Optimasi Hyperparameter pada Penerapan *Ensemble Regression Tree* untuk Simulasi Pewarnaan Bibir

http://dx.doi.org/10.28932/jutisi.v8i2.4611

Riwayat Artikel
Received: 11 Maret 2022 | *Final Revision*: 02 Agustus 2022 | *Accepted*: 02 Agustus 2022

Andi Hakim Arif ✉ #1, Achmad Solichin#2

Program Studi Magister Ilmu Komputer, Universitas Budi Luhur
Jalan Ciledug Raya, Jakarta Selatan, 12660, Indonesia.

[1]andi.hakim.arif@gmail.com

[2]achmad.solichin@budiluhur.ac.id

✉Corresponding author: andi.hakim.arif@gmail.com

Abstrak — **Teknologi membantu kita dalam banyak aktifitas dan terus mengalami kemajuan sehingga aktivitas lebih efisien, penggunaan waktu yang lebih hemat, sumberdaya yang lebih sedikit dan juga informasi serta hiburan lebih mudah didapatkan. Teknologi *Machine Learning* adalah bidang yang paling cepat berkembang dalam ilmu komputer yang penggunaannya mencakup berbagai bidang seperti pemasaran, perawatan kesehatan, manufaktur, keamanan informasi dan transportasi. Salah satu metode *machine learning* adalah *Ensemble Regression Tree* (ERT) yang telah berhasil mendeteksi fitur wajah pada bagian alis, mata, hidung dan bibir. Akan tetapi, belum ditemukan penggunaan metode ERT untuk mendeteksi spesifik area bibir saja. Maka akan dilakukan penelitian untuk ekstraksi *dataset* anotasi fitur wajah dari *dataset* iBUG 300W dengan fitur 68 titik wajah menjadi 20 titik area bibir. Hasil dari ekstraksi tersebut yaitu *error rate* berkurang, penggunaan *resource* lebih sedikit, fitur bibir dapat terdeteksi dan simulasi pewarnaan bibir berhasil dilakukan dengan menggunakan konfigurasi nilai *hyperparameter* yaitu *tree* = 4, *regularization* = 0,25, *cascade* = 8, *feature pool* = 500, *oversampling* = 40 dan *translation jitter* = 0. Dari pengamatan juga diketahui bahwa penghematan *resource hardisk* sebesar 69,36%, RAM 30,8% dan CPU 3,8%; mengurangi *error rate* sebesar 0,058%; serta meningkatkan *inference speed* sebesar 39%.**

Kata kunci— ***ensemble regression tree*; *hyperparameter*; pendeteksi bibir; simulasi pewarnaan bibir.**

Hyperparameter Optimization on Ensemble Regression Tree for Lip Coloring Simulation

Abstract — ***Technology helps us in many activities and keeps growing, so it makes activities more efficient, time-saving, using fewer resources and also information and entertainment are accessible. Machine Learning technology is the fastest-growing field in computer science that is used in many areas such as marketing, healthcare, manufacturing, information security, and transportation. One of the machine learning methods is the Ensemble of Regression Tree (ERT) which has succeeded in detecting facial features on the eyebrows, eyes, nose, and lips. However, the utilization ERT method has not been found to detect specific areas such as lips only for gaining optimization. Then this research will be conducted to extract the facial feature annotation dataset from the iBUG 300W dataset with 68 facial features to 20 lip area points. The results of the extraction are reduced error rate, resources saving, lip features still detected and lip coloring simulation was successfully carried out using the configuration of hyperparameter values, tree = 4, regularization = 0.25, cascade = 8, feature pool = 500, oversampling = 40 and translation jitter = 0. From observations also discovered optimization that hard disk resource savings are 69.36%, RAM 30.8%, and CPU 3.8%; reduce the error rate by 0.058%; and increase inference speed by 39%.***

Keywords— ***ensemble regression tree; hyperparameter; lip coloring simulation; lip detection.***

I. PENDAHULUAN

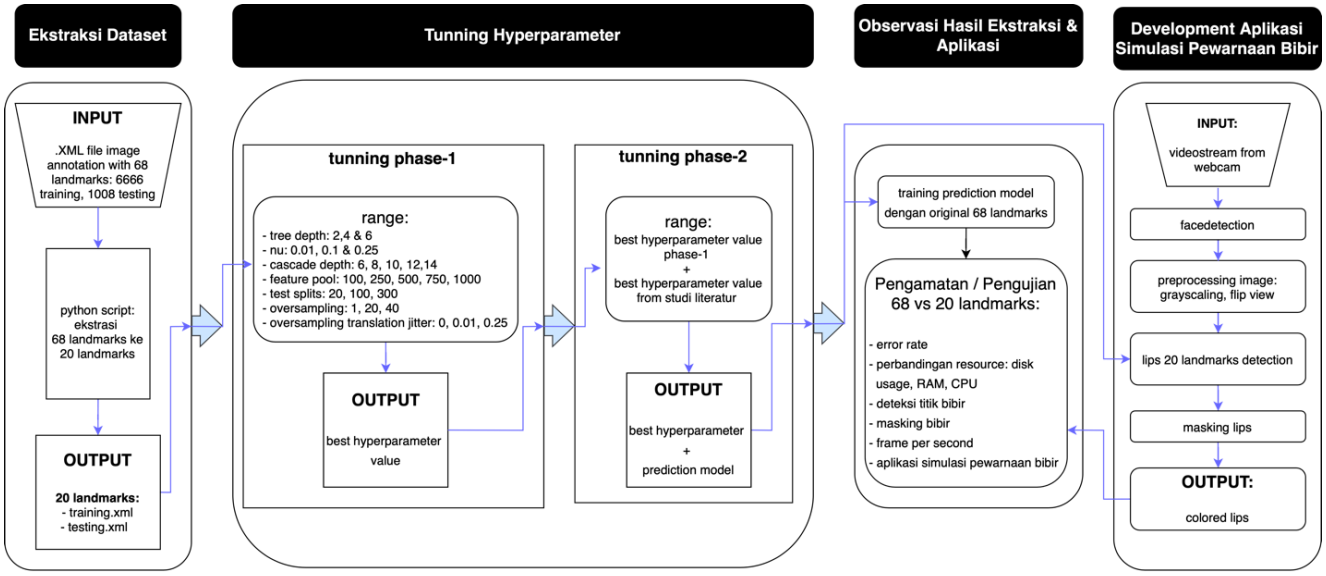
Teknologi dan mesin memudahkan kita dalam bekerja dan beraktivitas sehingga lebih efisien. Saat ini laptop dan *smartphone* telah menjadi kebutuhan umum masyarakat sebagai alat bantu pekerjaan maupun kegiatan belajar mengajar. Dalam ilmu teknologi komputer, banyak sekali cabang ilmu diantaranya adalah *computer vision* (penglihatan computer) yaitu cabang ilmu yang mempelajari tentang bagaimana suatu sistem dapat mengenali suatu objek yaitu kombinasi antara *image processing* (pengolahan citra) dan *pattern recognition* (pengenalan pola), salah satu contoh aplikasi penglihatan komputer adalah pendeteksi suatu objek atau wajah. Deteksi wajah didasarkan pada identifikasi dan menemukan lokasi citra wajah manusia pada gambar, terlepas dari ukuran, posisi, dan kondisi [1]. Pendeteksian wajah menjadi peranan yang penting dalam pembuatan aplikasi pengenalan wajah. Deteksi wajah didefinisikan sebagai proses mengekstrak wajah dari *scenes* (wilayah). Ekstraksi fitur melibatkan perolehan fitur wajah yang relevan dari data. Fitur-fitur ini dapat berupa daerah wajah tertentu, variasi, sudut atau ukuran, yang relevan dengan manusia atau tidak, misalnya jarak antara mata. Dalam tugas identifikasi, sistem akan melaporkan identitas dari database / *dataset* seperti iBUG 300W [2][3][4][5]. Fase ini melibatkan metode perbandingan, algoritma klasifikasi dan ukuran akurasi. Fase-fase ini dapat digabungkan, atau yang baru dapat ditambahkan. Karena itu, kita bisa temukan banyak pendekatan teknik yang berbeda untuk masalah pengenalan wajah. Tetapi untuk simulasi pewarnaan bibir hanya membutuhkan deteksi bibir, apakah metode *Ensemble of Regression Tree* dapat mendeteksi area bibir, alih-alih mendeteksi seluruh fitur wajah?

Machine Learning merupakan adalah studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa menggunakan instruksi eksplisit, dengan mengandalkan pola dan inferensi sebagai gantinya[6]. *Dlib-ml* merupakan salah satu *library* untuk *machine learning* [7] yang dapat mendeteksi wajah dengan membedakan antara mata, hidung dan mulut yang menggunakan metode *Ensemble Regression Tree* (ERT) [8]. Tetapi, belum ditemukan pemanfaatan metode ERT untuk pendeteksian spesifik area bibir saja, yang benar-benar tanpa menghiraukan area selain bibir pada hasil *prediction* modelnya. Dan dengan bantuan *OpenCV*, *library* dari fungsi pemrograman untuk *computer vision realtime* menjadi alat bantu untuk input *videostream* melalui *webcam* dan pengolahan citra. Sehingga, dengan dua *library* tersebut pada bagian bibir dapat disimulasikan warna seolah-olah menggunakan *makeup* lipstik secara *virtual*. Untuk melakukan tugasnya mendeteksi wajah, *machine learning* perlu belajar dari contoh sebelumnya dari tugas yang diberikan selama proses yang disebut *training* atau pelatihan. Setelah pelatihan, tugas dapat dilakukan pada data baru dalam proses yang disebut inferensi.

Akan tetapi, seperti konsep *pagination* atau *lazy-load* dimana hanya mengambil, menampilkan dan menggunakan data yang dibutuhkan saja. Maka diduga akan lebih cepat proses inferensi, jika terdapat sebuah *prediction* model yang hanya mendeteksi area bibir, dibandingkan harus mendeteksi titik-titik wajah (*face landmarks*) secara keseluruhan. Belum ditemukan penggunaan metode *Ensemble Regression Tree* untuk mendeteksi spesifik area bibir menjadi salah satu masalah pada penelitian ini, maka akan dikembangkan melalui proses ekstraksi titik anotasi wajah pada *dataset*. Kemudian, akan disajikan kegunaan dari ekstraksi jumlah anotasi *dataset* dengan metode *Ensemble of Regression Tree* untuk deteksi bibir dalam FPS [9][10][11]. Tujuan dari penelitian adalah menyajikan data hasil *tuning hyperparameter* dan *hyperparameter* terbaik untuk mendeteksi bibir saja. Maka konfigurasi nilai *hyperparameter* terbaik untuk pendeteksian spesifik area bibir akan dituliskan pada bagian kesimpulan.

II. METODE PENELITIAN

Machine Learning metode *Ensemble of Regression Trees (ERT)* digunakan untuk melakukan deteksi wajah dan pengenalan bibir. *Tools* / *framework* menggunakan *Dlib* untuk proses *training* dan *testing*, kemudian untuk aplikasi simulasi pewarnaan bibir digunakan *library* tambahan yaitu OpenCV. Alur sistematika proses dari penelitian ini dituangkan dalam Gambar 1:



Gambar 1. Alur penelitian

Berdasarkan alur penelitian dan pengembangan aplikasi di Gambar 1 dapat dijelaskan tahapan-tahapannya sebagai berikut:

- data *training* dan data *testing* berupa gambar dan anotasi lokasi *landmark* wajah didapatkan dari situs resmi iBug-300W, yang pada penelitian sebelumnya menggunakan *dataset* HELEN [8]
- anotasi lokasi landmark yang awalnya berjumlah 68 titik (key points) akan diekstraksi menjadi 20 titik bibir saja.
- untuk proses *testing* dan *training* pendeteksian bibir menggunakan metode *Ensemble of Regression Tree* (ERT) akan digunakan DLib *library*, yang sebelumnya telah mengimplementasikan metode ERT pada *library*-nya
- *Tuning hyperparameter* tahap kedua mengacu pada halaman 4 section 3 dari *paper* [8], berupa: *cascade* T (*cascade depth*) = 10, K = 500, *tree depth* = 5, P (*feature pool size*) = 400 piksel, S (*num test splits*) = 20
- proses *masking* bibir dan simulasi pewarnaan bibir akan menggunakan *OpenCV library*

A. *Dataset*

Dataset dari https://ibug.doc.ic.ac.uk/resources/300-W digunakan sebagai data utama. D*ataset* tersebut terdapat folder seperti: AFW, Helen, iBug dan FLPW dengan total file size 1,89 Gb. [12][13], kemudian dilakukan ekstraksi (pengurangan) keypoint *landmark* yang awalnya mendeteksi seluruh fitur wajah dengan total 68 titik, menjadi hanya *landmark* bibir 20 titik.

Dari data tersebut, berikut jumlah detail dari anotasi gambar wajah pada Tabel 1:

TABEL 1
DATA ANOTASI GAMBAR WAJAH

Jenis Gambar	Jumlah Data *Training*	Jumlah Data *Testing*
iBug 300W	6666	1008

B. *Ekstraksi Keypoint Landmark*

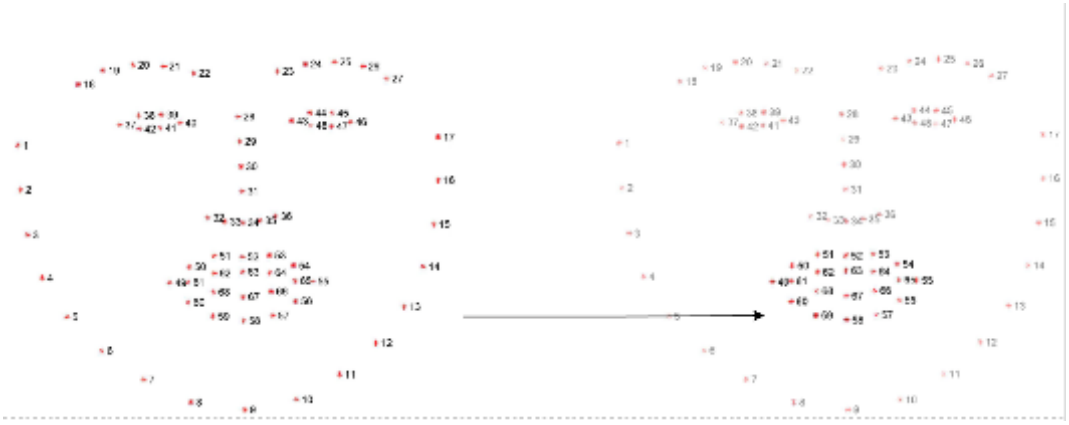
Dataset dari iBUG-300W dapat digunakan untuk memprediksi lokasi dari masing-masing 68 (x, y) pasangan koordinat *face landmark* untuk setiap titik pada wajah (alis, dagu, mata, hidung dan bibir). Anotasi *dataset training* dan *testing* tersedia dalam format XML. Detail titik (*keypoint*) dari anotasi wajah pada *dataset* Tabel 2:

TABEL 2
DETAIL TITIK (*KEYPOINT*)

Bagian	Rentang Titik	Jumlah Titik
Dagu (*jaw*)	1 – 17	17
Alis kanan (*right eyebrow*)	18 – 22	5
Alis kiri (*left eyebrow*)	23 – 27	5

Hidung (*nose*)	28 – 36	9
Mata kanan (*right eye*)	37 – 42	6
Mata kiri (*left eye*)	43 – 48	6
Mulut / bibir (*lips*)	49 – 68	20

Maka, dilakukan modifikasi / ekstraksi jumlah titik menjadi 20 titik *lips landmark*, seperti pada Gambar 2. Yaitu dengan cara *parsing* file anotasi 68 titik wajah yang berformat XML dengan jumlah 6666 data *training* dan 1008 data *testing*, sehingga akan menghasilkan file XML baru dengan 20 titik bibir saja menggunakan *python script*.



Gambar 2. Ekstraksi *keypoint dataset*

Untuk memahami bagaimana melakukan ekstraksi *keypoint dataset*, mari kita lihat bagaimana *keypoint face landmarks* dianotasi dalam *dataset* iBUG-300W dengan memeriksa potongan file **labels_ibug_300W_train.xml** kode sumber 1 pada folder *training*:

```
...
<images>
  <image file='lfpw/trainset/image_0457.png'>
    <box top='78' left='74' width='138' height='140'>
      <part name='00' x='55' y='141'/>
      <part name='01' x='59' y='161'/>
      …
      <part name='66' x='164' y='192'/>
      <part name='67' x='160' y='192'/>
    </box>
  </image>
</images>
...
```

Kode sumber 1. Potongan file labels_ibug_300W_train.xml

Semua data *training* dalam *dataset* iBUG-300W dituliskan dalam format XML. Setiap gambar memiliki penanda tag ***image***. Di dalam tag ***image*** memiliki atribut ***file*** yang merujuk lokasi penyimpanan gambar. Selain itu, setiap ***image*** memiliki tag ***box*** yang terkait dengannya. tag ***box*** mewakili koordinat kotak pembatas (*bounding box*) wajah pada gambar. Untuk memahami bagaimana tag ***box*** mewakili kotak *bounding box*, berikut adalah empat atributnya:

1. *Top* : koordinat y awal dari *bounding box*.
2. *Left* : koordinat x awal dari *bounding box*.
3. *Width* : lebar *bounding box*.
4. *Height* : tinggi *bounding box*.

Dan setiap ***part*** memiliki tiga atribut: *Name*: indeks/nama *landmark* wajah, x: koordinat x dari *landmark*. y: koordinat y. Maka dengan *Python script* akan menghasilkan 20 titik bibir mulai dari *part* 48 sampai 67 yang merupakan indeks dari koordinat x,y titik bibir. Hasilnya seperti kode sumber 2 (tidak ada titik wajah dagu, mata, alis dan hitung part 0 – 46):

```
...
<images>
  <image file='lfpw/trainset/image_0457.png'>
    <box top='78' left='74' width='138' height='140'>
      <part name='48' x='139' y='194'/>
      <part name='49' x='151' y='186'/>
      <part name='50' x='159' y='180'/>
      <part name='51' x='163' y='182'/>
      <part name='52' x='168' y='180'/>
      <part name='53' x='173' y='183'/>
      <part name='54' x='176' y='189'/>
      <part name='55' x='174' y='193'/>
      <part name='56' x='170' y='197'/>
      <part name='57' x='165' y='199'/>
      <part name='58' x='160' y='199'/>
      <part name='59' x='152' y='198'/>
      <part name='60' x='143' y='194'/>
      <part name='61' x='159' y='186'/>
      <part name='62' x='163' y='187'/>
      <part name='63' x='168' y='186'/>
      <part name='64' x='174' y='189'/>
      <part name='65' x='168' y='191'/>
      <part name='66' x='164' y='192'/>
      <part name='67' x='160' y='192'/>
    </box>
  </image>
  ...
</images>
...
```

Kode sumber 2. Potongan file hasil ekstraksi *keypoint* dataset

C. *Pendeteksian Titik Bibir dengan Dlib Ensemble Regression Tree*

Training dilakukan dengan *dataset* iBug 300W dan dilakukan konfigurasi *hyperparameter* yang digunakan dalam *training* [14]. Telah diketahui bahwa *Machine Learning* merupakan studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa harus diprogram secara eksplisit [15]. Sebaliknya, ia belajar dari contoh sebelumnya dari tugas yang diberikan selama proses yang disebut *training* atau pelatihan. Setelah pelatihan, tugas dapat dilakukan pada data baru dalam proses yang disebut inferensi [16]. Maka dari *dataset* yang telah diekstraksi dari 68 *keypoint face landmark* menjadi 20 *keypoint lips landmark*, selanjutnya akan digunakan *library* DLib untuk melakukan *training dan testing* model pendeteksian area bibir. Gambar 3 adalah gambaran yang bersumber dan hasil modifikasi dari [8] untuk proses *training* pada metode *Ensemble Regression Tree* (ERT):

(a) T=0 (b) T=1 (c) T=2 (d) T=3 (e) T=10

Gambar 3. Proses regresi deteksi bibir metode ERT

Dari Gambar 3 memperlihatkan proses *ensemble learning* (pembelajaran bertahap) menggunakan *Ensemble Regression Tree* (ERT) untuk mendeteksi *landmark* bibir. Metode *ensemble* yang dilakukan menggunakan *cascade regressor* sebanyak T-kali *regressor* saat proses *training* pada *dataset* anotasi *landmark* bibir. Untuk *dataset* akan memiliki koordinat x,y dari *landmark* yang dinotasikan / disimbolkan dengan x_i pada sebuah gambar I. Kemudian sebuah *shape* $S = x\frac{T}{1}, x\frac{T}{2}, \ldots, x\frac{T}{p}$ adalah kumpulan dari semua koordinat p untuk titik bibir pada sebuah gambar I. Terlihat pada regresi pertama / T=0 hasil

deteksi bibir masih kurang akurat karena posisi *landmark* (titik) bibir yang terlalu ke kiri. Kemudian secara bertahap proses *fine tuning* berlangsung sampai 10-kali regresi sehingga menghasilkan deteksi bibir yang lebih akurat.

Untuk proses *training* pendeteksian bibir jika dituangkan dalam bentuk *pseudocode* seperti kode sumber 3:

for I = *1 to* 10
  secara acak *generate* 400 lokasi piksel terhadap Sm di area Sm
  petakan 400 piksel ke gambar $I1$, ..., IN menggunakan *similarity transform* dari Sm ke $Sc1$, ..., ScN
  for j = 1 *to* 500
    for n = 1 *to* N, *set* $\Delta n = Sn - Scn$, *end for*
      generate fitur *perbedaan* piksel dan ambang batas (*threshold*) untuk mengelompokkan $\Delta n's$ menjadi 16 *leaf regression tree*
      for m = 1 to 16
      set Δ dari *leaf* Δm ke-m sebagai rata-rata Δn's yang dikelompokkan ke-m *leaf*
      update Scn pada *leaf* ke-m sebagai $Scn = Scn + \Delta m$
    end for
  end for
end for

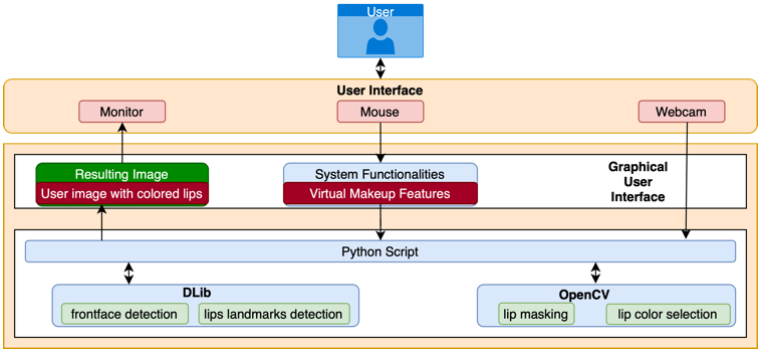
Kode sumber 3. Untuk proses *training* deteksi bibir

Penjelasan dari *pseudo code*:

- *Input* berupa gambar wajah $I1$, $I2$, ..., IN berbentuk persegi dan bentuk *ground truth* $S1$, $S2$, ..., SN
- normalisasi $S1$, …, SN ke $S1n$, ..., SNn dengan memperbesar persegi wajah yang sesuai ke ukuran $1x1$
- atur *mean* wajah Sm sebagai rata-rata $S1n$, ..., SNn
- petakan *mean* wajah S ke bentuk wajah persegi sebagai bentuk saat ini $Sc1$, $Sc2$, ..., ScN

D. *Block Diagram Aplikasi Simulasi Pewarnaan Bibir*

User dapat berinteraksi dengan aplikasi simulasi pewarnaan bibir melalui bantuan perangkat layar monitor, mouse dan webcam seperti pada Gambar 4. Webcam berfungsi untuk menangkap objek yang ada di depannya, kemudian objek akan dideteksi sebagai wajah atau non-wajah, jika terdeteksi wajah maka selanjutnya akan diproses face detection dan deteksi landmark bibir. Dan dengan menggunakan mouse, user dapat mengganti warna bibir sesuai keinginannya.



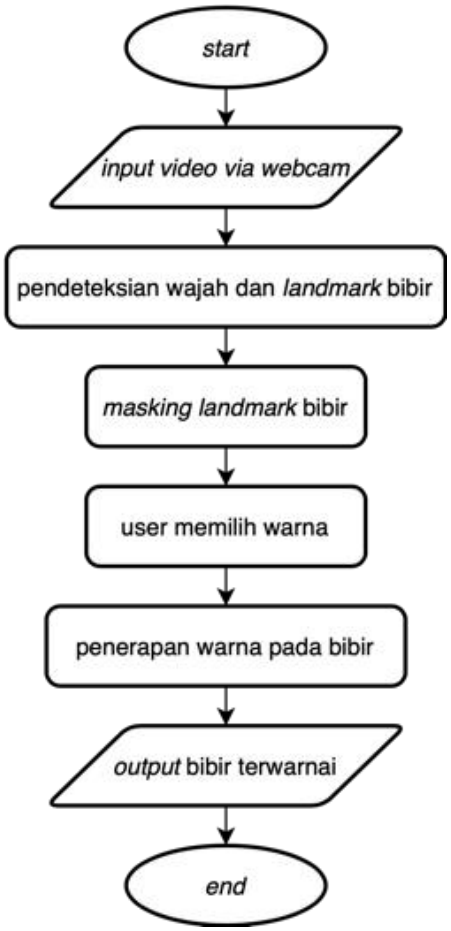
Gambar 4. *Blockdiagram* aplikasi simulasi pewarnaan bibir

III. HASIL DAN PEMBAHASAN

A. *Flowchart Overview Aplikasi Simulasi Pewarnaan Bibir*

Aplikasi simulasi pewarnaan bibir bekerja dengan cara yang awalnya kamera *webcam* mengambil *input* berupa video. Kemudian aplikasi akan mendeteksi apakah terdapat wajah dari video *stream*, jika ditemukan adanya wajah akan diberikan *landmark* bibir. Kemudian, setelah proses pemberian *landmark* pada bibir selesai, dilanjutkan dengan pembuatan *masking* pada bibir. Dan selanjutnya *user* dapat memilih warna untuk bibir, setelah user memilih warna, maka warna tersebut akan

diterapkan pada *masking* bibir. *Output* yang dihasilkan oleh aplikasi berupa video wajah pengguna dengan pewarnaan bibir yang sudah diaplikasikan seperti digambarkan pada *Flowchart* Gambar 5.



Gambar 5. *Flowcart* keseluruhan aplikasi simulasi pewarnaan bibir

B. Tuning Hyperparameter

Hyperparameter digunakan untuk mengkonfigurasi berbagai aspek dari algoritma *machine learning* dan dapat memiliki efek yang sangat bervariasi pada model yang dihasilkan dan kinerjanya. Beberapa kondisi yang digunakan yaitu:

- *tree_depth*: *tree* yang digunakan pada satu kali *cascade*. Akan ada 2^*tree_depth leaf* di setiap *tree*. Nilai tree_depth yang lebih kecil akan menghasilkan *tree* yang lebih pendek yang lebih cepat, tetapi berpotensi kurang akurat.
- nu: *regularization* parameter untuk membantu model melakukan *generalize* sebuah *shape*. Nilai yang mendekati 1 membuat model lebih *fit* dengan data pelatihan, tetapi berpotensi menyebabkan *overfitting*.
- *cascade_depth*: jumlah *cascade* untuk *refine and tune* (menyempurnakan) prediksi awal. Parameter ini memiliki dampak dramatis pada akurasi dan ukuran *output* file model. Semakin banyak *cascade*, maka semakin besar ukuran model (dan berpotensi lebih akurat).
- *feature_pool_size*: mengontrol jumlah piksel acak yang digunakan untuk menghasilkan fitur di dalam *cascade*. Semakin banyak piksel, semakin lambat model akan berjalan (tetapi menghasilkan prediktor *shape* yang lebih akurat).
- *num_test_splits*: jumlah pemisahan tes mempengaruhi waktu pelatihan dan akurasi model. Semakin banyak jumlahnya, semakin besar kemungkinan menghasilkan prediktor yang akurat, tapi nilai yang besar akan menyebabkan waktu pelatihan membengkak dan membutuhkan waktu lebih lama untuk menyelesaikan pelatihan prediktor.
- *oversampling_amount*: mengontrol jumlah "*jitter*" untuk diterapkan saat *training shape predictor*. Misalnya nilai 5 akan menghasilkan peningkatan 5x dalam data waktu *training* model. Maka semakin besar *oversampling_amount*, semakin lama waktu yang dibutuhkan model untuk melakukan *training*.

- *oversampling_translation_jitter*: Mengontrol jumlah *translation* "*jitter*" atau augmentasi yang diterapkan pada data.

Kemudian penjelasan dari kolom untuk *output* percobaan *tuning hyperparameter* yaitu:

- *Inference speed* adalah seberapa cepat model dapat memprediksi bibir pada satu file gambar
- *Accuracy* adalah seberapa tepat dan akurat model dalam prediksinya
- *Model size*, semakin besar modelnya, semakin banyak ruang penyimpanan (*disk space*) yang dibutuhkan, dan semakin banyak sumber daya komputasi yang dibutuhkan. Oleh karena itu, model yang lebih kecil lebih disukai.

Seluruh proses *tuning hyperparameter* dilakukan pada file *Python script* "*tune_predictor_hyperparams*.py". *Inference_speed* didapatkan dari *prediction model* yang melakukan deteksi / prediksi pada satu buah gambar *placeholder* "*image*_0237_*mirror*.jpg" yang dilakukan pada sebuah kode fungsi bernama "*evaluate_model_speed*". Prediksi *inference speed* dilakukan secara *default* yaitu sebanyak 10x dan pada setiap prediksi akan dilakukan:

- *Convert* gambar ke *grayscale*
- Deteksi wajah dari *input* gambar
- Jika paling tidak terdapat 1 wajah terdeteksi, maka akan dicatat waktu *start* prediksi, kemudian dilakukan prediksi menggunakan *prediction model*. Terakhir setelah selesai prediksi akan dicatat waktu *end*, yang selanjutkan akan dikurangi waktu *end* dan *start*, hasilnya disimpan kembali ke dalam sebuah *list* / *array* sebanyak 10x yang kemudian akan dirata-rata.

Untuk pencarian *hyperparameter* terbaik, pada *trials* pertama sebagai dasar *tuning hyperparameter* yang digunakan adalah tree_*depth* = 2, 4 dan 6; nu = 0,01, 0,1 dan 0,25; *cascade_depth* = 6, 8, 10, 12 dan 14; *feature_pool_size* = 100, 250, 500, 750 dan 1000; *num_test_splits* = 20, 100 dan 300; *oversampling_amount* = 1, 20 dan 40; *oversampling_translation_jitter* = 0; 0,1 dan 0,25

Dari *hyperparameter* tersebut, didapatkan hasil *tuning* seperti pada Tabel 3:

TABEL 3
HASIL PERTAMA *TUNING HYPERPARAMETER*

Tr	Td	Nu	Cd	Fpz	Nts	Oa	Otj	Is	Tt	Tre	Tee	Mz	Vse
1	2	0,1	6	1000	300	40	0	0,0002	16387	12,5	12,79	8110229	1
2	2	0,25	6	100	20	1	0,25	0,0003	14174	9,85	14,68	8022892	1
3	4	0,01	12	100	300	40	0	0,001	35745	13,5	13,56	21277347	1
4	6	0,01	10	250	20	40	0,25	0,0013	16328	20	18,44	56665155	1
5	6	0,1	8	100	20	20	0	0,0013	14894	7,24	10,71	46516006	1
6	6	0,1	14	1000	100	40	0,25	0,002	23155	7,48	10,14	77663269	1
7	2	0,1	10	1000	20	1	0,1	0,0004	14202	10,1	13,97	9424345	1
8	6	0,25	6	250	100	40	0,1	0,0007	16801	7,21	10,13	36436306	1
9	4	0,25	14	500	20	1	0,25	0,0014	14249	2,87	31,81	23898905	11
10	6	0,25	10	1000	300	40	0,1	0,0014	25611	4,71	8,918	57188350	2
11	4	0,25	8	100	100	40	0,1	0,0008	16414	9,55	10,62	16229978	1
12	2	0,1	12	500	100	20	0,1	0,0003	15326	11,6	11,79	9991311	1
13	4	0,01	12	250	20	20	0	0,0009	15274	15,1	15,11	21301724	1
14	2	0,01	12	750	20	20	0	0,0003	14786	19,1	18,48	10038019	1
15	2	0,1	6	750	20	1	0,1	0,0002	14182	12,4	15,11	8088199	1
16	4	0,25	8	500	300	40	0	0,0009	19966	7,53	9,043	16331771	1
17	4	0,01	6	750	300	40	0	0,0004	18321	15,9	15,88	13820921	1
18	2	0,01	10	250	20	40	0,25	0,0002	15188	29,2	26,79	9300593	1
19	2	0,1	10	750	20	20	0	0,0005	14743	12,3	12,65	9388099	1
20	4	0,01	14	500	20	1	0,1	0,0015	14231	11,8	15,91	23964694	1
21	2	0,1	6	100	20	1	0,1	0,0001	14184	13,1	15,74	8023328	1
22	6	0,1	6	100	20	40	0,1	0,0005	15349	10	11,38	36434030	1
23	4	0,01	12	1000	20	20	0,25	0,0009	15271	22,9	21,12	21558826	1
24	6	0,01	8	100	100	1	0	0,0009	14289	7	16,32	46220141	2
25	2	0,25	6	750	20	20	0,1	0,0002	14486	13,1	13,04	8088374	1
26	2	0,25	14	1000	300	40	0,25	0,0004	19252	14,7	13,96	10739451	1
27	2	0,1	14	100	100	20	0,1	0,0004	15125	11,8	11,97	10536784	1
28	4	0,1	12	750	100	40	0	0,001	35849	7,99	9,46	21501596	1
29	6	0,1	12	250	20	1	0,25	0,0017	14256	2,8	83,39	66321571	30
30	4	0,01	6	750	100	40	0,25	0,0005	16223	25,3	22,86	13820694	1
31	2	0,25	10	100	300	20	0	0,0003	15849	10,2	10,99	9280210	1
32	4	0,01	6	500	100	20	0,25	0,0005	15165	24,9	22,8	13784783	1

Tr	Td	Nu	Cd	Fpz	Nts	Oa	Otj	Is	Tt	Tre	Tee	Mz	Vse
33	2	0,1	12	750	100	20	0,25	0,0003	15430	17,1	16,07	10038335	1
34	2	0,1	8	1000	100	1	0	0,0001	14197	9,65	13,69	8767172	1
35	6	0,01	10	1000	20	20	0,25	0,0014	15623	19,3	18,22	57225504	1
36	6	0,1	6	250	20	40	0,25	0,0006	15681	13,4	14,33	36449228	1
37	6	0,1	14	500	100	1	0	0,002	14433	2,8	86,16	76554834	31
38	2	0,01	8	750	300	40	0,25	0,0001	17381	28,7	26,37	8738264	1
39	6	0,01	8	1000	100	20	0	0,001	16924	12,1	13,1	47006761	1
40	4	0,01	14	750	300	1	0,1	0,0014	14481	9	14,57	24040584	2
41	6	0,01	8	500	300	1	0,25	0,0011	14621	6,21	16,08	46473606	3
42	4	0,1	12	1000	20	40	0	0,001	17151	8,73	10,26	21557908	1
43	4	0,1	14	250	100	40	0	0,0013	18670	7,81	9,505	23828548	1
44	2	0,1	10	1000	100	20	0,1	0,0002	15470	12	12,1	9424808	1
45	6	0,01	8	1000	20	1	0,25	0,001	14277	8,27	16,64	46734754	2
46	4	0,25	14	1000	20	1	0,25	0,0013	14253	2,87	30,33	24046427	11
47	2	0,1	10	250	20	1	0,1	0,0002	14215	10,2	13,89	9300226	1
48	2	0,25	10	500	100	1	0,1	0,0002	14238	6,21	13,56	9347668	2
49	6	0,1	14	250	100	40	0,25	0,0018	21726	7,57	10,32	76859997	1
50	2	0,25	12	250	300	20	0,1	0,0002	16053	10,1	11,12	9932950	1
51	2	0,01	12	500	300	1	0,25	0,0002	14261	16,8	17,5	9990925	1
52	4	0,01	10	250	100	20	0	0,0007	15528	14,4	14,5	18774515	1
53	6	0,01	12	750	300	1	0,25	0,0016	14533	4,95	15,93	66815076	3

Akronim dari nama kolom: TR: *trials*, TD: *tree depth*, NU, CD: *cascade depth*, FPZ: *feature pool size*, NTS: *num test splits*, OA: *oversampling amount*, OTJ: *oversampling translation jitter*, IS: *inference speed*, TT: *training time*, TRE: *training error*, TEE: *testing error*, MZ: *model size*, VSE: *versus error, testing error divided by training error*.

Dari *trials* dengan *hyperparameter* Tabel 3, dilakukan *sorting descending* nilai *training error*, pertama untuk *tuning hyperparameter* didapatkan hasil *training error* terkecil pada percobaan ke-37 dengan nilai **2,80**. Nilai tersebut sangat baik karena nilai *Sum of Square Error* semakin mendekati 0, maka akan semakin baik. Akan tetapi, model ini terlihat *overfitting* karena pada saat dievaluasi pada data *testing* memiliki nilai *testing error*-nya yaitu **86,16**, yang berarti jika nilai *testing error* dibagi dengan *training error* menghasilkan nilai **31** kali. Yang artinya *error rate* naik 31 kali lipat atau performa model semakin menurun 31 kali pada saat melakukan prediksi pada *video* / *image* yang belum pernah digunakan / dilihat oleh *prediction model*. *Overfitting* ini terjadi juga pada trials ke-29, 9 dan 46 dengan nilai masing-masing *training error*: 2,80; 2,86; 2,86. Nilai *testing error*: 83,39; 31,80; 30,33. Dengan perbandingan *training* dan *testing error*: **30** kali; **11** kali; **11** kali.

Nilai perbandingan *training* dan *testing error*, pada percobaan ke-29 dan ke-9 turun cukup drastis dari angka puluhan ke belasan, tetapi terlihat jauh lebih kecil pada percobaan ke-10 dengan nilai **2** kali, dengan nilai *training error* **4,70** dan *testing error* **8,91**. Akan tetapi, untuk *file size* dari *prediction model* masih terlihat cukup besar yakni 57,18 MB. Pengamatan *file size* untuk mendapatkan *hyperparameter* terbaik, mulai mengecil di urutan ke- 48 dengan *file size* 9.34MB.

Pengamatan terakhir dilakukan dengan *sorting* / mengurutkan data hasil *trials* pada kolom *testing_error* secara *ascending*, menunjukkan bahwa *trials* ke-16 adalah yang terbaik jika dilihat dari sisi *testing_error*, *model_size* dan hasil pembagian antara *testing training error* seperti pada Gambar 6:

	trials	tree_depth	nu	cascade_depth	feature_pool_si	num_test_splits	oversampling_a	oversampling_tr	inference_speed	training_time	training_error	testing_error	model_siz	testing / training
2	37	6	0,1	14	500	100	1	0	0,002025723	14433,21445	2,804966894	86,16431928	76554834	31
3	29	6	0,1	12	250	20	1	0,25	0,001667404	14256,00998	2,804969763	83,39018955	66321571	30
4	9	4	0,25	14	500	20	1	0,25	0,001396274	14248,6035	2,865275815	31,80795445	23898905	11
5	46	4	0,25	14	1000	20	1	0,25	0,00126729	14253,43552	2,865899087	30,33404961	24046427	11
6	10	6	0,25	10	1000	300	40	0,1	0,001360893	25611,48332	4,707032812	8,917515415	57188350	2
7	53	6	0,01	12	750	300	1	0,25	0,001576113	14532,71093	4,947951144	15,92995626	66815076	3
8	41	6	0,01	8	500	300	1	0,25	0,001061511	14620,76156	6,211989478	16,08250867	46473606	3
9	48	2	0,25	10	500	100	1	0,1	0,000160336	14238,11231	6,213139091	13,5644184	9347668	2
10	24	6	0,01	8	100	100	1	0	0,000925159	14289,21818	7,003087099	16,32297729	46220141	2
11	8	6	0,25	6	250	100	40	0,1	0,000686645	16801,24467	7,209111893	10,12585976	36436306	1
12	5	6	0,1	8	100	20	20	0	0,001345372	14893,56776	7,235306267	10,71469504	46516006	1
13	6	6	0,1	14	1000	100	40	0,25	0,002027726	23154,5505	7,478094503	10,14097043	77663269	1
14	16	4	0,25	8	500	300	40	0	0,000879955	19965,74196	7,52887763	9,043093333	16331771	1

Gambar 6. Hasil *1st tunning hyperparameter sort by testing error ascending*

Hasil *research paper* [8] dan situs DLib: *cascade* T (*cascade_depth*) = 10, K = 500, *tree depth* = 5, P (*feature_pool_size*) = 400 piksel, S (*num_test_splits*) = 20, nu: 0,05, *oversampling_amount* = 300, *oversampling_translation_jitter* = 0.1.

Hasil *hyperparameter* terbaik pada 1st *batch-trials* urutan ke-16 menjadi yang terbaik, karena memiliki perbandingan *train_test* dan *testing_error* rendah yaitu 1:1, ukuran file yang cukup kecil 16.33MB, nilai terendah urutan ke-2 untuk *testing_error* dengan angka 9.04 dan *inference_speed* dengan angka 0,00087. Nilai *hyperparameter* pada *trial* ke-16 jika di-*summary* yaitu: *tree_depth* = 4; nu = 0,25; *cascade_depth* = 8; *feature_pool_size* = 500; *num_test_splits* = 300; *oversampling_amount* = 40; *oversampling_translation_jitter* = 0.

Maka untuk mendapatkan hasil *hyperparameter* terbaik dilakukan penggabungan dari kedua hasil *trials* dan *research*, kemudian digunakan kembali untuk *tuning hyperparameter* dengan *range* nilai sebagai berikut: *tree_depth* = 4 dan 5; nu = 0,05 dan 0,25; *cascade_depth* = 8 dan 10; *feature_pool_size* = 400 dan 500; *num_test_splits* = 20 dan 300; *oversampling_amount* = 40 dan 300; *oversampling_translation_jitter* = 0 dan 0,1. Dengan hasil pada Tabel 4:

TABEL 4
HASIL KEDUA *TUNING HYPERPARAMETER*

Tr	Td	Nu	Cd	Fpz	Nts	Oa	Otj	Is	Tt	Tre	Tee	Ms	Vse
1	5	0,05	10	500	20	40	0,1	0,0013	4677	11,2	11,54	40798224	1
2	4	0,25	10	500	20	40	0	0,0011	3867	8,15	9,996	20414390	1
3	4	0,05	8	500	300	40	0,1	0,0009	20045	12	11,82	16333286	1
4	5	0,25	10	500	20	40	0	0,0013	4514	6,13	9,516	40785289	2
5	5	0,05	10	400	300	40	0	0,0014	30636	8,5	9,497	40744066	1
6	4	0,05	8	400	20	40	0	0,001	2970	12,9	12,69	16307850	1
7	5	0,25	8	400	20	40	0	0,0017	5451	6	10,08	106233046	2
8	5	0,25	8	500	300	40	0	0,0017	40738	5,04	9,565	106262279	2
9	5	0,05	8	400	20	40	0,1	0,0016	5529	10,4	10,79	106268384	1
10	4	0,25	8	500	300	40	0	0,0009	19966	7,53	9,043	16331771	1

Kesimpulannya *trials* (TR) ke-10 adalah parameter terbaik dengan memperhatikan *trials error* (TEE) terkecil urutan pertama, *inference speed* (IS) tercepat urutan pertama dan *model size* (MS) terkecil urutan kedua.

Selanjutnya pengamatan dan perbandingan antara *prediction model* hasil ekstraksi *dataset* dan original *dataset* (68 *face landmarks*), maka dilakukan *training* dengan *hyperparameter* yang sama terhadap *training dataset*, dengan hasil Tabel 5:

TABEL 5
HASIL TRAINING PADA *ORIGINAL DATASET* (68 TITIK WAJAH)

No	Td	Nu	Cd	Fpz	Nts	Oa	Otj	Is	Tt	Tre	Tee	Ms	Vse
1	4	0,25	8	500	300	40	0	0,0015	32868	6,8	9,602	53165277	1

C. Pengujian Error Rate dari Prediction Model

Untuk menguji akurasi *prediction* model hasil *training*, maka dilakukan evaluasi terhadap 85% *training* dan 15% *training* pada Tabel 6:

TABEL 6
PENGUJIAN *EROR RATE*

Model Name	***Evaluate Error Rate On***	
	Training Dataset	***Testing Dataset***
68 face landmarks	6,799369984	9,602137643
20 face landmarks	7,52887763	9,043093333
Summary	>0,729 (-0,096%)	<0,559 (+0,058%)

Evaluation atau pengujian *prediction* model dilakukan pada kedua model yaitu *training* dan *testing* untuk melakukan verifikasi bahwa model yang dikembangkan tidak *overfitting* dan idealnya *prediction* model dapat mendeteksi gambar di luar *dataset training*. Nilai evaluasi dari *training loss* akan lebih kecil daripada *testing loss*. Hal ini tidak berarti bahwa model yang dikembangkan memiliki performa yang buruk, akan tetapi secara sederhana berarti bahwa model tersebut melakukan pekerjaan prediksi yang lebih baik pada *training* data daripada *testing* data.

Dari hasil pengujian disimpulkan bahwa *error rate prediction model* 20 *face landmark* yang dikembangkan pada *testing dataset* memiliki *error rate* yang lebih baik jika dibandingkan dengan *prediction model* original pada fitur 68 titik wajah.

D. Pengamatan Ukuran File dari Prediction Model

Pada *device resource* terbatas seperti *embedded system*, *IoT* seperti *Raspberry Pi*, ukuran file dari *prediction* model dapat berpengaruh terhadap kinerja sistem. Semakin kecil ukuran file akan menghemat *resource* terutama media penyimpanan file. Dari Tabel 7 dapat ditarik kesimpulan bahwa, pengurangan jumlah *face landmark* berbanding lurus dengan pengurangan ukuran file dari *prediction model*. Yaitu awalnya 68 titik wajah memiliki ukuran file 53,2 MB, setelah diekstraksi menjadi 20 titik, ukuran file menyusut menjadi 16,3 MB atau **69,36%** lebih kecil.

TABEL 7
PERBANDINGAN *PREDICTION MODEL FILE SIZE*

Model Name	***Prediction* Model *File Size***
68 *face landmarks*	53,2 MB
20 *face landmarks*	16,3 MB
Summary	< 69,36% (16,3 MB)

E. Pengamatan Penggunaan Resource CPU dan Memory

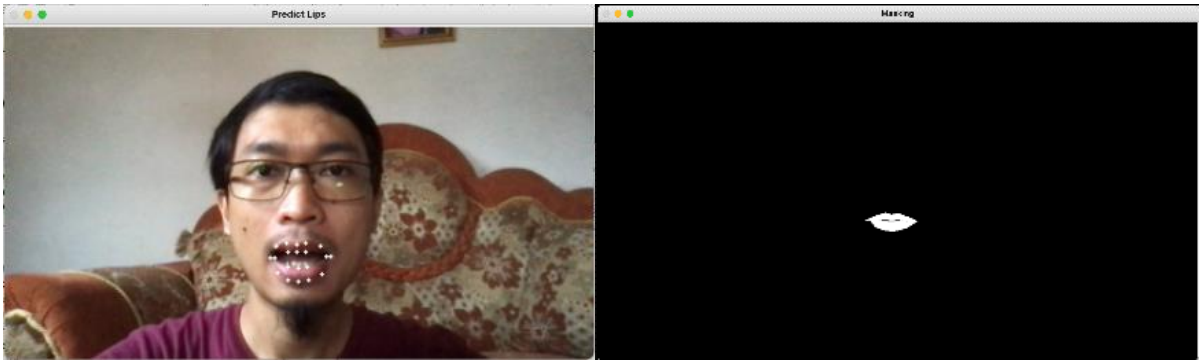
Dari Tabel 8 diketahui dengan mengurangi jumlah *face landmark* akan berbanding lurus dengan pengurangan penggunaan *resource* sebesar **30,8%** untuk *memory / RAM*, **65,1%** untuk penggunaan *resource hardisk* dan **3,8%** CPU *resource*. Spesifikasi perangkat yang digunakan adalah MacBook Pro (13-inch, 2019), prosesor 2,4 GHz Quad-Core Intel Core i5 dan RAM 16 GB 2133 MHz LPDDR3.

TABEL 8
PERBANDINGAN PENGGUNAAN *RESOURCE*

Model Name	***Memory***	***Disk***	**CPU**
20 *face landmarks*	119,5MB	16,3MB	76,8 %
68 *face landmarks*	172,7MB	53,2MB	80,6 %
Summary	< 30.8% (53,2MB)	< 69,36% (36,9 MB)	< 3.8%

F. Pengujian Deteksi Titik Bibir (Face Localization) dan Masking

Setelah dilakukan *face detection* maka gambar wajah akan di-*convert* ke warna *grayscale*, selanjutnya dilakukan *face localization* berupa 20 titik pada area bibir. Titik-titik bibir tersebut berjumlah 20 titik yang didapatkan dari hasil *training* yang kemudian digunakan pada *function shape_predictor* dari DLib *library*. Pada saat menjalankan aplikasi, diberikan titik lingkaran berwarna putih pada titik bibir sehingga dapat terlihat posisinya secara *visual*. Dari 20 titik bibir tersebut pada *Python script* akan dilakukan pengkoneksian sehingga dari awalnya berbentuk titik-titik saja akan berubah menjadi bentuk pola bibir (*masking*) seperti Gambar 7.



Gambar 7. Pengujian pendeteksian titik bibir (kiri) dan *Masking* bibir (kanan)

Masking bentuk bibir berwarna putih, melalui proses sebagai berikut:

- gambar wajah akan ditranslasi ke dalam bentuk *matrix array*
- setiap *value* dari *matrix array* akan diganti dengan 0 agar berwarna hitam

- *masking* titik-titik bibir hasil dari *prediction model* dengan warna putih
- kemudian hasil *masking* dan gambar wajah aslinya akan digabungkan sehingga menghasilkan warna latar belakang hitam dan *masking* area bibir saja yang berwarna putih

Masking bibir kemudian akan digunakan pada saat interaksi dilakukan oleh *user* dengan melakukan *dragging* pada pilihan warna. Karena pewarnaan tidak langsung dilakukan pada *input videostream webcam*, akan tetapi pewarnaan pada bibir akan diberikan pada bagian / *layer masking* ini.

G. Pengujian FPS pada Berbagai Kondisi

Di Tabel 9 pengujian kondisi pertama yaitu "pencahayaan cukup" terlihat terdapat peningkatan hampir dua FPS yaitu sebesar 1,89, yang awalnya untuk 68 *landmarks* wajah menghasilkan FPS sebesar 12,51, kemudian untuk 20 *landmarks* bibir menghasil FPS sebesar 14,40. Peningkatan FPS hampir terjadi pada semua kondisi pada percobaan ini, kecuali percobaan pada "mulut terbuka" dan "wanita usia anak", hipotesis untuk hal ini (tidak adanya peningkatan FPS) dapat disebabkan oleh banyak hal seperti: objek yang terlalu jauh dari kamera *webcam*, kurangnya jumlah data wajah pada proses *training* untuk kondisi tersebut, pergerakan terlalu signifikan, pencahayaan yang kurang baik.

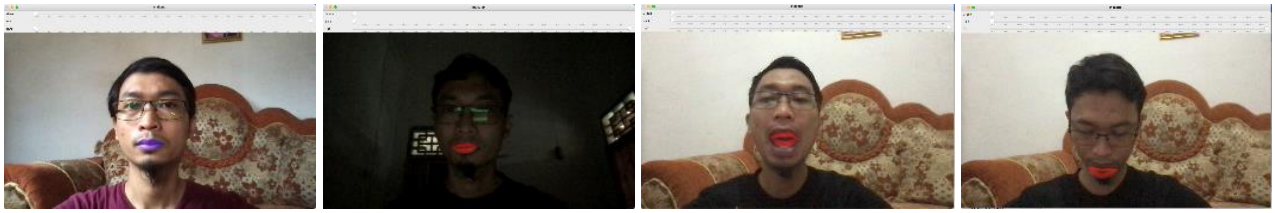
TABEL 9
HASIL PENGUJIAN FPS

No	Kondisi / Objek	FPS 20 *Landmarks*	FPS 68 *Landmarks*	*Delta*
1	Pencahayaan cukup	14,40	12,51	> 1,89 (13%)
2	Mulut terbuka	15,66	15,68	< 0,02 (-0,1%)
3	Kepala miring	15,27	12,78	> 2,49 (16%)
4	Kurang pencahayaan	14,31	14,28	> 0,03 (0,2%)
5	Menghadap atas	15,27	15,14	> 0,13 (0,8%)
6	Kepala miring kiri	15,41	14,69	> 0,72 (0,4)
7	Wanita usia sedang	14,88	14,87	> 0,01 (0,06%)
8	Wanita usia anak	14,92	14,92	= (0%)

Dari hasil pengujian dan pengamatan, dapat disimpulkan bahwa dengan ekstraksi *dataset* meningkatkan kecepatan *frame per second* (FPS) atau paling minimal sama. Akan tetapi, hasilnya terlihat ada penurunan pada kondisi mulut terbuka, disarankan untuk explorasi metode lain untuk pengujian FPS. Sebagai catatan, bahwa kedua program yaitu 20 *landmark* dan 68 *landmark* dijalankan secara bersamaan kemudian di-*capture* secara bersamaan pula, guna mendapatkan nilai FPS *linear*.

H. Pengujian Aplikasi Simulasi Pewarnaan Bibir

Dari hasil pengujian aplikasi simulasi pewarnaan bibir pada Gambar 8, dapat dilihat bahwa bibir dapat diwarnai sesuai dengan keinginan user mengikuti *masking* bentuk bibir. Untuk kondisi minim cahaya, simulasi pewarnaan bibir masih tetap dapat diterapkan. Akan tetapi, pada kondisi "wajah menghadap bawah" dan "mulut terbuka lebar" secara visual untuk deteksi bibir tidak sesuai dengan posisi yang seharusnya.



(a) warna biru (b) minim cahaya (c) mulut terbuka lebar (d) wajah menghadap bawah

Gambar 8. Pengujian aplikasi simulasi pewarnaan bibir

Langkah terakhir untuk aplikasi simulasi pewarnaan bibir, maka dari hasil *masking* bibir dapat diberikan warna tertentu sesuai keinginan user, sehingga seolah-olah seperti mengaplikasikan warna lipstik pada bibir. Berikut hasil pengujian aplikasi yang meliputi proses awal menjalankan aplikasi simulasi pewarnaan bibir menggunakan metode *Ensemble Regression Tree*.

Setelah melalui beberapa pengujian untuk kriteria dan skenario, maka hasil yang diperoleh pada Tabel 10:

TABEL 10
HASIL PENGUJIAN

No	Kriteria / Skenario Pengujian	Hasil
1	*Sum of square error*	Berhasil sebagian. Performa p*rediction model* lebih buruk pada *training dataset* sebesar -0,096%. Akan tetapi, terbukti lebih baik pada *testing dataset* sebesar 0,058%.
2	Ukuran file	Berhasil. Ukuran file *prediction* model berkurang 69,36%
3	*Resource* CPU & RAM	Berhasil. Penggunaan CPU berkurang 3,8% dan RAM 30,8%
4	*Face detection*	Berhasil. Wajah berhasil terdeteksi
5	*lips landmarks localization* (titik bibir)	Berhasil. *Prediction* model dapat mendeteksi titik-titik (*landmarks*) bibir saja
6	*Masking* bibir	Berhasil. Aplikasi mendeteksi pola bibir dengan mengkoneksikan titik-titik bibir.
7	*Frame per Second* (FPS)	Berhasil sebagian. Karena hasil tidak konsisten, ekstraksi *dataset* menghasilkan FPS yang lebih baik, tetapi terkadang sebaliknya.
8	Simulasi pewarnaan bibir	Berhasil sebagian. Aplikasi dapat mendeteksi bibir dan mewarnai *masking* bibir, tetapi tidak akurat secara visual untuk kondisi tertentu.

Dari Tabel 10 dapat ditarik kesimpulan bahwa terdapat dua kategori yaitu berhasil dan berhasil sebagian. Untuk kategori pengujian yang "Berhasil" yaitu dengan ekstraksi *dataset* anotasi wajah dari 68 *landmarks* menjadi 20 *landmarks* tetap dapat mendeteksi fitur bibir berupa 20 titik bibir. Kemudian berhasil pula menurunkan ukuran *prediction model* yang sangat signifikan sebesar 69,36% dari 53,2 MB menjadi 16,3 MB. Kemudian berhasil pula menurunkan penggunaan *resource* sebesar 3,8% CPU dan 30,8% RAM. Ketiga hal ini diharapkan dapat memberikan gambaran kepada perusahaan atau *software developer* bahwa terdapat keuntungan yang didapatkan dari ekstraksi ini, sehingga lebih memudahkan pada saat akan implementasi / pengembangan aplikasi lain pada perangkat yang mungkin memiliki *resource* terbatas seperti *Raspberry Pi*.

Kategori kedua dari hasil penelitian adalah "berhasil sebagian", artinya ada beberapa kondisi yang tidak terpenuhi. Pertama pada *Sum of Square Error* menunjukkan penurunan terhadap *training error* pada perbandingan antara *prediction model original* dengan hasil ekstraksi, tetapi tidak menjadi masalah, karena peningkatan terjadi pada hasil pengujian terhadap *testing dataset*. Yang artinya *prediction model* memiliki performa yang lebih baik dalam memprediksi sebuah objek yang belum pernah dilihatnya.

Kemudian untuk pengujian *frame per second* (FPS) juga termasuk dalam kategori "berhasil sebagian" karena pada pengujian terdapat penurunan FPS pada *prediction model* hasil ekstraksi, walaupun pada hasil *training* menunjukkan peningkatan *inference speed* sebesar 39%, maka perlu dilakukan penelitian dan pengujian lebih lanjut mengenai pengujian FPS.

IV. SIMPULAN

Berdasarkan hasil penelitian dapat disimpulkan metode *Ensemble Regression Tree* dan ekstraksi anotasi *dataset* menghasilkan *prediction* model yang dapat mendeteksi spesifik area bibir. Dengan konfigurasi *hyperparameter* yaitu: *tree_depth* = 4; nu = 0,25; *cascade_depth* = 8; *feature_pool_size* = 500; *num_test_splits* = 300; *oversampling_amount* = 40; *oversampling_translation_jitter* = 0. Dengan *sum of square error:* 9,043 pada *testing dataset*, *inference speed* $1{,}464\times10^{-3}$ dan ukuran file 16,3 MB dari *prediction* model.

Prediction model dapat digunakan untuk simulasi pewarnaan bibir. Hasil ekstraksi anotasi *dataset* berpengaruh terhadap beberapa hal yaitu: penghematan *resource hardisk* 69,36%, RAM 30,8% dan CPU 3,8%. Serta, mengurangi *error rate* sebesar 0,058% dan meningkatkan *inference speed* sebesar 39%.

Pada penelitian lanjutan disarankan beberapa hal, yaitu: 1). pengujian kecepatan dalam *frame per second* (FPS) masih sangat perlu dilakukan pengujian lanjutan, karena hasilnya masih menunjukkan penurunan pada beberapa kondisi. Hal ini mungkin dapat dicapai dengan awalnya langsung mendeteksi bibir, alih-alih mendeteksi seluruh area wajah. 2). melakukan *cropping input* gambar untuk *training* dan *testing* berdasarkan lokasi koordinat x dan y yang terdapat pada file anotasi, sehingga diharapkan dapat mengurangi beban komputasi. 3). Pendeteksian wajah dapat dilakukan dengan metode terkini berbasis *neural network* seperti CNN (*convolutional neural network*) yang dapat memanfaatkan GPU untuk percepatan proses *training* dan metode ini juga memiliki akurasi yang baik. Maka disarankan untuk dilakukan penelitian lanjutan untuk pendeteksi bibir, simulasi pewarnaan bibir atau virtual *makeup* dengan metode CNN. 4). terdapat beberapa *hyperparameter* yang tidak tereksplorasi karena *resource* RAM yang digunakan masih kurang besar yakni 16GB, diharapkan dengan mencoba memperbesar *resource* maka *hyperparameter* dapat tereksplorasi lebih banyak dan lebih baik.

DAFTAR PUSTAKA

[1] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[2] R. U. Delfana, A. R. Andrie, and L. Nadhifatul, "Penerapan Facial Landmark Point Untuk Klasifikasi Jenis Kelamin Berdasarkan Citra Wajah," *J. Inform. Polinema*, vol. 6, no. 1, pp. 55–60, 2020.
[3] C. Álvarez Casado and M. Bordallo López, "Real-time face alignment: evaluation methods, training strategies and implementation optimization,"

J. Real-Time Image Process., vol. 18, no. 6, pp. 2239–2267, 2021.
[4] Z. Liu *et al.*, "Semantic alignment: Finding semantically consistent ground-truth for facial landmark detection," in *Prosiding IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3462–3471.
[5] T. Soukupová and C. Jan, "Real-Time Eye Blink Detection using Facial Landmarks," *21st Computer Vision Winter Workshop*, 2016, pp. 1–8.
[6] R. R. Pratama, "Analisis Model Machine Learning Terhadap Pengenalan Aktifitas Manusia," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 19, no. 2, pp. 302–311, 2020.
[7] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.
[8] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Prosiding IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874.
[9] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2019.
[10] A. D. F. S. Borges and C. H. Morimoto, "A virtual makeup augmented reality system," in *Prosiding 2019 21st Symposium on Virtual and Augmented Reality, SVR 2019*, 2019, pp. 34–42.
[11] Y. Kartynnik and A. Ablavatski, "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs," in *Prosiding Computer Vision for Augmented and Virtual Reality 2019, IEEE*, 2019, pp. 2–5.
[12] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark Localization Challenge," in *Prosiding IEEE International Conference on Computer Vision*, 2013, pp. 397–403.
[13] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 Faces In-The-Wild Challenge: database and results," *Image Vis. Comput.*, vol. 47, pp. 3–18, 2016.
[14] J. Omar, N. H. Shabrina, A. N. Bhakti, and A. Patria, "Emotion Recognition using Convolutional Neural Network on Virtual Meeting Image," *Ultim. Comput. J. Sist. Komput.*, vol. 13, no. 1, 2021.
[15] B. Rao, "Machine Learning Algorithms: A Review," *Int. J. Comput. Sci. Inf. Technol.*, vol. 7, no. 3, pp. 1174–1179, 2016.
[16] E. Mjolsness and D. DeCoste, "Machine learning for science: State of the art and future prospects," *Science*, vol. 293, no. 5537, pp. 2051–2055, 2001.