PENGEMBANGAN MODEL KLASIFIKASI MONKEYPOX MENGGUNAKAN CNN TRANSFER LEARNING VGG16 DENGAN OPTIMIZER ADAM DAN AKSELERASI CUDA

Imam Afandy 21081010290

RUMUSAN MASALAH

- Bagaimana meningkatkan akurasi dan efisiensi deteksi monkeypox menggunakan transfer learning?
- Bagaimana penggunaan arsitektur VGG16 dan akselerasi CUDA dapat membantu mengatasi keterbatasan data dan waktu komputasi?

RESEARCH GAP

- Sebagian besar penelitian sebelumnya fokus pada penyakit kulit umum dan penyakit infeksi lain, tetapi masih sedikit penelitian tentang klasifikasi otomatis monkeypox menggunakan deep learning.
- kurangnya dataset besar dan model khusus yang dipotimalkan untuk klasifikasi monkeypox.

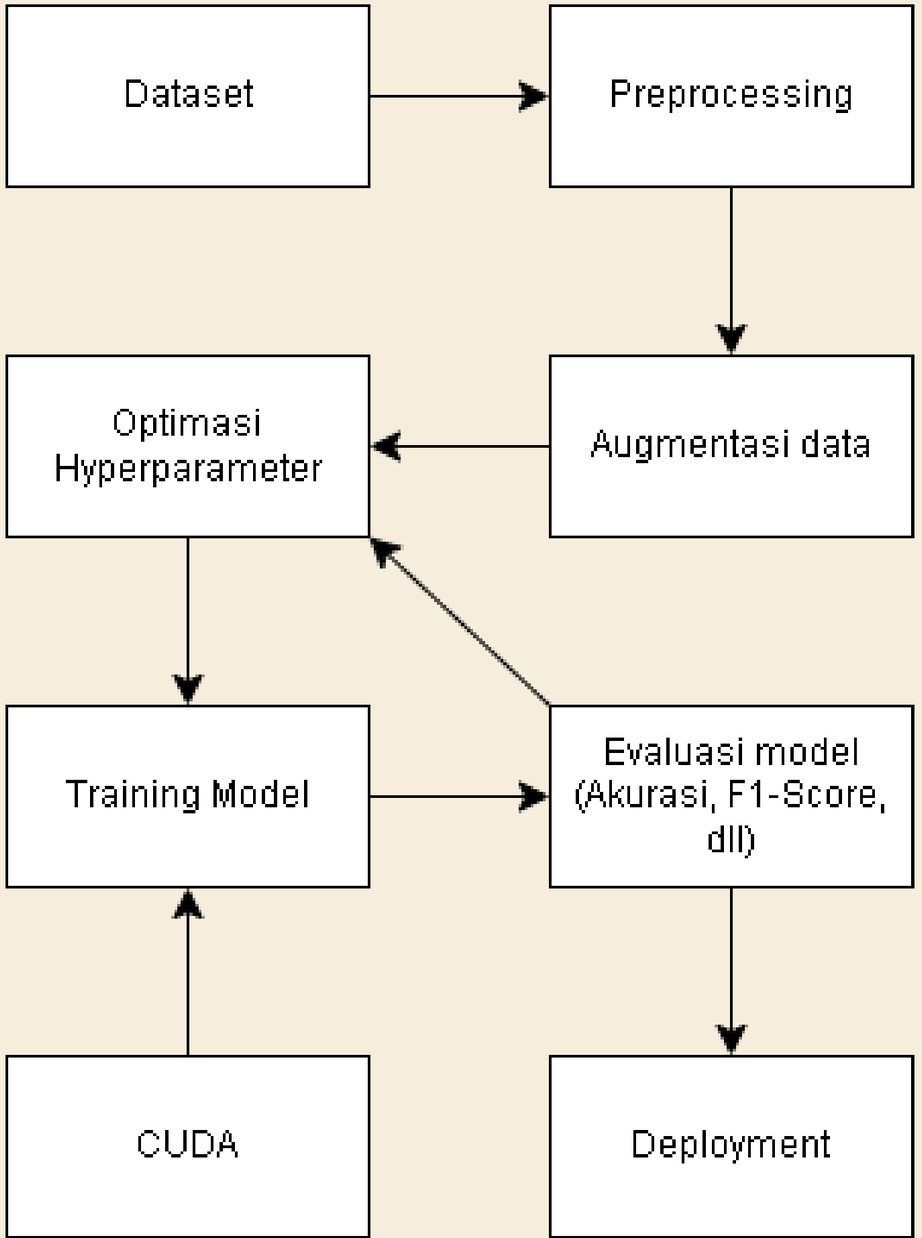
MIND MAPPING REFERENSI

Monkeypox Detection Using CNN with Transfer Learning (Murat Altun 1 , Hüseyin Gürüler 1 , Osman Özkaraca 1 , Faheem Khan 2,* , Jawad Khan 3
and Youngmoon Lee 3,*
1)

1. Dataset : Gambar lesi kulit dari dataset penelitian terdahulu
2. Model : Pre-Trained VGG16 sebagai feature extractor
3. Optimizer : Adam untuk efisiensi dan akurasi
4. Implementasi CUDA : Mempercepat proses training dan validasi

- Output : Model Klasifikasi dengan metrik seperti akurasi, F1-Score, dan waktu komputasi.

Metodologi


Dataset
Preprocessing
Augmentasi data
Optimasi Hyperparameter
Training Model
Evaluasi model (Akurasi, F1-Score, dll)
CUDA
Deployment

```python
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
from tensorflow.keras.models import load_model
import numpy as np

# Muat model
model = load_model('Model/my_model.h5')

# Fungsi untuk memprediksi gambar
def predict_image(img_path):
    img = Image.open(img_path)
    img = img.resize((224, 224))
    img_array = np.array(img, dtype=np.float32)  # Ubah tipe data ke float32
    img_array = np.expand_dims(img_array, axis=0)  # Menambahkan dimensi batch
    img_array /= 255.0  # Normalisasi

    # Memprediksi
    prediction = model.predict(img_array)
    if prediction[0] > 0.5:
        return "Monkeypox detected"
    else:
        return "Normal Skin"

# Fungsi untuk menangani klik tombol
def upload_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png")])
    if file_path:
        result = predict_image(file_path)
        result_label.config(text=result)

        # Tampilkan gambar
        try:
            img = Image.open(file_path)
            img = img.convert('RGB')  # Pastikan gambar dalam format RGB
            img = img.resize((150, 150))
            img_tk = ImageTk.PhotoImage(img)
            img_label.config(image=img_tk)
            img_label.image = img_tk
        except Exception as e:
            result_label.config(text=f"Error loading image: {e}")

# Membuat UI
root = tk.Tk()
root.title("Monkeypox Detector")

upload_button = tk.Button(root, text="Upload Image", command=upload_image)
upload_button.pack(pady=20)

img_label = tk.Label(root)
img_label.pack(pady=10)

result_label = tk.Label(root, text="", font=("Helvetica", 16))
result_label.pack(pady=10)

root.mainloop()

```

```python
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# Augmentasi data lebih lanjut
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

# Memuat data training dan validasi
train_data = datagen.flow_from_directory(
    'G:/My Drive/SKRIPSI/2/Dataset/Monkeypox Skin Image Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training'


validation_data = datagen.flow_from_directory(
    'G:/My Drive/SKRIPSI/2/Dataset/Monkeypox Skin Image Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

# Load VGG16 tanpa fully connected layer teratas
vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Membekukan layer VGG16
for layer in vgg16_model.layers:
    layer.trainable = False

# Membuat model Sequential dengan dropout
model = Sequential()
model.add(vgg16_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))  # Dropout layer untuk regularisasi
model.add(Dense(1, activation='sigmoid'))

# Compile model dengan learning rate yang lebih kecil
model.compile(optimizer=Adam(lr=0.00001),
              loss='binary_crossentropy',
              metrics=["accuracy"])

# Train model
history = model.fit(
    train_data,
    epochs=50,
    validation_data=validation_data
)

# Evaluasi model
val_loss, val_acc = model.evaluate(validation_data)
print(f"Validation accuracy: {val_acc:.4f}")

# Plotting Akurasi dan Loss
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(1, len(acc) + 1)

plt.figure(figsize=(12, 6))

# Plot Akurasi
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')

plt.show()

# Menyimpan model
model.save('Model/my_model.h5')

# Menambahkan confusion matrix
# Mengambil prediksi dari model
validation_labels = validation_data.classes
validation_predictions = (model.predict(validation_data) > 0.5).astype("int32")

# Menghitung confusion matrix
cm = confusion_matrix(validation_labels, validation_predictions)

# Menampilkan confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=validation_data.class_indices.keys())
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()

```

PROGRESS PENGERJAAN

THANK YOU