Laporan Pengolahan Citra dan Video
Deteksi Kartu Remi

Dibuat oleh :

Nama : Jihad Amal Farid
NRP : 5024211072
Kelas : Pengolahan Citra dan Video A

INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2023

Daftar Isi

1 Pendahuluan

1.1 Latar Belakang

Pada tugas akhir mata kuliah Pengolahan Citra dan video yaitu membuat deteksi pada kartu, tidak hanya deteksi kartu saja, selain itu kartu yang dideteksi dibuatkan menjadi permainan. Permainan ini adalah implementasi sederhana dari permainan kartu di mana pemain bersaing dengan bot untuk mencapai total poin sebesar mungkin tanpa melebihi 21. Sistem deteksi kartu menggunakan model neural network untuk mengenali nilai dan jenis kartu dari input kamera.

1.2 Tujuan

Tujuan dari projek ini adalah untuk menciptakan sebuah program yang dapat mengenali jenis dan nilai kartu remi yang ditampilkan melalui kamera. Program ini memanfaatkan model machine learning untuk deteksi kartu dan memberikan pengalaman bermain kartu yang interaktif.

2 Metodologi

2.1 Model Neural Network:

- Menggunakan Keras untuk membangun dan melatih model neural network.
- Model ini dilatih untuk mengenali nilai dan jenis kartu menggunakan dataset yang sesuai.

2.2 Deteksi Kartu:

- Menggunakan OpenCV untuk mengambil frame dari kamera.
- Menentukan Region of Interest (ROI) pada frame untuk mendeteksi kartu.
- Mengubah ukuran dan normalisasi ROI untuk persiapan prediksi menggunakan model.

2.3 Permainan Kartu:

- Implementasi permainan kartu sederhana antara pemain dan bot.
- Pemain memilih untuk ”stand” atau ”hit” menggunakan tombol keyboard.
- Bot memilih ”hit” atau ”stand” berdasarkan logika tertentu.

3 Implementasi kode

Pada projek ini saya menggunakan 3 files yaitu file untuk dataset, training, dan deteksi kartu. berikut ada implementasi kodenya:

3.1 Dataset

Berikut adalah file pemograman dari dataset

```
import cv2
import numpy as np
import time
import os

# Inisialisasi kamera
cap = cv2.VideoCapture(0)  # Ganti angka 0 dengan indeks kamera jika memiliki lebih dari satu kamera

j = 1

# Tentukan ranks dan suits untuk setiap putaran
ranks = [str(i) for i in range(2, 11)] + ["jack", "queen", "king", "ace"]
suits = ["diamonds", "hearts", "spades", "clubs"]

# Pilih indeks peringkat dan jenis kartu awal
current_rank_index = 0
current_suit_index = 0
# Ukuran kartu remi (lebar x tinggi)
card_width, card_height = 240, 360

# Menghitung koordinat untuk menempatkan kotak di tengah frame
frame_width, frame_height = int(cap.get(3)), int(cap.get(4))
roi_x = (frame_width - card_width) // 2
roi_y = (frame_height - card_height) // 2

# Loop untuk mengambil gambar dari kamera
while True:
    # Ambil frame dari kamera
    ret, frame = cap.read()

    # Pilih ROI (Region of Interest) pada tengah frame
    roi = frame[roi_y:roi_y + card_height, roi_x:roi_x + card_width]

    # Konversi ROI ke skala abu-abu dan ukuran yang lebih kecil
    roi_gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    roi_resized = cv2.resize(roi_gray, (128, 128), interpolation=cv2.INTER_AREA)

    # Tampilkan ROI yang telah diproses
    cv2.imshow("Processed ROI", roi_resized)

    # Duplikat frame untuk menggambar kotak pada frame asli
    copy = frame.copy()

    # Gambar kotak pada frame
    cv2.rectangle(copy, (roi_x, roi_y), (roi_x + card_width, roi_y + card_height), (0, 0, 0), 5)

    # Tampilkan frame
    cv2.imshow("Capture Images - Press 'P' to capture, 'O' to change rank", copy)

    # Tangkap tombol keyboard
    key = cv2.waitKey(1)

    # Tombol 'p' untuk mengambil gambar
    if key == ord('p'):
        rank = ranks[current_rank_index]
        suit = suits[current_suit_index]

        # Nama file gambar
        file_name = f"{rank}_{suit}_{j}.jpg"

        # Cek apakah file dengan nama tersebut sudah ada
        while os.path.exists(os.path.join("dataset", f"{rank}_{suit}", file_name)):
            j += 1
            file_name = f"{rank}_{suit}_{j}.jpg"

```

```
        # Simpan gambar di dalam subfolder
        folder_path = os.path.join("dataset", f"{rank}_{suit}")
        image_path = os.path.join(folder_path, file_name)

        # Buat subfolder jika belum ada
        os.makedirs(folder_path, exist_ok=True)

        cv2.imwrite(image_path, roi_resized)

        print(f"Image {j} captured: {file_name}")

    # Tombol 'o' untuk mengubah peringkat
    elif key == ord('o'):
        # Ganti indeks peringkat dan jenis kartu
        current_rank_index = (current_rank_index + 1) % len(ranks)
        if ranks[current_rank_index] == "2":
            current_suit_index = (current_suit_index + 1) % len(suits)

        j = 1

    # Hentikan program ketika tombol 'q' ditekan
    elif key == ord('q'):
        # Tambahkan satu ke jumlah percobaan setelah break
        break

# Bebaskan sumber daya dan tutup jendela
cap.release()
cv2.destroyAllWindows()
```

berikut adalah penjelasan dari kode ini :

3.1.1 Penamaan Kartu

```
ranks = [str(i) for i in range(2, 11)] + ["jack", "queen", "king", "ace"]
suits = ["diamonds", "hearts", "spades", "clubs"]
```

Untuk penamaan kartu diawali dari inisiasi variabel Ranks dan Suits dimana rank kartu adalah 2-10,jack,queen,king, dan ace sedangkan suits adalah lambang dari kartunya seperti diamonds,clubs(clover),hearts, dan spades.

3.1.2 Pemilihan Region Of Interest

```
    # Pilih ROI (Region of Interest) pada tengah frame
    roi = frame[roi_y:roi_y + card_height, roi_x:roi_x + card_width]

    # Konversi ROI ke skala abu-abu dan ukuran yang lebih kecil
    roi_gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    roi_resized = cv2.resize(roi_gray, (128, 128), interpolation=cv2.INTER_AREA)

    # Tampilkan ROI yang telah diproses
    cv2.imshow("Processed ROI", roi_resized)
```

Dari kode diatas adalah untuk pemilihan area yang akan dideteksi kartu, setelah itu gambar diubah menjadi warna abu-abu dan diperkecil gambarnya agar lebih efisien

3.2 Pengambilan dan penyimpanan file Gambar

```
    # Tombol 'p' untuk mengambil gambar
    if key == ord('p'):
        rank = ranks[current_rank_index]
        suit = suits[current_suit_index]

        # Nama file gambar
        file_name = f"{rank}_{suit}_{j}.jpg"

        # Cek apakah file dengan nama tersebut sudah ada
        while os.path.exists(os.path.join("dataset", f"{rank}_{suit}", file_name)):
            j += 1
            file_name = f"{rank}_{suit}_{j}.jpg"

        # Simpan gambar di dalam subfolder
        folder_path = os.path.join("dataset", f"{rank}_{suit}")
        image_path = os.path.join(folder_path, file_name)

```

```
        # Buat subfolder jika belum ada
        os.makedirs(folder_path, exist_ok=True)

        cv2.imwrite(image_path, roi_resized)

        print(f"Image {j} captured: {file_name}")

    # Tombol 'o' untuk mengubah peringkat
    elif key == ord('o'):
        # Ganti indeks peringkat dan jenis kartu
        current_rank_index = (current_rank_index + 1) % len(ranks)
        if ranks[current_rank_index] == "2":
            current_suit_index = (current_suit_index + 1) % len(suits)

        j = 1
```

Untuk pengambilan gambar pada kartu agar dapat dijadikan dataset yaitu menekan tombol P dan O untuk pengubahan rank, jadi sistemnya adalah diawali dari ranknya 2 dan suitnya adalah diamond. Saat itu kita mengambil gambar 2 diamond dengan menekan tombol P, jika sudah cukup pengambilan gambarnya tekan tombol O untuk mengubah rank dan menjadi 3 diamond. jika diteruskan maka 2-10, jack queen, king, ace diamond, setelah ace diamond lalu kita menekan tombol P maka ranknya kembali menjadi 2 tetapi suitsnya berubah menjadi hearts. Urutan perubahannya berdasarkan line 12-13. Gambar yang sudah diambil akan otomatis tersimpan dalam file dengan nama file yang sesuai rank dan suit kartunya.

3.3 Training

Berikut adalah file pemograman dari training

```
import os
import cv2
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten,MaxPooling2D,Input


def LoadCitraTraining(sDir,LabelKelas):
  JumlahKelas=len(LabelKelas)
  TargetKelas = np.eye(JumlahKelas)
  # Menyiapkan variabel list untuk data menampung citra dan data target
  X=[]#Menampung Data Citra
  T=[]#Menampung Target
  for i in range(len(LabelKelas)):
    #Membaca file citra di setiap direktori data set
    DirKelas = os.path.join(sDir, LabelKelas[i])
    files = os.listdir(DirKelas)
    for f in files:
      ff=f.lower()
      print(f)
      #memilih citra dengan extensi jpg,jpeg,dan png
      if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.endswith('.png')):
         NmFile = os.path.join(DirKelas,f)
         img= np.double(cv2.imread(NmFile,1))
         img=cv2.resize(img,(128,128));
         img= np.asarray(img)/255;
         img=img.astype('float32')
         #Menambahkan citra dan target ke daftar
         X.append(img)
         T.append(TargetKelas[i])
     #--------akhir loop :Pfor f in files-----------------
  #-----akhir  loop :for i in range(len(LabelKelas))----

  #Mengubah List Menjadi numppy array
  X=np.array(X)
  T=np.array(T)
  X=X.astype('float32')
  T=T.astype('float32')
  return X,T

def ModelDeepLearningCNN(JumlahKelas):

    model = Sequential()
    model.add(Input((128,128,3)))
    model.add(Conv2D(32, kernel_size=3, activation='relu',padding="same"))
```

```
    model.add(MaxPooling2D((2, 2), padding='same'))
    model.add(Conv2D(32, kernel_size=3, activation='relu',padding="same"))
    model.add(MaxPooling2D((2, 2), padding='same'))
    model.add(Conv2D(32, kernel_size=3, activation='relu',padding="same"))

    model.add(Flatten())
    model.add(Dense(JumlahKelas, activation='softmax'))


    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    #ModelCNN.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
    return model

def TrainingCNN(JumlahEpoh,DirektoriDataSet,LabelKelas,NamaFileBobot):
    #Membaca Data training dan label Kelas
    X,D=LoadCitraTraining(DirektoriDataSet,LabelKelas)
    JumlahKelas = len(LabelKelas)
    #Membuat Model CNN
    ModelCNN =ModelDeepLearningCNN(JumlahKelas)
    #Trainng
    history=ModelCNN.fit(X, D,epochs=JumlahEpoh,shuffle=True)
    #Menyimpan hasil learning
    ModelCNN.save(NamaFileBobot)
    #Mengembalikan output
    return ModelCNN,history


def Klasifikasi(DirDataSet,DirKlasifikasi,LabelKelas,ModelCNN=[]):
#Menyiapkan Data input Yang akan di kasifikasikan
  X=[]
  ls = []
  DirKelas = DirDataSet+"/"+DirKlasifikasi
  print(DirKelas)
  files = os.listdir(DirKelas)
  n=0
  for f in files:
      ff=f.lower()
      print(f)
      if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.endswith('.png')):
         ls.append(ff)
         NmFile = os.path.join(DirKelas,f)
         img= cv2.imread(NmFile,1)
         img=cv2.resize(img,(128,128))
         img= np.asarray(img)/255
         img=img.astype('float32')
         X.append(img)


  X=np.array(X)
  X=X.astype('float32')
  #Melakukan prediksi Klasifikasi
  hs=ModelCNN.predict(X)

  LKlasifikasi=[]
  LKelasCitra =[]
  n = X.shape[0]
  for i in range(n):
      v=hs[i,:]
      if v.max()>0.5:
          idx = np.max(np.where( v == v.max()))
          LKelasCitra.append(LabelKelas[idx])
      else:
          idx=-1
          LKelasCitra.append("-")
      #------akhir if
      LKlasifikasi.append(idx)
  #----akhir for
  LKlasifikasi = np.array(LKlasifikasi)
  return ls, hs, LKelasCitra


#Menentukan Direktori Yang menyimpan Data set
DirektoriDataSet="dataset"

#Label Data Set
```

```
LabelKelas = []

# Tentukan ranks dan suits untuk setiap putaran
ranks = [str(i) for i in range(2, 11)] + ["jack", "queen", "king", "ace"]
suits = ["diamonds", "hearts", "spades", "clubs"]

# Loop untuk menghasilkan label
for rank in ranks:
    for suit in suits:
        label = f"{rank}_{suit}"
        LabelKelas.append(label)

# Cetak hasil label
print(LabelKelas)

#c. Inisialisasi parameter Training
JumlahEpoh = 12
FileBobot = "DeteksiKartu.h5"
#d. training
ModelCNN,history = TrainingCNN(JumlahEpoh,DirektoriDataSet,LabelKelas ,FileBobot)
ModelCNN.summary()
```

berikut adalah penjelasan dari kode ini :

3.3.1 Fungsi LoadCitraTraining

Membaca dataset citra untuk training. Mengambil citra dari setiap kelas (peringkat dan jenis kartu) dan menyimpannya dalam bentuk array numpy.

3.3.2 Fungsi ModelDeepLearningCNN

Membangun model Convolutional Neural Network (CNN) menggunakan Keras untuk klasifikasi citra. Model ini memiliki lapisan konvolusi, max pooling, dan lapisan fully connected.

3.3.3 Fungsi TrainingCNN

Melakukan training pada model CNN dengan citra-citra yang telah dibaca sebelumnya. Hasil training disimpan dalam file bobot.

3.3.4 Fungsi Klasifikasi

Mengklasifikasikan citra-citra yang terdapat dalam direktori klasifikasi tertentu menggunakan model CNN yang telah dilatih sebelumnya. Hasil klasifikasi berupa label kelas untuk setiap citra.

3.4 Deteksi Kartu

Berikut adalah file pemograman dari deteksi kartu

```
import cv2
import numpy as np
import random
from keras.models import load_model

# Load model
model = load_model("DeteksiKartu.h5")

# Inisialisasi kamera
cap = cv2.VideoCapture(0)  # Ganti angka 0 dengan indeks kamera jika memiliki lebih dari satu kamera
detected_player_frame = np.zeros((400, 400, 3), dtype=np.uint8)
detected_bot_frame = np.zeros((400, 400, 3), dtype=np.uint8)

# Tentukan ranks dan suits untuk setiap putaran
ranks = [str(i) for i in range(2, 11)] + ["jack", "queen", "king", "ace"]
suits = ["diamonds", "hearts", "spades", "clubs"]

# Ukuran kartu remi (lebar x tinggi)
card_width, card_height = 240, 360

# Menghitung koordinat untuk menempatkan kotak di bawah tengah frame
frame_width, frame_height = int(cap.get(3)), int(cap.get(4))
```

```
roi_x = (frame_width - card_width) // 2
roi_y = frame_height - card_height - 50

# Inisialisasi total poin
player_points = 0
bot_points = 0

# Dimulai dengan giliran player
player_turn = random.choice([True, False])
player_stand = False


# List untuk menyimpan kartu yang sudah terdeteksi
detected_cards = set()
last_detected_card = None

# Direktori tempat gambar kartu disimpan
card_images_dir = "PNG/"

def read_card_image(label):
    # Dapatkan nama file gambar kartu sesuai label
    image_path = f"{card_images_dir}{label}.png"
    # Baca gambar kartu
    card_image = cv2.imread(image_path)
    # Pastikan gambar terbaca dengan benar
    if card_image is None:
        raise FileNotFoundError(f"Image not found: {image_path}")
    return card_image


# Loop untuk mengambil gambar dari kamera
while True:
    # Ambil frame dari kamera
    key = cv2.waitKey(1)
    ret, frame = cap.read()
    roi = frame[roi_y:roi_y + card_height, roi_x:roi_x + card_width]
    # Konversi ROI ke ukuran yang lebih kecil
    roi_resized = cv2.resize(roi, (128, 128), interpolation=cv2.INTER_AREA)
    roi_resized = roi_resized.astype("float32") / 255.0
    roi_resized = np.expand_dims(roi_resized, axis=0)

    # Prediksi kelas kartu menggunakan model
    pred = model.predict(roi_resized)

    # Loop untuk menghasilkan label
    LabelKelas = []
    for rank in ranks:
        for suit in suits:
            label = f"{rank}_{suit}"
            LabelKelas.append(label)

    # Tampilkan hasil prediksi
    label = LabelKelas[np.argmax(pred)]
    # Jika prediksi memiliki kepercayaan di atas ambang batas tertentu, tampilkan label
    jika tidak, tampilkan "Unknown"
    confidence_threshold = 0.5
    if np.max(pred) > confidence_threshold:
        # Hitung poin kartu
        rank = label.split("_")[0]
        if rank in ["2", "3", "4", "5", "6", "7", "8", "9"]:
            points = int(rank)
        elif rank in ["10", "jack", "queen", "king"]:
            points = 10
        else:
            points = 11

        # Pemain memilih "stand" atau "hit"
        key = cv2.waitKey(1)
        if player_turn:
            if key == ord('s'):  # Jika tombol 's' ditekan, pemain memilih "stand"
                player_stand = True
            elif key == ord('h'):  # Jika tombol 'h' ditekan, pemain memilih "hit"
                if label not in detected_cards:
                    player_points += points
                    frame_to_display = detected_player_frame
                    detected_cards.add(label)
```

```
                    # Tambahkan tulisan kartu ke frame detected_card_frame
                    card_image = read_card_image(label)
                    resized_card_image = cv2.resize(card_image, (75, 75))
                    detected_player_frame[len(detected_cards) * 20:len(detected_cards) *
20 + resized_card_image.shape[0], 10:10 + resized_card_image.shape[1]] =
resized_card_image
                    player_turn = False

        if not player_turn:
            # Keputusan bot hit atau stand
            if bot_points < 17:
                # Bot memilih "hit" dan secara acak memilih kartu
                random_rank = random.choice(ranks)
                random_suit = random.choice(suits)
                label = f"{random_rank}_{random_suit}"

                # Jika kartu belum terdeteksi sebelumnya, tambahkan poin ke total bot
                if label not in detected_cards:
                    detected_cards.add(label)
                    # Tambahkan tulisan kartu ke frame detected_card_frame
                    card_image = read_card_image(label)
                    resized_card_image = cv2.resize(card_image, (75, 75))
                    detected_bot_frame[len(detected_cards) * 20:len(detected_cards) * 20
+ resized_card_image.shape[0], 10:10 + resized_card_image.shape[1]] =
resized_card_image

                    # Update total poin bot
                    rank = label.split("_")[0]
                    if rank in ["2", "3", "4", "5", "6", "7", "8", "9"]:
                        points = int(rank)
                    elif rank in ["10", "jack", "queen", "king"]:
                        points = 10
                    else:
                        points = 11
                    bot_points += points
                    # Ganti giliran ke pemain setelah bot "hit"
                    player_turn = True
            else:
                player_turn = True

        if player_stand:
            # Keputusan bot hit atau stand
            player_turn = False
            if bot_points < 17:
                # Bot memilih "hit" dan secara acak memilih kartu
                random_rank = random.choice(ranks)
                random_suit = random.choice(suits)
                label = f"{random_rank}_{random_suit}"

                # Jika kartu belum terdeteksi sebelumnya, tambahkan poin ke total bot
                if label not in detected_cards:
                    detected_cards.add(label)
                    # Tambahkan tulisan kartu ke frame detected_card_frame
                    card_image = read_card_image(label)
                    resized_card_image = cv2.resize(card_image, (75, 75))
                    detected_bot_frame[len(detected_cards) * 20:len(detected_cards) * 20
+ resized_card_image.shape[0], 10:10 + resized_card_image.shape[1]] =
resized_card_image

                    # Update total poin bot
                    rank = label.split("_")[0]
                    if rank in ["2", "3", "4", "5", "6", "7", "8", "9"]:
                        points = int(rank)
                    elif rank in ["10", "jack", "queen", "king"]:
                        points = 10
                    else:
                        points = 11
                    bot_points += points
                    # Ganti giliran ke pemain setelah bot "hit"
                    player_turn = True
                else:
                    player_stand = False  # Atur player_stand menjadi False di sini
            else:
                # Tentukan pemenang setelah loop selesai
                if player_points == bot_points or (bot_points > 21 and player_points >
21):
```

```
                        winner = "Draw"
                elif player_points > 21 or (bot_points <= 21 and bot_points > player_points):
                    winner = "Bot"
                elif bot_points > 21 or (player_points <= 21 and player_points > bot_points):
                    winner = "Player"
                else:
                    # Pilih pemenang berdasarkan total poin tertinggi
                    winner = "Player" if player_points > bot_points else "Bot"
                    # Tampilkan hasil permainan pada frame terakhir
                cv2.putText(frame, f"The winner is: {winner}", (220,180), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 3)

        # Update last_detected_card
        last_detected_card = label
        cv2.putText(detected_player_frame, "Player Card:", (10, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
        cv2.putText(detected_bot_frame, "Bot Card:", (10, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
        cv2.putText(frame, f"{label} ({points})", (roi_x + 5, roi_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    else: # jika tidak terdeteksi kartu
        if player_turn:
            if key == ord('s'):  # Jika tombol 's' ditekan, pemain memilih "stand"
                player_stand = True
        if player_stand:
            if bot_points >= 17:
              # Tentukan pemenang setelah loop selesai
                if player_points == bot_points or (bot_points > 21 and player_points > 21):
                        winner = "Draw"
                elif player_points > 21 or (bot_points <= 21 and bot_points > player_points):
                    winner = "Bot"
                elif bot_points > 21 or (player_points <= 21 and player_points > bot_points):
                    winner = "Player"
                else:
                    # Pilih pemenang berdasarkan total poin tertinggi
                    winner = "Player" if player_points > bot_points else "Bot"
                    # Tampilkan hasil permainan pada frame terakhir
                cv2.putText(frame, f"The winner is: {winner}", (220,180), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 3)

    # Gambar kotak pada frame
    cv2.rectangle(frame, (roi_x, roi_y), (roi_x + card_width, roi_y + card_height), (0, 0, 0), 5)

    # Tampilkan player & bot poin
    cv2.putText(frame, f"Player Points: {player_points}/21", (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    cv2.putText(frame, f"Bot Points: {bot_points}/21", (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    #Tampilan Text Instruksi
    cv2.putText(frame, f"Hit press H and Stand press S", (250, 250), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    # Tampilkan frame
    cv2.imshow("Detected Cards - Player", detected_player_frame)
    cv2.imshow("detection card", frame)
    cv2.imshow("Detected Cards - Bot", detected_bot_frame)


    # Hentikan program ketika tombol 'q' ditekan
    if key == ord('q'):
        break



# Tutup jendela
cap.release()
cv2.destroyAllWindows()
```

berikut adalah penjelasan dari kode ini :

3.4.1 Penamaan Kartu

```
ranks = [str(i) for i in range(2, 11)] + ["jack", "queen", "king", "ace"]
suits = ["diamonds", "hearts", "spades", "clubs"]
```

Untuk penamaan kartu diawali dari inisiasi variabel Ranks dan Suits dimana rank kartu adalah 2-10,jack,queen,king, dan ace sedangkan suits adalah lambang dari kartunya seperti diamonds,clubs(clover),hearts, dan spades.

3.4.2 Deteksi Kartu

```
    roi = frame[roi_y:roi_y + card_height, roi_x:roi_x + card_width]
    # Konversi ROI ke ukuran yang lebih kecil
    roi_resized = cv2.resize(roi, (128, 128), interpolation=cv2.INTER_AREA)
    roi_resized = roi_resized.astype("float32") / 255.0
    roi_resized = np.expand_dims(roi_resized, axis=0)

    # Prediksi kelas kartu menggunakan model
    pred = model.predict(roi_resized)

    # Loop untuk menghasilkan label
    LabelKelas = []
    for rank in ranks:
        for suit in suits:
            label = f"{rank}_{suit}"
            LabelKelas.append(label)

    # Tampilkan hasil prediksi
    label = LabelKelas[np.argmax(pred)]
    # Jika prediksi memiliki kepercayaan di atas ambang batas tertentu, tampilkan label
    jika tidak, tampilkan "Unknown"
    confidence_threshold = 0.5
    if np.max(pred) > confidence_threshold:
        cv2.putText(frame, f"{label} ({points})", (roi_x + 5, roi_y - 10), cv2.
    FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

Untuk pendeteksi kartu, diawali dari pengambilan ROI atau area mana yang akan terdeteksi kartu. Setelah itu ROI diubah ukurannya menjadi kecil dan dinormalisasi lalu memprediksi kelas kartu berdasarkan ROI yang telah diubah ukurannya. Adapun untuk loop label kelas yang digunakan untuk label sebagai penamaan kartunya. Untuk penampilan hasil prediksi kartu adalah berdasarkan label dengan probabilitas tertinggi dari hasil prediksi, jika probabilitas tertinggi melebihi ambang batas tertentu (confidence threshold), label kartu dan nilai poinnya ditampilkan pada frame utama menggunakan cv2.putText.

3.4.3 Permainan Kartu

1. Player

```
        if player_turn:
            if key == ord('s'):  # Jika tombol 's' ditekan, pemain memilih "stand"
                player_stand = True
            elif key == ord('h'):  # Jika tombol 'h' ditekan, pemain memilih "hit"
                if label not in detected_cards:
                    player_points += points
                    frame_to_display = detected_player_frame
                    detected_cards.add(label)
                    # Tambahkan tulisan kartu ke frame detected_card_frame
                    card_image = read_card_image(label)
                    resized_card_image = cv2.resize(card_image, (75, 75))
                    detected_player_frame[len(detected_cards) * 20:len(
    detected_cards) * 20 + resized_card_image.shape[0], 10:10 + resized_card_image.
    shape[1]] = resized_card_image
                    player_turn = False
```

kode ini adalah untuk player (pemain) dimana ketika player menekan tombol S adalah untuk stand dan H adalah untuk Hit. Ketika player hit kartu maka jika kartu tidak berada list detected card player akan mendapatkan dari kartu tersebut dan kartu tersebut akan masuk list detected card yang dimana tidak akan bisa digunakan lagi.

2. Bot

```
        if not player_turn:
            # Keputusan bot hit atau stand
            if bot_points < 17:
                # Bot memilih "hit" dan secara acak memilih kartu
                random_rank = random.choice(ranks)
                random_suit = random.choice(suits)
```

```
                label = f"{random_rank}_{random_suit}"

                # Jika kartu belum terdeteksi sebelumnya, tambahkan poin ke total bot
                if label not in detected_cards:
                    detected_cards.add(label)
                    # Tambahkan tulisan kartu ke frame detected_card_frame
                    card_image = read_card_image(label)
                    resized_card_image = cv2.resize(card_image, (75, 75))
                    detected_bot_frame[len(detected_cards) * 20:len(detected_cards) * 20 + resized_card_image.shape[0], 10:10 + resized_card_image.shape[1]] = resized_card_image

                    # Update total poin bot
                    rank = label.split("_")[0]
                    if rank in ["2", "3", "4", "5", "6", "7", "8", "9"]:
                        points = int(rank)
                    elif rank in ["10", "jack", "queen", "king"]:
                        points = 10
                    else:
                        points = 11
                    bot_points += points
                    # Ganti giliran ke pemain setelah bot "hit"
                    player_turn = True
            else:
                player_turn = True
```

Kode ini adalah untuk bot, dalam permainannya bot akan memilih kartu secara acak jika point bot masih dibawah 17.

3. Point

```
player_points = 0
bot_points = 0
        # Hitung poin kartu
        rank = label.split("_")[0]
        if rank in ["2", "3", "4", "5", "6", "7", "8", "9"]:
            points = int(rank)
        elif rank in ["10", "jack", "queen", "king"]:
            points = 10
        else:
            points = 11
```

kedua point bot dan player diawali dengan nilai 0, kode diatas ini adalah point dari kartu dimana poin kartu 2-10 yaitu sesuai ranknya, jack,queen,king poinnya 10, dan ace poinnya 11.

```
                if player_points == bot_points or (bot_points > 21 and player_points > 21):
                        winner = "Draw"
                elif player_points > 21 or (bot_points <= 21 and bot_points > player_points):
                    winner = "Bot"
                elif bot_points > 21 or (player_points <= 21 and player_points > bot_points):
                    winner = "Player"
                else:
                    # Pilih pemenang berdasarkan total poin tertinggi
                    winner = "Player" if player_points > bot_points else "Bot"
                    # Tampilkan hasil permainan pada frame terakhir
                cv2.putText(frame, f"The winner is: {winner}", (220,180), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 3)
```

kode diatas ini adalah untuk menentukan pemenangnya siapa, jika poin player dan bot sama atau poin player dan bot keduanya melebihi 21 maka hasilnya draw, jika poin player melebihi 21 atau poin bot kurang dari 21 dan lebih besar dari player maka pemenangnya adalah bot, jika poin bot melebihi 21 atau poin player kurang dari 21 dan lebih besar dari bot maka pemenangnya adalah player.

4 Hasil

Hasil dari deteksi kartu ini dapat dimainkan dengan baik berikut adalah hasilnya :

4.1 Hasil Dataset

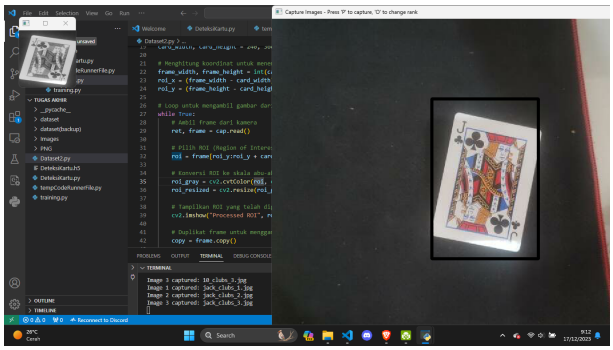

Figure 1: Ketika pengambilan Dataset

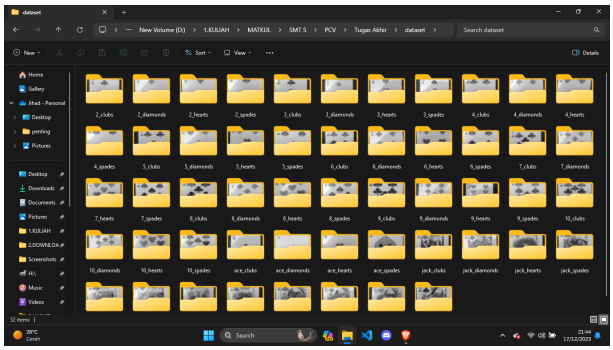

Figure 2: Hasil Dataset

4.2 Hasil Training

Hasil Training adalah berupa file dengan format .h5 yang nantinya akan diload model di deteksi kartu

4.3 Hasil Deteksi Kartu

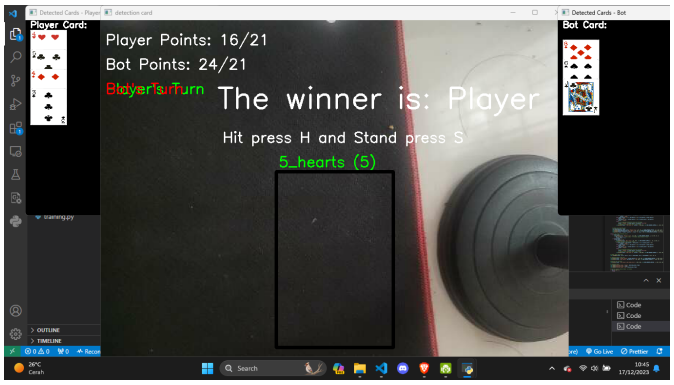

Figure 3: Ketika permainan dimenangkan player

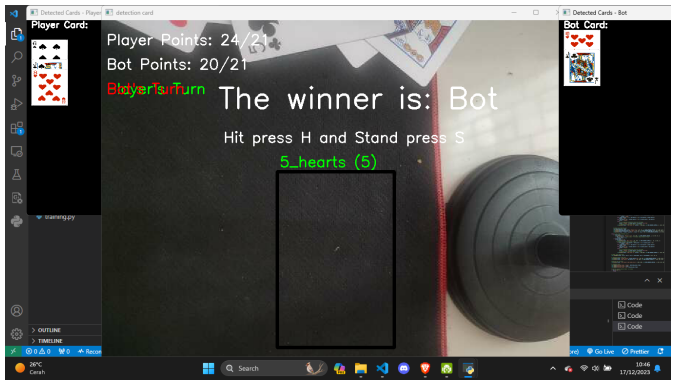

Figure 4: Ketika permainan dimenangkan player

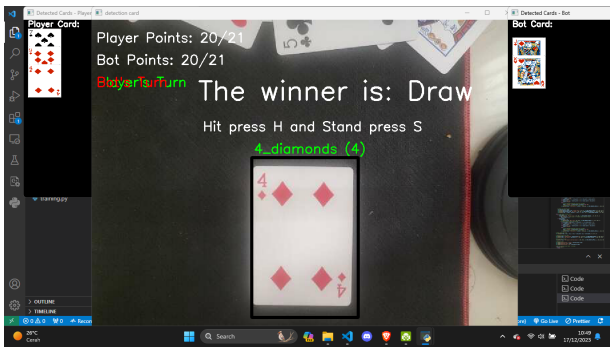

Figure 5: Ketika permainan seri

5 Kesimpulan

Projek deteksi kartu yang saya buat terdapat 3 file pemograman yaitu dataset, training, dan deteksi kartu.

1. Dataset : Ini berfungsi untuk pengambilan gambar yang gambarnya dikonversi ke warna abu-abu dan ukuran gambar diperkecil. Nantinya gambar yang sudah diambil akan digunakan untuk training.

2. Training : Training ini digunakan untuk melatih dan menerapkan model Convolutional Neural Network (CNN) dalam konteks deteksi dan klasifikasi citra

3. Deteksi Kartu : Setelah ditraining, deteksi kartu ini akan load model yang sudah ditraining oleh training agar hasil kartu yang ditraining dapat dideteksi

4. Permainan Blackjack : Cara kerja permainan ini yaitu :

    - Urutan giliran pertama diacak
    - Hit untuk mengambil kartu dan menambahkan point dan stand berarti tidak mengambil kartu
    - Bot akan hit/mengambil kartu secara random tetapi akan stand jika poinnya lebh dari 17
    - Point tidak boleh sampai melebihi 21 tetapi harus lebih besar dari bot
    - Jika poin player dan bot sama atau point keduanya melebihi dari 21 makan akan seri