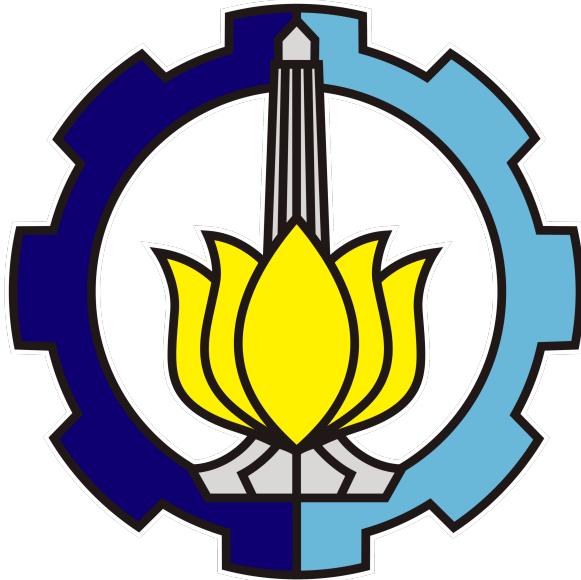


PLAYING CARD



Muhammad Daffa' Fisabilillah (5024211006)

Pengolahan Citra Video (B)

Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember Surabaya

1 Pendahuluan

1.1 Deskripsi Tugas

Dalam konteks pembuatan sebuah permainan kartu pada program yang penulis buat, pengolahan citra video merupakan salah satu aspek kunci. Proses pengolahan citra memungkinkan program untuk mendeteksi dan memproses kartu-kartu yang diambil melalui kamera. Ini melibatkan serangkaian langkah, mulai dari mengambil gambar dari kamera, melakukan operasi pemrosesan citra seperti pengaturan warna, deteksi kontur, penggunaan metode Canny Edge Detection, penggunaan transformasi perspektif, hingga proses thresholding adaptif dan identifikasi warna. Hal tersebut dilakukan untuk membuat pengambilan dataset dapat dilakukan dengan baik atau konstan dari gangguan berbagai pencahayaan. Dataset yang baik akan menghasilkan prediksi yang baik pula, sehingga setiap kartu yang dimainkan dapat berjalan sesuai keinginan. Prediksi dilakukan dengan menggunakan machine learning dengan model Convolutional Neural Network yang didesain untuk mengidentifikasi pola dalam data visual melalui serangkaian layer yang terdiri dari convolutional layers, pooling layers dan fully connected layers.

Interaksi pengguna melalui input keyboard juga menjadi bagian penting dari pembuatan permainan kartu ini. Penulis menggunakan modul "msvcrt" yang berfungsi untuk mengecek apakah ada tombol yang telah ditekan pada keyboard tanpa mengharuskan program menunggu input dari pengguna. Pengguna dapat memilih berbagai opsi, seperti mengambil kartu, menyimpan data, mengecek kartu, atau memulai permainan, dengan menginputkan tombol-tombol yang telah ditentukan.

1.2 Alur Program

Sebelum game playing card dapat dimainkan kita diharuskan terlebih dahulu mengumpulkan dataset sebanyak 20 jenis foto '.jpg' pada setiap jenis kartu yang ada yaitu 52 kartu. Hal tersebut dapat dilakukan dengan menekan tombol 'w' pada keyboard secara manual, setiap 20 photo yang telah diambil akan dilanjutkan ke label berikutnya hingga habis. Pengambilan dataset ini dijalankan pada file 'gameplay.py' pada fungsi 'def saveData()'. Jika dataset telah diambil semua maka Anda telah memiliki total 1.040 foto, selanjutnya Anda akan beralih ke file program '2_Training_Label.py'. Didalam program tersebut Anda akan melakukan latihan pada dataset yang telah diambil menggunakan machine learning dengan model Convolution Neural Network, sebelum melakukan hal tersebut Anda terlebih dahulu harus menyimpan semua dataset dan label yang dimiliki kedalam sebuah list yang berbeda lalu ubah keduanya kedalam array dengan type float32. Hasil dari machine learning tersebut berupa file 'Fix.h5'.

Kembali kedalam program 'gameplay.py', Anda harus memasukkan file 'Fix.h5' kedalam fungsi inisiasi 'def __init__(self)' agar model dapat di load dengan pustaka Keras. Untuk memulai game, hal pertama yang harus dilakukan

adalah mengambil kartu terlebih dahulu dengan menekan tombol 'e' maka akan otomatis mengambil gambar secara real time kemudian akan diprediksi dengan 'np.argmax' setelah itu akan muncul pernyataan mengenai kebenaran label jika salah Anda dapat mengulangnya. Jika kartu dinilai sudah benar maka anda dapat memasukkannya kedalam list player dengan tombol 'j' atau komputer dengan tombol 'k' lakukan hal tersebut sebanyak yang diinginkan sebelum memulai permainan.

Setelah list kartu terisi pada player dan komputer maka, fungsi yang akan berjalan selanjutnya adalah 'def play(self)'. Permainan telah berjalan hal pertama yang dilakukan adalah membuka kartu pertama untuk menentukan kartu yang akan dikeluarkan pada player dan komputer. Kartu bukaan akan disimpan pada list open_cards. Urutan bermain pertama akan selalu komputer karena urutan tidak berpengaruh pada hasil penentuan pemenang. Komputer akan mengecek apakah jenis bentuk kartu dari open_cards sama dengan yang ada di list computer_cards jika komputer tidak memiliki matching cards maka ia akan mengambil kartu lagi hingga menemukan kartu yang sama. Kartu yang sama akan ditambahkan ke open_cards sekaligus dihapus dari list computer_cards jika tidak sama maka akan tetap ditambahkan akan tetapi komputer akan mengambil kartu kembali dengan menekan tombol 't' hingga mendapat matching cards. Kondisi komputer lainnya adalah telah menemukan matching cards dari awal maka ia akan memilih terlebih dahulu dalam list computer_cards tersebut apakah ada yang memiliki precedence yang terbesar, jika sudah didapatkan maka akan dihapus dari list computer_cards dan ditambahkan pada open_cards

Selanjutnya adalah giliran player, player memiliki 2 pilihan menu yaitu menghapus kartu dan mengambil kartu. Ketika memilih menu 1 maka kita diharuskan menuliskan index yang ingin diinginkan, jika jenis bentuk tidak sesuai maka akan diberi peringatan dan menampilkan 2 menu diawal kembali serta jika jenis bentuk sesuai maka akan dihapus dari list player_cards lalu ditambahkan kedalam open_cards. Keadaan selanjutnya iyalah pengambilan kartu, setelah mengambil kartu maka akan langsung masuk kedalam list dan menampilkan kembali 2 menu yang ada sebelumnya. Giliran player akan berhenti jika iya sudah menambahkan kartunya ke dalam list open_cards

Keadaan setelah open_cards memiliki panjang 3 maka semua list yang ada dalam open_cards akan dipindahkan kedalam list trash sehingga tidak tersisa apapun maka keadaan game dapat berulang mulai dari pengambilan kartu open_cards kembali. Permainan akan berakhir jika terdapat 6 kondisi:

1. Jika panjang list trash, player_cards dan computer_cards telah mencapai 52 maka akan dihitung kartu dengan precedence yang paling banyak maka dialah pemenangnya dan jika seimbang maka akan berakhir (Draw !)
2. Jika panjang kartu player_cards telah kosong tidak berisi maka player

menang

3. Jika panjang kartu computer_cards telah kosong tidak berisi maka komputer menang

2 Pembahasan

2.1 Source Code

2.1.1 Main Program

```
1 # Akses file gameplay.py
2 import gameplay
3
4 # Program utama
5 # Memjadikan kelas Gameplay() dari file gameplay.py
6 # sebagai objek
6 game = gameplay.Gameplay()
7 # Memanggil fungsi start pada kelas Gameplay()
8 game.start()
```

2.1.2 Game Program

```
1 import os
2 import cv2
3 import numpy as np
4
5 from keras.models import load_model
6
7 import msvcrt
8
9 import utils
10
11 class Gameplay:
12     # Inisialisasi penyimpanan semua list, file, variabel
13     # , dan VideoCapture yang akan digunakan pada fungsi
14     # lain
15     def __init__(self):
16         # Menandakan apakah game telah dimulai atau
17         # belum
18         self.game_started = False
19
20         self.cap = cv2.VideoCapture(0)
21
22         self.Past = np.array(0)
```

```

20
21     self.photos_taken = 0
22
23     # Deklarasi list kartu yang dimiliki pemain
24     # dan komputer
24     self.player_cards = []
25     self.computer_cards = []
26     # Deklarasi list kartu open card selalu
27     # berisi 1
27     self.open_cards = []
28     # Inisialisasi list self.trash
29     self.trash = []
30
31     self.model = load_model('Fix.h5')
32
33     # Fungsi yang pertama kali dijalankan
34     def start(self):
35         print(
36             "#####"
37         )
36         print("##### PlayingCard#####")
37         print(
38             "#####"
39         )
38         print("Menu")
39         print("1. Press 'q' = Quit")
40         print("2. Press 'w' = Save datasheet")
41         print("3. Press 'e' = Take a card")
42         print("4. Press 'p' = Check card")
43         print("5. Press 'z' = Start the Game")
44
45     # Inisialisasi kamera pada droid cam
46     self.cap = cv2.VideoCapture(2)
47
48     while True:
49         ret, frame = self.cap.read()
50         if not ret:
51             print("Tidak dapat membaca frame dari
51                 kamera.")
52             break
53
54         # Menangkap Kartu dari greenscreen
55         self.readImage(frame)
56

```

```

57         # Menekan menu yang diinginkan pada
58         # terminal
59         err = self.readInput()
60
61         # Exit lewat terminal
62         if err == 'exit':
63             break
64
65         # Exit pada cv2.imshow agar imshow tidak
66         # abu-abu
67         if cv2.waitKey(1) & 0xFF == ord('q'):
68             break
69
70         self.cap.release()
71         cv2.destroyAllWindows()
72
73         # Inisialisasi Indeks yang terus bertambah
74         # saat mengambil foto
75         self.photos_taken = 0
76
77     # Fungsi yang mengelola bagian greenscreen, kontur
78     # kartu dan deteksi warna
79     def readImage(self, frame):
80         # Proses citra untuk menghasilkan maska warna
81         # hijau
82         hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
83         lower_green1 = np.array([35, 100, 100])
84         upper_green1 = np.array([100, 255, 255])
85         lower_green2 = np.array([40, 40, 40])
86         upper_green2 = np.array([80, 255, 255])
87         mask1 = cv2.inRange(hsv, lower_green1,
88                             upper_green1)
89         mask2 = cv2.inRange(hsv, lower_green2,
90                             upper_green2)
91
92         combined_mask = cv2.bitwise_or(mask1, mask2)
93         kernel = np.ones((3, 3), np.uint8)
94         combined_mask = cv2.erode(combined_mask,
95                                   kernel, iterations=3)
96         combined_mask = cv2.dilate(combined_mask,
97                                   kernel, iterations=7)
98
99         # Gunakan Canny Edge Detection pada maska
100        edges = cv2.Canny(combined_mask, 50, 150)

```

```

93     # Deteksi kontur pada citra hasil edge
94     # detection
95     contours, _ = cv2.findContours(edges, cv2.
96                                     RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
97
98     for contour in contours:
99         area = cv2.contourArea(contour)
100        # Threshold area sesuai kebutuhan
101        if area > 10000:
102            # Menghitung kotak berbentuk persegi
103            # panjang
104            rect = cv2.minAreaRect(contour)
105            # Ambil koordinat keempat sudut kartu
106            box = cv2.boxPoints(rect)
107            box = np.asarray(box, dtype=int)
108
109            # POTONG BAGIAN KARTUNYA SAJA
110            # Mendapatkan lebar dan tinggi dari
111            # persegi panjang terkecil
112            width = int(rect[1][0])
113            height = int(rect[1][1])
114
115            # Perbarui lebar dan tinggi menjadi
116            # dua kali lipat
117            width *= 2
118            height *= 2
119
120            # Jika lebar > tinggi, lakukan
121            # transformasi perspektif
122            if width > height:
123                width, height = height, width    #
124                # Tukar lebar dan tinggi
125
126                src_pts = box.astype("float32")
127                dst_pts = np.array([[0, height - 1],
128                                    [0, 0], [width - 1, 0], [width - 1,
129                                     height - 1]], dtype="float32")
130
131                matrix = cv2.getPerspectiveTransform(
132                    src_pts, dst_pts)
133                warped = cv2.warpPerspective(frame,
134                                             matrix, (width, height))
135
136                # Lakukan adaptive thresholding
137                gray_warped = cv2.cvtColor(warped,
138                                           cv2.COLOR_BGR2GRAY)

```

```

127     blurred = cv2.GaussianBlur(
128         gray_warped, (11, 11), 0)
129     biner = cv2.adaptiveThreshold(blurred,
130         255, cv2.
131         ADAPTIVE_THRESH_GAUSSIAN_C, cv2.
132         THRESH_BINARY, 11, 2)
133
134     # Resize
135     self.Past = cv2.resize(biner, (200,
136         300))
137
138     # Tampilkan gambar warped yang telah
139     # ditandai dengan kontur persegi
140     # panjang warna hitam dan merah
141     # cv2.imshow('AdaptiveThresholding',
142     # self.Past)
143
144     # Gambar kotak yang mengelilingi
145     # kontur kartu
146     cv2.drawContours(frame, [box], 0, (0,
147         255, 0), 2)
148     cv2.imshow('Deteksi_Kartu', frame)
149
150
151     # MEMBERIKAN KONTUR WARNA MERAH DAN
152     # HITAM
153     # Mengubah ke skema warna HSV
154     hsv_warped = cv2.cvtColor(warped, cv2
155         .COLOR_BGR2HSV)
156
157     # Definisikan rentang warna merah dan
158     # hitam di HSV
159     lower_red = np.array([0, 120, 70])
160     upper_red = np.array([10, 255, 255])
161
162     lower_black = np.array([0, 0, 0])
163     upper_black = np.array([179, 255,
164         30])
165
166     # Buat maska untuk warna merah dan
167     # hitam
168     mask_red = cv2.inRange(hsv_warped,
169         lower_red, upper_red)
170     mask_black = cv2.inRange(hsv_warped,
171         lower_black, upper_black)

```

```

156      # Gabungkan maska warna merah dan
157      # hitam
158      mask_color = cv2.bitwise_or(mask_red,
159          mask_black)
160      kernel = np.ones((3, 3), np.uint8)
161      mask_color = cv2.erode(mask_color,
162          kernel, iterations=3)
163      mask_color = cv2.dilate(mask_color,
164          kernel, iterations=7)
165
166      # Temukan kontur pada warna yang
167      # terdeteksi
168      contours_color, _ = cv2.findContours(
169          mask_color, cv2.RETR_EXTERNAL, cv2.
170          CHAIN_APPROX_SIMPLE)
171
172      # Loop melalui setiap kontur warna
173      # yang terdeteksi
174      for contour in contours_color:
175          area = cv2.contourArea(contour)
176          if area > 100:  # Sesuaikan
177              # dengan threshold area yang
178              # sesuai
179              # Tentukan warna berdasarkan
180              # maska yang terdeteksi
181              if np.any(cv2.bitwise_and(
182                  mask_red, mask_black)):
183                  color_detected = ""
184                  Campuran_Merah_dan_Hitam"
185
186                  # np.any digunakan untuk
187                  # mengetahui apakah ada pixel
188                  # hitam atau merah di dalam
189                  # kontur
190
191                  # Jika ada pixel hitam dan
192                  # merah, maka warna yang
193                  # terdeteksi adalah campuran
194                  # merah dan hitam
195
196                  # Jika hanya ada pixel merah
197                  # maka warna yang terdeteksi
198                  # adalah merah
199
200                  # Jika hanya ada pixel hitam
201                  # maka warna yang terdeteksi
202                  # adalah hitam
203
204
205                  # Gambar kotak mengelilingi
206                  # kontur warna hitam dan
207                  # merah
208                  rect = cv2.minAreaRect(
209                      contour)
210                  box = cv2.boxPoints(rect)
211                  box = np.int0(box)
212                  cv2.drawContours(warped, [box],
213                      0, (0, 255, 0), 2)

```

```

184                                     # Tambahkan teks pada kontur
185                                     cv2.putText(warped, f"Warna\u20ac
186                                         adalah:{color_detected}",      ,
187                                         (box[0][0], box[0][1]), cv2
188                                         .FONT_HERSHEY_SIMPLEX, 0.5,
189                                         (255, 255, 255), 2)
190
191                                     # Tampilkan gambar warped yang telah
192                                     # ditandai dengan kontur persegi
193                                     # panjang warna hitam dan merah
194                                     cv2.imshow('Deteksi\u20acWarna\u20acdengan\u20acTeks',
195                                     , warped)
196
197                                     # Fungsi yang mengatur bagian tombol
198                                     def readInput(self):
199                                         if msvcrt.kbhit():
200                                             inp = msvcrt.getch()
201
202                                         if inp == b'q':
203                                             return 'exit'
204                                         elif inp == b'e':
205                                             card = self.takeCard()
206
207                                         if card:
208                                             # Mencocokkan label benar, kartu
209                                             # dapat disimpan ke dalam pemain
210                                             # atau komputer
211                                             save_card = input("Press\u20ac'j'\u20acto\u20ac
212                                                 add\u20acto\u20acplayer\u20acor\u20ac'k'\u20acto\u20acadd\u20acto\u20ac
213                                                 computer\u20ac:")
214                                             if save_card == 'j': # Saat
215                                                 tombol 'j' ditekan (untuk
216                                                 pemain)
217                                                 self.player_cards.append(card
218                                                     )
219                                                 print("Card\u20acadded\u20acto\u20acplayer\u20ac
220                                                     cards.")
221                                             elif save_card == 'k': # Saat
222                                                 tombol 'k' ditekan (untuk
223                                                 komputer)
224                                                 self.computer_cards.append(
225                                                     card)
226                                                 print("Card\u20acadded\u20acto\u20accomputer
227                                                     cards.")
228
229                                         elif inp == b'w':

```

```

211         self.saveData()
212     elif inp == b'p':
213         utils.display_cards(self.player_cards,
214                             , "Player")
215         utils.display_cards(self.
216                             computer_cards, "Computer")
217         utils.display_cards(self.open_cards,
218                             , "Open")
219     elif inp == b'z':
220         return self.play()

221 # Fungsi untuk mengambil gambar yang telah diberikan
222 # label ke list pemain
223 def takeCard(self):
224     cv2.imwrite(f'tmp.jpg', self.Past)
225     img = np.array(cv2.imread('tmp.jpg'))
226     img = img/255
227     img = np.array(cv2.resize(img, (128, 128)))
228     img = img.reshape(1,128,128,3)

229
230 # Prediksi
231 label_prediction = self.model.predict(img)

232
233 # Mendapatkan indeks label dengan nilai
234 # maksimum prediksi
235 predicted_label_index = np.argmax(
236     label_prediction)
237 # Menampilkan label yang sesuai dengan indeks
238 # prediksi
239 predicted_label = utils.labels[
240     predicted_label_index]

241
242 print(predicted_label)

243
244 # Jika predicted_label sudah benar, lanjutkan
245 # dengan menyimpan kartu
246 label_correct = input("Is the predicted label
247 correct? (y/n): ")
248 if label_correct.lower() == 'y':
249     card = {'image': self.Past, 'label':
250             predicted_label}

251
252     self.Past = None
253     predicted_label = None
254     return card
255 else:

```

```

246         print("Label not confirmed.")
247         return None
248
249 # Fungsi untuk Membuat Datasheet
250 def saveData(self):
251     # Cek apakah sudah diambil 10 foto, jika iya,
252     # pindah ke label berikutnya
253     if self.photos_taken >= 20:
254         self.photos_taken = 0 # Reset jumlah
255         foto yang diambil
256         label_state += 1 # Pindah ke label
257         berikutnya
258         print(f'Switched to label {utils.labels[label_state]}' )
259
260     # Cek apakah file ada
261     if not os.path.exists(f'{utils.labels[label_state]}/'):
262         os.mkdir(f'{utils.labels[label_state]}/')
263
264     # Mengambil foto
265     cv2.imwrite(f'{utils.labels[label_state]}/{utils.labels[label_state]}_{self.photos_taken}.jpg', self.Past)
266     self.photos_taken += 1 # Menambah jumlah
267     foto yang telah diambil
268
269     print(f'Photo {self.photos_taken} taken for label {utils.labels[label_state]}')
270
271 # Game Dimulai
272 def play(self):
273     print("#####")
274     print("#### GAME STARTED ####")
275     print("#####")
276
277     self.game_started = True
278
279     # 0 = com turn, 1 = player turn
280     state = 0
281
282     while True:
283         ret, frame = self.cap.read()
284         self.readImage(frame)
285
286         if cv2.waitKey(1) & 0xFF == ord('q'):

```

```

283         break
284
285     utils.display_cards(self.player_cards, "Player")
286     utils.display_cards(self.computer_cards, "Computer")
287     utils.display_cards(self.open_cards, "Open")
288
289     if len(self.open_cards) == 0:
290         print('Tekan tombol U membuka kartu ...')
291         # Menunggu tombol 'u' ditekan membuka kartu baru
292     while True:
293         ret, frame = self.cap.read()
294         self.readImage(frame)
295
296         if cv2.waitKey(1) & 0xFF == ord('q'):
297             break
298
299         if msvcrt.kbhit() and msvcrt.getch() == b'u':
300             card = self.takeCard()
301             self.open_cards.append(card)
302             print("Card added to open card.")
303             break
304
305 ######
306 # Computer #
307 #####
308 if state == 0:
309     print("Computer Take it First!")
310     # OpenCard hanya ada satu kartu dan ketika sudah berjumlah 3 ia akan dipindahkan ke trash
311     label_to_remove = self.open_cards[0][label].split('_')[-1]
312
313     precedence = {}
314     # Pilih kamus precedence berdasarkan jenis bentuk
315     if label_to_remove == 'Club':

```

```

316         precedence = utils.
317             precedence_Club
318     elif label_to_remove == 'Diamond':
319         precedence = utils.
320             precedence_Diamond
321     elif label_to_remove == 'Heart':
322         precedence = utils.
323             precedence_Heart
324     elif label_to_remove == 'Spades':
325         precedence = utils.
326             precedence_Spades
327
328     matching_cards = []
329     for c in self.computer_cards:
330         # cek seluruh indeks pada list
331         # computer_cards yang berakhiran
332         # label_to_remove
333     if c['label'].endswith(
334         label_to_remove):
335         # Jika terdapat kondisi
336         # diatas maka akan di
337         # tambahkan pada list
338         # matching_cards
339     matching_cards.append(c)
340
341     # Keadaan ketika matching_cards tidak
342     # ditemukan pada list computer_cards
343     if len(matching_cards) == 0:
344         # Komputer tidak memiliki kartu
345         # yang sesuai, maka ambil kartu
346         # sampai mendapatkan yang sesuai
347     print(f"Computer doesn't have any
348         matching card for {label_to_remove}. Press 't' to
349         take a card.")
350     # Menunggu tombol 't' ditekan
351     # sebelum mengambil kartu
352     while True:
353         if self.checkEnd():
354             break
355
356         ret, frame = self.cap.read()
357         self.readImage(frame)
358
359         if cv2.waitKey(1) & 0xFF ==
360             ord('q'):

```

```

344         break
345
346     if msvcrt.kbhit() and msvcrt.
347         getch() == b't':
348             c = self.takeCard()
349
350         if c:
351             # Jika komputer
352                 berhasil mengambil
353                 kartu yang sesuai,
354                 keluar dari loop
355             if c['label'].split(',')
356                 [1] ==
357                 label_to_remove:
358                 print("Computer\u
359                     got\ua\umatching\u
360                     card!")
361
362             # Hapus kartu
363                 dengan label
364                 yang sama dari
365                 self.
366                 computer_cards
367                 dengan
368                 precedence
369                 terbesar pada
370                 jenis bentuk
371                 tertentu
372             # Pindahkan kartu
373                 -kartu yang
374                 ingin dihapus
375                 ke dalam list
376                 self.trash
377             self.open_cards.
378                 append(c)
379
380             # Komputer
381                 memiliki kartu
382                 yang sesuai
383             print(f"Computer\u
384                 has\ua\umatching\u
385                 card\ufor\u{label_to_remove
386                 }.\")
387
388             break
389         else:
390             self.
391             computer_cards.

```

```

361                                         append(card)
362                                         print(f"Computer\u
363                                         doesn't have\uau
364                                         matching\u card\u
365                                         for\u{
366                                         label_to_remove
367                                         }.uPress\u't'\u to
368                                         \take\u a\u card.\")
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384

```

```

                                         append(card)
print(f"Computer\u
doesn't have\uau
matching\u card\u
for\u{
label_to_remove
}.uPress\u't'\u to
\take\u a\u card.\")
else:
    # Hapus kartu dengan label yang
    # sama dari self.computer_cards
    # dengan precedence terbesar pada
    # jenis bentuk tertentu
    max_precedence = -1
    max = 0
    for i in range(len(self.
        computer_cards)):
        card = self.computer_cards[i]
        if card['label'].split('_')
            [1] == label_to_remove and
            precedence[card['label']] >
            max:
            max_precedence = i
            max = precedence[card[,
                'label']]
if max_precedence != -1:
    # Pindahkan kartu-kartu yang
    # ingin dihapus ke dalam list
    # self.trash
    self.open_cards.append(self.
        computer_cards.pop(i))
    # Hapus kartu-kartu tersebut
    # dari self.computer_cards
    # Komputer memiliki kartu
    # yang sesuai
    print(f"Computer\u has\uau
        matching\u card\u for\u{
        label_to_remove}.")
state = 1
#####
# PLAYER TURN #
#####

```

```

385     elif state == 1:
386         # Player Turn
387         while True:
388             if self.checkEnd():
389                 break
390
391             # Pilih Menu yang ingin diambil
392             print("Menu:")
393             print("1. Remove a card")
394             print("2. Take a card")
395             player_choice = input("Enter your
396             choice (1 or 2): ")
397             # Jika pemain memilih untuk
398             # menghapus kartu
399             if player_choice == '1':
400                 # Setelah input dari pemain
401                 # untuk memilih kartu yang
402                 # ingin dihapus
403                 selected_card_index = int(
404                     input("Enter the index of
405                     the card you want to choose
406                     : "))
407
408                 # Pilih kartu yang ingin
409                 # dihapus berdasarkan indeks
410                 # yang dimasukkan pemain
411                 selected_card = self.
412                     player_cards[
413                         selected_card_index]
414
415                 # Periksa jenis bentuk kartu
416                 # yang dipilih pemain
417                 selected_card_shape =
418                     selected_card['label'].
419                     split('_')[-1]
420
421                 # Mencari kartu yang sesuai
422                 # dengan jenis bentuk pada
423                 # self.open_cards
424                 if self.open_cards[0]['label',
425                     ].endswith(
426                     selected_card_shape):
427                     # Memindahkan kartu dari
428                     # self.player_cards ke
429                     # self.trash

```

```

410                         self.open_cards.append(
411                             self.player_cards.pop(
412                                 selected_card_index))
413                             print(f"Card{selected_card_index+1} moved from player's cards to open card.")
414                             state = 0
415                             break
416                         else:
417                             print("Selected card doesn't match any shape in open cards. Please choose another card.")
418
419                         if player_choice == '2':
420                             print("Press 't' to take a card.")
421                             while True:
422                                 ret, frame = self.cap.read()
423                                 self.readImage(frame)
424
425                                 if cv2.waitKey(1) & 0xFF == ord('q'):
426                                     break
427
428                                 if msvcrt.kbhit() and msvcrt.getch() == b't':
429                                     card = self.takeCard()
430
431                                     if card:
432                                         self.player_cards.append(card)
433                                         print("Card added to player cards.")
434                                         break
435
436                         ## If open_card length >= 3, then
437                         open_card = 0
438                         if len(self.open_cards) == 3:
439                             for c in self.open_cards:

```

```

439             self.trash.append(c)
440
441         self.open_cards.clear()
442         print("open_cards moved to trash")
443
444         state = 0
445
446
447     if self.checkEnd():
448         print("Game Over!")
449         utils.display_cards(self.player_cards,
450                             "Player:")
451         utils.display_cards(self.
452                             computer_cards, "Computer:")
453         utils.display_cards(self.trash, "Open
454             ")
455         # Hitung total precedence dari setiap
456         # list kartu
457         total_player_precedence = utils.
458             calculate_total_precedence(self.
459             player_cards)
460         total_computer_precedence = utils.
461             calculate_total_precedence(self.
462             computer_cards)
463         total_trash_precedence = utils.
464             calculate_total_precedence(self.
465             trash)
466         # Tentukan pemenang berdasarkan total
467         # precedence terkecil di antara self.
468         # .player_cards dan self.
469         # .computer_cards
470
471     if total_player_precedence <=
472         total_computer_precedence:
473         print("Congratulations! You win!")
474
475     elif total_computer_precedence <=
476         total_player_precedence:
477         print("Computer wins!")
478     elif total_computer_precedence ==
479         total_player_precedence:
480         # Ketika jumlah kartu pada pada
481         print("Draw!")
482
483     break
484
485
486     if len(self.player_cards) == 0 and len(
487         self.computer_cards) == 0:

```

```

467         print("Draw!")
468         break
469     elif len(self.player_cards) == 0:
470         print("Player Wins")
471         break
472     elif len(self.computer_cards) == 0:
473         print("Computer Wins")
474         break
475
476     print("Game Over")
477
478 # Kondisi 52 kartu telah terpakai
479 def checkEnd(self):
480     if len(self.trash) + len(self.player_cards) +
481         len(self.computer_cards) == 52:
482         return True
483
484     return False

```

2.1.3 Label Program

```

1 # Label yang digunakan untuk mengambil dataset
2 labels = (
3     "2_Club", "3_Club", "4_Club", "5_Club", "6
4         _Club", "7_Club", "8_Club", "9_Club", "10
5         _Club", "ace_Club",
6     "J_Club", "Q_Club", "K_Club", "2_Diamond", "3
7         _Diamond", "4_Diamond", "5_Diamond", "6
8         _Diamond", "7_Diamond",
9     "8_Diamond", "9_Diamond", "10_Diamond", "
10     ace_Diamond", "J_Diamond", "Q_Diamond", "
11     K_Diamond", "2_Heart",
12     "3_Heart", "4_Heart", "5_Heart", "6_Heart", "
13     7_Heart", "8_Heart", "9_Heart", "10_Heart",
14     "ace_Heart",
15     "J_Heart", "Q_Heart", "K_Heart", "2_Spades",
16     "3_Spades", "4_Spades", "5_Spades", "6
17         _Spades", "7_Spades",
18     "8_Spades", "9_Spades", "10_Spades", "
19     ace_Spades", "J_Spades", "Q_Spades", "
20     K_Spades"
21 )
22
23 # Label yang digunakan untuk menghitung nilai sebuah
24     kartu

```

```

12 precedence_Club = {
13     '2_Club': 2, '3_Club': 3, '4_Club': 4, '5_Club': 5,
14     '6_Club': 6, '7_Club': 7, '8_Club': 8, '9_Club': 9,
15     'ace_Club': 14, 'J_Club': 11, 'Q_Club': 12, 'K_Club': 13,
16 }
17 precedence_Diamond={
18     '2_Diamond': 2, '3_Diamond': 3, '4_Diamond': 4, '5_Diamond': 5,
19     '6_Diamond': 6, '7_Diamond': 7, '8_Diamond': 8, '9_Diamond': 9,
20     '10_Diamond': 10, 'ace_Diamond': 14, 'J_Diamond': 11, 'Q_Diamond': 12,
21     'K_Diamond': 13,
22 }
23 precedence_Heart= {
24     '2_Heart': 2, '3_Heart': 3, '4_Heart': 4, '5_Heart': 5,
25     '6_Heart': 6, '7_Heart': 7, '8_Heart': 8, '9_Heart': 9,
26     '10_Heart': 10, 'ace_Heart': 14, 'J_Heart': 11, 'Q_Heart': 12,
27     'K_Heart': 13,
28 }
29 precedence_Spades={
30     '2_Spades': 2, '3_Spades': 3, '4_Spades': 4, '5_Spades': 5,
31     '6_Spades': 6, '7_Spades': 7, '8_Spades': 8, '9_Spades': 9,
32     '10_Spades': 10, 'ace_Spades': 14, 'J_Spades': 11, 'Q_Spades': 12,
33     'K_Spades': 13
34 }

# Fungsi untuk menampilkan kartu-kartu yang ada pada list
35 # utils.display_cards(self.player_cards, "Player")
36 def display_cards(cards, player_type):
37     print(f"{player_type} Cards:")
38     if not cards:
39         print("No cards yet.")
40     else:
41         for idx, card in enumerate(cards, 1):
42             print(f"Card {idx}:")
43             #print(f"Image: {card['image']}") # Dalam bentuk array matriks
44             print(f"Label: {card['label']}")
45             print("-----")

```

```

43 # Fungsi menghitung precedence di setiap list player
44 # maupun komputer
45 def calculate_total_precedence(cards):
46     total_precedence = 0
47     precedences = [precedence_Club,
48                     precedence_Diamond, precedence_Heart,
49                     precedence_Spades]
50
51     for card in cards:
52         shape = card['label']
53         for precedence in precedences:
54             if shape in precedence:
55                 total_precedence += precedence[shape]
56                 break
57
58     return total_precedence

```

2.1.4 Training Program

```

1 import os
2 from keras.models import load_model
3 import cv2
4 import numpy as np
5 from keras.layers import Input, Dense
6 from keras.layers import Conv2D, MaxPooling2D,
7     Flatten
8 from keras.models import Model
9 import matplotlib.pyplot as plt
10 from datetime import datetime
11 from numpy import expand_dims
12 from keras.utils import load_img
13 from keras.utils import img_to_array
14 from keras.preprocessing.image import
15     ImageDataGenerator
16 from matplotlib import pyplot
17
18 def ModelDeepLearningCNN(JumlahKelas):
19     # Ukuran citra 128x128 piksel dan tiga saluran
20     # warna (RGB).
21     input_img = Input(shape=(128, 128, 3))
22     # Menambahkan layer konvolusi 32 filter dengan
23     # kernel 3*3
24     # ReLU (Rectified Linear Activation)
25     # ReLU mengonversi nilai input menjadi nol
26     # jika nilainya kurang dari nol,

```

```

23     # dan mempertahankan nilai inputnya
24     # jika nilainya lebih besar dari nol.
25     x = Conv2D(32, (3, 3), activation='relu', padding
26         ='same')(input_img)
27     x = MaxPooling2D((2, 2), padding='same')(x)
28     x = Conv2D(32, (3, 3), activation='relu', padding
29         ='same')(x)
30     x = MaxPooling2D((2, 2), padding='same')(x)
31     x = Conv2D(32, (3, 3), activation='relu', padding
32         ='same')(x)
33     x = Flatten()(x)
34     x = Dense(100, activation='relu')(x)
35     x=Dense(JumlahKelas,activation='softmax')(x)
36     ModelCNN = Model(input_img, x)
37     ModelCNN.compile(loss='categorical_crossentropy',
38         optimizer='adam', metrics=['accuracy'])
39     #ModelCNN.compile(loss='mse', optimizer='adam',
40         #metrics=['accuracy'])
41     return ModelCNN
42
43 DirektoriDataSet="E:\\University\\Material\\Semester\\
44      5\\Pengolahan_Citra_Video\\Tugas_Ahir"
45 # Data Set disimpan dalam direktori yang sama
46 dengan nama kelas
47
48 #b. Label Data Set
49 LabelKelas = (
50     "2_Club", "3_Club", "4_Club", "5_Club", "6
51         _Club", "7_Club", "8_Club", "9_Club", "10
52         _Club", "ace_Club",
53     "J_Club", "Q_Club", "K_Club", "2_Diamond", "3
54         _Diamond", "4_Diamond", "5_Diamond", "6
55         _Diamond", "7_Diamond",
56     "8_Diamond", "9_Diamond", "10_Diamond", "
57         ace_Diamond", "J_Diamond", "Q_Diamond", "
58         K_Diamond", "2_Heart",
59     "3_Heart", "4_Heart", "5_Heart", "6_Heart", "
60         7_Heart", "8_Heart", "9_Heart", "10_Heart",
61         "ace_Heart",
62     "J_Heart", "Q_Heart", "K_Heart", "2_Spades",
63     "3_Spades", "4_Spades", "5_Spades", "6
64         _Spades"

```

```

52         _Spades", "7_Spades",
53     "8_Spades", "9_Spades", "10_Spades", "
54     ace_Spades", "J_Spades", "Q_Spades", "
55     K_Spades"
56 )
57
58 # One-hot Encoding ada 52 kelas vektor [1. 0. 0. (
59 # hingga 52 angka)
60 t = np.eye(len(LabelKelas))
61
62 for i in range(len(LabelKelas)):
63     label = LabelKelas[i]
64
65     if not os.path.exists(f'{label}/'):
66         continue
67
68     j = 0
69     while os.path.exists(f'{label}/{label}_{j}.jpg'):
70         filename = f'{label}/{label}_{j}.jpg'
71         img = np.double(cv2.imread(filename))
72         img = cv2.resize(img,(128,128))
73         img = np.asarray(img)/255
74         img = img.astype('float32')
75         x.append(img)
76
77         y.append(t[i])
78
79         j+=1
80
81 x = np.array(x).astype('float32')
82 y = np.array(y).astype('float32')
83
84 # # CEK APAKAH DATASET DAN LABEL BENAR
85 # print("Jumlah dataset (gambar):", len(x))
86 # print("Jumlah dataset (label):", len(y))
87
88 # for i in range(len(LabelKelas)):
89 #     label = LabelKelas[i]
90
91 #     if not os.path.exists(f'{label}/'):
92 #         continue
93 #     j = 0

```

```

94     #         while os.path.exists(f'{label}/{label}_{j}.jpg'):
95     #             :
96     #                 filename = f'{label}/{label}_{j}.jpg'
97     #                 print("File Gambar:", filename)
98     #                 print("Label:", label)
99     #             j += 1
100
101 #c. Inisialisasi parameter Training
102 JumlahEpoh = 10
103 FileBobot = "Fix.h5"
104 #d. training
105 modelCNN = ModelDeepLearningCNN(len(LabelKelas))
106 #Trainng
107 history=modelCNN.fit(x, y, epochs=JumlahEpoh, shuffle=
    True)
108 #Menyimpan hasil learning
109 modelCNN.save(FileBobot)
110
111 #Mengembalikan output
112 modelCNN.summary()
113 model = Model.load_weights
114
115 #c. Menampilkan Grafik Loss dan accuracy
116 plt.plot(history.history['loss'])
117 plt.plot(history.history['accuracy'])
118
119 plt.title('model.loss')
120 plt.ylabel('loss/accuracy')
121 plt.xlabel('epoch')
122 plt.legend(['loss', 'acc'], loc='upper left')
123 plt.show()

```

2.2 Penjelasan Bagian pada Source Code

Pada project game ini, terdapat implementasi yang menggunakan beberapa library Python untuk membuat sebuah permainan kartu. Kode program ini terstruktur dalam beberapa bagian yang memiliki fungsi spesifik. Program permainan kartu dibagi kedalam 4 jenis file program, yaitu 'Game.py' sebagai program utama yang akan menjalankan semua fungsi yang ada pada kelas 'Gameplay()'.

```

1 # Akses file gameplay.py
2 import gameplay
3
4 # Program utama

```

```

5 # Memjadikan kelas Gameplay() dari file gameplay.py
6     sebagai objek
7 game = gameplay.Gameplay()
8 # Memanggil fungsi start pada kelas Gameplay()
9 game.start()

```

Selanjutnya adalah 'utils.py' yang berfungsi untuk sebagai inisiasi seluruh label yang digunakan untuk pengambilan dataset dan label yang telah diberikan nilai, lalu di bagi berdasarkan jenis bentuknya kemudian digunakan untuk menghitung nilai sebuah kartu pada fungsi selanjutnya.

```

1 # Label yang digunakan untuk mengambil dataset
2 labels = (
3     "2_Club", "3_Club", "4_Club", "5_Club", "6
4         _Club", "7_Club", "8_Club", "9_Club", "10
5         _Club", "ace_Club",
6     "J_Club", "Q_Club", "K_Club", "2_Diamond", "3
7         _Diamond", "4_Diamond", "5_Diamond", "6
8         _Diamond", "7_Diamond",
9     "8_Diamond", "9_Diamond", "10_Diamond", "
10     ace_Diamond", "J_Diamond", "Q_Diamond", "
11     K_Diamond", "2_Heart",
12     "3_Heart", "4_Heart", "5_Heart", "6_Heart", "
13     7_Heart", "8_Heart", "9_Heart", "10_Heart",
14     "ace_Heart",
15     "J_Heart", "Q_Heart", "K_Heart", "2_Spades",
16     "3_Spades", "4_Spades", "5_Spades", "6
17         _Spades", "7_Spades",
18     "8_Spades", "9_Spades", "10_Spades", "
19     ace_Spades", "J_Spades", "Q_Spades", "
20     K_Spades"
21 )
22
23 # Label yang digunakan untuk menghitung nilai sebuah
24     kartu
25 precedence_Club = {
26     '2_Club': 2, '3_Club': 3, '4_Club': 4, '5_Club': 5,
27     '6_Club': 6, '7_Club': 7, '8_Club': 8, '9
28         _Club': 9, '10_Club': 10,
29     'ace_Club': 14, 'J_Club': 11, 'Q_Club': 12, '
30         K_Club': 13,
31 }
32 precedence_Diamond={...
33 precedence_Heart= {...
34 precedence_Spades={...

```

Selain itu, terdapat dua fungsi lain yang berfungsi untuk menampilkan list

kartu player, komputer dan opencard. Fungsi selanjutnya adalah untuk menghitung nilai setiap kartu yang terdeapat pada list player dan komputer yang digunakan untuk menentukan pemenang.

1. **def display_cards(cards, player_type):**

Menampilkan list kartu player, komputer dan opencard. Parameter cards merujuk pada list penyimpanan kartu player, komputer, open-card ataupun trash sedangkan player_type adalah string untuk penamaan jenis list yang telah disimpan. Jika cards memiliki isi, fungsi akan melakukan iterasi pada setiap kartu dalam list menggunakan perulangan for. Selama iterasi, fungsi akan menampilkan informasi setiap kartu yang terdapat di cards. Setiap kartu akan ditampilkan dalam format "Card idx:" dimana idx adalah nomor urutan kartu dalam list yang dimulai dari angka 1. Kemudian, fungsi akan menampilkan label dari kartu tersebut dengan menggunakan print(f"Label: card['label'])")

```
1 # Fungsi untuk menampilkan kartu-kartu yang ada pada
  list
2 def display_cards(cards , player_type):
3     print(f"{player_type} Cards:")
4     if not cards:
5         print("No cards yet.")
6     else:
7         for idx , card in enumerate(cards , 1):
8             print(f"Card {idx}:")
9             #print(f"Image: {card[ 'image' ]}") #
10            Dalam bentuk array matriks
11            print(f"Label: {card[ 'label' ]}")
12            print("-----")
```

2. **def calculate_total_precedence(cards):**

Menghitung nilai keseluruhan pada sebuah list kartu player dan komputer untuk menentukan pemenang jika jumlah list player, komputer dan trash sudah terpenuhi (52 kartu). Fungsi ini melakukan iterasi pada setiap kartu dalam cards. Pada setiap iterasi, ia mengambil bentuk kartu yang ada (shape = card['label']) dan memeriksa apakah bentuk tersebut terdapat dalam salah satu dari dictionary precedences. Jika bentuk kartu tersebut terdapat dalam salah satu dictionary, maka nilai "precedence" dari kartu tersebut ditambahkan ke dalam total_precedence. Kemudian, iterasi berikutnya dimulai untuk kartu selanjutnya. Pada akhirnya, fungsi ini mengembalikan total_precedence, yaitu jumlah total nilai "precedence" dari seluruh kartu yang ada dalam list cards. Dengan demikian, fungsi ini memungkinkan untuk menghitung total nilai "precedence" dari sejumlah kartu yang diproses dalam suatu permainan kartu, seperti menentukan nilai total dari setiap jenis kartu dalam tumpukan pemain atau komputer dalam suatu permainan.

```

1 # Fungsi menghitung precedence di setiap list player
2 maupun komputer
3 def calculate_total_precedence(cards):
4     total_precedence = 0
5     precedences = [precedence_Club,
6                     precedence_Diamond, precedence_Heart,
7                     precedence_Spades]
8
9     for card in cards:
10         shape = card['label']
11         for precedence in precedences:
12             if shape in precedence:
13                 total_precedence += precedence[shape]
14             break
15
16     return total_precedence

```

Ketiga adalah program 'gameplay.py' yang terdiri dari kelas 'Gameplay' yang memiliki delapan fungsi sebagai berikut:

1. **def __init__(self):**

Fungsi __init__ yang terdapat dalam kelas Gameplay digunakan untuk melakukan inisialisasi atribut-atribut yang akan digunakan dalam permainan. Pada awalnya, atribut game_started diatur menjadi False, yang akan berubah menjadi True ketika permainan dimulai. Objek self.cap dibuat menggunakan cv2.VideoCapture(0) untuk mengakses kamera. Kemudian, terdapat inisialisasi variabel self.Past yang diatur sebagai 'np.array(0)', dan self.photos_taken yang awalnya diatur sebagai 0 untuk menghitung jumlah foto yang telah diambil. Selanjutnya, terdapat empat list yang dideklarasikan, yaitu self.player_cards, self.computer_cards, self.open_cards, dan self.trash, yang akan digunakan untuk menyimpan kartu-kartu dalam permainan. Terakhir, terdapat pemanggilan fungsi load_model('Fix.h5') dari modul Keras untuk memuat model yang akan digunakan dalam permainan dari file yang disebut 'Fix.h5'. Model tersebut adalah hasil dari file program lain yang khusus untuk melatih machine learning dengan Convolution Neural Networks, akan dibahas pada bagian selanjutnya. Semua inisialisasi ini dilakukan agar atribut-atribut tersebut siap digunakan dalam menjalankan permainan dan penyimpanan data yang dibutuhkan.

```

1 def __init__(self):
2     # Menandakan apakah game telah dimulai atau
3     # belum
4     self.game_started = False

```

```

5         self.cap = cv2.VideoCapture(0)
6
7         self.Past = np.array(0)
8
9         self.photos_taken = 0
10
11        # Deklarasi list kartu yang dimiliki pemain
12        # dan komputer
13        self.player_cards = []
14        self.computer_cards = []
15        # Deklarasi list kartu open card selalu
16        # berisi 1
17        self.open_cards = []
18        # Inisialisasi list self.trash
19        self.trash = []

self.model = load_model('Fix.h5')

```

2. def start(self):

Fungsi start(self) adalah fungsi yang pertama kali dijalankan ketika program dimulai. Fungsi ini bertanggung jawab untuk menampilkan pesan-pesan informasi awal serta menu-menu permainan kartu kepada pengguna. Pesan tersebut mencakup judul permainan dan daftar menu dengan instruksi yang harus dilakukan untuk berinteraksi dengan permainan. Selain itu, fungsi ini juga melakukan inisialisasi kamera menggunakan cv2.VideoCapture(2) untuk mendapatkan akses ke perangkat kamera dengan indeks 2 (DroidCam).

Selama program berjalan dalam loop while True, fungsi ini akan terus membaca setiap frame yang diambil dari kamera dan memprosesnya dengan memanggil fungsi self.readImage(frame), yang bertujuan untuk menangkap kartu dari layar hijau (greenscreen). Kemudian, fungsi akan meminta input dari pengguna melalui self.readInput(), yang mengatur menu yang dapat dipilih melalui terminal. Terakhir, fungsi melakukan inisialisasi ulang variabel self.photos_taken menjadi 0, yang berperan sebagai indeks yang terus bertambah saat mengambil foto dalam permainan.

```

1 def start(self):
2     print('##### Playing Card #####')
3     print('##### \n')
4     print('##### \n')
5     print("Menu")
6     print("1. Press 'q' = Quit")

```

```

7      print ("2. Press 'w' = Save datasheet")
8      print ("3. Press 'e' = Take a card")
9      print ("4. Press 'p' = Check card")
10     print ("5. Press 'z' = Start the Game")

11
12     # Inisialisasi kamera pada droid cam
13     self.cap = cv2.VideoCapture(2)

14
15     while True:
16         ret, frame = self.cap.read()
17         if not ret:
18             print("Tidak dapat membaca frame
19                     dari kamera.")
20             break

21     # Menangkap Kartu dari greenscreen
22     self.readImage(frame)

23
24     # Menekan menu yang diinginkan pada
25         terminal
26     err = self.readInput()

27
28     # Exit lewat terminal
29     if err == 'exit':
30         break

31     # Exit pada cv2.imshow agar imshow tidak
32         abu-abu
33     if cv2.waitKey(1) & 0xFF == ord('q'):
34         break

```

3. def readImage(self, frame):

Fungsi ‘readImage(self, frame)’ mengimplementasikan serangkaian proses pemrosesan citra untuk mendeteksi dan mengenali kartu dalam gambar yang diberikan ('frame'). Proses dimulai dengan mengonversi citra ke ruang warna HSV untuk mendapatkan maska warna hijau yang mengidentifikasi kartu. Setelah itu, dilakukan proses bitwise OR pada maska untuk menggabungkan hasil deteksi. Dilanjutkan dengan erosi dan dilasi untuk membersihkan dan memperkuat hasil maska.

Pada langkah selanjutnya, dilakukan deteksi tepi (edge detection) menggunakan Canny Edge Detection pada maska yang telah dihasilkan sebelumnya. Hasil deteksi tepi digunakan untuk menemukan kontur (contour) yang ada pada citra menggunakan metode ‘cv2.findContours’.

Setiap kontur yang terdeteksi kemudian dianalisis berdasarkan area yang dihasilkan. Kontur yang memiliki luas area lebih besar dari nilai threshold tertentu kemudian diambil untuk diolah lebih lanjut. Di dalam loop kontur tersebut, dilakukan berbagai operasi seperti pengukuran kotak persegi panjang terkecil yang mengelilingi kontur, transformasi perspektif untuk menyesuaikan dimensi kartu, penerapan adaptive thresholding, dan menampilkan hasil dalam jendela ‘Adaptive Thresholding’.

Selanjutnya, program menentukan warna kartu dengan memproses citra warped. Proses ini melibatkan konversi citra ke ruang warna HSV, pembuatan maska untuk warna merah dan hitam, dilanjutkan dengan penggabungan maska, dilasi, dan erosi. Kontur yang teridentifikasi dengan warna yang sesuai diproses untuk menggambar persegi panjang mengelilingi area warna tersebut dan menambahkan teks berisi informasi tentang warna yang terdeteksi pada citra warped.

Semua hasil proses ini ditampilkan dalam beberapa jendela yang berbeda, termasuk jendela untuk deteksi kartu dengan kotak kontur, deteksi warna dengan kontur dan teks, serta beberapa visualisasi tambahan untuk membantu pengguna dalam mengidentifikasi kartu dan warna yang ada.

```
1 # Fungsi yang mengelola bagian greenscreen , kontur
2 # kartu dan deteksi warna
3 def readImage(self , frame):
4     # Proses citra untuk menghasilkan maska warna
5     # hijau
6     hsv = cv2.cvtColor(frame , cv2.COLOR_BGR2HSV)
7     lower_green1 = np.array([35 , 100 , 100])
8     upper_green1 = np.array([100 , 255 , 255])
9     lower_green2 = np.array([40 , 40 , 40])
10    upper_green2 = np.array([80 , 255 , 255])
11    mask1 = cv2.inRange(hsv , lower_green1 ,
12                          upper_green1)
13    mask2 = cv2.inRange(hsv , lower_green2 ,
14                          upper_green2)
15
16    combined_mask = cv2.bitwise_or(mask1 , mask2)
17    kernel = np.ones((3 , 3) , np.uint8)
18    combined_mask = cv2.erode(combined_mask ,
19                               kernel , iterations=3)
20    combined_mask = cv2.dilate(combined_mask ,
21                               kernel , iterations=7)
22
23    # Gunakan Canny Edge Detection pada maska
24    edges = cv2.Canny(combined_mask , 50 , 150)
```

```

20      # Deteksi kontur pada citra hasil edge
21      # detection
22      contours , _ = cv2 . findContours (edges , cv2 .
23      RETR_EXTERNAL, cv2 . CHAIN_APPROX_SIMPLE)
24
25      for contour in contours:
26          area = cv2 . contourArea (contour)
27          # Threshold area sesuai kebutuhan
28          if area > 10000:
29              # Menghitung kotak berbentuk persegi
30              # panjang
31              rect = cv2 . minAreaRect (contour)
32              # Ambil koordinat keempat sudut
33              # kartu
34              box = cv2 . boxPoints (rect)
35              box = np . asarray (box , dtype = int )
36
37              # POTONG BAGIAN KARTUNYA SAJA
38              # Mendapatkan lebar dan tinggi dari
39              # persegi panjang terkecil
40              width = int (rect [1] [0])
41              height = int (rect [1] [1])
42
43              # Perbarui lebar dan tinggi menjadi
44              # dua kali lipat
45              width *= 2
46              height *= 2
47
48              # Jika lebar > tinggi , lakukan
49              # transformasi perspektif
50              if width > height:
51                  width , height = height , width ##
52                  # Tukar lebar dan tinggi
53
54                  src_pts = box . astype (" float32 ")
55                  dst_pts = np . array ([[0 , height - 1],
56                  [0 , 0], [width - 1 , 0], [width -
57                  1, height - 1]], dtype = " float32 "
58                  )
59
60                  matrix = cv2 . getPerspectiveTransform
61                  (src_pts , dst_pts)
62                  warped = cv2 . warpPerspective (frame ,
63                  matrix , (width , height))
64
65                  # Lakukan adaptive thresholding

```

```

53     gray_warped = cv2.cvtColor(warped,
54         cv2.COLOR_BGR2GRAY)
55     blurred = cv2.GaussianBlur(
56         gray_warped, (11, 11), 0)
57     biner = cv2.adaptiveThreshold(
58         blurred, 255, cv2.
59             ADAPTIVE_THRESH_GAUSSIAN_C, cv2.
60             THRESH_BINARY, 11, 2)
61
62     # Resize
63     self.Past = cv2.resize(biner, (200,
64         300))
65
66     # Tampilkan gambar warped yang telah
67     # ditandai dengan kontur persegi
68     # panjang warna hitam dan merah
69     cv2.imshow('Adaptive Thresholding',
70             self.Past)
71
72     # Gambar kotak yang mengelilingi
73     # kontur kartu
74     cv2.drawContours(frame, [box], 0,
75         (0, 255, 0), 2)
76     cv2.imshow('Deteksi Kartu', frame)
77
78
79     # MEMBERIKAN KONTUR WARNA MERAH DAN
80     # HITAM
81     # Mengubah ke skema warna HSV
82     hsv_warped = cv2.cvtColor(warped,
83         cv2.COLOR_BGR2HSV)
84
85     # Definisikan rentang warna merah
86     # dan hitam di HSV
87     lower_red = np.array([0, 120, 70])
88     upper_red = np.array([10, 255, 255])
89
90     lower_black = np.array([0, 0, 0])
91     upper_black = np.array([179, 255,
92         30])
93
94     # Buat maska untuk warna merah dan
95     # hitam
96     mask_red = cv2.inRange(hsv_warped,
97         lower_red, upper_red)
98     mask_black = cv2.inRange(hsv_warped,
99

```

```

    lower_black , upper_black)

82
83     # Gabungkan maska warna merah dan
84         hitam
85     mask_color = cv2.bitwise_or(mask_red
86         , mask_black)
87     kernel = np.ones((3, 3), np.uint8)
88     mask_color = cv2.erode(mask_color,
89         kernel, iterations=3)
90     mask_color = cv2.dilate(mask_color,
91         kernel, iterations=7)

92     # Temukan kontur pada warna yang
93         terdeteksi
94     contours_color, _ = cv2.findContours(
95         mask_color, cv2.RETR_EXTERNAL,
96             cv2.CHAIN_APPROX_SIMPLE)

97     # Loop melalui setiap kontur warna
98         yang terdeteksi
99     for contour in contours_color:
100         area = cv2.contourArea(contour)
101         if area > 100: # Sesuaikan
102             dengan threshold area yang
103                 sesuai
104             # Tentukan warna berdasarkan
105                 maska yang terdeteksi
106             if np.any(cv2.bitwise_and(
107                 mask_red, mask_black)):
108                 color_detected = ""
109                     Campuran Merah dan
                         Hitam"
110
111             # np.any digunakan untuk
112             elif np.any(mask_red):
113                 color_detected = "Merah"
114             elif np.any(mask_black):
115                 color_detected = "Hitam"

116
117             # Gambar kotak mengelilingi
118                 kontur warna hitam dan
119                     merah
120             rect = cv2.minAreaRect(
121                 contour)
122             box = cv2.boxPoints(rect)
123             box = np.int0(box)
124             cv2.drawContours(warped, [

```

```

110                                         box] , 0, (0, 255, 0), 2)
111
112                                         # Tambahkan teks pada kontur
113                                         cv2.putText(warped, f"Warna
114                                         adalah: {color_detected
115                                         }", (box[0][0], box
116                                         [0][1]), cv2.
117                                         FONT_HERSHEY_SIMPLEX,
118                                         0.5, (255, 255, 255), 2)
119
120                                         # Tampilkan gambar warped yang telah
121                                         ditandai dengan kontur persegi
122                                         panjang warna hitam dan merah
123                                         cv2.imshow('Deteksi Warna dengan
124                                         Teks', warped)

```

4. def readInput(self):

Fungsi readInput merupakan bagian dari sebuah program yang mengatur respons terhadap input yang diterima dari pengguna dalam bentuk tombol yang ditekan pada keyboard. Fungsi ini menggunakan modul msvcrt untuk memeriksa adanya tombol yang ditekan tanpa mengharuskan program untuk menunggu input dari pengguna. Jika ada tombol yang ditekan, program akan menjalankan serangkaian aksi tergantung pada karakter yang diterima dari fungsi msvcrt.getch().

Karakter yang diterima, seperti b'q', b'e', b'w', b'p', dan b'z', memiliki respons yang berbeda. Penggunaan b sebelum karakter dalam Python menandakan bahwa karakter tersebut adalah representasi dari bytes literals (literal byte). Dalam Python, b digunakan untuk menunjukkan bahwa nilai yang diberikan adalah byte data atau byte string, bukan string biasa yang terdiri dari karakter unicode. Karakter yang dimulai dengan b diikuti dengan tanda kutip tunggal atau ganda (misalnya, b'q', b'e', b'w', b'p', b'z') menunjukkan bahwa karakter tersebut adalah representasi dari byte sesuai dengan ASCII atau nilai byte lainnya, bukan karakter unicode atau string biasa. Dalam konteks penggunaan modul msvcrt.getch(), fungsi ini mengembalikan nilai byte yang mewakili karakter yang ditekan pada keyboard. Oleh karena itu, karakter-karakter yang diterima oleh getch() adalah byte literals (b'character') yang dapat dibandingkan dengan nilai-nilai byte lainnya untuk menentukan respons yang sesuai dalam program.

Jika karakter yang diterima adalah b'q', program akan mengembalikan string 'exit'. Jika karakter adalah b'e', program akan memanggil fungsi self.takeCard() untuk mengambil kartu. Selanjutnya, program akan meminta input dari pengguna (save_card) untuk menentukan apakah kartu

yang diambil akan disimpan dalam kartu pemain (self.player_cards) atau kartu komputer (self.computer_cards) berdasarkan input 'j' atau 'k'. Selain itu, karakter b'w' akan menjalankan fungsi self.saveData() untuk menyimpan data, karakter b'p' akan menampilkan kartu pemain, komputer, dan kartu yang terbuka menggunakan fungsi utils.display_cards, dan karakter b'z' akan memulai permainan dengan menjalankan fungsi self.play().

```
1 # Fungsi yang mengatur bagian tombol
2     def readInput(self):
3         if msvcrt.kbhit():
4             inp = msvcrt.getch()
5
6             if inp == b'q':
7                 return 'exit'
8             elif inp == b'e':
9                 card = self.takeCard()
10
11            if card:
12                # Mencocokkan label benar, kartu
13                # dapat disimpan ke dalam
14                # pemain atau komputer
15                save_card = input("Press 'j' to
16                add to player or 'k' to add
17                to computer : ")
18                if save_card == 'j': # Saat
19                    tombol 'j' ditekan (untuk
20                    pemain)
21                    self.player_cards.append(
22                        card)
23                    print("Card added to player
24                    cards.")
25
26                elif save_card == 'k': # Saat
27                    tombol 'k' ditekan (untuk
28                    komputer)
29                    self.computer_cards.append(
30                        card)
31                    print("Card added to
32                    computer cards.")
33
34
35            elif inp == b'w':
36                self.saveData()
37            elif inp == b'p':
38                utils.display_cards(self.
39                    player_cards, "Player")
40                utils.display_cards(self.
```

```

26         computer_cards , "Computer")
27     utils . display_cards ( self . open_cards ,
28             "Open")
27     elif inp == b'z':
28         return self . play ()

```

5. **def takeCard(self):**

Fungsi ‘takeCard’ bertanggung jawab untuk mengambil gambar yang telah diberi label dan menyimpannya ke dalam list pemain atau komputer. Pertama, fungsi ini menyimpan gambar yang telah diberi label ke file ‘tmp.jpg’. Kemudian, gambar tersebut dibaca kembali menggunakan OpenCV, diubah ke dalam bentuk array, dan dinormalisasi dengan membaginya dengan 255. Setelah itu, gambar diubah menjadi ukuran 128x128 piksel dan direformasi ke dalam bentuk yang sesuai untuk dapat diproses oleh model. Selanjutnya, model yang telah dimuat sebelumnya digunakan untuk melakukan prediksi terhadap gambar yang telah diubah tersebut. Dari hasil prediksi, fungsi ini akan mengidentifikasi label yang memiliki nilai prediksi tertinggi dan menampilkan label tersebut. Setelah menampilkan label, pengguna diminta untuk memastikan apakah label yang diprediksi sudah benar atau tidak dengan menanyakan pertanyaan kepada pengguna. Jika pengguna mengonfirmasi bahwa label yang diprediksi sudah benar, maka gambar beserta labelnya akan disimpan dalam bentuk objek kartu yang memiliki atribut ‘image’ dan ‘label’. Jika tidak, program akan mencetak pesan bahwa label yang diprediksi belum dikonfirmasi dan mengembalikan nilai ‘None’.

```

1 # Fungsi untuk mengambil gambar yang telah diberikan
2     label ke list pemain
3     def takeCard ( self ) :
4         cv2 . imwrite ( f 'tmp . jpg ' , self . Past )
5         img = np . array ( cv2 . imread ( 'tmp . jpg ' ) )
6         img = img / 255
7         img = np . array ( cv2 . resize ( img , ( 128 , 128 ) ) )
8         img = img . reshape ( 1 , 128 , 128 , 3 )
9
9     # Prediksi
10    label_prediction = self . model . predict ( img )
11
12    # Mendapatkan indeks label dengan nilai
13        maksimum prediksi
14    predicted_label_index = np . argmax (
15        label_prediction )
14    # Menampilkan label yang sesuai dengan
15        indeks prediksi
15    predicted_label = utils . labels [
16        predicted_label_index ]

```

```

16         print(predicted_label)
17
18
19     # Jika predicted_label sudah benar,
20     # lanjutkan dengan menyimpan kartu
21     label_correct = input(" Is the predicted
22     label correct? (y/n): ")
23     if label_correct.lower() == 'y':
24         card = {'image': self.Past, 'label':
25             predicted_label}
26
27         self.Past = None
28         predicted_label = None
29         return card
30     else:
31         print(" Label not confirmed. ")
32         return None

```

6. def saveData(self):

Fungsi saveData() pada program tersebut bertanggung jawab untuk menyimpan data gambar ke dalam dataset berdasarkan kondisi tertentu. Pertama, fungsi ini memeriksa jumlah foto yang telah diambil (self.photos_taken). Jika jumlah foto yang telah diambil sudah mencapai atau melebihi 20, variabel label_state akan ditambah 1, menunjukkan perpindahan ke label berikutnya dalam dataset. Fungsi akan mencetak pesan yang memberitahukan perpindahan tersebut dengan menampilkan label baru yang dipilih. Selanjutnya, fungsi akan memeriksa apakah direktori untuk label yang ditentukan sudah ada atau belum. Jika belum ada, fungsi akan membuat direktori baru sesuai dengan label yang sedang diproses. Setelah itu, fungsi akan mengambil foto menggunakan cv2.imwrite() dengan format penamaan yang sesuai berdasarkan label saat ini dan jumlah foto yang telah diambil. Kemudian, variabel self.photos_taken akan ditambah 1 untuk menandai pengambilan foto selanjutnya.

```

1 # Fungsi untuk Membuat Datasheet
2     def saveData(self):
3         # Cek apakah sudah diambil 10 foto , jika iya
4         # , pindah ke label berikutnya
5         if self.photos_taken >= 20:
6             self.photos_taken = 0 # Reset jumlah
7             foto yang diambil
8             label_state += 1 # Pindah ke label
9             berikutnya
10            print(f'Switched to label {utils.labels[
11                label_state]}')

```

```

9      # Cek apakah file ada
10     if not os.path.exists(f'{utils.labels[
11         label_state]}/'):
12         os.mkdir(f'{utils.labels[label_state]}')
13
14     # Mengambil foto
15     cv2.imwrite(f'{utils.labels[label_state]}/{{
16         utils.labels[label_state]}_self.
17         photos_taken}.jpg', self.Past)
18     self.photos_taken += 1 # Menambah jumlah
19     foto yang telah diambil
20
21     print(f'Photo {self.photos_taken} taken for
22         label {utils.labels[label_state]}'')

```

7. def play(self):

Fungsi play yang diberikan merupakan inti dari permainan yang sedang dimainkan. Saat dipanggil, permainan dimulai dengan pencetakan header yang menandakan dimulainya permainan. Variabel state bertindak sebagai penanda giliran pemain, di mana nilai 0 menunjukkan giliran komputer dan nilai 1 menunjukkan giliran pemain.

Selama permainan berlangsung, program membaca input dari pemain (tombol yang ditekan) dan menangani berbagai skenario. Input dari pemain dipantau melalui msvcrt.kbhit() dan msvcrt.getch() untuk memastikan responsifitas terhadap aksi pemain tanpa harus menghentikan jalannya program. Bagian awal dari fungsi tersebut menampilkan kartu-kartu yang dimiliki oleh pemain, komputer, dan kartu terbuka. Jika tidak ada kartu terbuka, pemain diminta untuk menekan tombol 'u' untuk membuka kartu. Selanjutnya, program membagi giliran antara komputer dan pemain. Komputer memeriksa dan mengambil kartu yang sesuai dengan kartu terbuka, jika tidak ada kartu yang sesuai, maka komputer akan mengambil kartu dari tumpukan kartu hingga menemukan yang sesuai.

Selama giliran pemain, pemain dapat memilih untuk menghapus kartu atau mengambil kartu baru. Jika pemain memilih untuk menghapus kartu, program meminta input untuk memilih kartu yang sesuai dengan kartu terbuka. Jika pemain memilih untuk mengambil kartu baru, pemain diminta untuk menekan tombol 't' untuk mengambil kartu baru. Program juga mengatur kondisi-kondisi lain, seperti ketika jumlah kartu terbuka mencapai tiga, semua kartu terbuka akan dipindahkan ke tumpukan sampah dan giliran dikembalikan kepada komputer. Juga, saat kondisi permainan berakhir (kartu pemain atau komputer habis), program akan menampilkan pemenang berdasarkan total prioritas terendah

antara kartu-kartu pemain dan komputer. Jika jumlah total prioritas kartu pemain dan komputer sama, akan dianggap seri. Fungsi ini berjalan dalam loop hingga kondisi permainan berakhir dan mencetak pesan "Game Over" di akhir permainan. Fungsi ini secara efektif mengatur alur permainan, memungkinkan interaksi antara pemain dan komputer untuk mengambil atau memindahkan kartu, serta menentukan pemenang berdasarkan kriteria yang ditetapkan.

```

1 # Game Dimulai
2     def play(self):
3         print('##### GAME STARTED #####')
4
5         self.game_started = True
6
7         # 0 = com turn , 1 = player turn
8         state = 0
9
10        while True:
11            ret, frame = self.cap.read()
12            self.readImage(frame)
13
14            if cv2.waitKey(1) & 0xFF == ord('q'):
15                break
16
17            utils.display_cards(self.player_cards, "Player")
18            utils.display_cards(self.computer_cards, "Computer")
19            utils.display_cards(self.open_cards, "Open")
20
21            if len(self.open_cards) == 0:
22                print('Tekan tombol U membuka kartu ...')
23                # Menunggu tombol 'u' ditekan
24                membuka kartu baru
25                while True:
26                    ret, frame = self.cap.read()
27                    self.readImage(frame)
28
29                    if cv2.waitKey(1) & 0xFF == ord('q'):
30                        break
31
32

```

```

33         if msvcrt.kbhit() and msvcrt.
34             getch() == b'u':
35                 card = self.takeCard()
36                 self.open_cards.append(card)
37                 print(" Card added to open
38                     card .")
39
39 ######
40 # Computer #
41 #####
42 if state == 0:
43     print(" Computer Take it First!")
44     # OpenCard hanya ada satu kartu dan
45     # ketika sudah berjumlah 3 ia akan
46     # dipindahkan ke trash
47     label_to_remove = self.open_cards
48         [0][ 'label '].split(' - ')[-1]
49
50 precedence = {}
51 # Pilih kamus precedence berdasarkan
52 # jenis bentuk
53 if label_to_remove == 'Club':
54     precedence = utils.
55         precedence_Club
56 elif label_to_remove == 'Diamond':
57     precedence = utils.
58         precedence_Diamond
59 elif label_to_remove == 'Heart':
60     precedence = utils.
61         precedence_Heart
62 elif label_to_remove == 'Spades':
63     precedence = utils.
64         precedence_Spades
65
66 matching_cards = []
67 for c in self.computer_cards:
68     # cek seluruh indeks pada list
69     # computer_cards yang
70     # berakhiran label_to_remove
71     if c[ 'label '].endswith(
72         label_to_remove):
73         # Jika terdapat kondisi
74         # diatas maka akan di
75         # tambahkan pada list
76         matching_cards

```

```

63                         matching_cards.append(c)

64

65             # Keadaan ketika matching_cards
66             # tidak ditemukan pada list
67             computer_cards
68             if len(matching_cards) == 0:
69                 # Komputer tidak memiliki kartu
70                 # yang sesuai , maka ambil kartu
71                 # sampai mendapatkan yang
72                 # sesuai
73                 print(f"Computer doesn't have a
74                     matching card for {
75                         label_to_remove}. Press 't',
76                     to take a card.")
77             # Menunggu tombol 't' ditekan
78             # sebelum mengambil kartu
79             while True:
80                 if self.checkEnd():
81                     break

82

83                 ret , frame = self.cap.read()
84                 self.readImage(frame)

85                 if cv2.waitKey(1) & 0xFF ==
86                     ord('q'):
87                     break

88                 if msvrt.kbhit() and msvrt
89                     .getch() == b't':
90                     c = self.takeCard()

91

92                 if c:
93                     # Jika komputer
94                     # berhasil
95                     # mengambil kartu
96                     # yang sesuai ,
97                     # keluar dari loop
98                     if c['label'].split
99                         ('_')[1] ==
100                         label_to_remove:
101                             print("Computer
102                                 got a
103                                 matching card
104                                 !")
105             # Hapus kartu
106             # dengan label

```

```

88         yang sama
89         dari self.
90         computer_cards
91         dengan
92         precedence
93         terbesar pada
94         jenis bentuk
95         tertentu
# Pindahkan
# Kartu-kartu
# yang ingin
# dihapus ke
# dalam list
# self.trash
self.open_cards.
append(c)
# Komputer
# memiliki
# kartu yang
# sesuai
print(f"Computer
has a
matching card
for {
label_to_
remove}.")

break
else:
    self.
    computer_cards
    .append(card)
print(f"Computer
doesn't have
a matching
card for {
label_to_
remove}.
Press 't' to
take a card
.")

else:
# Hapus kartu dengan label yang
# sama dari self.computer_cards
# dengan precedence terbesar
# pada jenis bentuk tertentu

```

```

99         max_precedence = -1
100        max = 0
101        for i in range(len(self.
102            computer_cards)):
103            card = self.computer_cards[i
104                ]
105                if card[ 'label '].split( ' - ')
106                    [1] == label_to_remove
107                    and precedence[card[ 'label ']] > max:
108                        max_precedence = i
109                        max = precedence[card[ 'label ']]
110
111
112
113
114        state = 1
115
116        #####
117        # PLAYER TURN #
118        #####
119        elif state == 1:
120            # Player Turn
121            while True:
122                if self.checkEnd():
123                    break
124
125                # Pilih Menu yang ingin diambil
126                print("Menu:")
127                print("1. Remove a card")
128                print("2. Take a card")
129                player_choice = input("Enter
130                    your choice (1 or 2): ")
# Jika pemain memilih untuk

```

```

131         menghapus kartu
132     if player_choice == '1':
133         # Setelah input dari pemain
134         untuk memilih kartu yang
135         ingin dihapus
136         selected_card_index = int(
137             input("Enter the index of
138                 the card you want to
139                 choose: ")) - 1
140
141         # Pilih kartu yang ingin
142         dihapus berdasarkan
143         indeks yang dimasukkan
144         pemain
145         selected_card = self.
146             player_cards[
147                 selected_card_index]
148
149         # Periksa jenis bentuk kartu
150         yang dipilih pemain
151         selected_card_shape =
152             selected_card['label']
153             .split('-')[-1]
154
155         # Mencari kartu yang sesuai
156         dengan jenis bentuk pada
157         self.open_cards
158         if self.open_cards[0]['label']
159             .endswith(
160                 selected_card_shape):
161             # Memindahkan kartu dari
162                 self.player_cards ke
163                 self.trash
164             self.open_cards.append(
165                 self.player_cards.pop(
166                     selected_card_index)
167             )
168             print(f"Card {
169                 selected_card_index +
170                     1} moved from player
171                     's cards to open card
172                         .")
173             state = 0
174             break
175         else:
176             print(" Selected card

```

```
      doesn't match any
      shape in open cards.
      Please choose another
      card.”)

150
151     if player_choice == '2':
152         print("Press 't' to take a
153             card.")
154         while True:
155             ret, frame = self.cap.
156                 read()
157                 self.readImage(frame)
158
159
160         if cv2.waitKey(1) & 0xFF
161             == ord('q'):
162             break
163
164         if msvcrt.kbhit() and
165             msvcrt.getch() == b't
166             ':
167                 card = self.takeCard
168                     ()
169
170                 if card:
171                     self.
172                         player_cards.
173                             append(card)
174                         print("Card
175                             added to
176                             player cards
177                             .")
178                         break
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
998
999
```

```

180
181     if self.checkEnd():
182         print("Game Over!")
183         utils.display_cards(self.
184             player_cards, "Player :")
185         utils.display_cards(self.
186             computer_cards, "Computer :")
187         utils.display_cards(self.trash, "
188             Open :")
189 # Hitung total precedence dari
190 # setiap list kartu
191 total_player_precedence = utils.
192     calculate_total_precedence(self.
193         player_cards)
194 total_computer_precedence = utils.
195     calculate_total_precedence(self.
196         computer_cards)
197 total_trash_precedence = utils.
198     calculate_total_precedence(self.
199         trash)
200 # Tentukan pemenang berdasarkan
201 # total precedence terkecil di
202 # antara self.player_cards dan self.
203 # .computer_cards
204 if total_player_precedence <=
205     total_computer_precedence:
206     print("Congratulations! You win
207         !")
208 elif total_computer_precedence <=
209     total_player_precedence:
210     print("Computer wins!")
211 elif total_computer_precedence ==
212     total_player_precedence:
213     # Ketika jumlah kartu pada pada
214     # print("Draw !")
215     break
216
217 if len(self.player_cards) == 0 and len(
218     self.computer_cards) == 0:
219     print("Draw !")
220     break
221 elif len(self.player_cards) == 0:
222     print("Player Wins")
223     break
224 elif len(self.computer_cards) == 0:
225     print("Computer Wins")

```

```

208     break
209
210     print ("Game Over")

```

8. **def checkEnd(self):** Fungsi ‘checkEnd‘ dalam kode bertugas untuk memeriksa apakah seluruh kartu dalam permainan telah digunakan atau tidak. Kondisi ini terpenuhi saat total jumlah kartu yang telah dibuang ('self.trash'), jumlah kartu pada tumpukan pemain ('self.player_cards'), dan jumlah kartu pada tumpukan komputer ('self.computer_cards') sama dengan 52, yaitu jumlah total kartu dalam satu set kartu standar. Fungsi ini melakukan pengecekan dengan cara menjumlahkan panjang (jumlah elemen) dari ketiga tumpukan kartu tersebut. Jika jumlah kartu mencapai 52, fungsi mengembalikan nilai ‘True‘, menandakan bahwa semua kartu telah terpakai dalam permainan. Sebaliknya, jika kondisi tersebut tidak terpenuhi, fungsi akan mengembalikan nilai ‘False‘, menandakan bahwa masih ada kartu yang belum digunakan dalam permainan. Hal ini berguna sebagai kondisi terminasi yang mengindikasikan berakhirnya permainan setelah seluruh kartu telah dimainkan.

```

1 # Kondisi 52 kartu telah terpakai
2     def checkEnd( self ):
3         if len( self . trash ) + len( self . player_cards )
4             + len( self . computer_cards ) == 52:
5                 return True
6
7         return False

```

Keempat adalah implementasi dari model jaringan saraf tiruan (neural network) menggunakan arsitektur Convolutional Neural Network (CNN) dengan menggunakan pustaka Keras. Tujuannya adalah untuk memproses dataset yang berisi gambar-gambar kartu dari berbagai jenis dan angka pada setumpuk kartu remi. Fungsi ModelDeepLearningCNN(JumlahKelas) mendefinisikan arsitektur CNN dengan lapisan-lapisan konvolusi dan lapisan terkait seperti Conv2D, MaxPooling2D, Flatten, dan Dense. Model ini dirancang untuk mengolah gambar berukuran 128x128 piksel dengan tiga saluran warna (RGB). Terdapat empat lapisan konvolusi yang masing-masing memiliki 32 filter, menggunakan aktivasi ReLU, dan diikuti oleh lapisan MaxPooling2D untuk mengurangi resolusi gambar. Output dari model menggunakan fungsi aktivasi softmax dengan jumlah kelas yang ditentukan oleh parameter JumlahKelas.

Selanjutnya, proses pemrosesan dataset dimulai dengan memuat gambar-gambar kartu dari direktori yang telah ditentukan. Gambar-gambar ini diubah ukurannya menjadi 128x128 piksel, dinormalisasi, dan diubah menjadi larik numpy. Selain itu, label pada dataset diubah menggunakan vektor one-hot encoding untuk setiap kelas pada gambar kartu.

Proses pelatihan model CNN dilakukan dengan menggunakan metode fit pada data masukan (x) dan label (y) yang telah diproses sebelumnya. Proses pelatihan ini dilakukan sebanyak JumlahEpoh yang telah ditentukan sebelumnya.

Kode yang diberikan mengimplementasikan model Convolutional Neural Network (CNN) menggunakan pustaka Keras untuk memproses dataset berisi gambar-gambar kartu remi. Fungsi ModelDeepLearningCNN(JumlahKelas) mendefinisikan arsitektur CNN dengan lapisan-lapisan konvolusi dan lapisan terkait seperti Conv2D, MaxPooling2D, Flatten, dan Dense. Model ini dirancang untuk mengolah gambar berukuran 128x128 piksel dengan tiga saluran warna (RGB). Terdapat empat lapisan konvolusi yang masing-masing memiliki 32 filter, menggunakan aktivasi ReLU, dan diikuti oleh lapisan MaxPooling2D untuk mengurangi resolusi gambar. Output dari model menggunakan fungsi aktivasi softmax dengan jumlah kelas yang ditentukan oleh parameter JumlahKelas.

Selanjutnya, proses pemrosesan dataset dimulai dengan memuat gambar-gambar kartu dari direktori yang telah ditentukan. Gambar-gambar ini diubah ukurannya menjadi 128x128 piksel, dinormalisasi, dan diubah menjadi larik numpy. Selain itu, label pada dataset diubah menggunakan vektor one-hot encoding untuk setiap kelas pada gambar kartu. Proses pelatihan model CNN dilakukan dengan menggunakan metode fit pada data masukan (x) dan label (y) yang telah diproses sebelumnya. Proses pelatihan ini dilakukan sebanyak JumlahEpoh yang telah ditentukan sebelumnya.

Hasil dari proses pembelajaran disimpan ke dalam sebuah file dengan nama FileBobot dalam format H5, yang merupakan format standar untuk menyimpan model Keras. Terakhir, grafik loss dan akurasi dari proses pelatihan ditampilkan menggunakan pustaka matplotlib, menampilkan nilai loss dan akurasi pada sumbu y dan jumlah epok pada sumbu x. Keseluruhan fungsi ini bertujuan untuk membuat, melatih, dan menyimpan model CNN yang mampu mengklasifikasikan gambar-gambar kartu remi ke dalam berbagai label yang telah ditentukan sebelumnya.

```
1 import os
2 from keras.models import load_model
3 import cv2
4 import numpy as np
5 from keras.layers import Input, Dense
6 from keras.layers import Conv2D, MaxPooling2D,
7 Flatten
8 from keras.models import Model
9 import matplotlib.pyplot as plt
from datetime import datetime
```

```

10     from numpy import expand_dims
11     from keras.utils import load_img
12     from keras.utils import img_to_array
13     from keras.preprocessing.image import
14         ImageDataGenerator
15     from matplotlib import pyplot
16
17     def ModelDeepLearningCNN(JumlahKelas):
18         # Ukuran citra 128x128 piksel dan tiga
19             saluran warna (RGB).
20         input_img = Input(shape=(128, 128, 3))
21         # Menambahkan layer konvolusi 32 filter
22             dengan kernel 3*3
23         # ReLU (Rectified Linear Activation)
24         # ReLU mengonversi nilai input menjadi nol
25             # jika nilainya kurang dari nol,
26             # dan mempertahankan nilai inputnya
27             # jika nilainya lebih besar dari nol.
28         x = Conv2D(32, (3, 3), activation='relu',
29             padding='same')(input_img)
30         x = MaxPooling2D((2, 2), padding='same')(x)
31         x = Conv2D(32, (3, 3), activation='relu',
32             padding='same')(x)
33         x = MaxPooling2D((2, 2), padding='same')(x)
34         x = Conv2D(32, (3, 3), activation='relu',
35             padding='same')(x)
36         x = MaxPooling2D((2, 2), padding='same')(x)
37         x = Conv2D(32, (3, 3), activation='relu',
38             padding='same')(x)
39         x = Flatten()(x)
40         x = Dense(100, activation='relu')(x)
41         x=Dense(JumlahKelas, activation='softmax')(x)
42         ModelCNN = Model(input_img, x)
43         ModelCNN.compile(loss='
44             categorical_crossentropy', optimizer='
45                 adam', metrics=['accuracy'])
46         #ModelCNN.compile(loss='mse', optimizer='
47             adam', metrics=['accuracy'])
48         return ModelCNN
49
50
51
52     DirektoriDataSet="E:\\\\University\\\\Material\\\\
53         Semester 5\\\\Pengolahan Citra Video\\\\Tugas
54             Akhir"
55
56     # Data Set disimpan dalam direktori yang sama

```

```

dengan nama kelas

44
45 #b. Label Data Set
46 LabelKelas = (
47     "2_Club", "3_Club", "4_Club", "5_Club",
48     "6_Club", "7_Club", "8_Club", "9_Club",
49     ", "10_Club", "ace_Club",
50     "J_Club", "Q_Club", "K_Club", "2_Diamond",
51     ", "3_Diamond", "4_Diamond", "5
52     _Diamond", "6_Diamond", "7_Diamond",
53     ", "8_Diamond", "9_Diamond", "10_Diamond",
54     ", "ace_Diamond", "J_Diamond", "
55     Q_Diamond", "K_Diamond", "2_Heart",
56     ", "3_Heart", "4_Heart", "5_Heart", "6
57     _Heart", "7_Heart", "8_Heart", "9
58     _Heart", "10_Heart", "ace_Heart",
59     "J_Heart", "Q_Heart", "K_Heart", "2
60     _Spades", "3_Spades", "4_Spades", "5
61     _Spades", "6_Spades", "7_Spades",
62     ", "8_Spades", "9_Spades", "10_Spades",
63     ", "ace_Spades", "J_Spades", "Q_Spades",
64     ", "K_Spades"
65 )
66
67 x = []
68 y = []
69
70 # One-hot Encoding ada 52 kelas vektor [1. 0.
71     0. (hingga 52 angka)]
72 t = np.eye(len(LabelKelas))
73
74 for i in range(len(LabelKelas)):
75     label = LabelKelas[i]
76
77     if not os.path.exists(f'{label}/'):
78         continue
79
80     j = 0
81     while os.path.exists(f'{label}/{label}-{j}.jpg'):
82         filename = f'{label}/{label}-{j}.jpg'
83         img = np.double(cv2.imread(filename))
84         img = cv2.resize(img,(128,128))
85         img = np.asarray(img)/255
86         img = img.astype('float32')
87         x.append(img)
88
89
90
91
92
93
94

```

```

75
76         y.append( t[ i ] )
77
78         j+=1
79
80         x = np.array( x ).astype( 'float32' )
81         y = np.array( y ).astype( 'float32' )
82
83 # # CEK APAKAH DATASET DAN LABEL BENAR
84 # print(" Jumlah dataset (gambar):" , len(x))
85 # print(" Jumlah dataset (label):" , len(y))
86
87 # for i in range(len(LabelKelas)):
88 #     label = LabelKelas[ i ]
89
90 #     if not os.path.exists(f'{label}/'):
91 #         continue
92
93 #     j = 0
94 #     while os.path.exists(f'{label}/{label}-{j}.jpg'):
95 #         filename = f'{label}/{label}-{j}.jpg'
96 #         print(" File Gambar:" , filename)
97 #         print(" Label:" , label)
98
99 #         j += 1
100
101 #c. Inisialisasi parameter Training
102 JumlahEpoh = 10
103 FileBobot = "Fix.h5"
104 #d. training
105 modelCNN =ModelDeepLearningCNN( len( LabelKelas ) )
106 #Trainng
107 history=modelCNN.fit( x , y , epochs=JumlahEpoh ,
108                         shuffle=True )
109 #Menyimpan hasil learning
110 modelCNN.save( FileBobot )
111
112 #Mengembalikan output
113 modelCNN.summary()
114 model = Model.load_weights
115
116 #c. Menampilkan Grafik Loss dan accuracy
117 plt.plot(history.history[ 'loss' ])
118 plt.plot(history.history[ 'accuracy' ])

```

```

119     plt.title('model loss')
120     plt.ylabel('loss/accuracy')
121     plt.xlabel('epoch')
122     plt.legend(['loss', 'acc'], loc='upper left')
123     plt.show()

```

2.3 Hasil Output Program

Berikut ini adalah hasil ketika game dijalankan dengan pembagian 2 kartu disetiap player dan komputer. Hal pertama yang dilakukan adalah mengambil kartu dan meletakkannya pada list player, kemudian baru komputer lalu game baru dapat dimulai.

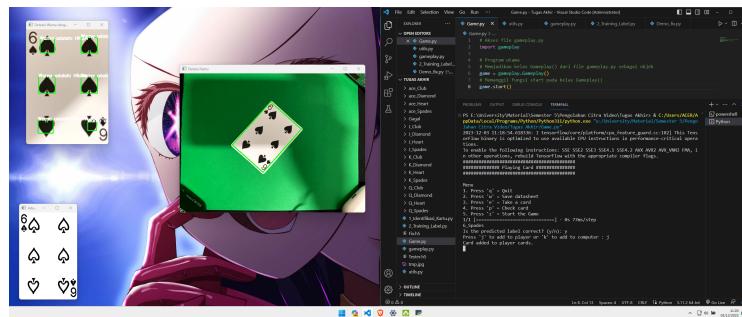


Figure 1: Player menerima 6_Spade

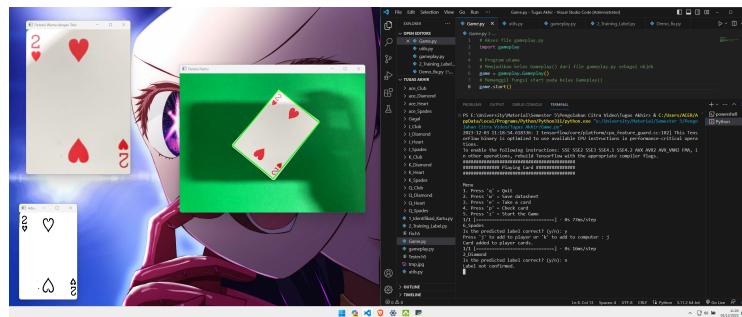


Figure 2: Ketika kartu yang diambil tidak sesuai

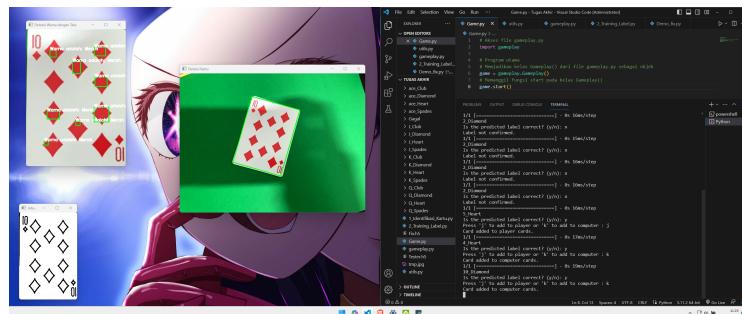


Figure 3: Komputer menerima 10_Diamond

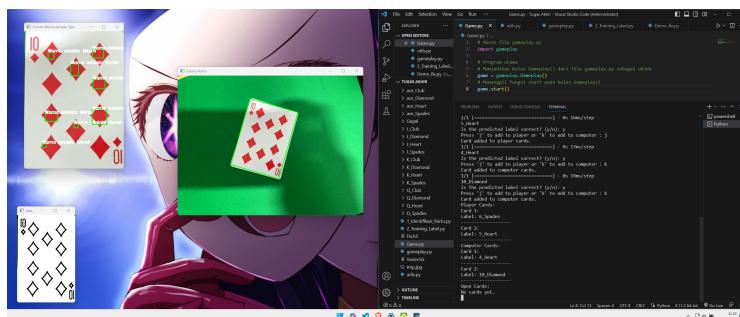


Figure 4: Semua pemain mendapatkan 2 kartu

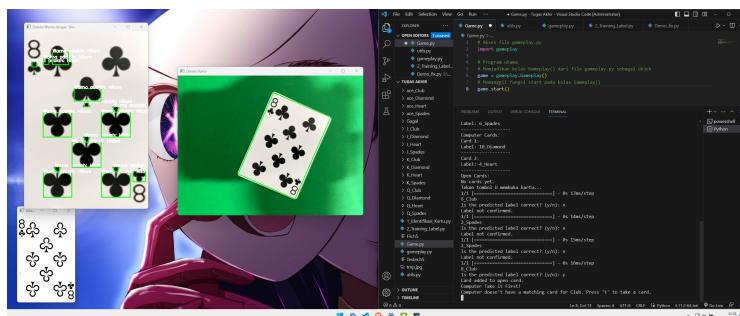


Figure 5: Komputer tidak memiliki jenis kartu yang sama dengan open_cards

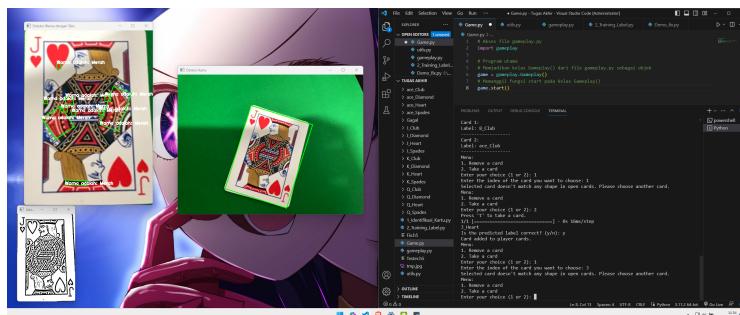


Figure 6: Giliran player bermain

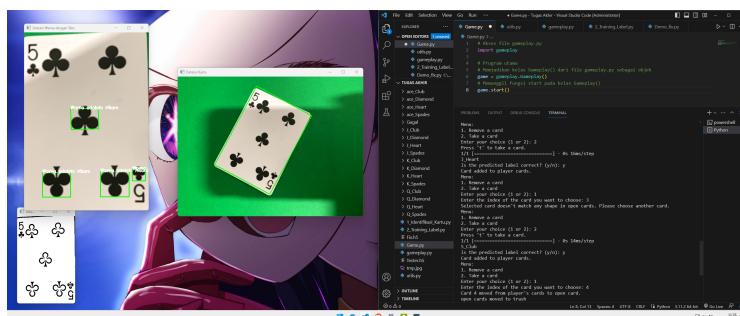


Figure 7: Membuang 3 kartu open_cards ke dalam list trash

3 Penutupan

3.1 Kesimpulan

Dalam tugas mata kuliah pengolahan citra video, telah diimplementasikan sebuah program yang menggunakan teknologi pemrosesan citra dan machine learning untuk membuat permainan kartu interaktif. Program tersebut memiliki tiga bagian utama yang saling terkait untuk menciptakan pengalaman bermain yang menarik. Pertama, program mengakses kamera untuk mendeteksi kartu yang diletakkan di depannya. Dengan menggunakan teknik pemrosesan citra seperti Canny Edge Detection, deteksi kontur, transformasi perspektif, dan thresholding adaptif, program dapat secara akurat memisahkan kartu dari latar belakang dan mendeteksi bentuk serta warna kartu yang ada. Kedua, program menggunakan model machine learning untuk mengidentifikasi dan memvalidasi kartu yang telah diambil dari kamera. Dengan memanfaatkan model yang telah dilatih sebelumnya, program dapat memprediksi jenis kartu yang diambil, memeriksa kebenaran prediksi tersebut, dan menyimpan informasi kartu ke dalam list pemain atau komputer sesuai dengan keputusan pengguna. Terakhir, program mengimplementasikan logika permainan kartu yang dinamis. Dengan menyediakan berbagai opsi untuk pengguna, seperti mengambil kartu, menyimpan data foto, mengecek kartu, atau memulai permainan, program mengintegrasikan interaksi pengguna dengan alur permainan yang cerdas. Melalui logika permainan yang terstruktur dengan baik, termasuk penentuan giliran pemain dan komputer, pengelolaan kartu, dan penentuan pemenang berdasarkan nilai precedence kartu, program menciptakan pengalaman bermain yang menyenangkan dan menantang.