'Long-Snouted Pipefish',
       'Mosquito Fish', 'Mudfish', 'Mullet', 'Pangasius', 'Perch', 'Scat Fish', 'Silver Barb', 'Silver Carp',
      'Silver Perch', 'Snakehead', 'Tenpounder', 'Tilapia']
```

```
images, labels = test_generator.next()
preds = model.predict(images)
fig,axes=plt.subplots(nrows=2,ncols=4,figsize=(25,25))
dic={i:ax for i,ax in enumerate(axes.flat)}
for i in range(0,8):
   label = np.argmax(labels[i])
   pred = np.argmax(preds[i])
   image = images[i]
     dic[i].set_title("real label: " + str(class_name[label]) + " v.s " + "predictedd lable: " +
str(class_name[pred]))
   dic[i].imshow(image)
plt.tight_layout()
plt.show()
```

Setelah model selesai melakukan prediksi terhadap *testing dataset*, langkah selanjutnya adalah untuk menghitung akurasi. *Output* dari program nantinya berupa akurasi dari prediksi yang telah dilakukan.

Segmen Program 4.6 Hasil Akurasi

```
results = model.evaluate(test_generator, verbose=1)



print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

4.7 Implementasi Program

Pada sub bab ini akan dibahas mengenai proses dari pembuatan implementasi program.

Segmen Program 4.7 Implementasi Program

```
import streamlit as st
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt

# Load the Keras model and define class names
model = load_model('saved_modellnew', compile=False)
class_name = ['Bangus', 'Big Head Carp', 'Black Spotted Barb', 'Catfish', 'Climbing Perch', 'Fourfinger Threadfin',
              'Freshwater Eel', 'Glass Perchlet', 'Goby', 'Gold Fish', 'Gourami', 'Grass Carp', 'Green Spotted Puffer',
                   'Indian Carp', 'Indo-Pacific Tarpon', 'Jaguar Gapote', 'Janitor Fish', 'Knifefish', 'Long-Snouted Pipefish',
                'Mosquito Fish', 'Mudfish', 'Mullet', 'Pangasius', 'Perch', 'Scat Fish', 'Silver Barb', 'Silver Carp',
           'Silver Perch', 'Snakehead', 'Tenpounder', 'Tilapia']

# Function to make predictions
def predict(path):
    img = load_img(path, target_size=(224, 224, 3))  # Convert image size and declare the kernel
   img = img_to_array(img)  # Convert the image to an image array
   img = img / 255  # RGB is 255
   img = np.expand_dims(img, [0])
   answer = model.predict(img)
   x = list(np.argsort(answer[0])[::-1][:5])

   result = []
   for i in x:
      result.append(f"{class_name[i]}: {float(answer[0][i]) * 100:.2f}%")
```

```
    y_class = answer.argmax(axis=-1)
    y = " ".join(str(x) for x in y_class)
    y = int(y)
    res = class_name[y]

    return res, result

# Streamlit App
st.title("Fish Classifier App")
st.write("Upload an image of a fish to classify its species.")

uploaded_file = st.file_uploader("Choose a fish image...", type="jpg")

if uploaded_file is not None:
    # Display the uploaded image
    st.image(uploaded_file, caption="Uploaded Image.", use_column_width=True)

    # Make predictions when the user clicks the "Predict" button
    if st.button("Predict"):
        result_class, top_predictions = predict(uploaded_file)

        # Display the results
        st.success(f"Prediction: {result_class}")
        st.write("Top Predictions:")
        for pred in top_predictions:
            st.write(pred)
```