

# Laporan Final Project

## Implementasi Permainan Kartu Domino Menggunakan Convolutional Neural Network (CNN) untuk Permainan Manusia - Mesin

Agus Rasi Doanta Ginting  
NRP 5024211018  
Pengolahan Citra Video A

17 Desember, 2023

## 1 Pendahuluan

Dalam semester gasal 2023 saat mengikuti mata kuliah Pengolahan Citra Visual (PCV), mahasiswa diberikan tugas proyek akhir untuk menciptakan sebuah permainan kartu yang menggunakan kamera. Proyek ini melibatkan penerapan teknologi pemrosesan citra dengan filter yang telah dipelajari selama satu semester menggunakan bahasa pemrograman Python dan pustaka OpenCV.

## 2 Dasar Teori

### 2.1 Domino

Permainan kartu domino melibatkan aturan dasar di mana pemain mencocokkan kartu dengan jumlah bulatan yang sama. Strategi permainan terfokus pada pengelolaan kartu dan penempatan yang optimal. Teori probabilitas membantu memprediksi kartu yang tersisa, sementara teori game menganalisis skenario dan strategi terbaik. Matematika dasar, seperti penjumlahan bulatan di kartu, juga penting dalam memahami potensi skor dan peluang permainan. Kombinasi dari aturan, strategi, probabilitas, teori game, dan matematika dasar membantu pemain mengambil keputusan yang lebih baik saat bermain kartu domino.

### 2.2 OpenCV

Pemrosesan citra digital menggunakan OpenCV merupakan praktik penggunaan perpustakaan sumber terbuka yang terfokus pada manipulasi, analisis, dan pemahaman gambar secara komputerisasi. OpenCV memungkinkan pengguna untuk melakukan sejumlah operasi dasar seperti resize, rotasi, deteksi tepi, serta aplikasi filter untuk menghilangkan

noise atau mengekstraksi fitur penting dari citra. Dengan dukungan algoritma yang luas, termasuk deteksi objek, pengenalan pola, dan analisis video, OpenCV memainkan peran penting dalam bidang-bidang seperti visi komputer, penglihatan mesin, dan pengenalan citra. Penggunaannya sering terkait dengan bahasa pemrograman seperti Python, memungkinkan para pengembang untuk membuat aplikasi yang melibatkan pengolahan citra secara efisien dalam berbagai skenario, mulai dari aplikasi medis hingga kecerdasan buatan dan kendaraan otonom.

## 2.3 Convolutional Neural Network (CNN)

CNN (Convolutional Neural Network) adalah jenis arsitektur jaringan saraf tiruan yang dirancang khusus untuk tugas-tugas pengolahan citra dan pengenalan pola. Dengan menggunakan lapisan konvolusi, CNN dapat mengekstraksi fitur-fitur penting dari data gambar, seperti tepi, tekstur, dan pola-pola visual lainnya. Lapisan pengelompokan digunakan untuk mengurangi dimensi spasial dari representasi gambar. Setelah melalui serangkaian lapisan konvolusi dan pengelompokan, hasilnya diteruskan ke lapisan terhubung penuh untuk klasifikasi atau regresi. Aktivasi seperti ReLU digunakan untuk menambahkan non-linearitas, dan pelatihan dilakukan melalui algoritma backpropagation. CNN telah menjadi arsitektur standar dalam berbagai aplikasi pengolahan citra, mulai dari pengenalan objek, klasifikasi gambar, hingga segmentasi gambar, karena kemampuannya dalam mengekstraksi fitur dari data gambar yang kompleks secara otomatis.

## 2.4 Proses Implementasi dalam Proyek

- Pengambilan Citra dari Kamera: Mengambil citra secara real-time dari kamera sebagai input permainan.
- Pengenalan Kartu: Menggunakan teknik pengolahan citra untuk mengenali dan memahami kartu yang ditampilkan dalam citra.
- Logika Permainan Blackjack: Implementasi permainan kartu domino dengan perubahan dimana pemain yang memiliki kartu terbesar yang pertama main. Dan hanya menggunakan satu kali penyesuaian untuk kartu yang dimasukkan, setelah itu kembali ke awal untuk memulai permainan sampai kartu habis.
- Filter Citra: Menambahkan filter gaussian blur, sobel, dan threshold untuk meningkatkan tampilan permainan

# 3 SourceCode

## 3.1 ModulKlasifikasiCitraCNN.py

```
1 #####  
2 # Progrm Ini dijadikan modul dengan nama
```

```

3      # ModulKlasifikasiCitraCNN.py
4      #####
5
6      import os
7      from keras.models import load_model
8      import cv2
9      import numpy as np
10     from keras.layers import Input, Dense
11     from keras.layers import Conv2D, MaxPooling2D, Flatten
12     from keras.models import Model
13     import matplotlib.pyplot as plt
14     from datetime import datetime
15     from numpy import expand_dims
16     from keras.utils import load_img
17     from keras.utils import img_to_array
18     from keras.preprocessing.image import ImageDataGenerator
19     from matplotlib import pyplot
20
21     # load the image
22     def LoadCitraTraining(sDir,LabelKelas):
23         JumlahKelas=len(LabelKelas)
24         TargetKelas = np.eye(JumlahKelas)
25         # Menyiapkan variabel list untuk data menampung citra
26         # dan data target
27         X=[] #Menampung Data Citra
28         T=[] #Menampung Target
29         for i in range(len(LabelKelas)):
30             #Membaca file citra di setiap direktori data set
31             DirKelas = os.path.join(sDir, LabelKelas[i])
32             files = os.listdir(DirKelas)
33             for f in files:
34                 ff=f.lower()
35                 print(f)
36                 #memilih citra dengan ekstensi jpg,jpeg,dan png
37                 if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.endswith
38                     ('.png')):
39                     NmFile = os.path.join(DirKelas,f)
40                     img= np.double(cv2.imread(NmFile,1))
41                     img=cv2.resize(img,(128,128))
42                     img= np.asarray(img)/255
43                     img=img.astype('float32')
44                     #Menambahkan citra dan target ke daftar
45                     X.append(img)

```

```

44 T.append(TargetKelas[i])
45 #-----akhir loop :Pfor f in files-----
46 #-----akhir loop :for i in range(len(LabelKelas))----
47
48 #Mengubah List Menjadi numpy array
49 X=np.array(X)
50 T=np.array(T)
51 X=X.astype('float32')
52 T=T.astype('float32')
53 return X,T
54
55 def ModelDeepLearningCNN(JumlahKelas):
56     input_img = Input(shape=(128, 128, 3))
57     x = Conv2D(32, (3, 3), activation='relu', padding='same'
58         )(input_img)
59     x = MaxPooling2D((2, 2), padding='same')(x)
60     x = Conv2D(32, (3, 3), activation='relu', padding='same'
61         )(x)
62     x = MaxPooling2D((2, 2), padding='same')(x)
63     x = Conv2D(32, (3, 3), activation='relu', padding='same'
64         )(x)
65     x = Flatten()(x)
66     x = Dense(100,activation='relu')(x)
67     x=Dense(JumlahKelas,activation='softmax')(x)
68     ModelCNN = Model(input_img, x)
69     ModelCNN.compile(loss='categorical_crossentropy',
70         optimizer='adam', metrics=['accuracy'])
71     #ModelCNN.compile(loss='mse', optimizer='adam', metrics
72     =['accuracy'])
73     return ModelCNN
74
75 def TrainingCNN(JumlahEpoh,DirektoriDataSet,LabelKelas,
76     NamaFileBobot):
77     #Membaca Data training dan label Kelas
78     X,D=LoadCitraTraining(DirektoriDataSet,LabelKelas)
79     JumlahKelas = len(LabelKelas)
80     #Membuat Model CNN
81     ModelCNN =ModelDeepLearningCNN(JumlahKelas)
82     #Trainng
83     history=ModelCNN.fit(X, D,epochs=JumlahEpoh,shuffle=True
84         )
85     #Menyimpan hasil learning
86     ModelCNN.save(NamaFileBobot)

```

```

80     #Mengembalikan output
81     return ModelCNN,history
82
83
84     def Klasifikasi(DirDataSet,DirKlasifikasi,LabelKelas,
85                     ModelCNN=[]):
86         #Menyiapkan Data input Yang akan di kasifikasikan
87         X=[]
88         ls = []
89         DirKelas = DirDataSet+"/"+DirKlasifikasi
90         print(DirKelas)
91         files = os.listdir(DirKelas)
92         n=0
93         for f in files:
94             ff=f.lower()
95             print(f)
96             if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.endswith
97                 ('.png')):
98                 ls.append(ff)
99                 NmFile = os.path.join(DirKelas,f)
100                 img= cv2.imread(NmFile,1)
101                 img=cv2.resize(img,(128,128))
102                 img= np.asarray(img)/255
103                 img=img.astype('float32')
104                 X.append(img)
105                 #----Akhir if-----
106                 #---Akhir For
107                 X=np.array(X)
108                 X=X.astype('float32')
109                 #Melakukan prediksi Klasifikasi
110                 hs=ModelCNN.predict(X)
111
112                 LKlasifikasi=[]
113                 LKelasCitra =[]
114                 n = X.shape[0]
115                 for i in range(n):
116                     v=hs[i,:]
117                     if v.max(>0.5:
118                         idx = np.max(np.where( v == v.max()))
119                         LKelasCitra.append(LabelKelas[idx])
120                     else:
121                         idx=-1
122                     LKelasCitra.append("-")

```

```

121     #-----akhir if
122     LKlasifikasi.append(idx)
123     #----akhir for
124     LKlasifikasi = np.array(LKlasifikasi)
125     return ls, hs, LKelasCitra
126
127     def LoadModel(sf):
128         ModelCNN=load_model(sf)
129         return ModelCNN
130
131     def ImageAugmentation(SPath,Kelas):
132         parent_dir = SPath
133         directory = Kelas
134         directoryExt =directory + "_ext"
135         sdirExt = os.path.join(parent_dir, directoryExt)
136         if not os.path.exists(sdirExt):
137             os.mkdir(sdirExt)
138         directory_image =directory
139         sDir = os.path.join(parent_dir, directory_image)
140         files = os.listdir(sDir)
141         ii=0
142         for f in files:
143             ff=f.lower()
144             if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.endswith(
145                 '.png')):
146                 print(ff)
147                 sfs= os.path.join(sDir,ff)
148                 img = load_img(sfs)
149                 img2 = np.array(img)
150                 sfn= os.path.join(sdirExt, ff)
151                 cv2.imwrite(sfn,img2)
152                 data = img_to_array(img)
153                 samples = expand_dims(data, 0)
154                 datagen = ImageDataGenerator(rotation_range=90,
155                     brightness_range=[0.2,2.0],zoom_range=[0.5,2.0],
156                     width_shift_range=0.2,height_shift_range=0.2)
157                 it = datagen.flow(samples, batch_size=1)
158                 for i in range(9):
159                     batch = it.next()
160                     image = batch[0].astype('uint8')
161                     now = datetime.now()
162                     ii=ii+1
163                     sf = now.strftime("%Y%m%d%H%M%S")+"_"+str(ii)+".jpg"

```

```
161 sfn= os.path.join(sdirExt, sf)
162 cv2.imwrite(sfn,image)
```

## 3.2 Buat Dataset

```
1  import cv2
2  import time
3  import os
4
5  def preprocess_image(image):
6      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
7      blurred = cv2.GaussianBlur(gray, (3, 3), 0)
8
9      # Filter Sobel untuk mendeteksi tepi
10     sobel_x = cv2.Sobel(blurred, cv2.CV_64F, 1, 0, ksize=3)
11     sobel_y = cv2.Sobel(blurred, cv2.CV_64F, 0, 1, ksize=3)
12
13     edge = cv2.bitwise_or(cv2.convertScaleAbs(sobel_x), cv2.
14                           convertScaleAbs(sobel_y))
15
16     # edges = cv2.Canny(edge, 50, 150)
17     _, edges = cv2.threshold(edge, 100, 255, cv2.
18                           THRESH_BINARY + cv2.THRESH_OTSU)
19
20     return edges, blurred
21
22 def detect_contours(edges, image):
23     contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
24                                   cv2.CHAIN_APPROX_SIMPLE)
25     detected_image = image.copy()
26     detected_x, detected_y = 0, 0 # Inisialisasi nilai x
27     dan y
28     cropped_images = []
29
30     for contour in contours:
31         perimeter = cv2.arcLength(contour, True)
32         approx = cv2.approxPolyDP(contour, 0.04 * perimeter,
33                                   True)
34
35     if len(approx) == 4:
36         x, y, w, h = cv2.boundingRect(contour)
37         aspect_ratio = float(w) / h
```

```

34     aspect_ratio_a = float(h) / w
35
36     if 0.3 <= aspect_ratio <= 0.8 or 0.3 <= aspect_ratio_a
37         <= 0.8:
38         cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0)
39             , 2)
40         detected_x, detected_y = x, y # Update nilai x dan y
41
42         # Crop gambar di dalam kontur yang terdeteksi
43         cropped_image = edges[y:y+h, x:x+w]
44         cropped_images.append(cropped_image)
45
46     return edges, detected_image, cropped_images, detected_x
47         , detected_y
48
49     def CreateDataSet(sDirektoriData, sKelas, NoKamera,
50         FrameRate, recording_duration):
51         sDirektoriKelas = os.path.join(sDirektoriData, sKelas)
52         if not os.path.exists(sDirektoriKelas):
53             os.makedirs(sDirektoriKelas)
54
55         cap = cv2.VideoCapture(NoKamera)
56         TimeStart = 0
57         Recording = False
58         recording_end_time = 0
59         x, y = 0, 0 # Inisialisasi nilai x dan y
60
61         while cap.isOpened():
62             success, image = cap.read()
63
64             if not success:
65                 print("Ignoring empty camera frame.")
66                 continue
67
68             cv2.imshow('Original', image)
69
70             key = cv2.waitKey(1) & 0xFF
71             if key == ord('q'):
72                 break
73             elif key == ord('t'):
74                 if not Recording:
75                     Recording = True
76                     TimeStart = time.time()

```

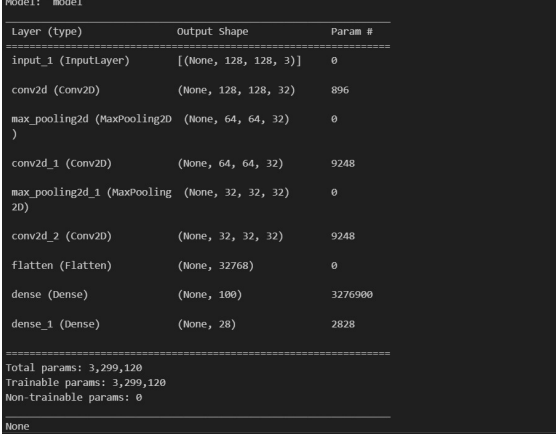


```

73     recording_end_time = TimeStart + recording_duration #
       Waktu akhir pencatatan
74
75     if Recording:
76         edges, detected_image, cropped_images, x, y =
           detect_contours(*preprocess_image(image))
77         cv2.imshow('Filtered_Edges', edges)
78         cv2.imshow('Detected_Contours', detected_image)
79
80         TimeNow = time.time()
81         if TimeNow - TimeStart > 1 / FrameRate and TimeNow <
           recording_end_time:
82             for i, cropped_image in enumerate(cropped_images):
83                 sfFile = os.path.join(sDirektoriKelas, f"cropped_{int(
                   TimeNow*1000)}_{x}_{y}_{i}.jpg")
84                 cv2.imwrite(sfFile, cropped_image) # Simpan gambar yang
                   di-crop
85
86                 cv2.imshow(f'Cropped_Image_{i}', cropped_image)
87
88             TimeStart = TimeNow
89
90             if TimeNow >= recording_end_time:
91                 print("Recording_duration_reached._Stopping_the_camera."
                   )
92                 break
93
94             cap.release()
95             cv2.destroyAllWindows()
96
97             if _name_ == "_main_":
98                 sDirektoriData = "D:/PCV/project/dataset"
99                 sKelas = "4x0"
100                 NoKamera = 1 # Menggunakan kamera 1, sesuaikan jika
                   menggunakan kamera lain
101                 FrameRate = 7
102                 recording_duration = 7 # Waktu pencatatan dalam detik
103
104                 # Inisialisasi variabel Recording
105                 Recording = False
106
107                 # Membuat dataset

```

```
CreateDataSet(sDirektoriData, sKelas, NoKamera,
              FrameRate, recording_duration)
```



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 128, 128, 3)]	0
conv2d (Conv2D)	(None, 128, 128, 32)	896
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 100)	3276900
dense_1 (Dense)	(None, 28)	2828

Total params: 3,299,120  
 Trainable params: 3,299,120  
 Non-trainable params: 0

Figure 1: Gambar Train

### 3.3 Train Dataset

```

1  import os
2  import cv2
3  import ModulKlasifikasiCitraCNN as mCNN
4  from sklearn.model_selection import train_test_split
5
6  sDir = "D:/PCV/project/dataset"
7  files = os.listdir(sDir)
8  for f in files:
9      print(f)
10
11  LabelKelas = ['0x0', '1x0', '1x1', '2x0', '2x1', '2x2',
12                '3x0', '3x1', '3x2', '3x3', '4x0', '4x1', '4x2', '4x3',
13                '4x4', '5x0', '5x1', '5x2', '5x3', '5x4', '5x5', '6x0',
14                '6x1', '6x2', '6x3', '6x4', '6x5', '6x6']
15
16  X, T = mCNN.LoadCitraTraining(sDir, LabelKelas)
17
18  JumlahEpoh = 10
```

```

19     # Tetapkan nama file untuk menyimpan model setelah
20     training
21     FileBobot = "domino2.h5"
22
23     # Lakukan training model dengan memanggil fungsi
24     TrainingCNN dari modul
25     ModelCNN, history = mCNN.TrainingCNN(JumlahEpoh, sDir,
26     LabelKelas, FileBobot)
27
28     # Tampilkan summary dari model setelah proses training
29     selesai
30     print(ModelCNN.summary())

```

### 3.4 Kode Game

```

1     from ModulKlasifikasiCitraCNN import LoadModel
2     import cv2
3     import numpy as np
4     import time
5
6     label_kelas = ['0x0', '1x0', '1x1', '2x0', '2x1', '2x2',
7     '3x0', '3x1', '3x2', '3x3', '4x0', '4x1', '4x2', '4
8     x3',
9     '4x4', '5x0', '5x1', '5x2', '5x3', '5x4', '5x5', '6x0',
10    '6x1', '6x2', '6x3', '6x4', '6x5', '6x6']
11
12
13    def preprocess_image(image):
14        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
15        blurred = cv2.GaussianBlur(gray, (3, 3), 0)
16
17        # Filter Sobel untuk mendeteksi tepi
18        sobel_x = cv2.Sobel(blurred, cv2.CV_64F, 1, 0, ksize=3)
19        sobel_y = cv2.Sobel(blurred, cv2.CV_64F, 0, 1, ksize=3)
20
21        edge = cv2.bitwise_or(cv2.convertScaleAbs(sobel_x), cv2.
22        convertScaleAbs(sobel_y))
23
24        _, edges = cv2.threshold(edge, 100, 255, cv2.
25        THRESH_BINARY + cv2.THRESH_OTSU)
26
27        return edges, blurred
28
29    def jumlahkan_angka_list(label_kelas):

```

```

24     hasil_jumlah_list = {}
25     for label in label_kelas:
26         angka = int(label.split('x')[0]) + int(label.split('x')
27             [1])
28     hasil_jumlah_list[label] = angka
29     return hasil_jumlah_list
30
31     def find_valid_card_for_computer(computer_card,
32         player_play_card):
33         valid_cards = []
34
35         # Periksa setiap kartu pada computer_card
36         for card in computer_card:
37             # Periksa setiap kartu yang sudah dimainkan oleh pemain
38             atau komputer
39             for played_card in player_play_card:
40                 # Dapatkan nilai dari kartu
41                 card_values = card.split('x')
42                 played_card_values = played_card.split('x')
43
44                 # Periksa apakah salah satu sisi kartu yang akan
45                 dimainkan oleh komputer sama dengan sisi kartu yang
46                 telah dimainkan oleh pemain atau komputer
47                 if card_values[0] == played_card_values[0] or
48                     card_values[1] == played_card_values[1]:
49                     valid_cards.append(card)
50                 break # Hentikan loop jika kartu valid ditemukan
51
52         return valid_cards
53
54     def check_validity(card_entered):
55         # Pisahkan angka-angka dalam kartu
56         card1 = list(map(int, card_entered[0].split('x')))
57         card2 = list(map(int, card_entered[1].split('x')))
58
59         # Periksa kesamaan antara dua list angka
60         return any(item in card1 for item in card2)

```

```

61     return True
62 else: # Jika objek tersebut tidak kosong
63     return False
64
65     def is_not_empty(obj):
66         return not is_empty(obj)
67
68     # Fungsi untuk menampilkan jendela permainan pemain
69     def show_player_play_window(kartu_terbesar_player):
70         # Implementasi sederhana, menampilkan jendela kosong
71         player_window = np.ones((300, 500, 3), dtype=np.uint8) *
            255 # Membuat jendela kosong
72         cv2.putText(player_window, f"Player's Turn: {
            kartu_terbesar_player}", (50, 150),
73             cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2, cv2.LINE_AA)
74         return player_window
75
76     # Fungsi untuk menampilkan jendela permainan komputer
77     def show_computer_play_window(kartu_terbesar_komputer):
78         # Implementasi sederhana, menampilkan jendela dengan
            instruksi komputer untuk memainkan kartu terbesar
79         computer_window = np.ones((300, 500, 3), dtype=np.uint8)
            * 255 # Membuat jendela kosong
80         cv2.putText(computer_window, f"Computer's Turn: {
            kartu_terbesar_komputer}", (10, 150),
81             cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2, cv2.LINE_AA
            )
82         return computer_window
83
84     def show_computer_play(kartu_komputer):
85         # Implementasi sederhana, menampilkan jendela dengan
            instruksi komputer untuk memainkan kartu terbesar
86         computer_play_window = np.ones((300, 500, 3), dtype=np.
            uint8) * 255 # Membuat jendela kosong
87         cv2.putText(computer_play_window, f"Computer's Turn: {
            Play_{kartu_komputer}", (10, 150),
88             cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2, cv2.LINE_AA
            )
89         return computer_play_window
90
91     player_play_window_open = True
92     computer_play_window_open = True
93     computer_play_open = True

```

```

94
95
96 def main():
97     cap = cv2.VideoCapture(1) # Ubah ke 0 jika webcam utama
98         , 1 jika webcam eksternal
99     model_path = "domino2.h5"
100     model = LoadModel(model_path)
101
102     global player_play_window_open,
103         computer_play_window_open, computer_play_open
104
105     game_start = False
106
107     hasil_jumlah_list = jumlahkan_angka_list(label_kelas)
108
109     while True:
110         ret, frame = cap.read()
111         height, width, _ = frame.shape
112
113         card_entered = []
114         player_play_card = []
115         computer_play_card = []
116         player_card = []
117         computer_card = []
118
119         live_frame = frame.copy()
120         edges, preprocessed_frame = preprocess_image(frame)
121
122         cv2.line(live_frame, (0, height // 3), (width, height //
123             3), (0, 255, 0), 2)
124         cv2.line(live_frame, (0, height // 3 * 2), (width,
125             height // 3 * 2), (0, 255, 0), 2)
126
127         cv2.line(live_frame, (width // 2, width // 2), (width //
128             2, height//3), (0, 255, 0), 2)
129
130         contours, _ = cv2.findContours(edges.copy(), cv2.
131             RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
132
133         for contour in contours:
134             perimeter = cv2.arcLength(contour, True)

```

```

131     approx = cv2.approxPolyDP(contour, 0.04 * perimeter,
132                                True)
133
134     if len(approx) == 4:
135         x, y, w, h = cv2.boundingRect(contour)
136         aspect_ratio = w
137         aspect_ratio_a = h
138
139         if 0 <= aspect_ratio <= 80 and 80 <= aspect_ratio_a <=
140             170:
141             cv2.rectangle(live_frame, (x, y), (x + w, y + h), (0,
142                 255, 0), 2)
143
144             cropped_image = edges[y:y+h, x:x+w]
145             cropped_image = cv2.cvtColor(cropped_image, cv2.
146                 COLOR_GRAY2BGR)
147
148             X = []
149             img = cv2.resize(cropped_image, (128, 128))
150             img = np.asarray(img)/255
151             img = img.astype('float32')
152             X.append(img)
153             X = np.array(X)
154             X = X.astype('float32')
155
156             hs = model.predict(X, verbose=0)
157             predicted_class_idx = np.argmax(hs)
158             predicted_class = label_kelas[predicted_class_idx]
159
160             if y < height // 3:
161                 player_card.append(predicted_class)
162                 cv2.putText(live_frame, f"player:_{predicted_class}", (x
163                     -30 , y - 5),
164                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.
165                         LINE_AA)
166             elif y > height // 3 * 2:
167                 computer_card.append(predicted_class)
168                 cv2.putText(live_frame, f"Komputer:_{predicted_class}",
169                     (x -30, y - 5),
170                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.
171                         LINE_AA)

```

```

165 elif height // 3 < y < height // 3 * 2 and x < width //
    2:
166 player_play_card.append(predicted_class)
167 cv2.putText(live_frame, f"player_play:_{predicted_class}"
    ", (x -30, y - 5),
168 cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.
    LINE_AA)
169
170 else :
171 computer_play_card.append(predicted_class)
172 cv2.putText(live_frame, f"computer_play:_{
    predicted_class}", (x -30, y - 5),
173 cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2.
    LINE_AA)
174
175 # Logika untuk menampilkan jendela berdasarkan kartu
    terbesar yang terdeteksi
176 if game_start:
177 print("computer_play", computer_play_card)
178 print("player_play", player_play_card)
179 card_entered = player_play_card + computer_play_card
180 print(card_entered)
181 valid_cards = find_valid_card_for_computer(computer_card
    , player_play_card)
182 text_computer = (100, height // 2)
183 text_player = (50, height//2)
184 if len(player_card) == 0 or len(computer_card) == 0:
185 print("Kartu_tidak_terdeteksi")
186 continue
187 contours, _ = cv2.findContours(edges.copy(), cv2.
    RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
188 most_common_label = max(player_card + computer_card, key
    =lambda x: hasil_jumlah_list.get(x, 0))
189
190 if most_common_label in player_card:
191 # computer_play_window_open = False
192 global player_play_window_open
193 if player_play_window_open:
194 player_window = show_player_play_window(
    most_common_label)
195 cv2.imshow("Player_Play", player_window)
196 if cv2.waitKey(1) == ord('_'):
197 cv2.destroyWindow("Player_Play")

```



```

198 player_play_window_open = False
199 if is_empty(player_play_card):
200     cv2.putText(live_frame, "masukkan_kartu_player",
        text_player, cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0)
        , 2, cv2.LINE_AA)
201 if is_not_empty(player_play_card):
202     print("xxxx")
203     global computer_play_open
204     if computer_play_open:
205         computer_valid = show_computer_play(valid_cards)
206         cv2.imshow("Computer_Play2", computer_valid)
207         if cv2.waitKey(1) == ord('_'):
208             cv2.destroyAllWindows("Computer_Play2")
209             computer_play_open = False
210         if is_empty(computer_play_card):
211             cv2.putText(live_frame, "Masukkan_Kartu_Komputer",
                text_player, cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0)
                , 2, cv2.LINE_AA)
212         if is_not_empty(computer_play_card):
213             print("guss")
214             # if cv2.waitKey(1) == ord(' '):
215             is_valid = check_validity(card_entered)
216             if is_valid:
217                 print("fakk")
218                 computer_play_open = True
219                 player_play_window_open = True
220                 cv2.putText(live_frame, "lanjut", text_player, cv2.
                    FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
221             else :
222                 print("salah")
223                 cv2.putText(live_frame, "salahsalah", text_player, cv2.
                    FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
224             # if is_empty(player_play_card) and is_empty(
                computer_play_card):
225                 # computer_play_open = True
226                 # player_play_window_open = True
227
228
229 elif most_common_label in computer_card:
230     computer_play_window_open
231     if computer_play_window_open:
232         computer_window = show_computer_play_window(
            most_common_label)

```

```

233 cv2.imshow("Computer_Play", computer_window)
234 if cv2.waitKey(1) == ord('_'):
235 cv2.destroyAllWindows("Computer_Play")
236 computer_play_window_open = False
237 if is_empty(computer_play_card):
238 cv2.putText(live_frame, "masukkan_kartu_computer",
    text_computer, cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0,
    0), 2, cv2.LINE_AA)
239 # if cv2.waitKey(1) == ord(' '):
240 #     break
241 if is_not_empty(computer_play_card):
242 print("masuk")
243 if is_not_empty(player_play_card) and is_valid:
244 computer_play_window_open = True
245 else:
246 cv2.putText(live_frame, "tidak_sesuai", text_player, cv2
    .FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA
    )
247
248
249
250 cv2.imshow("Deteksi_Kartu_Kotak", live_frame)
251 cv2.imshow("Frame_Hasil_Pra-Pemrosesan_Tepi", edges)
    # Tampilkan tepi Canny
252
253 if cv2.waitKey(1) == ord('t') and not game_start:
254 game_start = True
255 print("mulai")
256
257 if cv2.waitKey(1) == ord('q'):
258 break
259
260 if cv2.waitKey(1) == ord('r'): # Tekan 'r' untuk
    merestart permainan
261 cv2.destroyAllWindows()
262 computer_play_window_open = True
263 player_play_window_open = True
264 game_start = True
265
266 cap.release()
267 cv2.destroyAllWindows()
268
269 if __name__ == "__main__":

```

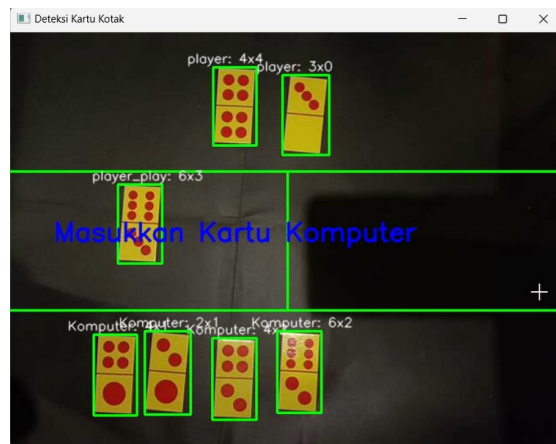


Figure 2: Game Domino

## 4 Kesimpulan

Melalui proyek ini, banyak hal yang perlu dipertimbangkan dan langkah-langkah yang harus dilakukan untuk berhasil mengintegrasikan konsep-konsep Pengolahan Citra Video yang telah dipelajari selama semester ini ke dalam pembuatan game interaktif antara manusia dan mesin. Untuk saat ini masih ada beberapa logika permainan yang belum berhasil dijalankan sehingga masih perlu perbaikan lebih lanjut.