

## Final Project Pengolahan Citra dan Video

Permainan Blackjack dengan *Image Classification* menggunakan *Deep Learning Training*

Rayhan Narindran Cendikia - 5024211022

DEPARTEMEN TEKNIK KOMPUTER  
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA  
CERDAS  
INSTITUT TEKNOLOGI SEPULUH NOPEMBER  
2023

# BAB I

## Pendahuluan

### 1 Latar Belakang

Permainan Blackjack adalah salah satu permainan kartu yang paling populer, Blackjack dimainkan dengan menghitung angka pada pemain dan dealer, dimana kartu yang digunakan harus mendekati atau sama dengan 21.

*Deep Learning* adalah cabang dari *Machine Learning* yang menggunakan *layers* untuk melakukan *Training* terhadap suatu dataset untuk mendapatkan hasil tertentu, pada penelitian ini penulis menggunakan *Image Multi-Class Classification*.

Permainan Blackjack pada dunia nyata dihitung dan dinilai oleh dealer pada tiap permainan, dengan menggunakan *Image Classification* hal ini dapat dipermudah tanpa adanya campur tangan manusia, sehingga permainan lebih mudah dimainkan oleh orang awam.

### 2 Konsep Project

#### 2.1 Konsep Game

Blackjack bertujuan untuk mendapatkan kartu mendekati 21 (oleh pemain dan dealer), namun jika melebihi angka tersebut maka pemegang kartu tersebut kalah atau disebut dengan *Bust*. Jika dealer melebihi 21, maka semua pemain menang. Jika kartu tepat di 21, maka pemegang kartu tersebut menang atau disebut dengan *Blackjack*. Jika dealer mendapatkan *Blackjack* maka semua pemain kalah, kecuali pemain tersebut juga mendapatkan *Blackjack*. Jika semua pemain (termasuk dealer) memiliki kartu dibawah 21, maka pemain dengan angka tertinggi menang.

Agar permainan lebih menantang, penulis memasukkan Computer sebagai lawan dari Player, dan setelah permainan selesai, maka jika keduanya menang melawan dealer, jumlah kartu antar computer dan player juga akan dibandingkan, dan kartu yang lebih tinggi menang.

#### 2.2 Konsep Program

Program akan ditulis dengan menggunakan bahasa Python dan beberapa library untuk melakukan pengolahan pada gambar, terutama menggunakan OpenCV, dan library pembantu yaitu Numpy (untuk melakukan kalkulasi pada matriks citra). OpenCV digunakan untuk membaca dan mengolah gambar dari sumber yang sudah ditentukan.

Tensorflow dan Keras juga digunakan pada proyek ini untuk melakukan training terhadap dataset, dimana keluaran dari training ini adalah sebuah model yang dapat digunakan untuk melakukan identifikasi kelas terhadap sebuah citra yang telah ditentukan.

Kedua konsep ini kemudian akan digabungkan untuk menghasilkan dan menjalankan sebuah permainan *Blackjack*, lengkap dengan Player, Computer, dan Dealer.

## BAB II

### Dasar Teori

#### 1 Computer Vision (*OpenCV*)

Computer Vision atau Visi Komputer adalah sebuah cabang kecerdasan buatan, yang menggunakan informasi dari sebuah citra untuk mendapatkan informasi tertentu mengenai citra tersebut. Dengan harapan bisa mendapatkan sebuah kesimpulan dari banyak input sehingga dapat diambil konklusi tertentu. Visi Komputer digunakan untuk berbagai hal dalam teknologi modern agar dapat mempermudah pekerjaan manusia dalam aspek visi, sebagai contoh dalam aspek Keamanan, Otomotif, Logistik, dll.[1]

OpenCV adalah sebuah library pada bahasa pemrograman Python (terdapat juga pada C++), yang dapat membantu pemrogram dalam melakukan pengolahan pada sebuah citra dengan harapan output visi komputer dengan tujuan tertentu. Secara luas OpenCV digunakan dalam beberapa seperti *Image Processing*, *Pattern Recognition*, dan *Photogrammetry*, dan secara spesifik sering digunakan untuk *Object Tracking*, Rekonstruksi Gambar, dan *Video Processing*. [2] Pada bahasa Python, OpenCV dapat diinstall melalui terminal dengan *pip*, dengan *pip install opencv-python*, kemudian dapat digunakan pada sebuah program dengan menuliskan *import cv2*.

#### 2 Machine Learning (*TensorFlow*)

Machine Learning adalah salah satu komponen penting dalam Kecerdasan Buatan, dimana sebuah teknik yang digunakan agar sebuah komputer dapat mengenali sebuah pola dari sebuah data tertentu, tanpa/sedikit campur tangan manusia. Machine learning memiliki beberapa cara untuk mengenali sebuah pola tertentu, dikelompokkan menjadi Supervised, Unsupervised, dan Reinforced Learning. Supervised learning adalah ketika sebuah mesin diberikan dataset tertentu yang sudah memiliki hasil tertentu, dan kemudian dilatih dengan dataset tanpa hasil tersebut dengan harapan dapat mengelompokkan data - data baru yang diberikan, unsupervised learning adalah cara mesin belajar tanpa adanya dataset yang sudah di labelkan, sehingga dataset yang diberikan akan dicari pola/kemiripan diantaranya sehingga dapat dikelompokkan menjadi satu, dan terakhir adalah reinforced learning, yaitu ketika sebuah program diberikan tugas tertentu dan diberikan "reward" ketika berhasil melakukan tugas tersebut, sehingga komputer akan terus berusaha mendapatkan "reward" tersebut. [3]

TensorFlow adalah sebuah library dan framework yang digunakan untuk membuat program mengenai Machine Learning, terdapat pada program Python, C++, Java, dan JavaScript. TensorFlow dapat memudahkan sebuah programmer untuk membuat arsitektur dan mengimplementasikan Machine Learning kedalam program mereka masing - masing.

#### 3 Deep Learning (*Keras*)

Deep learning adalah sebuah sub-bidang dari machine learning yang menggunakan artificial neural networks untuk mempelajari data dan membuat prediksi atau keputusan. ANN adalah jaringan yang terinspirasi oleh otak manusia, yang terdiri dari lapisan neuron yang saling terhubung. Neuron adalah unit dasar dari neural network, yang menerima masukan dari neuron lain, memprosesnya, dan kemudian mengirimkan keluaran ke neuron lain. Deep learning telah terbukti sangat efektif untuk berbagai tugas, seperti pengenalan gambar, pengenalan suara, dan terjemahan bahasa. Hal ini karena deep learning dapat mempelajari pola dan hubungan yang kompleks dalam data yang tidak dapat dipelajari oleh algoritma machine learning tradisional.

Keras adalah sebuah library dari TensorFlow yang memiliki syntax yang mudah dipelajari dan diimplementasikan untuk menghasilkan model dan arsitektur neural network. Dimana tiap layer dapat kita tentukan sesuai dengan kebutuhan dari program/model. [4]

# BAB III

## Program

### 1 Image Classification

#### 1.1 Mengambil dataset

Klasifikasi pada gambar digunakan angka yang terdapat pada kartu, dimana dataset dijadikan 13 kelas. Pengambilan dataset dilakukan dengan cara mengambil kartu pada gambar kemudian, mengambil ujung (area lokasi angka) kemudian melakukan inverted thresholding pada gambar sehingga gambar menjadi hitam putih. Dataset diambil dari kamera video, dimana setiap 10 frame disimpan satu gambar. Berikut program yang digunakan.

```
1 import cv2
2 import numpy as np
3 import deteksiKartu as dk
4 import os
5
6 # Lebar dan Tinggi Angka
7 LEBAR_ANGKA = 70
8 TINGGI_ANGKA = 125
9
10 # Set Angka Kartu yang akan diambil
11 card_check = "A"
12 folder_save = "A"
13 framePerImages = 10
14
15 # Set Direktori
16 video_path = "C:/Users/rayha/Pictures/Camera Roll/"
17 image_destination = "../dataset/"
18 os.chdir(image_destination + folder_save + "/")

19 # Membuka Video
20 cap = cv2.VideoCapture(video_path + card_check + ".mp4")      # Sesuaikan Video
21         dengan Kartu
22 videoFrames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

23
24 if not cap.isOpened():
25     print("Error opening video file")
26     exit()

27
28 # Frame Counter (tiap 10 frame akan diambil)
29 count = 0
30
31 while True:
32     ret, frame = cap.read()
33
34     if not ret:
35         break
36
37     # Gambar diproses dan disimpan tiap 10 frame
38     if count % framePerImages == 0:
39         # Preprocessing
40         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Mengubah gambar ke
41         grayscale
42         blur = cv2.GaussianBlur(gray,(5,5),0) # Menghilangkan noise dengan blurring
43         retval, thresh = cv2.threshold(blur,100,255,cv2.THRESH_BINARY) # Mengubah
44         gambar ke binary
45
46         # Mencari contour di frame
47         cnts,hier = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
48         cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
49
50         # Mengambil contour terbesar (dianggap sebagai kartu)
51         image2 = frame.copy()
52
53         # Jika tidak ada contour, lanjutkan
```

```

52     if len(cnts) == 0:
53         continue
54
55     card = cnts[0]
56
57     # Mencari sudut dari kartu
58     peri = cv2.arcLength(card,True)
59     approx = cv2.approxPolyDP(card,0.01*peri,True)
60     pts = np.float32(approx)
61
62     # Membuat bounding box dari kartu
63     x,y,w,h = cv2.boundingRect(card)
64
65     # Warp gambar kartu agar rata
66     warp = dk.warpKartu(frame,pts,w,h)
67
68     # Mengambil sudut kiri atas dari kartu (termasuk angka dan tingkat)
69     corner = warp[0:84, 0:32]
70     corner_zoom = cv2.resize(corner, (0,0), fx=4, fy=4)
71     corner.blur = cv2.GaussianBlur(corner_zoom,(5,5),0)
72     retval, corner_thresh = cv2.threshold(corner.blur, 155, 255, cv2.
THRESH_BINARY_INV)
73
74     angka = corner_thresh[20:185, 0:128] # Mengambil angka dari sudut kiri atas
saja
75     angka_cnts, hier = cv2.findContours(angka, cv2.RETR_TREE, cv2.
CHAIN_APPROX_SIMPLE)
76     if len(angka_cnts) < 1:
77         continue
78     angka_cnts = sorted(angka_cnts, key=cv2.contourArea, reverse=True)
79     x,y,w,h = cv2.boundingRect(angka_cnts[0])
80
81     # Crop angka dari sudut kiri atas
82     angka_roi = angka[y:y+h, x:x+w]
83     angka_sized = cv2.resize(angka_roi, (LEBAR_ANGKA, TINGGI_ANGKA), 0, 0) #
Resize angka agar sama ukurannya
84     final_img = angka_sized
85
86     # Menyimpan gambar sesuai direktori
87     cv2.imwrite(str(count // framePerImages) + ".jpg", final_img)
88
89     # Increment Counter Frame
90     if count >= videoFrames:
91         break
92     count += 1
93
94     if cv2.waitKey(25) & 0xFF == ord('q'):
95         break
96
97 cap.release()
98 cv2.destroyAllWindows()

```

Pada program ini digunakan juga *wrapping/flattening* pada gambar kartu, tepatnya dengan fungsi *dk.warpKartu*, Fungsi ini bekerja dengan cara mengambil 4 titik sudut kartu, kemudian meratakannya sehingga dapat memotong sudut kartu dan mengambil angka kartu dengan mudah. Berikut fungsi berikut. [5]

```

1 def warpKartu(image, pts, w, h):
2     pts_source = np.zeros((4, 2), dtype = "float32")
3
4     # Mengambil titik sudut dari kartu
5     s = np.sum(pts, axis = 2)
6     kiri_atas = pts[np.argmin(s)]
7     kanan_bawah = pts[np.argmax(s)]
8
9     diff = np.diff(pts, axis = -1)
10    kanan_atas = pts[np.argmin(diff)]
11    kiri_bawah = pts[np.argmax(diff)]
12
13    # Mengatur titik sudut agar rata
14    if w <= 0.6*h:

```

```

15     pts_source[0] = kiri_atas
16     pts_source[1] = kanan_atas
17     pts_source[2] = kiri_bawah
18     pts_source[3] = kanan_bawah
19
20     if w >= 1.3*h:
21         pts_source[0] = kiri_atas
22         pts_source[1] = kiri_bawah
23         pts_source[2] = kanan_atas
24         pts_source[3] = kanan_bawah
25
26 # Jika kartu miring
27 if w > 0.6*h and w < 1.3*h:
28     if pts[1][0][1] <= pts[3][0][1]:
29         pts_source[0] = pts[1][0]
30         pts_source[1] = pts[0][0]
31         pts_source[2] = pts[3][0]
32         pts_source[3] = pts[2][0]
33
34     if pts[1][0][1] > pts[3][0][1]:
35         pts_source[0] = pts[0][0]
36         pts_source[1] = pts[3][0]
37         pts_source[2] = pts[2][0]
38         pts_source[3] = pts[1][0]
39
40 pts_dest = np.float32([[0, 0], [LEBAR_KARTU-1, 0],[LEBAR_KARTU-1, TINGGI_KARTU-1], [0, TINGGI_KARTU-1]])
41
42 matrix = cv2.getPerspectiveTransform(pts_source, pts_dest)
43 warp = cv2.warpPerspective(image, matrix, (LEBAR_KARTU, TINGGI_KARTU))
44
45 return warp

```

Hasil dataset yang didapatkan adalah sebagai berikut. Dimana angka kartu berwarna putih, dengan background hitam, output ini yang akan digunakan pada training model. Beberapa dataset yang tidak sesuai sudah di hapus secara manual oleh penulis. Didapatkan 2759 dataset secara keseluruhan di 13 kelas, dengan rata - rata tiap kelas 212 gambar.

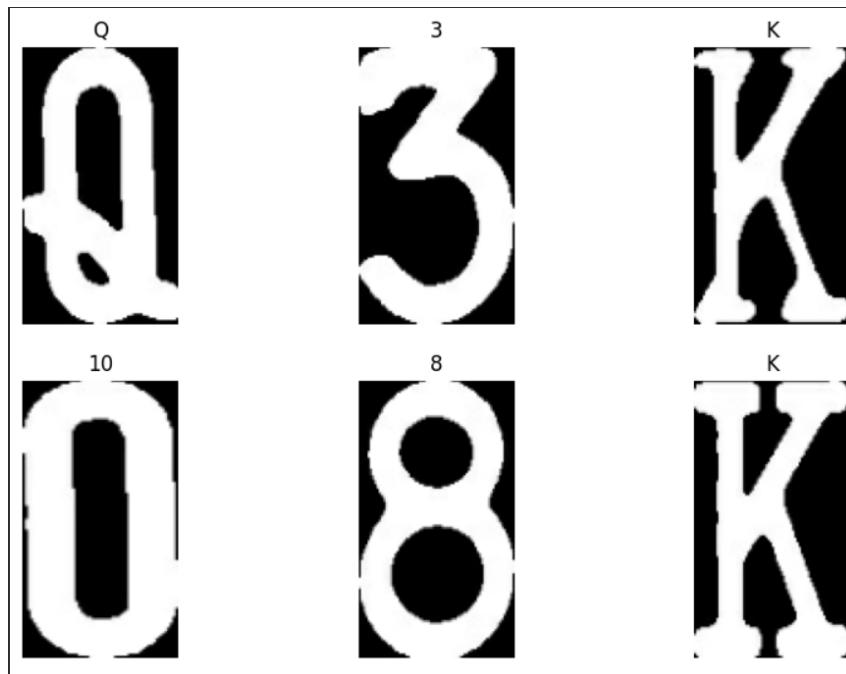


Figure 1: Contoh hasil pengambilan dataset

## 1.2 Training Model

Training model dilakukan pada platform Google Colab agar lebih efisien dan cepat. Pertama kita harus menyimpan dataset ke dalam Google Drive. Pertama harus dilakukan setup library dan settings google colab, pastikan menggunakan runtime *T4 GPU*.

### 1.2.1 Setup

```
1 import tensorflow as tf # Library Machine Learning
2 import keras # Library Deep Learning/Training
3 import numpy as np
4 from matplotlib import pyplot as plt
5
6 tf.config.list_physical_devices('GPU') # Cek runtime apakah GPU
7
8 # Memasukkan Google Drive ke environment/session Google Colab
9 from google.colab import drive
10 drive.mount('/content/drive')
```

### 1.2.2 Load Dataset

Setelah setup sudah dilakukan kita dapat mengambil dataset dari google drive, dan mulai melakukan proses training, disini kita juga melakukan setup pada dataset agar sesuai dengan konfigurasi training nanti, disini penulis menggunakan fungsi *tf.keras.utils.image\_dataset\_from\_directory* yang langsung memisahkan dataset menjadi kelas sesuai dengan nama folder datasetnya.

```
1 # Setting folder dataset sesuai dengan tempat menyimpan
2 data_dir = '/content/drive/MyDrive/Colab Notebooks/playing_card_dataset/rank'
3 train_dir = data_dir + '/train'
4 valid_dir = data_dir + '/valid'
5
6 batch = 32 # Batch sample tiap epoch training
7 img_height = 125 # Ukuran gambar dari dataset
8 img_width = 70
9
10 train_ds = tf.keras.utils.image_dataset_from_directory(
11     data_dir,
12     validation_split=0.2, # Training dataset diambil 80%
13     subset="training",
14     seed = 123,
15     image_size=(img_height, img_width),
16     batch_size=batch)
17
18 val_ds = keras.utils.image_dataset_from_directory(
19     data_dir,
20     validation_split=0.2, # Validation dataset diambil 20%
21     subset="validation",
22     seed=123,
23     image_size=(img_height, img_width),
24     batch_size=batch)
```

Dataset juga dimasukan kedalam algoritma tuning dari tensorflow dengan *tf.data.AUTOTUNE*, yang akan menyesuaikan dataset dengan mesin yang digunakan agar training lebih efisien. Berikut program tuning tersebut.

```
1 AUTOTUNE = tf.data.AUTOTUNE
2 train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
3 val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

### 1.2.3 Model dan Training

Pada model yang digunakan memiliki 2 layer konvolusi dengan kernel 3x3 yang keduanya menggunakan aktivasi relu dan keduanya dilakukan Max Pooling, yaitu mengambil beberapa elemen pada ukuran tertentu pada inputnya, kemudian mengambil nilai intensitas terbesar dari area tersebut. Pada layer konvolusi juga dilakukan Dropout, dimana beberapa node pada layer konvolusi secara acak akan diabaikan agar model tidak bergantung pada node - node yang terlalu

spesifik, digunakan agar model akhir tidak overfitting. Kemudian digunakan 2 layer tersembunyi (Dense Layer), dimana yang berfungsi sebagai "Neuron"/Nodes pada neural network, berbentuk array satu dimensi. Dense layer digunakan aktivasi relu dan softmax, dimana dense layer terakhir memiliki output 13 node, sesuai dengan 13 kelas dari dataset tersebut yang akan diprediksi. Berikut program yang menghasilkan model tersebut.

```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras import layers
3 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
4
5 num_classes = len(class_names)
6
7 # Menyusun layer model
8 model = Sequential([
9     #Mengubah rentang gambar dari 0 sampai 1
10    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
11
12    #Layer Konvolusi
13    layers.Conv2D(64, 3, padding='same', activation='relu'),
14    layers.MaxPooling2D(),
15    layers.Dropout(0.2),
16
17    layers.Conv2D(32, 3, padding='same', activation='relu'),
18    layers.MaxPooling2D(),
19    layers.Dropout(0.2),
20
21    # Menyatukan Konvolusi
22    layers.Flatten(),
23
24    # Layer Dense
25    layers.Dense(100, activation='relu'),
26    layers.Dense(num_classes, activation='softmax') # Output node!
27])
28
29 # Kompilasi model yang sudah disusun, dengan optimizer adam dan loss diukur dengan
# Sparse Categorical Crossentropy
30 model.compile('adam',
31               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
32               metrics=['accuracy'])
33
34 # Visualisasi hasil arsitektur model
35 model.summary()
```

Berikut hasil akhir dari arsitektur model yang telah disusun. Didapatkan 1.708.069 parameter berukuran 6.52MB

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 125, 70, 3)	0
conv2d_2 (Conv2D)	(None, 125, 70, 64)	1792
max_pooling2d_2 (MaxPooling2D)	(None, 62, 35, 64)	0
dropout_2 (Dropout)	(None, 62, 35, 64)	0
conv2d_3 (Conv2D)	(None, 62, 35, 32)	18464
max_pooling2d_3 (MaxPooling2D)	(None, 31, 17, 32)	0
dropout_3 (Dropout)	(None, 31, 17, 32)	0
flatten_1 (Flatten)	(None, 16864)	0
dense_2 (Dense)	(None, 100)	1686500
dense_3 (Dense)	(None, 13)	1313
<hr/>		
Total params: 1708069 (6.52 MB)		
Trainable params: 1708069 (6.52 MB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 2: Arsitektur Model

#### 1.2.4 Training Model

Setelah menyusun arsitektur model, kita dapat melanjutkan ke training dataset dengan fungsi *fit*. Pada training penulis menggunakan 10 epoch (perulangan), dan sudah mendapatkan hasil yang cukup akurat. Berikut program training tersebut.

```
1 log_dir = '/content/drive/MyDrive/Colab Notebooks/training_logs/playing_card_log'
2 tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)
3 hist = model.fit(train_ds,
4                   epochs=10,
5                   validation_data=val_ds,
6                   callbacks=[tensorboard_callback])
```

Hasil dari training akan dicantumkan pada Bab 4.

#### 1.2.5 Saving Model

Setelah model telah selesai di train, kita dapat menyimpan model ke *local device* agar dapat diimplementasikan ke dalam program. Ada beberapa tipe file yang dapat menyimpan model yaitu h5 dan keras, pada dokumentasi dianjurkan untuk menggunakan tipe keras. Berikut caranya.

```
1 from tensorflow.keras.models import load_model
2
3 model_extension = "keras"
4 if model_extension == "h5":
5     model.save("/content/drive/MyDrive/Colab Notebooks/models/" + model_name + ".h5")
6 elif model_extension == "keras":
7     model.save("/content/drive/MyDrive/Colab Notebooks/models/" + model_name)
```

## 2 Blackjack Game

### 2.1 Fungsi dan *Preprocessing*

Program dari game dibagi jadi dua file, dimana salah satu file bernama *deteksiKartu.py* yang berisikan fungsi dan kelas yang digunakan untuk melakukan preproses dan menjalankan program, program dipisah agar lebih terstruktur.

#### 2.1.1 Imports dan Konstanta

Berikut adalah beberapa library yang digunakan dalam program *deteksiKartu.py*, yaitu OpenCV, numpy, tensorflow, keras, dan os. OpenCV digunakan untuk melakukan pengolahan pada citra, numpy digunakan untuk melakukan proses dan perhitungan pada citra, tensorflow dan keras digunakan untuk melakukan *load* model dan deteksi dengan model tersebut yang sudah kita hasilkan sebelumnya. OS digunakan untuk mengakses data pada sistem/device, utamanya untuk merubah dan mengambil data dari direktori tertentu.

Kemudian ada beberapa konstanta yang digunakan dalam program ini, pertama adalah model CNN yang digunakan, sebelumnya penulis menggunakan dua model pada program untuk mendeteksi Angka dan Tingkat dari kartu, namun karena hanya dibutuhkan deteksi angka pada kartu, hanya model deteksi angka saja yang digunakan, beberapa kode untuk deteksi tingkat akan di *comment*, namun bisa diaktifkan jika dibutuhkan. Berikutnya ada Kelas Angka, konstanta array yang digunakan sebagai output dari deteksi dengan model CNN, karena output dari model CNN sebenarnya adalah score *confidence* dari seluruh index kelas, sehingga index yang diambil hanya index dengan score tertinggi. Kemudian ada konstanta array Sort, yang berfungsi untuk menyortir seluruh kartu yang telah dideteksi sesuai dengan array ini, hal ini dilakukan karena Angka As dapat bernilai dua dalam satu waktu, yaitu 11 dan 1, untuk mempermudah kalkulasi seluruh kartu selain As akan didahului untuk dihitung nilainya, sehingga nilai As dapat bergantung dengan nilai total kartu lainnya. Kemudian ada BKG Thresh, konstanta ini digunakan untuk melakukan adaptive thresholding pada gambar, fungsinya agar thresholding dapat menyesuaikan dengan lighting. Terakhir ada beberapa settings untuk ukuran dan area, ukuran - ukuran ini digunakan untuk menyesuaikan gambar agar pada satu ukuran tertentu, dan area digunakan untuk memastikan ukuran contour yang diambil sesuai dengan ukuran kartu, jika diluar range akan di keluarkan, hal ini memastikan yang terdeteksi hanya kartu saja. Berikut adalah program untuk Library dan konstanta tersebut.

```
1 import cv2
2 import numpy as np
3 import tensorflow as tf
4 from keras.models import load_model
5 import os
6 os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
7
8 # Model CNN
9 MODEL_ANGKA = load_model("./models/modelDeteksiRankKartu")
10 # MODEL_TINGKAT = load_model("./models/modelDeteksiSuitKartu")
11
12 KELAS_ANGKA = ['10', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'J', 'K', 'Q']
13 # KELAS_TINGKAT = ['Clubs', 'Diamonds', 'Hearts', 'Spades']
14
15 # Urutan Sorting Kartu
16 # ! Pastikan AS terakhir dalam penyortiran, sisanya bebas
17 SORT = ['0', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'K', 'Q', 'A']
18
19 # Adaptive thresholding
20 BKG_THRESH = 70
21
22 # Dimensi Kartu
23 LEBAR_KARTU = 200
24 TINGGI_KARTU = 300
25
26 # Area Kartu
27 AREA_KARTU_MAKS = 120000
28 AREA_KARTU_MIN = 25000
```

```

29
30 # Lebar dan Tinggi Ujung Kartu (Lokasi Angka dan Tingkat)
31 LEBAR_UJUNG = 32
32 TINGGI_UJUNG = 84
33
34 # Lebar dan Tinggi Angka
35 LEBAR_ANGKA = 70
36 TINGGI_ANGKA = 125
37
38 # Lebar dan Tinggi Tingkat
39 LEBAR_TINGKAT = 70
40 TINGGI_TINGKAT = 100

```

### 2.1.2 Kelas

Pada program ini, digunakan dua kelas, yaitu kelas Kartu dan kelas Game, dimana kelas Kartu berisikan informasi mengenai kartu yang dideteksi, seperti kontournya, titik sudut kartu, gambar rata dari kartu, prediksi deteksi, dan posisinya. Kemudian ada kelas gameState yang digunakan untuk menyimpan informasi dari permainan, seperti kartu player, dealer, dan komputer, poin player, dealer, dan komputer, dan state dari player, dealer, dan komputer. Berikut kelas-kelas tersebut.

```

1 # * Kelas Kartu
2 class Kartu:
3     def __init__(self):
4         self.contour = []
5         self.corners = [] # Titik sudut kartu
6         self.warp = []
7         self.corner = [] # Angka dan Tingkat Kartu
8         self.center = []
9         self.angka = []
10        self.tingkat = []
11        self.prediksi_angka = "0"
12        self.posisi_area = "Unknown"
13        # self.prediksi_tingkat = "0"
14
15 # * Kelas Game
16 class gameState:
17     def __init__(self):
18         self.player = []
19         self.computer = []
20         self.dealer = []
21         self.point_player = 0
22         self.point_computer = 0
23         self.point_dealer = 0
24         self.player_state = ""
25         self.computer_state = ""
26         self.dealer_state = ""

```

### 2.1.3 Fungsi Preprocessing

Berikut adalah beberapa fungsi yang digunakan dalam program untuk melakukan menyiapkan kartu sebelum dideteksi, fungsi *sortKartu* digunakan untuk menyortir kartu sesuai dengan konstanta array SORT. Hal ini juga dilakukan di program utama untuk menyortir kartu buffer. Fungsi *imageBinaryEdge* digunakan untuk mengubah gambar menjadi hitam putih (thresholding) juga mencari pinggiran dari bentuk kartu, *bkg\_level* adalah titik tengah atas dari gambar yang dijadikan titik acuan thresholding. Hal ini dilakukan agar memastikan pencahayaan tidak memengaruhi gambar yang diberikan. Kemudian *contoursImage* adalah fungsi mencari contour sebuah objek dari sisi-sisinya (Edge) yang telah dicari dengan Canny. Kemudian ada *cornersImage* adalah fungsi mencari titik-titik sudut dari sebuah contour objek. Karena contour menggunakan metode *cv2.CHAIN\_APPROX\_SIMPLE*, maka poly yang menjadi output lebih sederhana, sehingga lebih akurat jika hanya berbentuk persegi saja seperti kartu untuk dicari sudutnya. Kemudian ada fungsi warping yang digunakan untuk meratakan gambar sehingga menjadi ukuran dan orientasi gambar yang selalu sama pada sebuah objek tertentu.

```

1 # * Menyortir kartu berdasarkan angka (2 - A)
2 def sortKartu(kartu):
3     kartu = sorted(kartu, key=lambda kartu: SORT.index(kartu.prediksi_angka))
4     return kartu
5
6 # * Ubah Image menjadi Binary dan Cari Edge
7 def imageBinaryEdge(image):
8     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9     blur = cv2.GaussianBlur(gray, (5,5), 3, 3)
10
11    img_w, img_h = np.shape(image)[:2]
12    bkg_level = gray[int(img_h/100)][int(img_w/2)]
13    thresh_level = bkg_level + BKG_THRESH
14
15    _, thresh = cv2.threshold(blur, thresh_level, 255, cv2.THRESH_BINARY)
16    edge = cv2.Canny(image, 100, 200)
17
18    return thresh, edge
19
20 # * Mencari Contour dari Kartu
21 def contoursImage(edges):
22     contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.
23     CHAIN_APPROX_SIMPLE)
24     contours = sorted(contours, key=cv2.contourArea, reverse=True)
25
26     # Hilangkan contour di luar ukuran kartu
27     if len(contours) != 0:
28         for i in range(len(contours)-1, -1, -1):
29             if cv2.contourArea(contours[i]) > AREA_KARTU_MAKS or cv2.contourArea(
30             contours[i]) < AREA_KARTU_MIN:
31                 contours.pop(i)
32
33     return contours, hierarchy
34
35 # * Mencari titik sudut tiap kartu
36 def cornersImage(contours):
37     approx = []
38     for contour in contours:
39         peri = cv2.arcLength(contour, True)
40         approx.append(cv2.approxPolyDP(contour, 0.02 * peri, True))
41     return approx
42
43 # * Meratakan kartu agar dapat di prediksi angka
44 def warpKartu(image, pts, w, h):
45     pts_source = np.zeros((4, 2), dtype = "float32")
46
47     s = np.sum(pts, axis = 2)
48     kiri_atas = pts[np.argmin(s)]
49     kanan_bawah = pts[np.argmax(s)]
50
51     diff = np.diff(pts, axis = -1)
52     kanan_atas = pts[np.argmin(diff)]
53     kiri_bawah = pts[np.argmax(diff)]
54
55     if w <= 0.6*h:
56         pts_source[0] = kiri_atas
57         pts_source[1] = kanan_atas
58         pts_source[2] = kiri_bawah
59         pts_source[3] = kanan_bawah
60
61     if w >= 1.3*h:
62         pts_source[0] = kiri_atas
63         pts_source[1] = kiri_bawah
64         pts_source[2] = kanan_atas
65         pts_source[3] = kanan_bawah
66
67     if w > 0.6*h and w < 1.3*h:
68         if pts[1][0][1] <= pts[3][0][1]:
69             pts_source[0] = pts[1][0]
70             pts_source[1] = pts[0][0]

```

```

69         pts_source[2] = pts[3][0]
70         pts_source[3] = pts[2][0]
71
72     if pts[1][0][1] > pts[3][0][1]:
73         pts_source[0] = pts[0][0]
74         pts_source[1] = pts[3][0]
75         pts_source[2] = pts[2][0]
76         pts_source[3] = pts[1][0]
77
78     pts_dest = np.float32([[0, 0], [LEBAR_KARTU-1, 0],[LEBAR_KARTU-1, TINGGI_KARTU-1], [0, TINGGI_KARTU-1]])
79
80     matrix = cv2.getPerspectiveTransform(pts_source, pts_dest)
81     warp = cv2.warpPerspective(image, matrix, (LEBAR_KARTU, TINGGI_KARTU))
82
83     return warp

```

#### 2.1.4 Fungsi Deteksi

Kemudian ada fungsi - fungsi yang digunakan untuk melakukan deteksi dan penggambaran dari hasil deteksi. Pertama ada fungsi *prosesKartu*, yang bekerja dengan membuat sebuah objek dari kelas Kartu, kemudian melakukan preprocessing dengan fungsi - fungsi preprocessing diatas, yaitu dicari contour, titik sudut, titik tengah (moments), dan posisi pada gambar. Kemudian hasil dari proses tersebut di warping atau diratakan kemudian ujung dari kartu dilakukan cropping, lebih tepatnya bagian dari angka dari kartu yang sudah diratakan. Setelah semua itu didapatkan, akan tersimpan ke dalam objek Kartu yang kemudian akan disimpan di sebuah array di program utama. Hal ini akan berlaku pada setiap kartu yang dideteksi.

```

1 # * Inisialisasi Kelas Kartu
2 def prosesKartu(image, contour, approx, IM_WIDTH, IM_HEIGHT):
3     kartuQ = Kartu()
4     kartuQ.contour = contour
5     kartuQ.corners = approx
6
7     if len(kartuQ.corners) < 3:
8         return kartuQ
9
10    # ! Set titik tengah kartu, untuk menentukan area kartu diletakkan
11    M = cv2.moments(kartuQ.contour)
12    kartuQ.center = [int(M['m10']/M['m00']), int(M['m01']/M['m00'])]
13
14    if kartuQ.center[1] < IM_HEIGHT//2 and kartuQ.center[0] < IM_WIDTH//2 + 200:
15        kartuQ.posisi_area = "Player"
16    elif kartuQ.center[1] > IM_HEIGHT//2 and kartuQ.center[0] < IM_WIDTH//2 + 200:
17        kartuQ.posisi_area = "Computer"
18    else:
19        kartuQ.posisi_area = "Dealer"
20
21    x,y,w,h = cv2.boundingRect(kartuQ.contour)
22    kartuQ.warp = warpKartu(image, kartuQ.corners, w, h)
23
24    cornerKartu = kartuQ.warp[0:TINGGI_UJUNG, 0:LEBAR_UJUNG]
25    cornerKartuZoomed = cv2.resize(cornerKartu, (0,0), fx=4, fy=4)
26    kartuQ.corner = cornerKartuZoomed
27
28    cornerKartuZoomed = cv2.cvtColor(cornerKartuZoomed, cv2.COLOR_BGR2GRAY)
29    cornerKartuZoomed = cv2.adaptiveThreshold(cornerKartuZoomed, 255, cv2.
30                                              ADAPTIVE_THRESH_GAUSSIAN_C,
31                                              cv2.THRESH_BINARY_INV, 11, 2)
32
33    angkaCropped = cornerKartuZoomed[20:170, 0:120]
34    tingkatCropped = cornerKartuZoomed[171:300, 0:120]
35
36    angkaCroppedCnts, _ = cv2.findContours(angkaCropped, cv2.RETR_TREE, cv2.
37    CHAIN_APPROX_SIMPLE)
38    angkaCroppedCnts = sorted(angkaCroppedCnts, key=cv2.contourArea, reverse=True)
39
40    if len(angkaCroppedCnts) != 0:

```

```

39     xa, ya, wa, ha = cv2.boundingRect(angkaCroppedCnts[0])
40     angkaCropped = angkaCropped[ya:ya+ha, xa:xa+wa]
41     angkaCropped = cv2.resize(angkaCropped, (LEBAR_ANGKA, TINGGI_ANGKA), fx=0,
42     fy=0)
43     angkaCropped = np.repeat(angkaCropped[:, :, np.newaxis], 3, axis=2)
44     kartuQ.angka = angkaCropped
45
46     tingkatCroppedCnts, _ = cv2.findContours(tingkatCropped, cv2.RETR_TREE, cv2.
47     CHAIN_APPROX_SIMPLE)
48     tingkatCroppedCnts = sorted(tingkatCroppedCnts, key=cv2.contourArea, reverse=
49     True)
50
51     if len(tingkatCroppedCnts) != 0:
52         xt, yt, wt, ht = cv2.boundingRect(tingkatCroppedCnts[0])
53         tingkatCropped = tingkatCropped[yt:yt+ht, xt:xt+wt]
54         tingkatCropped = cv2.resize(tingkatCropped, (LEBAR_TINGKAT, TINGGI_TINGKAT))
55         , fx=0, fy=0)
56         tingkatCropped = np.repeat(tingkatCropped[:, :, np.newaxis], 3, axis=2)
57         kartuQ.tingkat = tingkatCropped
58
59     return kartuQ

```

Kemudian ada fungsi prediksi kartu, fungsi ini mengambil informasi gambar angka kartu yang sudah di crop dari fungsi *prosesKartu*, kemudian memasukannya ke dalam model yang sudah di train. Karena model menerima gambar dalam bentuk array satu dimensi, maka gambar tersebut harus diubah terlebih dahulu dengan fungsi *img\_to\_array* dari tf.keras, kemudian karna gambar yang diambil harus berbentuk 3 dimensi, maka array diperluas dengan fungsi *expand\_dims* dari numpy. Kemudian hasil yang telah didapatkan tersebut baru bisa dimasukkan ke model dengan fungsi *predict* dari tf.keras, sesuai dengan model yang didapatkan, kemudian jika prediksi sudah dilakukan maka akan di return oleh fungsi. Nanti pada program utama, hasil ini akan dimasukkan ke objek Kartu sesuai dengan prediksinya masing - masing. Terakhir adalah fungsi *drawKartu* yang digunakan untuk menggambarkan contour dan hasil prediksi ke layar utama permainan.

```

1 # * Prediksi kartu dengan model CNN
2 def prediksiKartu(kartu):
3     prediksi_angka = "0"
4     # prediksi_tingkat = "0"
5
6     if len(kartu.angka) != 0 and len(kartu.tingkat) != 0:
7         angka = kartu.angka
8         # tingkat = kartu.tingkat
9
10    angkaArray = tf.keras.utils.img_to_array(angka)
11    angkaArray = np.expand_dims(angkaArray, axis=0)
12    prediksi_angka = MODEL_ANGKA.predict(angkaArray, verbose=0)
13    prediksi_angka = KELAS_ANGKA[np.argmax(prediksi_angka)]
14
15    # tingkatArray = tf.keras.utils.img_to_array(tingkat)
16    # tingkatArray = np.expand_dims(tingkatArray, axis=0)
17    # prediksi_tingkat = MODEL_TINGKAT.predict(tingkatArray, verbose=0)
18    # prediksi_tingkat = KELAS_TINGKAT[np.argmax(prediksi_tingkat)]
19
20    return prediksi_angka #, prediksi_tingkat
21
22 # * Menggambar contour dan angka kartu
23 def drawKartu(image, kartu):
24     image = cv2.drawContours(image, [kartu.contour], 0, (255,0,0), 2)
25     angkaKartu = kartu.prediksi_angka
26     posisiTextAngka = (kartu.center[0], kartu.center[1] - 10)
27     # tingkatKartu = kartu.prediksi_tingkat
28     # posisiTextTingkat = (kartu.center[0] - 50, kartu.center[1] + 10)
29     image = cv2.putText(image, angkaKartu, posisiTextAngka, cv2.FONT_HERSHEY_PLAIN,
30     1.5, (0,0,0), 6)
31     image = cv2.putText(image, angkaKartu, posisiTextAngka, cv2.FONT_HERSHEY_PLAIN,
32     1.5, (255,0,255), 2)
33     # image = cv2.putText(image, tingkatKartu, posisiTextTingkat, cv2.
34     FONT_HERSHEY_PLAIN, 1.5, (0,0,0), 6)
35     # image = cv2.putText(image, tingkatKartu, posisiTextTingkat, cv2.
36     FONT_HERSHEY_PLAIN, 1.5, (255,0,255), 2)

```

```
33     return image
```

## 2.2 Game Utama

Berikut adalah fungsi dan program yang digunakan untuk menjalankan game *Blackjack* mulai dari *decision-making* dari Computer, perhitungan poin dan kemenangan, musik, sound effects, UI, dan status dari permainan.

### 2.2.1 Fungsi Game

Fungsi *game* digunakan sebuah membuat objek game dan segala hal yang berhubungan dengan sistem dari game, fungsi ini diletakkan pada file deteksiKartu.py, mulai dari pencantuman kartu sesuai posisi, penyortiran dan perhitungan kartu, dan status dari permainan. Berikut kode yang penulis hasilkan. Pertama adalah cara perhitungan kartu yaitu pertama dengan melihat posisi kartu pada gambar, penulis telah menetapkan posisi kartu dimana bagian atas adalah computer, bawah adalah player, dan kanan adalah dealer. Sehingga sesuai dengan titik tengah dari kartu penulis bisa mengetahui kepemilikan dari kartu dan mencantumkannya kedalam array pada objek dari kelas gameState,

```
1 # * Loop Game
2 def game(kartu):
3     permainan = gameState()
4
5     if len(kartu) == 0:
6         return permainan
7
8     for i in range(len(kartu)):
9         if kartu[i].posisi_area == "Player":
10             permainan.player.append(kartu[i])
11         elif kartu[i].posisi_area == "Computer":
12             permainan.computer.append(kartu[i])
13         else:
14             permainan.dealer.append(kartu[i])
```

kemudian penulis menyortir kartu sesuai dengan konstanta SORT, agar kartu As terletak terakhir pada array kartu tiap pemain, hal ini dilakukan untuk kebutuhan perhitungan kartu. Perhitungan kartu dilakukan dengan menambahkan jumlah nilai kartu, dimana 2 - 10 bernilai sesuai dengan angkanya Jack, Queen, dan King bernilai 10, dan As dapat bernilai dua, yaitu 11 atau 1, dan hal ini ditentukan dari kartu - kartu lainnya, jika total kartu lain berjumlah 10 atau kurang maka As bernilai 11 dan jika diatas dari 10 maka As akan bernilai 1, oleh karena itu As diletakkan terakhir pada array, sehingga nilainya dapat ditentukan setelah menjumlahkan kartu yang lain.

```
1 # ! Sortir kartu agar AS dihitung terakhir (untuk menentukan apakah AS bernilai
2     1 atau 11)
3     permainan.player = sorted(permainan.player, key=lambda kartu: SORT.index(kartu.
4     prediksi_angka))
5     permainan.computer = sorted(permainan.computer, key=lambda kartu: SORT.index(
6     kartu.prediksi_angka))
7     permainan.dealer = sorted(permainan.dealer, key=lambda kartu: SORT.index(kartu.
8     prediksi_angka))
9
10    # * Melakukan penjumlahan poin pada tiap pemain
11    for i in range(len(permainan.player)):
12        if permainan.player[i].prediksi_angka == "A":
13            if permainan.point_player + 11 > 21:
14                permainan.point_player += 1
15            else:
16                permainan.point_player += 11
17        elif permainan.player[i].prediksi_angka == "J" or permainan.player[i].
18        prediksi_angka == "Q" or permainan.player[i].prediksi_angka == "K":
19            permainan.point_player += 10
20        else:
21            permainan.point_player += int(permainan.player[i].prediksi_angka)
```

```

18     for i in range(len(permainan.computer)):
19         if permainan.computer[i].prediksi_angka == "A":
20             if permainan.point_computer + 11 > 21:
21                 permainan.point_computer += 1
22             else:
23                 permainan.point_computer += 11
24         elif permainan.computer[i].prediksi_angka == "J" or permainan.computer[i].prediksi_angka == "Q" or permainan.computer[i].prediksi_angka == "K":
25             permainan.point_computer += 10
26         else:
27             permainan.point_computer += int(permainan.computer[i].prediksi_angka)
28
29     for i in range(len(permainan.dealer)):
30         if permainan.dealer[i].prediksi_angka == "A":
31             if permainan.point_dealer + 11 > 21:
32                 permainan.point_dealer += 1
33             else:
34                 permainan.point_dealer += 11
35         elif permainan.dealer[i].prediksi_angka == "J" or permainan.dealer[i].prediksi_angka == "Q" or permainan.dealer[i].prediksi_angka == "K":
36             permainan.point_dealer += 10
37         else:
38             permainan.point_dealer += int(permainan.dealer[i].prediksi_angka)

```

Terakhir adalah pengecekan status dari game, yang berisikan semua kemungkinan yang dapat terjadi pada game blackjack. Dan mencantumkan status tersebut ke seluruh pemain pada game. Pada fungsi ini juga berisikan decision making dari komputer, dimana jika kartu dibawah 17 maka komputer akan meminta kartu lebih, dan jika diatas 17 maka akan menetapkan kartu, namun pada game ini dibuat jika kartu komputer diatas 17, namun masih dibawah kartu player (selagi player tidak lebih dari 21), komputer akan meminta kartu lebih, karena bertujuan untuk tidak hanya mengalahkan dealer namun juga player.

```

1 # * State menang dan kalah pemain
2 if len(permainan.dealer) >= 2:
3     if permainan.point_dealer == 21:
4         permainan.dealer_state = "Blackjack"
5     elif permainan.point_dealer > 21:
6         permainan.dealer_state = "Bust"
7     else:
8         permainan.dealer_state = "Safe"
9
10    if permainan.point_player == 21:
11        permainan.player_state = "Blackjack"
12    elif permainan.point_player > 21:
13        permainan.player_state = "Bust"
14    else:
15        permainan.player_state = "Safe"
16
17    if permainan.point_computer == 21:
18        permainan.computer_state = "Blackjack"
19    elif permainan.point_computer > 21:
20        permainan.computer_state = "Bust"
21    else:
22        permainan.computer_state = "Safe"
23
24    # Check Wins
25    if permainan.player_state == "Blackjack" and permainan.computer_state == "Blackjack":
26        permainan.player_state = "Draw"
27        permainan.computer_state = "Draw"
28    elif permainan.player_state == "Blackjack" and permainan.computer_state != "Blackjack":
29        permainan.player_state = "Win"
30        permainan.computer_state = "Lose"
31    elif permainan.player_state != "Blackjack" and permainan.computer_state == "Blackjack":
32        permainan.player_state = "Lose"
33        permainan.computer_state = "Win"
34    elif permainan.player_state == "Bust" and permainan.computer_state == "Bust":
35

```

```

35         permainan.player_state = "Lose"
36         permainan.computer_state = "Lose"
37     elif permainan.player_state == "Bust" and permainan.computer_state != "Bust"
38     :
39         permainan.player_state = "Lose"
40         permainan.computer_state = "Win"
41     elif permainan.player_state != "Bust" and permainan.computer_state == "Bust"
42     :
43         permainan.player_state = "Win"
44         permainan.computer_state = "Lose"
45     elif permainan.player_state == "Safe" and permainan.computer_state == "Safe"
46     :
47         if permainan.point_player > permainan.point_computer:
48             permainan.player_state = "Win"
49             permainan.computer_state = "Lose"
50         elif permainan.point_player < permainan.point_computer:
51             permainan.player_state = "Lose"
52             permainan.computer_state = "Win"
53         else:
54             permainan.player_state = "Draw"
55             permainan.computer_state = "Draw"
56     else:
57         # Computer Decision
58         if permainan.point_computer < 17:
59             permainan.computer_state = "Hit"
60         elif permainan.point_computer < permainan.point_player and permainan.
61         point_player < 20:
62             permainan.computer_state = "Hit"
63         elif permainan.point_computer >= 17 and permainan.point_computer <= 21:
64             permainan.computer_state = "Stand"
65
66         # Kartu Dealer kurang dari 2
67         if permainan.point_player == 21:
68             permainan.player_state = "Blackjack"
69         elif permainan.point_player > 21:
70             permainan.player_state = "Bust"
71
72         if permainan.point_computer == 21:
73             permainan.computer_state = "Blackjack"
74         elif permainan.point_computer > 21:
75             permainan.computer_state = "Bust"
76
77         if permainan.player_state == "Blackjack" and permainan.computer_state == "
78 Blackjack":
79             permainan.player_state = "Draw"
80             permainan.computer_state = "Draw"
81
82     return permainan

```

### 2.2.2 Imports, Variabel, dan Konstanta

Berikut adalah program yang digunakan pada kode utama, yaitu pada file gameKartu.py, sebelum program ditulis ada beberapa library, variabel, dan konstanta yang digunakan pada program ini, sebagai berikut.

Library yang digunakan program ini adalah OpenCV untuk membaca citra dari kamera, kemudian numpy untuk proses kalkulasi pada citra, kemudian PIL untuk menuliskan pada citra dengan font custom, time untuk melihat perubahan waktu pada program, pygame untuk memulai musik dan sound effect, copy untuk menduplikat citra, dan deteksiKartu adalah program fungsi yang ditulis diatas.

```

1 import cv2
2 import numpy as np
3 from PIL import ImageFont, ImageDraw, Image
4 import time
5 import pygame
6 from pygame import mixer
7 import copy
8 import deteksiKartu as dk

```

Kemudian ada beberapa variabel yang akan digunakan sebagai settings untuk permainan, mulai dari ukuran kamera, direktori dari file musik dan sound effect, direktori dari file font, variabel untuk setting warna, kemudian variabel sumber kamera, dan variabel mixer untuk musik dan sound effect.

```

1 # * SETTING KAMERA
2 IM_WIDTH = 1280
3 IM_HEIGHT = 720
4 FRAME_RATE = 10
5
6 # Source Video; 0 = Kamera ; path ke video
7 sourceVideo = 0
8 sourceAudio = "Casino.mp3"
9 winAudio = "Win.mp3"
10 loseAudio = "Lose.mp3"
11
12 # Set Text
13 fontpath_casino = "Casino.ttf"
14 fontpath_cards = "CardsFont.ttf"
15 font_casino = ImageFont.truetype(fontpath_casino, 40)
16 font_casino_state = ImageFont.truetype(fontpath_casino, 70)
17 font_cards = ImageFont.truetype(fontpath_cards, 40)
18
19 black = (0,0,0,0)
20 red = (0,0,200,0)
21 white = (255,255,255,0)
22 purple = (200,0,200,0)
23 green = (0,170,0,0)
24
25 # Set Video
26 video = cv2.VideoCapture(sourceVideo)
27 video.set(cv2.CAP_PROP_FRAME_WIDTH, IM_WIDTH)
28 video.set(cv2.CAP_PROP_FRAME_HEIGHT, IM_HEIGHT)
29 video.set(cv2.CAP_PROP_FPS, FRAME_RATE)
30
31 # Set Pygame
32 pygame.init()
33 mixer.music.load(sourceAudio)
34 mixer.music.play(-1)
35 winSFX = mixer.Sound(winAudio)
36 loseSFX = mixer.Sound(loseAudio)
```

Kemudian ada beberapa variabel dan buffer yang digunakan untuk jalannya permainan. Pertama ada frame counter, yang akan digunakan untuk menghitung frame yang sudah di proses, detectionTimer adalah variabel time yang akan digunakan untuk menghitung berapa lama program telah berjalan dan akan digunakan untuk menentukan kapan deteksi kartu dilakukan, kemudian ada winCheckCounter adalah variabel yang digunakan untuk melihat apakah state kartu yang dicek sama dengan setelahnya, hal ini dilakukan agar ketika permainan berakhir, tidak ada kesalahan dalam deteksi, kemudian ada bufferKartu yang digunakan sebagai buffer ketika selesai mendeteksi, bufferKartu inilah array kartu yang akan menjadi array kartu yang akan diproses dalam keseluruhan game, bufferKartuCheckWin adalah variabel yang akan sama dengan bufferKartu setiap 10 frame, hal ini memastikan bahwa tidak ada perubahan pada deteksi yang akan mengakibatkan state yang salah. Terakhir adalah variabel playerWins dan computerWins yang menghitung seberapa banyak para player memenangkan permainan.

```

1 # Timer, buffer, and game
2 frameCounter = 0
3 detectionTimer = time.time()
4 winCheckCounter = 0 # Jika state Win atau Lose, maka akan menunggu 60 frame untuk
                     # mengecek input
5 bufferKartu = []
6 bufferKartuCheckWin = []
7
8 # Wins
9 playerWins = 0
10 computerWins = 0
```

### 2.2.3 Deteksi Kartu

Di dalam loop game dan kamera, yang pertama dilakukan adalah mengambil semua contour objek pada gambar, kemudian mencari titik - titik sudutnya, karena kita telah menyantumkan area dan minimum dan maksimum. Kemudian pada diinisiasikan variabel kartu yang akan menyimpan semua data mengenai kartu dalam bentuk objek dari kelas Kartu. Pertama dimasukkan parameter gambar, contour, sudut, dan ukurannya yang akan diproses dengan fungsi prosesKartu, kemudian di append kedalam array kartu sebagai objek, setelah itu semua kartu di dalam array tersebut akan di prediksi angka dari kartu tersebut, hal ini dilakukan setiap 1 detik sekali untuk melancarkan jalannya program. Setelah itu semua contour dan prediksi angka dari kartu di tampilkan ke gambar.

```
1 while(True):
2     ret, image = video.read()
3
4     # * PROSES GAMBAR
5     thresh, edge = dk.imageBinaryEdge(image)
6     contours, hierarchy = dk.contoursImage(edge)
7     approx = dk.cornersImage(contours)
8
9     # * LOOP DETEKSI KARTU
10    kartu = []
11    if len(contours) != 0:
12        for i in range(len(contours)):
13            kartu.append(dk.prosesKartu(image, contours[i], approx[i], IM_WIDTH,
14                                         IM_HEIGHT))
15
16    # ! Program hanya mendekripsi setiap 1 detik (agar tidak lagging)
17    if time.time() - detectionTimer > 1:
18        bufferKartu = kartu
19        for i in range(len(bufferKartu)):
20            kartu[i].prediksi_angka = dk.prediksiKartu(bufferKartu[i])
21        bufferKartu = dk.sortKartu(bufferKartu)
22        detectionTimer = time.time()
23
24    # * GAMBAR CONTOUR DAN ANGKA KARTU
25    for i in range(len(bufferKartu)):
26        image = dk.drawKartu(image, bufferKartu[i])
```

### 2.2.4 User Interface

Pada user interface digambarkan semua yang berhubungan dengan gambar dan penulisan, dimana digambarkan poin pemain, state dari setiap pemain, kartu yang dimiliki pemain, dan seberapa banyak pemain memenangkan game. Juga digambarkan garis yang memisahkan area antar pemain.

```
1     # * LOOP PERMAINAN
2     game = dk.game(bufferKartu)
3
4     # ! Untuk Win/Lose/Draw Screen
5     imagePilSave = Image.fromarray(copy.deepcopy(image))
6     drawSave = ImageDraw.Draw(imagePilSave)
7
8     # * UI DAN POIN PEMAIN
9     image = cv2.line(image, (0, IM_HEIGHT//2), (IM_WIDTH//2 + 200, IM_HEIGHT//2),
10                      (190, 0, 0), 10)
11    image = cv2.line(image, (IM_WIDTH//2 + 200, 0), (IM_WIDTH//2 + 200, IM_HEIGHT),
12                      (190, 0, 0), 10)
13
14    image = cv2.line(image, (0, IM_HEIGHT//2), (IM_WIDTH//2 + 200, IM_HEIGHT//2),
15                      (255, 255, 255), 2)
16    image = cv2.line(image, (IM_WIDTH//2 + 200, 0), (IM_WIDTH//2 + 200, IM_HEIGHT),
17                      (255, 255, 255), 2)
18
19     # * Image Conversion (Untuk text dengan PIL)
20     image_pil = Image.fromarray(image)
21     draw = ImageDraw.Draw(image_pil)
22
23     # * STATE PEMAIN
```

```

20     if len(game.player_state) > 1:
21         textSize_P = cv2.getTextSize(game.player_state, cv2.FONT_HERSHEY_PLAIN, 3,
22                                     0)[0]
23         textX_P = ((IM_WIDTH//2 + 200) - (textSize_P[0]))//2
24         textY_P = (IM_HEIGHT//4) - (textSize_P[1]//2)
25         draw.text((textX_P, textY_P), game.player_state, font=font_casino_state,
26                   fill='red',
27                   stroke_width=4, stroke_fill='white')
28
29     if len(game.computer_state) > 1:
30         textSize_C = cv2.getTextSize(game.computer_state, cv2.FONT_HERSHEY_PLAIN, 3,
31                                     4)[0]
32         textX_C = ((IM_WIDTH//2 + 200) - textSize_C[0])//2
33         textY_C = ((IM_HEIGHT*3)//4) - (textSize_C[1]//2)
34         draw.text((textX_C, textY_C), game.computer_state, font=font_casino_state,
35                   fill='red',
36                   stroke_width=4, stroke_fill='white')
37
38     if len(game.dealer_state) > 1:
39         textSize_D = cv2.getTextSize(game.dealer_state, cv2.FONT_HERSHEY_PLAIN, 3,
40                                     0)[0]
41         textX_D = (IM_WIDTH//2 + 200)+(IM_WIDTH - (IM_WIDTH//2 + 200))//2 -
42         textSize_D[0]//2
43         textY_D = (IM_HEIGHT - textSize_D[1])//2
44         draw.text((textX_D, textY_D), game.dealer_state, font=font_casino_state,
45                   fill='red',
46                   stroke_width=4, stroke_fill='white')
47
48     # * POIN PEMAIN
49     draw.text((15, 15), "Player", font=font_casino, fill='white',
50               stroke_width=3, stroke_fill='black')
51     draw.text((15, 50), "pts: " + str(game.point_player), font=font_casino, fill=
52               'white',
53               stroke_width=3, stroke_fill='black')
54
55     draw.text((15, IM_HEIGHT - 40), "Computer", font=font_casino, fill='purple',
56               stroke_width=3, stroke_fill='white')
57     draw.text((15, IM_HEIGHT - 75), "pts: " + str(game.point_computer), font=
58               font_casino, fill='purple',
59               stroke_width=3, stroke_fill='white')
60
61     draw.text((IM_WIDTH - 150, IM_HEIGHT - 40), "Dealer", font=font_casino, fill=
62               'green',
63               stroke_width=3, stroke_fill='white')
64     draw.text((IM_WIDTH - 150, IM_HEIGHT - 75), "pts: " + str(game.point_dealer),
65               font=font_casino, fill='green',
66               stroke_width=3, stroke_fill='white')
67
68     # * KARTU PEMAIN
69     temp_kartu_player = []
70     temp_kartu_computer = []
71     temp_kartu_dealer = []
72
73     for i in range(len(game.player)):
74         temp_kartu_player.append(game.player[i].angka)
75         draw.text((IM_WIDTH//2 + 160 - (i*50), IM_HEIGHT//2 - 60), game.player[i].
76                   prediksi_angka, font=font_cards, fill='white',
77                   stroke_width=2, stroke_fill='black')
78
79     for i in range(len(game.computer)):
80         temp_kartu_computer.append(game.computer[i].angka)
81         draw.text((IM_WIDTH//2 + 160 - (i*50), IM_HEIGHT - 50), game.computer[i].
82                   prediksi_angka, font=font_cards, fill='purple',
83                   stroke_width=2, stroke_fill='white')
84
85     for i in range(len(game.dealer)):
86         temp_kartu_dealer.append(game.dealer[i].angka)
87         draw.text((IM_WIDTH//2 + 220, 20 + (i*50)), game.dealer[i].prediksi_angka,
88                   font=font_cards, fill='green',
89                   stroke_width=2, stroke_fill='white')

```

```

76     # * WINS PEMAIN
77     draw.text((IM_WIDTH//2 + 30, 10), "Wins: " + str(playerWins), font=font_casino,
78                 fill=white,
79                 stroke_width=3, stroke_fill=black)
80     draw.text((IM_WIDTH//2 + 30, IM_HEIGHT//2 + 15), "Wins: " + str(computerWins),
81                 font=font_casino, fill=purple,
82                 stroke_width=3, stroke_fill=white)

```

## 2.2.5 Game Over Screen

Terakhir adalah melakukan checking jika permainan sudah selesai, hal ini dilakukan dengan melihat bufferKartu dan membandingkannya dengan bufferKartuCheckWin dimana akan memiliki kartu yang berbeda jika terjadi perubahan deteksi, namun akan sama jika deteksi tetap, hal ini bertujuan untuk meminimalisir terjadinya kesalahan deteksi dan state kemenangan, setelah deteksi sudah stabil, maka akan ditampilkan pemenang dan dimainkan sound effect menang atau kalah berdasarkan state Player, kemudian bisa dipilih apakah mau main kembali atau keluar dari game.

```

1      # * Check Wins
2      if frameCounter % 10 == 0:
3          bufferKartuCheckWin = bufferKartu
4          bufferKartuCheckWin = dk.sortKartu(bufferKartuCheckWin)
5
6          if game.player_state == "Win" or game.player_state == "Lose" or game.
7              computer_state == "Win" or game.computer_state == "Lose" or game.player_state
8                  == "Draw" or game.computer_state == "Draw":
9              forceEnd = cv2.waitKey(1) # Jika state permainan sudah benar tekan E agar
10             permainan berakhir
11             if bufferKartuCheckWin == bufferKartu or forceEnd == ord("e") or forceEnd
12             == ord("E"):
13                 if winCheckCounter == 20 or forceEnd == ord("e") or forceEnd == ord("E")
14             ):
15                 mixer.music.pause()
16                 if game.player_state == "Win" and game.computer_state == "Lose":
17                     drawSave.text((IM_WIDTH//2 - 110, IM_HEIGHT//2 - 60), "You Win!",
18                         font=font_casino_state, fill=white,
19                         stroke_width=3, stroke_fill=black)
20                     mixer.Sound.play(winSFX)
21
22                 elif game.player_state == "Lose" and game.computer_state == "Win":
23                     drawSave.text((IM_WIDTH//2 - 110, IM_HEIGHT//2 - 60), "You Lose
24 !", font=font_casino_state, fill=white,
25                         stroke_width=3, stroke_fill=black)
26                     mixer.Sound.play(loseSFX)
27                 else:
28                     drawSave.text((IM_WIDTH//2-120, IM_HEIGHT//2 - 60), "Draw!", font=font_casino_state, fill=white,
29                         stroke_width=3, stroke_fill=black)
30                     mixer.Sound.play(winSFX)
31
32             drawSave.text((IM_WIDTH//2 - 200, IM_HEIGHT//2 + 20), "Press Y to
33             save state!", font=font_casino, fill=white,
34                         stroke_width=3, stroke_fill=black)
35             drawSave.text((IM_WIDTH//2 - 220, IM_HEIGHT//2 + 60), "Press N if
36             state is false!", font=font_casino, fill=white,
37                         stroke_width=3, stroke_fill=black)
38             drawSave.text((IM_WIDTH//2 - 185, IM_HEIGHT//2 + 100), "Press Q to
quit game!", font=font_casino, fill=white,
                         stroke_width=3, stroke_fill=black)
39
40             cv2.imshow("Video", np.array(imagePilSave))
41
42             state = cv2.waitKey(0)
43             if state == ord("y") or state == ord("Y"):
44                 if game.player_state == "Win" and game.computer_state == "Lose"
45                 :
46                     playerWins += 1
47                 elif game.player_state == "Lose" and game.computer_state == "
48 Win":

```

```

39             computerWins += 1
40             mixer.music.unpause()
41         elif state == ord("n") or state == ord("N"):
42             mixer.music.unpause()
43         elif state == ord("q") or state == ord("Q"):
44             break
45         else:
46             print("Invalid Input, continuing...")
47             pass
48         winCheckCounter = 0
49     else:
50         winCheckCounter += 1
51 else:
52     winCheckCounter = 0
53
54 # Show Video
55 image = np.array(image_pil)
56 cv2.imshow("Video", image)
57 frameCounter += 1
58
59 # Program Keluar
60 key = cv2.waitKey(1) & 0xFF
61 if key == ord("q") or key == ord("Q"):
62     break
63
64 cv2.destroyAllWindows()
65 video.release()

```

## BAB IV

# Hasil Training dan Game

### 1 Training

Berikut adalah hasil dari training yang didapatkan, dimana dilakukan 10 Epoch dalam training, dan dari 10 epoch tersebut didapatkan akurasi 1.0 dan loss  $2 \times 10^4$  dan akurasi validasi juga 1.0.

```
Epoch 1/10
69/69 [=====] - 362s 1s/step - loss: 0.1493 - accuracy: 0.9552 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 2/10
69/69 [=====] - 1s 17ms/step - loss: 4.5507e-04 - accuracy: 1.0000 - val_loss: 5.3181e-04 - val_accuracy: 1.0000
Epoch 3/10
69/69 [=====] - 1s 18ms/step - loss: 1.6189e-05 - accuracy: 1.0000 - val_loss: 4.4822e-04 - val_accuracy: 1.0000
Epoch 4/10
69/69 [=====] - 1s 18ms/step - loss: 6.9247e-06 - accuracy: 1.0000 - val_loss: 3.6707e-04 - val_accuracy: 1.0000
Epoch 5/10
69/69 [=====] - 1s 18ms/step - loss: 4.4074e-06 - accuracy: 1.0000 - val_loss: 3.0624e-04 - val_accuracy: 1.0000
Epoch 6/10
69/69 [=====] - 1s 16ms/step - loss: 2.5802e-06 - accuracy: 1.0000 - val_loss: 2.8655e-04 - val_accuracy: 1.0000
Epoch 7/10
69/69 [=====] - 1s 16ms/step - loss: 2.8698e-06 - accuracy: 1.0000 - val_loss: 2.7530e-04 - val_accuracy: 1.0000
Epoch 8/10
69/69 [=====] - 1s 16ms/step - loss: 2.1833e-06 - accuracy: 1.0000 - val_loss: 2.5757e-04 - val_accuracy: 1.0000
Epoch 9/10
69/69 [=====] - 1s 16ms/step - loss: 1.9141e-06 - accuracy: 1.0000 - val_loss: 2.2100e-04 - val_accuracy: 1.0000
Epoch 10/10
69/69 [=====] - 1s 16ms/step - loss: 1.7713e-06 - accuracy: 1.0000 - val_loss: 2.0189e-04 - val_accuracy: 1.0000
```

Figure 3: Epoch Training

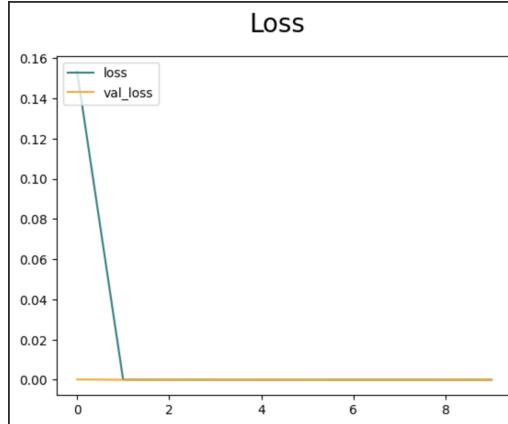


Figure 4: Loss Training

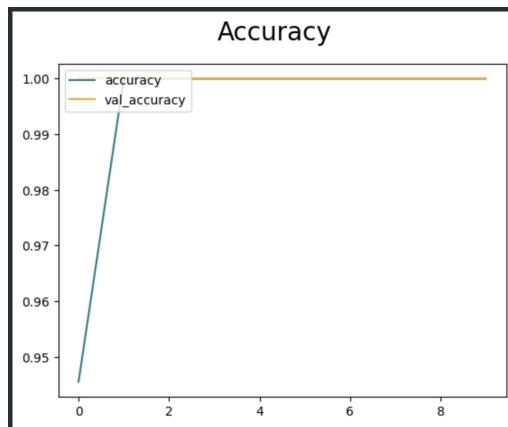


Figure 5: Akurasi Training

## 2 Game

Berikut adalah beberapa gambar yang menunjukkan hasil dari game. Mulai dari berjalannya permainan, pilihan dari komputer, state tiap player, game over screen, dan perubahan game dan kemenangan.



Figure 6: Permainan Berjalan



Figure 7: Player Kalah



Figure 8: Game Over Screen

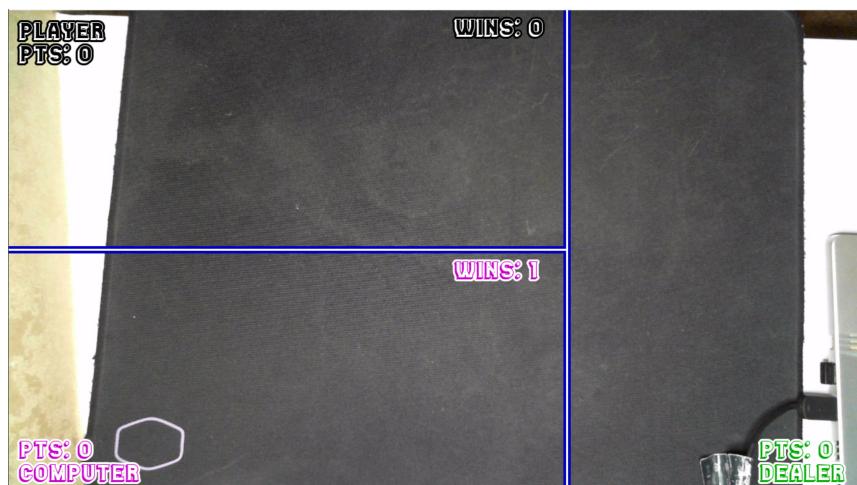


Figure 9: Perubahan Skor

## **BAB V**

### **Kesimpulan**

Kesimpulan yang dapat ditarik dari Project ini adalah Machine Learning adalah salah satu tool yang sangat *multi-purpose* dengan efisiensi tinggi yang dapat diimplementasikan ke banyak hal, salah satunya adalah Visi Komputer, dimana dengan peleburan dua hal tersebut dapat menghasilkan sebuah alat yang dapat mengenali pola gambar dan menjadikannya sebuah program yang bermanfaat, sebagai contoh adalah dengan permainan kartu Blackjack.

## References

- [1] IBM. What is Computer Vision? <https://www.ibm.com/topics/computer-vision>: :text=Computer Diakses: November 30, 2023.
- [2] Ravikiran A.S. OpenCV C: What is it, How to Use and its Applications. <https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/opencv-csharp>: :text=OpenCV Diakses: November 30, 2023.
- [3] Dicoding Intern. Apa itu Machine Learning? Beserta Pengertian dan Cara Kerjanya. <https://www.dicoding.com/blog/machine-learning-adalah/>. Diakses: Desember 4, 2023.
- [4] Fireship. TensorFlow in 100 Seconds. [https://www.youtube.com/watch?v=i8NETqtGHmsab\\_channel=Fireship](https://www.youtube.com/watch?v=i8NETqtGHmsab_channel=Fireship). Diakses : Desember 4, 2023.
- [5] Adrian Rosebrock. 4 Point OpenCV getPerspective Transform Example. <https://pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/>. Diakses: November 12, 2023.