Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*

TESIS

Oleh:
Andi Hakim Arif
1911601431

PROGRAM STUDI MAGISTER ILMU KOMPUTER
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

JAKARTA
GASAL 2021/2022

Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*

TESIS

Diajukan untuk memenuhi salah satu persyaratan memperoleh gelar Magister Ilmu Komputer (MKOM)

Oleh:
Andi Hakim Arif
1911601431

PROGRAM STUDI MAGISTER ILMU KOMPUTER
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

JAKARTA
GASAL 2021/2022

Abstrak

Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*

Oleh: Andi Hakim Arif (1911601431)

Teknologi membantu kita dalam banyak aktifitas seperti bekerja lebih efisien, informasi serta hiburan juga mudah didapatkan. Fenomena covid-19 menjadi titik dimana teknologi menjadi alat yang sangat penting, karena kegiatan sekolah, bekerja dan berdagang dapat dilakukan secara daring atau tanpa terbatas oleh jarak. Untuk jual beli produk fisik seperti baju, sepatu maupun *makeup* lebih nyaman dilakukan secara fisik, karena pengguna dapat mencoba produk secara langsung. Namun dengan kondisi pandemi menyebabkan mencoba *makeup* terutama lipstik akan sulit dilakukan karena menggunakan masker. Maka akan dikembangkan prototype (purwarupa) virtual *makeup* (*try-on*) sehingga user dapat mensimulasikan penggunaan lipstik dengan menggunakan algoritma *Ensemble of Regression Trees* (ERT). Kemudian isu selanjutnya, *dataset* yang tersedia adalah untuk seluruh bagian wajah dengan total 68 titik wajah, padahal untuk virtual *makeup* hanya membutuhkan titik bibir, sehingga akan dilakukan modifikasi / ekstraksi terhadap *dataset* sehingga menghasilkan *dataset* dan *prediction model* baru yang hanya berupa area bibir dengan total 20 titik. Dengan ekstraksi tersebut menghemat penggunaan *resource*: *hardisk* 69.36%, RAM 30.8% dan CPU 3.8%; mengurangi *error rate* sebesar 0.058%; dan juga meningkatkan *inference speed* sebesar 39%.

Kata Kunci : virtual *makeup*, virtual *try-on*, *ensemble of regression trees*, pendeteksi wajah

Abstract

Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*

By: Andi Hakim Arif (1911601431)

Technology helps us in many activities such as working efficiently, information and entertainment are also easily accessible. The Covid-19 pandemic was the point where technology becomes an important tool, because school, work and shopping activities can be done remotely or not limited by distance. To buy physical products such as clothes, shoes and makeup, it is more convenient to do it physically, because users can try the product directly. However, with the pandemic conditions causing makeup trial, especially lipstick, will be difficult because them using a mask. Then a virtual makeup prototype (try-on) will be developed so users can simulate applying lipstick using the Ensemble of Regression Trees (ERT) algorithm. Then the next issue, currently dataset is available only for all parts of the face with a total of 68 facial points, while for virtual makeup it only requires point lips, so modifications / extractions will be carried out on the dataset to produce a new dataset and prediction model which is only a lip area with a total of 20 point. This extraction saves resource usage: 69.36% hard disk, 30.8% RAM and 3.8% CPU; reduce the error rate by 0.058%; and also increases the inference speed by 39%.

Keywords: virtual makeup, virtual try-on, ensemble of regression trees, face detection

Kata Pengantar

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat-Nya penulis bisa menyelesaikan proposal penelitian tesis. Tujuan dari penulisan proposal penelitian tesis ini adalah sebagai salah satu syarat untuk menyusun tesis pada Program Studi Magister Ilmu Komputer, Universitas Budi Luhur Jakarta.

Rasa dan ucapan terima kasih penulis persembahkan kepada semua pihak yang telah membantu penulis dalam menyusun proposal penelitian tesis ini:

1. Bapak Dr. Deni Mahdiana, S.Kom., M.M., M.Kom, selaku Dekan Fakultas Teknologi Informasi Universitas Budi Luhur.
2. Ibu Dr. Rusdah, M.Kom, selaku Ketua Program Studi Magister Ilmu Komputer Universitas Budi Luhur, yang telah memverifikasi proposal tesis ini.
3. Bapak Dr. Ir. Wendi Usino, M.M., M.Sc, selaku Rektor Universitas Budi Luhur dan sekaligus sebagai dosen pembimbing proposal tesis yang telah membimbing dan memotivasi penulis dalam mengerjakan proposal penelitian tesis ini.
4. Bapak Dr. Achmad Solichin, S.Kom, M.T.I selaku dosen pembimbing tesis yang sudah dengan sabar mengarahkan dan banyak membantu dalam proses penyelesaian tesis ini.
5. Rekan-rekan mahasiswa MKOM Universitas Budi Luhur, terima kasih atas kebersamaan, kerja keras dan dukungan semangatnya.
6. Kedua orangtua yang menyarankan untuk melanjutkan studi ke jejang magister ini.
7. Kepada Istri tercinta atas perhatian, setia menunggu dan memberikan motivasi yang sangat besar.
8. Kepada rekan kantor di Tokopedia terutama leader yang sudah banyak membantu memberikan kelonggaran pekerjaan

Penulis menyadari masih banyak kekurangan dari proposal penelitian tesis ini, oleh karena itu penulis mengharapkan kritik dan saran yang bersifat membangun demi kesempurnaan penelitian tesis nantinya. Semoga proposal penelitian tesis ini masih dapat memberikan manfaat dari keterbatasannya. *Amin*.

Jakarta, 15 January 2022

Andi Hakim Arif

Daftar Isi

ABSTRAK II
ABSTRACT III
KATA PENGANTAR IV
DAFTAR ISI V
DAFTAR TABEL VII
DAFTAR GAMBAR VIII
BAB I: PENDAHULUAN 1
1.1. LATAR BELAKANG 1
1.2. MASALAH PENELITIAN 2
1.2.1 Identifikasi Masalah 2
1.2.2 Batasan Masalah 2
1.2.3 Rumusan Masalah 2
1.3. TUJUAN DAN MANFAAT PENELITIAN 3
1.3.1 Tujuan Penelitian 3
1.3.2 Manfaat Penelitian 3
1.4. SISTEMATIKA / TATA URUT PENULISAN 3
BAB II: LANSADAN TEORI DAN KERANGKA KONSEP/PEMIKIRAN 5
2.1. LANDASAN TEORI 5
2.1.1. *Machine Learning* (ML) 5
2.1.2. *Ensemble of Regression Trees* 6
2.1.3. Virtual *Makeup* 12
2.1.4. *Dataset* 13
2.1.5. DLib 14
2.1.6. OpenCV 15
2.1.7. *Face Recognition* 15
2.1.8. HOG *Face Recognition* 18
2.1.9. *Image Processing* 20
2.1.10. *Pattern Recognition* 20
2.2. KAJIAN LITERATUR 21
2.3. TINJAUAN OBJEK PENELITIAN 23
2.4. KERANGKA KONSEP PEMIKIRAN 24
2.5. HIPOTESIS 25
BAB III: METODOLOGI DAN RANCANGAN/DESAIN PENELITIAN 26
3.1. TAHAPAN PENELITIAN 26
3.2. METODE PENELITIAN 26
3.2.1. *Dataset* 27
3.3. TEKNIK ANALISIS, RANCANGAN DAN PENGUJIAN 27
3.4.1. Ekstraksi *keypoint landmark* 28
3.4.2. Pendeteksian bibir dengan DLib ERT 30
3.4.3. Perancangan aplikasi virtual *makeup* 31
3.4.4. Pengujian aplikasi 32
BAB IV: PEMBAHASAN DAN HASIL PENELITIAN 34
4.1. PERSIAPAN PENGUJIAN, ANALISIS DAN TEMUAN-TEMUAN 34
4.1.1. Analisis Bahan dan Pengujian Penelitian 34
4.1.2. Analisis Temuan-Temuan 34
4.1.3. Analisis Kebutuhan Lingkungan Pengembangan dan Pengujian 36

4.2. PENGEMBANGAN APLIKASI 37
4.2.1. *Tunning Hyperparameter* 37
4.2.2. *Use Case* Diagram 43
4.2.3. Diagram Proses Sistem 44
4.2.4. *Flowchart* Diagram 46
4.2.5. Konstruksi Antarmuka 49
4.3. PENGUJIAN APLIKASI 50
4.3.1. Pengujian *error rate* dari *prediction* model 50
4.3.2. Pengamatan ukuran file dari *prediction* model 51
4.3.3. Pengamatan penggunaan CPU & *memory usage* 51
4.3.4. Pengujian deteksi wajah (*face detection*) 51
4.3.5. Pengujian deteksi titik bibir (*face localization*) 52
4.3.6. Pengujian *masking* bibir 53
4.3.7. Pengamatan *Frame Per Second* (FPS) 53
4.3.8. Pengujian aplikasi virtual *makeup* 56
4.4. PEMBAHASAN HASIL PENELITIAN 58
4.5. IMPLIKASI PENELITIAN 59
4.4.1. Aspek penelitian lanjutan 60
4.4.2. Aspek pengembangan lanjutan 60

BAB V: PENUTUP 61

5.1. KESIMPULAN 61
5.2. SARAN 61

DAFTAR PUSTAKA 63

Daftar Tabel

Tabel 2. 1 Perbandingan regularisasi 9
Tabel 2. 2 Contoh aplikasi *face recognition* 15
Tabel 2. 3 Studi Literatur 21
Tabel 3. 1 Data anotasi gambar wajah 27
Tabel 3. 2 Detail titik (*keypoint*) 28
Tabel 4. 1 Hasil pertama *tunning hyperparameter* 39
Tabel 4. 2 Hasil kedua tunning hyperparameter 43
Tabel 4. 3 Hasil training pada *original dataset* 43
Tabel 4. 4 Pengujian Error Rate 50
Tabel 4. 5 Perbandingan prediction model *file size* 51
Tabel 4. 6 Perbandingan penggunaan *resource* 51
Tabel 4. 7 Pengujian FPS pada berbagai kondisi 54
Tabel 4. 8 Hasil pengujian FPS 55
Tabel 4. 9 Pengujian aplikasi virtual *makeup* 56
Tabel 4. 10 Hasil Pengujian 58

Daftar Gambar

Gambar 2. 1 Klasifikasi *machine learning* (Rao, 2016) 6
Gambar 2. 2 Hasil yang dipilih pada *dataset* HELEN. *Ensemble Regression Trees* digunakan untuk mendeteksi 194 *landmark* di wajah dari satu gambar (Kazemi & Sullivan, 2014) 7
Gambar 2. 3 Contoh *decision tree* (Rao, 2016) 8
Gambar 2. 4 *Overfitting* dan *underfitting* (Putra, 2019) 10
Gambar 2. 5 Hasil regresi ERT dan *ground truth* 11
Gambar 2. 6 Contoh model arsitektur dari *direct regression* model untuk mendeksi 68 titik landmark wajah. Persegi panjang menandakan *fully-connected layer* dan jumlah dari *nodes* pada *layer* yang ada di bawah nya (Hsu et al., 2020) 12
Gambar 2. 7 Titik Wajah dari *Dataset* iBUG 300-w 14
Gambar 2. 8 *Element* dari dlib 14
Gambar 2. 9 Sistem umum *face recognition* 16
Gambar2.10 Contoh HOG *descriptors*, *patch size*=8x 8. cell menunjukkan orientasi dari *gradient* (Déniz et al., 2011) 18
Gambar 2. 11 Deksriptor HOG untuk mata kanan 19
Gambar 2. 12 HOG deteksi manusia 19
Gambar 2. 13 Kerangka Pemikiran 24
Gambar 3. 1 Tahapan Penelitian 26
Gambar 3. 2 Alur aplikasi 27
Gambar 3. 3 Tahapan pengujian penelitian 28
Gambar 3. 4 Ekstraksi keypoint dataset 29
Gambar 3. 5 Perancangan antarmuka 31
Gambar 3. 6 *Blockdiagram* aplikasi virtual *makeup* 32
Gambar 3. 7 Contoh hasil pengujian average error (Kazemi & Sullivan, 2014) 33
Gambar 4. 1 proses *tunning hyperparameter* 35
Gambar 4. 2 Hasil *tunning* pertama *hyperparameter sort by testing error ascending* 41
Gambar 4. 3 *Use Case* diagram aplikasi 44
Gambar 4. 4 Proses aplikasi *prototype* virtual *makeup* 45
Gambar 4. 5 *Flowcart* keseluruhan aplikasi virtual *makeup* 46
Gambar 4. 6 *Flowchart* pendeteksian wajah dan *landmarks* bibir 47
Gambar 4. 7 Hasil ekstraksi *landmark* bibir 47
Gambar 4. 8 Flowchart pembuatan *masking* 48
Gambar 4. 9 *Flowcart* pembuatan virtual *makeup* 48
Gambar 4. 10 *Flowchart* pembuatan virtual *makeup* 49
Gambar 4. 11 konstruksi antarmuka saat dijalankan 49
Gambar 4. 12 Pengujian deteksi wajah 52
Gambar 4. 13 Pengujian pendeteksian titik bibir 52
Gambar 4. 14 *Masking* bibir 53

BAB I: PENDAHULUAN

1.1. Latar Belakang

Teknologi dan mesin memudahkan kita dalam bekerja dan beraktifitas sehingga lebih efisien. Saat ini laptop dan *smartphone* telah menjadi kebutuhan umum masyarakat sebagai alat bantu pekerjaan maupun kegiatan belajar mengajar. Dalam ilmu teknologi computer, 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 dalam gambar terlepas dari ukuran, posisi, dan kondisi. Pendeteksian wajah menjadi peranan yang penting dalam pembuatan aplikasi pengenalan wajah.

Fenomena covid-19 yang meresahkan seluruh penjuru dunia, pada situasi yang serba harus menjaga jarak antara satu orang dengan orang lainnya meng-haruskan semua kegiatan dilakukan secara aman dengan menjaga jarak. Baik itu *meeting*, kerja, kegiatan belajar atau mengajar, transaksi jual atau beli, semuanya dilakukan secara *daring*. Ini menegaskan bahwa keamanan setiap *individu* harus selalu dijaga. Alangkah baiknya jika ingin berbelanja atau mencoba makeup seperti lipstik dapat dilakukan secara daring, sehingga mengurangi resiko penularan virus, maka akan dikembangkan sebuah *prototype* aplikasi virtual *makeup* menggunakan *machine learning* sebagai alat untuk mendeteksi bibir.

Machine Learning merupakan studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa diprogram secara eksplisit. Dlib-ml merupakan salah satu *library* untuk *machine learning* diharapkan dapat mendeteksi wajah dengan membedakan antara mata, hidung dan mulut yang menggunakan algoritma *Ensemble of Regression Trees* (ERT). Dan dengan bantuan OpenCV, library dari fungsi pemrograman untuk *computer vision* (penglihatan computer) *realtime* menjadi alat bantu untuk *input videostream* melalui *webcam*. Sehingga, akhirnya dengan bantuan dua *tools* tersebut pada bagian bibir dapat diaplikasikan warna seolah-olah menggunakan *makeup* lipstik secara virtual.

Akan tetapi, seperti konsep *pagination* dimana hanya mengambil dan menampilkan data yang dibutuhkan saja. Maka diduga akan lebih efisien jika terdapat sebuah *prediction* model yang hanya mendeteksi area tertentu pada wajah, dibandingkan harus mendeteksi titik-titk wajah (*facelandmarks*) secara keseluruh-an. Belum tersedia model data untuk spesifik area wajah saja, dalam penelitian ini adalah titik-titik pada bibir. Maka, penulis dengan menggunakan model untuk 68 titik wajah, akan mengekstraksi *dataset* anotasi wajah dari semua 68 menjadi 20 titik bibir. Diharapkan akan mengurangi penggunaan *resource*, mengurangi *error rate / performance* dan meningkatkan kecepatan deteksi dalam satuan *milliseconds* atau dalam *Frame Per Second* (FPS) dari model yang akan dikembangkan.

1.2. Masalah Penelitian

Dimasa pandemi covid19 salah satu himbauannya adalah untuk menjaga jarak, maka untuk proses berbelanja lipstik juga mengalami penyesuaian, serta penulis juga belum menemukan adanya model *dataset* yang spesifik untuk area bibir saja.

Permasalahan penelitian ini dijabarkan dalam 3 (tiga) sub bab, yaitu Identifikasi Masalah, Ruang Lingkup Masalah dan Rumusan Masalah. Berikut adalah penjabarannya:

1.2.1 Identifikasi Masalah

Dimasa pandemi covid-19 customer sulit untuk mencoba *makeup* (lipstik) karena pandemi ini tidak boleh dilakukan kontak langsung. Berikut adalah beberapa indentifikasi permasalah yang terdapat pada penelitian ini adalah :

1. Saat ini belum ditemukan *dataset* anotasi dan *prediction model* untuk spesifik titik-titik (*landmarks*) bibir
2. Saat ini belum diketahui keuntungan mengembangkan aplikasi virtual *makeup* dengan *dataset* anotasi bibir
3. Saat ini belum ditemukan informasi lengkap *hyperparameter* untuk *developer* dan spesifik area bibir

1.2.2 Batasan Masalah

Dalam penelitian yang dilakukan, terdapat beberapa batasan masalah yang digunakan, diantaranya :

1. Menggunakan algoritma *Ensemble of Regression Trees*
2. *Dataset* yang digunakan adalah iBUG 300W trace 68 keypoint
3. Pendeteksian wajah hanya pada area bibir
4. Hasil akhir dari *hyperparameter* adalah yang terbaik menurut penulis
5. Antarmuka dari virtual *makeup* bersifat final dan hanya dapat digunakan pada aplikasi desktop, karena masih dalam tahap *prototyping*
6. Penelitian masih dalam tahap *prototyping* sehingga belum akan membahas untuk akurasi warna

1.2.3 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Apakah dari data *face annotation iBug-300W* yang sudah ada, kemudian diekstraksi menjadi deteksi spesifik bibir saja, maka akan berpengaruh terhadap hal apa saja dari hasil ekstraksi tersebut?
2. Apakah model yang dihasilkan dapat mensimulasikan penggunan virtual *makeup* lipstik secara *realtime* melalui video *webcam*?
3. Nilai berapa saja yang harus digunakan pada *hyperparameter* untuk pendeteksian bibir?

1.3. Tujuan dan Manfaat Penelitian

Tujuan penelitian yang akan dikerjakan dan manfaat yang akan diperoleh diarahkan lebih spesifik agar tidak melebar dari batasan yaitu sebagai berikut:

1.3.1 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Menghasilkan *dataset* anotasi dan *prediction model*
2. Menghasilkan prototype virtual *makeup* untuk mencoba lipstik secara *realtime* melalui video *webcam* dan mengetahui keuntungannya
3. Menyajikan data hasil *tunning hyperparameter* dan *hyperparameter* terbaik menurut kebutuhan penulis

1.3.2 Manfaat Penelitian

Manfaat dengan adanya penelitian ini yaitu:

1. Diharapkan penelitian ini dapat dikembangkan untuk aplikasi virtual *makeup* yang lebih sempurna
2. Diharapkan model berupa titik bibir dapat membantu pengembang atau penelitian selanjutnya
3. Diharapkan data hasil *tunning hyperparameter* dapat diteliti lebih lanjut oleh *researcher* maupun dapat langsung digunakan oleh *developer*

1.4. Sistematika / Tata Urut Penulisan

Adapun sistematika penulisan dalam penyusunan proposal memiliki urutan sebagai berikut:

BAB I PENDAHULUAN

Bab isi menjelaskan tentang penjabaran latar belakang, masalah yang terdapat saat proses penelitian, tujuan dan manfaat dari penelitian ini, tata urut penulisan.

BAB II LANDASAN TEORI DAN KERANGKA KONSEP

Yaitu meliputi tentang landasan teori dalam penyusuan penelitian sebagai landasan dasat teori yang meenjadi landasan ilmiah sebagai tinjauan pustaka yaitu dari pengertian *machine learning*, algoritma *Ensemble of Regression Trees*, virtual *makeup*, *image processing*, kajian literatur, tinjauan objek penelitian, kerangka konsep pemikiran dan hipotesis.

BAB III METODOLOGI DAN RANCANGAN PENELITIAN

Yaitu meliputi tentang tahapan dalam penelitian, metode penelitian, *dataset* yang digunakan dalam penelitian, teknik analisis, rancangan, pengujian dan jadwal penelitian yang terorganisir untuk menyelidiki masalah dalam penelitian ini.

BAB IV PEMBAHASAN DAN HASIL PENELITIAN

Bab ini membahas persiapan pengujian, analisis dan temuan-temuan pada saat proses penelitian. Kemudian perancangan aplikasi, pengujian aplikasi hasil dari pengujian penelitian dan implikasi penelitian.

BAB V PENUTUP

Yaitu meliputi lembar yang berisi tentang kesimpulan dari penulisan tesis dalam penelitian ini. Serta saran dari penulis untuk penelitian ada pengembangan aplikasi kedepannya agar lebih baik lagi.

BAB II: LANSADAN TEORI DAN KERANGKA KONSEP/PEMIKIRAN

2.1. Landasan Teori

2.1.1. *Machine Learning* (ML)

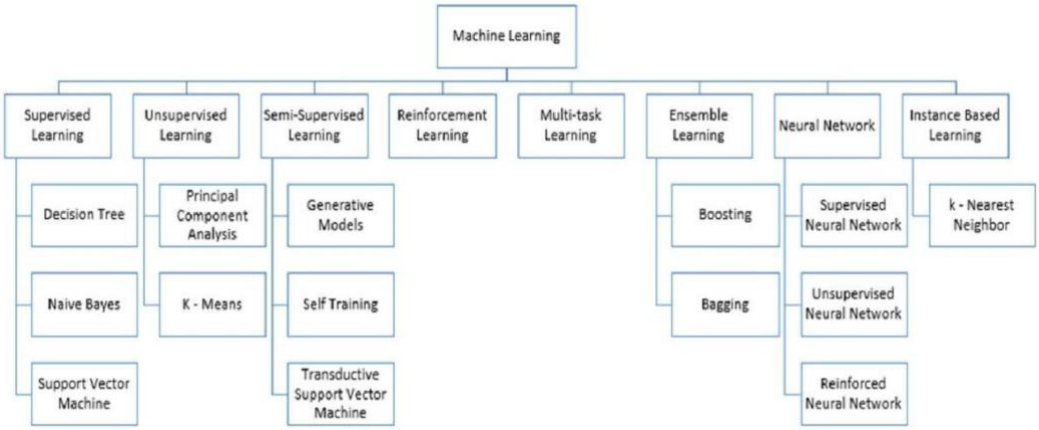
Saat ini, *Machine Learning* adalah bidang yang paling cepat berkembang dalam ilmu komputer yang mencakup berbagai bidang seperti pemasaran, perawatan kesehatan, manufaktur, keamanan informasi, dan transportasi (Wäldchen & Mäder, 2018). Alasan utama untuk "ledakan" literal teknik ini adalah ketersediaan dan pertemuan tiga hal berikut:

a) perangkat keras komputer yang lebih cepat dan lebih kuat, yaitu prosesor paralel besar-besaran dan unit pemrosesan grafis tujuan umum (GP-GPU)
b) algoritma perangkat lunak yang memanfaatkan arsitektur komputasi ini; dan
c) jumlah data pelatihan yang hampir tidak terbatas untuk masalah tertentu, seperti gambar digital, dokumen digital, posting media sosial, atau pengamatan dengan geolokasi.

Machine Learning merupakan studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa diprogram secara eksplisit (Rao, 2016). 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 (Mjolsness & DeCoste, 2001). *Machine Learning* membantu terutama dalam mengekstraksi informasi dari sejumlah besar data yang terus bertambah dan sangat berguna untuk aplikasi.

Sejak evolusinya, manusia telah menggunakan banyak jenis alat untuk menyelesaikan berbagai tugas dengan cara yang lebih sederhana. Kreativitas otak manusia menyebabkan penemuan mesin yang berbeda. Mesin-mesin ini memudahkan kehidupan manusia dengan memungkinkan manusia memenuhi berbagai kebutuhan hidup, termasuk bepergian, industri, dan komputasi. Dan *Machine Learning* adalah salah satunya.

Menurut Arthur Samuel *Machine learning* didefinisikan sebagai bidang studi yang memberikan komputer kemampuan untuk belajar tanpa diprogram secara eksplisit (Rao, 2016). Arthur Samuel sangat menyukai program permainan caturnya. Pembelajaran mesin (ML) digunakan untuk mengajari mesin cara menangani data dengan lebih efisien. Terkadang setelah melihat data, kami tidak dapat menginterpretasikan informasi ekstrak dari data. Dalam hal ini, kami menerapkan pembelajaran mesin. Dengan banyaknya kumpulan data tersedia, permintaan untuk pembelajaran mesin sedang meningkat. Banyak industri menerapkan pembelajaran mesin untuk mengekstrak data yang relevan. Tujuan dari pembelajaran mesin adalah untuk belajar dari data. Banyak penelitian telah dilakukan tentang bagaimana membuat mesin belajar sendiri tanpa diprogram secara eksplisit. Banyak ahli matematika dan pemrograman menerapkan beberapa pendekatan untuk menemukan solusi dari masalah ini yang memiliki kumpulan data yang sangat besar.



Gambar 2. 1 Klasifikasi *machine learning* (Rao, 2016)

Gambar di atas menjelaskan sekilas beberapa algoritme yang umum digunakan dalam pembelajaran mesin. *Machine learning* bergantung pada algoritma yang berbeda untuk memecahkan masalah data. Ilmuwan data ingin menunjukkan bahwa tidak ada satu jenis algoritma yang cocok untuk semua yang terbaik untuk memecahkan masalah. Jenis algoritma yang digunakan tergantung pada jenis masalah yang ingin kita pecahkan, jumlah variabel, jenis model yang paling cocok untuk itu, dan seterusnya.

2.1.2. *Ensemble of Regression Trees*

Ensemble Learning adalah proses di mana beberapa model, seperti pengklasifikasi atau system pakar, secara strategis dihasilkan dan digabungkan untuk memecahkan masalah kecerdasan komputasi tertentu (Rao, 2016). Pembelajaran *ensemble* terutama digunakan untuk meningkatkan kinerja model, atau mengurangi kemungkinan pemilihan model yang buruk. Aplikasi lain dari pembelajaran *ensemble* termasuk memberikan kepercayaan pada keputusan yang dibuat oleh model, memilih fitur yang optimal, fusi data, pembelajaran inkremental, pembelajaran non stasioner dan koreksi kesalahan.

Ensemble of Regression Trees (ERT) adalah framework pembelajaran berdasarkan *gradient boosting* untuk yang mengoptimalisasi *the sum of square error loss* dan secara alami menangani kehilangan *labelled* data secara seluruhnya atau sebagian. Dengan cara melakukan *training multiple base learner* sebagai bagian dari *ensemble* dan mendapatkan satu *output* hasil dari kombinasi *multiple* prediksi tersebut (Wang et al., 2018).

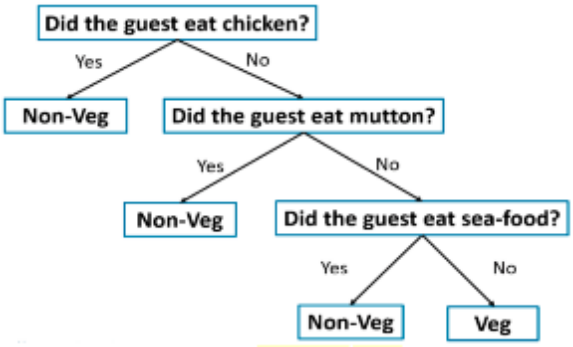
Gambar 2. 2 Hasil yang dipilih pada *dataset* HELEN. *Ensemble Regression Trees* digunakan untuk mendeteksi 194 *landmark* di wajah dari satu gambar (Kazemi & Sullivan, 2014)

Dari gambar di atas terlihat bahwa *Ensemble of Regression Trees* dapat mendeteksi landmark wajah dengan cukup baik, meskipun dengan posisi wajah yang beragam.

2.1.2.1. Decision Tree

Decision tree (DT) yang juga dikenal dengan *Classification and Regression Tree (CART)*. Itu tidak hanya dapat memecahkan masalah klasifikasi, tetapi juga dapat memecahkan masalah regresi (Wang et al., 2018). Proses membangun pohon regresi *(regression trees*) berdasarkan norma *minimum square error* adalah proses untuk menghasilkan pohon biner secara rekursif dengan memilih fitur dan titik split terbaik yang sesuai. Struktur pohon dapat disesuaikan sesuai dengan karakteristik kumpulan data. Oleh karena itu, kita tidak perlu mengatur struktur fungsi dan mampu menangani variabel diskrit dan kontinu pada waktu yang sama. Namun ketika struktur pohon regresi terlalu kompleks, masalah *over fitting* atau terjebak ke dalam titik minimum lokal mungkin muncul.

Algoritma *Decision Tree* adalah klasifikasi dasar dan metode regresi dan memiliki struktur pohon di mana setiap simpul (*node*) internal mewakili pengujian pada suatu atribut, setiap simpul daun (*leaf*) mewakili kategori, dan setiap cabang (*branch*) mewakili hasil pengujian. Singkatnya, sebagai algoritma pembelajaran mesin ensemble (*ensemble learning*), *Gradient Boostring Regression Trees* lebih baik daripada ANN dan SVM dalam perkiraan (*forecasting*) akurasi (Wang et al., 2018). Secara umum *decision tree* dapat berbentuk seperti berikut:



Gambar 2. 3 Contoh *decision tree* (Rao, 2016)

Node dalam grafik mewakili suatu peristiwa atau pilihan dan *edge* (tepi) grafik mewakili aturan atau kondisi keputusan. Setiap pohon terdiri dari node dan cabang. Setiap node mewakili atribut dalam grup yang akan diklasifikasikan dan setiap cabang mewakili nilai node tersebut.

2.1.2.2. Gradient Boosting

Menggunakan pohon keputusan (*decision tree*) sebagai pembelajar (*training*) yang lemah dan membangun *model* secara bertahap dengan meng-optimalkan *loss function* (Wang et al., 2018). Metode *boosting* merupakan jenis metode utama dalam metode *ensemble learning* dan secara bersama-sama membangun pembelajar yang kuat dengan menggabungkan pembelajar yang lemah melalui metode iteratif untuk mendapatkan hasil terbaik dalam waktu sesingkat mungkin. Algoritma *gradient boosting* menyelesaikan proses pembelajaran dengan memperbarui *loss function* dan gradien. Metode ini memecahkan masalah optimasi pada *function space* yang meniru metode gradien yang layak pada *numerical space*. Singkatnya, masalahnya adalah menemukan fungsi optimal yang meminimalkan nilai *loss function*. *Gradient-based method* mengubah-ubah parameter model w sehingga *loss* dapat diminimalkan.

2.1.2.3. Regularization parameter

Saat menggunakan algoritma *gradient boosting*, satu hal yang harus berhati-hati adalah menghindari *overfitting*. Untuk mendapatkan kesalahan pengujian yang lebih rendah perlu dilakukan beberapa bentuk regularisasi (Kazemi & Sullivan, 2014). Beberapa pendekatan untuk regularisasi adalah:

- Pendekatan paling sederhana adalah *srinkage*. Yaitu, melibatkan pengaturan tingkat pembelajaran dalam algoritma *gradient boosting* menjadi kurang dari 1 (disarankan nilainya = 0,1).
- Regularisasi juga dapat dicapai dengan *averaging* prediksi dari *multiple regression tree*. Pada setiap iterasi dari algoritma *gradient boosting* alih-alih memasang satu pohon regresi ke residual, maka digunakan beberapa pohon (10 pohon pada percobaan kami) dan hasilnya dirata-rata (Kazemi & Sullivan, 2014). Regularisasi dengan *averaging* memiliki keuntungan menjadi lebih ter-ukur, karena memungkinkan paralelisasi pada saat *training* pada skala masalah / jumlah training data yang lebih besar.

Tabel 2. 1 Perbandingan regularisasi

	Unregularized	Shrinkage	Averaging
Error	.103	.049	.049

Tujuan *machine learning* adalah membuat model yang mampu memprediksi data yang belum pernah dilihat (*unseen instances*) dengan tepat; disebut sebagai generalisasi (Putra, 2019). Pada *library DLib* parameter *regularization* dilambangkan dengan nu.

2.1.2.4. Sum of square error

Sum of square error adalah seberapa jauh antara data dari *regression line*. Mirip dengan *Mean squared error* yang digunakan untuk mengetahui perbedaan antara estimator dengan hasil estimasi. MSE digunakan sebagai tolok ukur suatu *estimator*. Hasil yang diperoleh selalu berupa angka positif. Semakin mendekati nol maka semakin baik kinerja *estimator* tersebut (Lazaro et al., 2017).

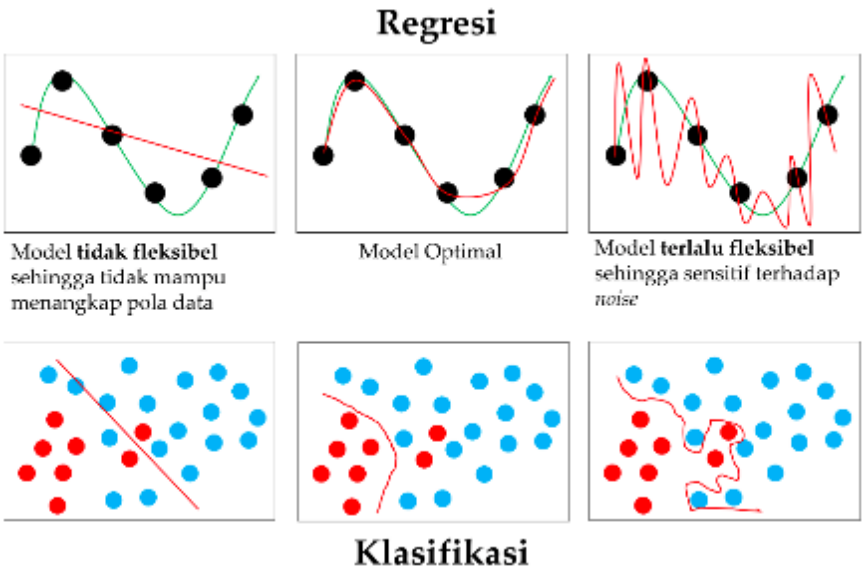
2.1.2.5. *Loss function*

Loss adalah ukuran seberapa dekat atau berbeda model yang dihasilkan dengan konsep asli, sementara *error* adalah salah satu fungsi untuk mengukur loss (Putra, 2019). Untuk mengukur nilai *loss* dapat diekspresikan dengan *loss function*. Secara umum ada dua macam *loss*, yaitu *generalization loss*/*error* dan *training loss*/*error*. *Generalization loss/error* adalah ukuran sejauh mana algoritma mampu memprediksi *unobserved* data dengan tepat, karena kita hanya membangun model dengan data yang terbatas, tentunya bisa saja terdapat ketidakcocokan dengan data yang asli. Sedangkan *training loss/error* seperti namanya, ukuran *loss* saat *training*.

Secara lebih detail, berkaitan dengan meminimalkan *loss*; tugas *machine learning* adalah untuk menemukan struktur tersembunyi (*discovering hidden structure*) (Putra, 2019). Hal ini sangat erat kaitannya dengan *knowledge discovery* dan data *mining*. Bila membuka forum di internet, kebanyakan akan membahas perihal *learning machine* yang memaksimalkan akurasi (meminimalkan *error*). Selain harus memaksimalkan akurasi (meminimalkan salah *assignment*), kita juga harus mampu membuat model yang cukup generik. Artinya tidak hanya memiliki kinerja tinggi pada training data, tapi juga mampu memiliki kinerja yang baik untuk *unseen* data. Hal ini dapat tercapai apabila model yang dihasilkan melakukan *inferensi* yang mirip dengan *inferensi* sebenarnya (konsep asli). Meminimalkan *loss* adalah hal yang lebih penting, dimana meminimalkan *error* dapat digunakan sebagai sebuah *proxy* untuk mengestimasi *loss* (pada banyak kasus).

2.1.2.6. *Overfitting* dan *underfitting*

Overfitting adalah keadaan ketika model memiliki kinerja baik hanya untuk *training* data/*seen examples* tetapi tidak memiliki kinerja baik untuk *unseen examples*. *Underftting* adalah keadaan ketika model memiliki kinerja buruk baik untuk *training data* dan *unseen examples* (Putra, 2019).



Gambar 2. 4 *overfitting* dan *underfitting* (Putra, 2019)

Dari gambar di atas, dataset asli diambil (*sampled*) dari fungsi polinomial orde-3. Model *underfitting* hanya mampu mengestimasi dalam orde-1 (kemampuan terlalu rendah), sedangkan model *overfitting* mampu mengestimasi sampai orde-9 (kemampuan terlalu tinggi).

2.1.2.7. Bias

Bias adalah kekeliruan *benchmark* yang mendorong pengembangan dan penerapan model yang berkinerja baik hanya pada subset data yang diwakili oleh data *benchmark* (Suresh & Guttag, 2021). γ adalah simbol yang melambangkan *bias* atau *noise* (Putra, 2019).

2.1.2.8. Variance

Variance (varian) adalah fungsi untuk mengetahui seberapa variasi nilai f(x) di sekitar nilai rata-ratanya. Bila nilai *variance* tinggi, secara umum banyak variabel yang nilainya jauh dari nilai rata-ratamya (Putra, 2019). Untuk fungsi dengan lebih dari satu *random variable*, maka yang dihitung adalah covariance. *Covariance* adalah *variance* untuk kombinasi variabel. Akar dari *variance* disebut *standard deviation* (sering disingkat "SD"). SD melambangkan seberapa jauh jarak dari masing-masing data point xi dengan rata-rata μ. Salah satu contoh cara seleksi fitur adalah menghapus atribut yang memiliki variance bernilai 0. Berdasarkan *information theory* atau *entropy*, fitur ini tidak memiliki nilai informasi yang tinggi. Dengan kata lain, atribut yang tidak dapat membedakan satu kelas dan lain bersifat tidak informatif.

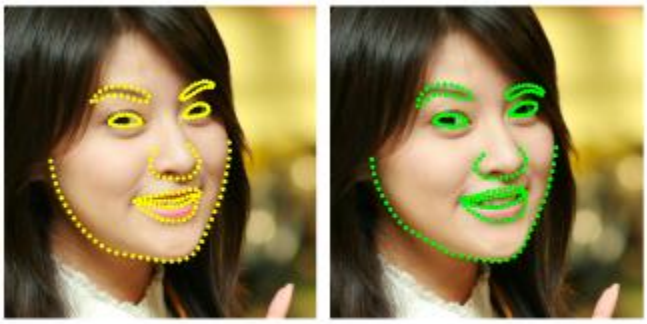
2.1.2.9. *Parameter tunning*

Pada *Ensemble of Regression Tree* terdapat parameter yang dapat di-optimalisasi, seperti *hyperparameters* untuk *gradient boost*. *Hyperparameter* adalah pengaturan yang jumlahnya banyak sekali untuk menghasilkan sebuah

model. Sayangnya, tidak ada jawaban yang mudah untuk tahu pengaturan *hyperparameter* mana yang harus digunakan. Menentukan kumpulan pengaturan dan kombinasi *hyperparameter* yang benar (optimal) seringkali merupakan tugas yang sulit dan memakan waktu bagi AI *programmer* dengan *trial and error* (Mukhopadhyay, 2018).

2.1.2.10.*Ground truth*

Saat ini, deteksi landmark wajah berbasis *deep learning* telah mencapai kesuksesan besar. Secara khusus, ambiguitas semantik berarti bahwa beberapa *landmark* (misalnya yang merata di sepanjang kontur wajah) tidak memiliki definisi yang jelas dan akurat, menyebabkan anotasi yang tidak konsisten oleh anotator. Dengan demikian, anotasi yang tidak konsisten ini, yang biasanya disediakan oleh *database* publik, biasanya berfungsi sebagai *ground truth* untuk mengawasi *training* jaringan, yang mengarah pada penurunan akurasi (Liu et al., 2019).



(e) $T = 10$ (f) Ground truth

Gambar 2. 5 Hasil regresi ERT dan *ground truth*

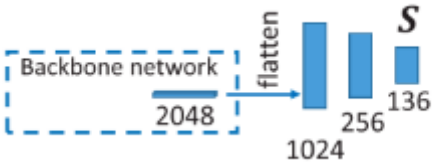
Gambar tersebut menunjukkan perbandingan secara visual antara hasil regressi dengan metode *ensemble of regression tree* dan *ground truth*-nya, yang mana terlihat kemiripan untuk setiap titik wajah.

2.1.2.11.Regression Approach

Regression Approach dapat dibagi lagi menjadi *Direct regression models* dan *Cascaded regression models*.

Direct regression models.

Mendeteksi *landmark* wajah dengan model regresi langsung dipelajari selama bertahun-tahun. Model ini mendeteksi *landmark* koordinat S direpresentasi-kan oleh vektor dari gambar wajah I. Dimensi dari vektor adalah dua kali dari jumlah landmark. Gambar berikut menunjukkan contoh dari *direct regression model* untuk mendeteksi 68 titik landmark wajah. *Backbone network* bisa menggunakan *network* arsitektur apa saja untuk melakukan ekstraksi fitur dari *input* gambar wajah. Dengan asumsi bahwa *layer* terakhir dari *backbone network* memiliki 2048 chanel (Hsu et al., 2020).



Gambar 2. 6 Contoh model arsitektur dari *direct regression* model untuk mendeksi 68 titik landmark wajah. Persegi panjang menandakan *fully-connected layer* dan jumlah dari *nodes* pada *layer* yang ada di bawah nya
(Hsu et al., 2020)

Cascaded regression models.

Tidak seperti *direct regression models* yang secara langsung mendeteksi titik koordinat landmark, *cascaded regression models* memperbaharui *landmark* secara berkala atau menggunakan *landmark* yang sudah dideteksi sebelumnya S_0 untuk mendeteksi *landmark*. *Sub-network* digunakan untuk menghasilkan vektor terbaru ΔS_i untuk memperbaharui posisi *landmark* pada setiap *stage* ke-i. Setelah *update* ke-n, model menghasilkan koordinat *landmark final* S_n. Untuk melakukan *training regression*, jarak L2 didapatkan untuk mengevaluasi *point-wise* perbedaan anatara hasil deteksi dan wajah sebenarnya.

Membandingkan *direct model* dan *cascaded model*, *cascaded regression model* secara umum lebih efektif dari *direct regression model*, karena *cascaded regression model* mengikuti strategi *coarse-to-fine* (kasar hingga halus). Bagai-manapun juga, tidak ada definisi standar berapa banyak *stages* yang harus dilakukan pada *cascaded model*_untuk mencapai deteksi paling akurat. Dan juga, tidak ada standar untuk menghasilkan bentuk wajah. Oleh karena itu, banyak penelitian memperoleh bentuk yang telah ditentukan dengan rata-rata bentuk wajah dari set pelatihan atau memprediksi bentuk dengan model tambahan.

Secara umum, *regression approach* mendapatkan keuntungan dari hubung-an kuat antar *landmark* karena informasi struktural wajah secara implisit tertanam dan dipelajari di *fully connected layer*. Karena itu, *landmark* yang terdeteksi masih dapat menggambarkan bentuk seperti wajah meskipun beberapa bagian penting dari wajah tertutup. Di sisi lain, karena keterkaitannya kuat, pendekatan ini mengalami posisi tengara (*landmark*) yang sedikit tidak akurat. Yaitu, pendekatan yang cenderung mempertahankan bentuk *landmark* sebagai wajah daripada mendeteksi posisi sebenarnya dari *landmark*. Setelah pendeteksian gagal, model dapat secara acak menempatkan *landmark* dengan bentuk seperti wajah (Hsu et al., 2020).

2.1.3. Virtual *Makeup*

Aplikasi virtual *makeup* memungkinkan pengguna untuk mencoba *makeup* dari jarak jauh, tanpa membuang produk kosmetik atau membuang waktu untuk membersihkan *makeup*-nya (*makeup* itu sendiri, produk *cleansing* dan *applicator*) nanti. Beberapa aplikasi virtual *makeup* hanya menampilkan hasil akhir *makeup* hanya pada wajah pengguna, sementara aplikasi lain memungkinkan interaksi pada saat melakukan makeup pada citra (Borges & Morimoto, 2019).

Contoh komersial dari aplikasi virtual *makeup* adalah ModiFace dan YouCam *makeup*. Sistem serupa telah menjadi incaran investasi dari banyak

produsen makeup seperti Natura, Avon, Clinique, Shiseido, dan lainnya. Dengan menggunakan *Augmented Reality* (AR) menawarkan beberapa keuntungan pada proses eksperimen, memungkinkan pengguna untuk memvisualisasikan hasil dengan cepat dan bersih, bahkan dari jarak jauh, tanpa pemborosan (Borges & Morimoto, 2019).

2.1.3.1. Augmented Reality

Augmented Reality (AR) adalah tampilan *real-time* secara langsung atau tidak langsung dari lingkungan fisik di dunia nyata yang telah ditingkatkan / di-tambah dengan menambahkan informasi yang dihasilkan komputer virtual ke dalamnya (Carmigniani & Furht, 2011).

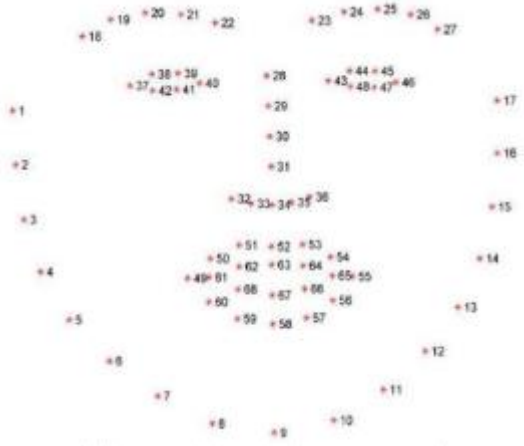
2.1.4. *Dataset*

Dataset pada *machine learning* merupakan kumpulan gambar dan anotasi titik wajah. Anotasi tersebut biasanya memiliki format extension .XML.

2.1.4.1. *Facial Landmark Point* dan Penerapan Antropometri

Landmark wajah didefinisikan sebagai deteksi dan lokalisasi titik-titik ter-tentu pada wajah yang merupakan fitur menonjol yang dapat memainkan peran diskriminatif atau dapat berfungsi sebagai titik jangkar pada grafik wajah.

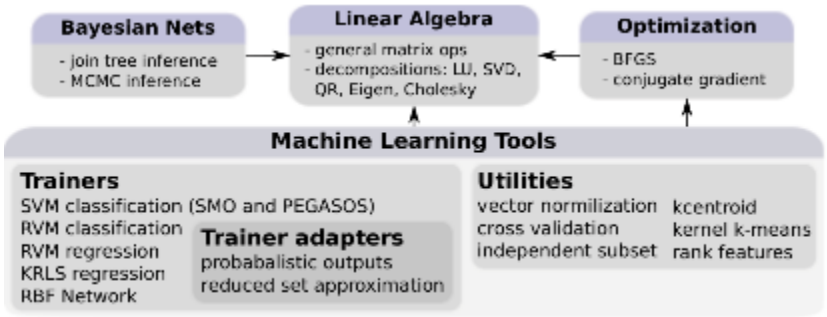
Dalam dunia kedokteran terdapat istilah antropometri yang merupakan ilmu yang mempelajari pengukuran dimensi tubuh manusia (ukuran, berat, volume, dan lain-lain) dan karakteristik khusus dari tubuh seperti ruang gerak. Pada pengukuran antropometri menunjukkan ada perbedaan bermakna antara laki-laki dan perempuan pada 21 titik pengukuran pada wajah yaitu pada ukuran lebar dasar kepala (t-t) , lebar mandibula (go-go), dalamnya wajah atas (n-t), dalamnya maksila (sn-t), tinggi wajah morfologi (n-gn), tinggi wajah bawah (sn-gn), tinggi mandibula anterior (sto-gn), tinggi wajah atas (n-sto) dan tinggi wajah fisiognomi (tr-gn), lebar nasal root (mf-mf), lebar hidung (al-al) , lebar dasar nostril (sbsal- sn), panjang cuping hidung kanan dan kiri (ac prn1 dan ac–prn2) serta panjang permukaan cuping hidung (ac-prn), tinggi bibir atas (sn-sto), tinggi vermilion bawah (sto-li), tinggi bibir bawah (sto-sl) dan tinggi bibir bagian lateral (sbsal-ls) dan tinggi vermilion atas (ls-sto), panjang telinga (sa-sba) dan jarak insersi telinga (obs-obi). Pada analisa jaringan (Elizabeth). (Rosiani et al., 2020)



Gambar 2. 7 Titik Wajah dari *Dataset* iBUG 300-w
(https://ibug.doc.ic.ac.uk/resources/300-W)

2.1.5. DLib

Dlib-ml adalah *library* perangkat lunak *open source* lintas platform yang ditulis dalam bahasa pemrograman C++. Desainnya sangat dipengaruhi oleh ide-ide dari desain dengan kontrak dan rekayasa perangkat lunak berbasis komponen. Ini berarti pertama dan terutama kumpulan komponen perangkat lunak independen, masing-masing disertai dengan dokumentasi ekstensif dan mode debugging menyeluruh. Selain itu, *library* dimaksudkan untuk berguna baik dalam penelitian dan proyek komersial dunia nyata dan telah dirancang dengan cermat untuk memudahkan integrasi ke dalam aplikasi C++ pengguna (King, 2009).



Gambar 2. 8 E*lement* dari Dlib

Library terdiri dari empat komponen berbeda yang ditunjukkan pada di atas. Komponen aljabar linier menyediakan satu set fungsionalitas inti sementara tiga lainnya mengimplementasikan berbagai *tool* yang berguna.

2.1.5.1. *Machine Learning Tools*

Tujuan desain utama dari bagian library DLib ini adalah untuk menyediakan arsitektur yang sangat modular dan sederhana untuk menangani algoritma *kernel*. Secara khusus, setiap algoritma diparameterisasi untuk memungkinkan pengguna memasok salah satu dari *kernel* dlib-ml yang telah ditentukan sebelumnya, atau *kernel* yang ditentukan pengguna baru. Selain itu, implementasi algoritma benar-benar terpisah dari data tempat mereka beroperasi. Ini membuat implementasi dlib-

ml cukup umum untuk beroperasi pada semua jenis data, baik itu *vectors*, gambar, atau bentuk lain dari data terstruktur. Yang diperlukan hanyalah *kernel* yang sesuai (King, 2009).

Ini adalah fitur unik untuk dlib-ml. Banyak perpustakaan mengizinkan *kernel* yang telah dikomputasi sebelumnya dan beberapa bahkan mengizinkan *kernel* yang ditentukan pengguna tetapi memiliki antarmuka yang membatasi mereka untuk beroperasi pada vektor kolom. Namun, tidak ada yang mengizinkan fleksibilitas untuk beroperasi secara langsung pada objek arbitrer, membuatnya lebih mudah untuk menerapkan *kernel* khusus dalam kasus di mana kernel beroperasi pada objek selain vektor dengan panjang tetap.

2.1.6. *OpenCV*

OpenCV (*Open Source Computer Vision*) adalah library dari fungsi pemrograman untuk *computer vision* (penglihatan computer) realtime. OpenCV menggunakan lisensi BSD dan bersifat gratis baik untuk penggunaan akademis maupun komersial (Lazaro et al., 2017). Library OpenCV dapat dijalankan pada beragam bahasa pemrograman C, C++, Python dan Java. OpenCV juga bersifat *cross platform*, yang dapat dijalankan pada sistem operasi Windows, Linux, Android, iOS dan Mac OS. OpenCV memiliki lebih dari 2500 algoritma yang telah dioptimalkan.

2.1.7. *Face Recognition*

Face recognition atau pengenalan wajah adalah salah satu aplikasi analisis citra yang paling relevan. Pengenalan wajah merupakan tantangan nyata untuk membangun sistem otomatis yang setara dengan kemampuan manusia untuk mengenali wajah. Meskipun manusia cukup baik dalam mengidentifikasi wajah yang dikenal, kita tidak terlalu terampil ketika kita harus berurusan dengan sejumlah besar wajah yang tidak dikenal. Komputer, dengan memori dan kecepatan komputasi yang hampir tak terbatas, seharusnya dapat mengatasi keterbatasan manusia. Pengenalan wajah adalah subjek yang relevan dalam pengenalan pola (*pattern matching*), jaringan saraf (*neural network*), grafik computer *(computer graphics)*, pemrosesan gambar (*image processing*) dan psikologi (Graña, 2010).

Tabel 2. 2 Contoh aplikasi *face recognition*

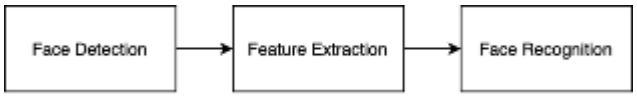
Area	Aplikasi
Information security	*Access security (OS, data bases)* *Data privacy (e.g. medical records)* *User authentication (trading, on line banking)*
Manajemen Akses	*Secure access authentication (restricted facilities)* *Permission based systems Access log or audit trails*
Biometrics	*Person identification (national IDs, Passports, voter registrations, driver licenses)* *Automated identity verification (border controls)*
Law Enforcement	*Video surveillance* *Suspect identification* *Suspect tracking (investigation)*

	Simulated aging *Forensic Reconstruction of faces from remains*
Personal security	*Home video surveillance systems* *Expression interpretation (driver monitoring system)*
Entertainment - Leisure	*Home video game systems* *Photo camera applications*

Face recognition adalah istilah yang mencakup beberapa *sub-problems*, diantaranya adalah:

2.1.7.1. Sistem Umum *Face Recognition*

Input dari sistem pengenalan wajah selalu berupa gambar atau video *stream*. *Output*-nya berupa identifikasi atau verifikasi dari subyek yang muncul pada gambar atau video. Beberapa pendekatan mendefinisikan sistem *face recognition* dengan proses tiga langkah (*face detection* dan *face extraction* dapat berjalan secara bersamaan) (Graña, 2010)



Gambar 2. 9 Sistem umum *face recognition*

Deteksi wajah didefinisikan sebagai proses mengekstrak wajah dari *scenes* (tempat / layar). Jadi, sistem secara positif mengidentifikasi wilayah gambar tertentu sebagai wajah. Prosedur ini memiliki banyak aplikasi seperti pelacakan wajah, estimasi pose. Langkah selanjutnya -ekstraksi fitur- melibatkan perolehan fitur wajah yang relevan dari data. Fitur-fitur ini dapat berupa daerah wajah tertentu, variasi, sudut atau ukuran, yang dapat relevan dengan manusia (misalnya jarak mata) atau tidak. Fase ini memiliki aplikasi lain seperti pelacakan fitur wajah atau pengenalan emosi. Akhirnya, sistem akan mengenali wajah. Dalam tugas identifikasi, sistem akan melaporkan identitas dari *database*. Fase ini melibatkan metode perbandingan, algoritma klasifikasi dan ukuran akurasi.

Fase-fase ini dapat digabungkan, atau yang baru dapat ditambahkan. Karena itu, kita bisa menemukan banyak pendekatan teknik yang berbeda untuk masalah pengenalan wajah. Deteksi dan pengenalan wajah dapat dilakukan bersama-sama, atau dilanjutkan ke analisis ekspresi sebelum wajah dinormalisasi. Beberapa algoritma lebih baik, beberapa kurang akurat, beberapa lebih fleksibel dan yang lainnya terlalu mahal secara komputasi. Terlepas dari keragaman ini, pengenalan wajah menghadapi beberapa masalah yang melekat pada definisi masalah, kondisi lingkungan, dan kendala perangkat keras.

2.1.7.2. Masalah Umum *Face Recognition*

Secara lebih detail beberapa masalah umum pada deteksi wajah adalah:

1. *Illumination* / pencahayaan

   Banyak algoritma mengandalkan informasi warna untuk mengenali wajah. Fitur diekstraksi dari gambar berwarna, meskipun beberapa diantara-nya mungkin berwarna abu-abu. Warna yang didapat dari permukaan objek tidak hanya bergantung pada sifat dasar (bentuk) objek, tetapi juga pada

cahaya di atasnya (Graña, 2010). Intensitas warna dalam piksel dapat sangat bervariasi tergantung pada kondisi pencahayaan.

Nilai piksel bukan hanya bergantung berdasarkan perubahan cahaya, hubungan antar piksel juga dapat bervariasi. Kebanyakan metode ekstraksi fitur bergantung pada keberagaman warna/intensitas antar piksel untuk mendapatkan data yang relevan, metode ini menunjukkan ketergantungan penting pada perubahan pencahayaan. Tetapi, tidak hanya sumber cahaya yang dapat bervariasi, melainkan juga intensitas cahaya dapat meningkat atau menurun. Seluruh daerah wajah menjadi kabur, tidak fokus atau berbayang, sehingga ekstraksi fitur menjadi tidak. Ringkasnya, iluminasi adalah salah satu tantangan besar sistem pengenalan wajah otomatis.

2. *Pose variation*

   Variasi pose dan iluminasi adalah dua masalah utama yang dihadapi oleh peneliti pengenalan wajah. Sebagian besar metode pengenalan wajah didasarkan pada gambar wajah bagian depan. Kumpulan gambar ini dapat memberikan dasar penelitian yang kuat untuk pengenalan wajah bagian depan. Banyak sistem pengenalan wajah, seperti video pengawas, sistem keamanan video atau sistem hiburan rumah *augmented reality* mengambil *input* data dari lingkungan yang tidak terkendali. Kendala lingkungan yang tidak terkendali melibatkan beberapa kendala untuk pengenalan wajah: Masalah pencahayaan di atas adalah salah satunya. Dan masalah selanjutnya adalah variasi pose.

3. Masalah lain

   Terdapat masalah lain yang harus dihadapi oleh penelitian pengenalan wajah otomatis, seperti:

   a. *Occlusion* / penghalang

      Dalam konteks pengenalan wajah, masalah ini dapat berupa beberapa bagian wajah tidak dapat diperoleh. Misalnya, foto wajah yang diambil dari kamera pengintai yang tertutup sebagian. Proses pengenalan dapat sangat bergantung pada ketersediaan *input* wajah secara utuh. Oleh karena itu, tidak adanya beberapa bagian wajah dapat menyebabkan klasifikasi yang buruk. Masalah ini mendukung pendekatan sedikit demi sedikit (*piecemeal approach*) untuk ekstraksi fitur, yang tidak bergantung pada keseluruhan wajah.

      Ada juga benda-benda yang dapat menutupi fitur wajah seperti kacamata, topi, janggut, potongan rambut tertentu, dll.

   b. *Optical technology*

      Sistem pengenalan wajah juga bergantung pada format *input* gambar. Kamera yang berbeda akan memiliki fitur yang berbeda, kelemahan dan masalah yang berbeda. Biasanya, sebagian besar proses pengenalan melibatkan langkah *preprocessing* yang berhubungan dengan masalah ini.

   c. *Expression*

      Ekspresi wajah juga merupakan masalah, namun tidak sebesar pencahayaan atau pose wajah. Beberapa algoritma tidak menangani masalah ini secara eksplisit, tetapi menunjukkan kinerja

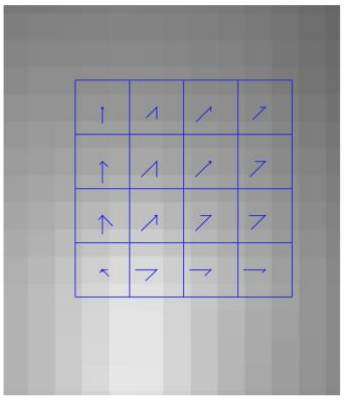
yang baik ketika ada ekspresi wajah yang berbeda. Di sisi lain, penambahan variabilitas ekspresi, pose dan masalah pencahayaan dapat menjadi hambatan nyata untuk pengenalan wajah yang akurat.

d. *Algorithm evaluation*

Tidak mudah untuk mengevaluasi efektivitas algoritma pengenalan wajah. Terutama faktor inti seperti: *hit ratio, error rate, computational speed* dan *memory usage*

2.1.8. HOG *Face Recognition*

Histogram of Oriented Gradient (HOG) telah terbukti menjadi deskriptor yang efektif untuk pengenalan objek secara umum dan pengenalan wajah pada khususnya (Déniz et al., 2011). Algoritma HOG menghitung kemunculan orientasi tepi terdekat dari suatu gambar, kemudian fitur HOG diekstraksi dari semua lokasi grid padat (*dense grid*) pada wilayah gambar dan menggunakan *Linear Support Vector Machine* (SVM) untuk mengklasifikasikan fitur gabungan (Jamshed et al., 2015).



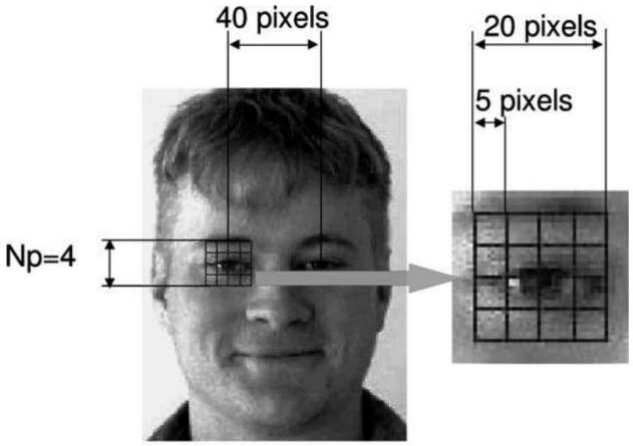
Gambar 2. 10 Contoh HOG *descriptors*, *patch size*=8x 8. cell menunjukkan orientasi dari *gradient* (Déniz et al., 2011)

Setiap piksel dalam cell berkontribusi pada saat dilakukan voting bobot (w / *weight*) untuk membangun sebuah histogram yang berorientasi pada nilai-nilai gradien yang dihitung. Sehingga dapat diambil asumsi yang menunjukkan bahwa suatu objek dapat direpresentasikan dengan baik berdasarkan bentuknya. Untuk memperoleh informasi pembeda maka gambar akan dibagi menjadi *cell* dan setiap *cell* akan dihitung sebagai *histogram of oriented gradients*. Wajah sebelumnya dinormalisasi dalam skala dan orientasi (kemiringan wajah), sehingga langkah-langkah untuk deteksi dan orientasi skala-ruang (*scale-space*) ekstrim tidak diperlukan. Satu set 25 landmark wajah dilokalisasi menggunakan kerangka *Elastic Bunch Graph Matching* dengan fitur HOG (Déniz et al., 2011). Fitur HOG yang diekstraksi dari sekitar masing-masing 25 landmark wajah digunakan untuk klasifikasi, menggunakan *cell* terdekat dan *Euclidean Distance*.

Langkah-langkah untuk deteksi wajah menggunakan HOG terbagi menjadi tiga hal yaitu: Pertama, untuk mengkompensasi kesalahan dalam deteksi fitur wajah karena oklusi/penghalang, perubahan pose dan iluminasi/pencahayaan dengan mengekstrak deskriptor HOG dari grid biasa. Kedua, perpaduan deskriptor HOG

pada skala yang berbeda memungkinkan untuk menangkap struktur penting untuk pengenalan wajah. Ketiga, mengidentifikasi perlunya melakukan pengurangan dimensi untuk menghilangkan kebisingan/*noise* dan membuat proses klasifikasi tidak *overfitting*. Ini sangat penting jika fitur HOG diekstraksi dari *cell* yang tumpang tindih (Déniz et al., 2011).

Active Appearance Model dan *Elastic Bunch Graph Matching* adalah salah satu metode berbasis fitur yang mencoba mengenali wajah menggunakan komponen wajahnya: mata, hidung, mulut, dll (Albiol et al., 2008). Deskriptor HOG adalah kumpulan histogram yang terdiri dari orientasi piksel yang diberikan oleh gradiennya. Jumlah kemungkinan orientasi (histogram *bins*) disebut sebagai No. Setiap histogram dalam bundel menggambarkan area spesifik di sekitar keypoint. Daerah-daerah ini sesuai dengan sel-sel Np? Grid kuadrat Np berpusat pada keypoint seperti pada gambar di bawah:



Gambar 2. 11 Deksriptor HOG untuk mata kanan

Namun, penulis belum menemukan contoh *output* dari HOG untuk hasil deteksi wajah, berikut adalah contoh *output* HOG untuk deteksi manusia:

(a) (b) (c)

Gambar 2. 12 HOG deteksi manusia

Dari Gambar (a) di atas menunjukkan sebuah contoh *input* gambar manusia, (b) rata-rata gradien pada *input* gambar, (c) hasil perhitungan R-HOG *descriptor*.

2.1.9. *Image Processing*

Image Processing merupakan metode untuk melakukan beberapa operasi pada gambar untuk mendapatkan gambar yang disempurnakan atau untuk meng-ekstrak informasi yang dibutuhkan. Proses sinyal di mana inputnya adalah gambar dan *output*-nya adalah gambar atau karakteristik / fitur yang terkait dengan gambar tersebut. Pemrosesan gambar mencakup tiga langkah; yang pertama adalah mengimpor gambar melalui alat akuisisi gambar, yang kedua adalah menganalisis dan memanipulasi gambar, dan yang terakhir adalah output di mana hasilnya dapat berupa gambar yang diubah atau laporan yang didasarkan pada analisis gambar (Rosalina & Wijaya, 2020).

2.1.10. *Pattern Recognition*

Proses *pattern recognition* (pengenalan pola) adalah untuk membuat representasi baru atau mengubah satu representasi menjadi representasi lain, dimulai dari representasi objek sebagai fungsi pola. Kita juga dapat mengatakan bahwa tujuan pengenalan pola oleh komputer adalah untuk menghasilkan dan mengubah representasi informasi yang berguna menggunakan kecepatan tinggi dan kemampuan pemrosesan data yang masif dari komputer (Anzai, 2012).

Pattern recognition berkaitan dengan deskripsi dan klasifikasi pengukuran yang diambil dari proses fisik. Untuk memberikan gambaran *pattern recognition* yang efektif dan efisien, pra-pemrosesan sering diperlukan untuk menghilangkan *noise* dan redundansi dalam pengukuran. Kemudian serangkaian pengukuran karakteristik, yang dapat berupa numerik dan / atau non numerik, dan hubungan antara pengukuran ini, diekstraksi untuk representasi pola. Klasifikasi dan atau deskripsi pola sehubungan dengan tujuan tertentu dilakukan atas dasar representasi.

Banyak teknik matematika berbeda yang digunakan untuk memecahkan masalah pengenalan pola mungkin dikelompokkan menjadi dua pendekatan umum. Mereka adalah pendekatan teori keputusan (atau diskriminan) dan pendekatan sintaksis (atau struktural). Dalam pendekatan teori keputusan, seperangkat peng-ukuran karakteristik, yang disebut fitur, diekstraksi dari pola. Setiap pola diwakili oleh vektor fitur, dan pengenalan setiap pola biasanya dibuat dengan mempartisi ruang fitur. Di sisi lain, dalam pendekatan sintaksis, setiap pola dinyatakan sebagai komposisi komponennya, yang disebut subpola atau pola primitif. Pendekatan ini menarik analogi antara struktur pola dan sintaksis suatu bahasa. Pengenalan setiap pola biasanya dibuat dengan menguraikan struktur pola menurut seperangkat aturan sintaks yang diberikan. Dalam beberapa aplikasi, kedua pendekatan ini mungkin digunakan. Misalnya, dalam masalah yang berhubungan dengan pola kompleks, pendekatan teori keputusan biasanya efektif dalam pengenalan pola primitif, dan pendekatan sintaksis kemudian digunakan untuk pengenalan subpola dan pola itu sendiri (Fu & Rosenfeld, 1976).

Untuk menentukan seperangkat pengukuran karakteristik yang baik dan hubungannya untuk representasi pola sehingga kinerja pengenalan yang baik dapat diharapkan, diperlukan analisis yang cermat dari pola yang diteliti. Pengetahuan tentang karakteristik statistik dan struktur pola harus dimanfaatkan sepenuhnya.

Dari sudut pandang ini, studi pengenalan pola mencakup analisis karakteristik pola dan desain sistem pengenalan.

2.2. Kajian Literatur

Studi mengenai deteksi wajah, *landmark* wajah dan *landmark* bibir, telah penulis sajikan dalam bentuk tabel sebagai berikut:

Tabel 2. 3 Studi Literatur

Judul	**Tujuan Penelitian / Permasalahan**	**Dataset, algoritma, arsitektur**	**kesimpulan**
Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs (Kartynnik & Ablavatski, 2019)	Menyajikan model neural network untuk memperkirakan representasi mesh 3D dari wajah manusia dari input kamera tunggal	Model: 30K in-the-wild mobile camera photos Algoritma: *Neural network*	IOD MAD 3.96% pada full model. 5.29% pada model terkecil. Kecepatan deteksi wajah maksimal 15FPS
BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs (Bazarevsky et al., 2019)	Lightweight face detector (Pendeteksi wajah yang ringan)	Data: - Algoritma: SSD	Kecepatan deteksi wajah maksimal 24FPS, dengan deteksi 4 titik wajah
Biological landmark vs quasi-landmarks for 3D face recognition and gender classification (Abbas et al., 2019)	Perbandingan ide *Biological landmark* vs *quasi-landmarks* untuk klasifikasi *gender*	Data: - Algoritma: *Biological landmark, quasi-landmarks*	Akurasi klasifikasi gender 92% untuk quasi-*landmarks* dan 90% untuk *biological landmarks*
A review of image-based automatic facial landmark identification techniques (Johnston & Chazal, 2018)	*provide a review of the current facial landmarking literature*	Data: BioID, MUCT, HELEN, 300W, Menpo	*Dlib, HOG-based face detector outperformed all of the OpenCV variants with greater accuracy and fewer false positives.*
Semantic Alignment: Finding Semantically Consistent Ground-truth for Facial	Menemukan *ground-truth* yang konsisten semantik untuk deteksi *landmark* wajah	Data: IBug 300W, AFLW Algoritma: -	*global heatmap correction unit* (GHCU) untuk mengoreksi *outlier* dengan

Landmark Detection (Liu et al., 2019)			mempertimbangkan bentuk wajah *global* sebagai *constraint.* GHCU secara efektif meningkatkan akurasi deteksi *landmark* dan mencapai kebaruan kinerja
Virtual Makeup Application Using Image Processing Methods (Yolcu Oztel & Kazan, 2015)	studi bertujuan untuk mencoba produk secara virtual untuk mencoba *lipstick*	Data: Algoritma: Matlab Functions	Sistem bekerja realistis, tetapi masih terjadi delay
One millisecond face alignment with an ensemble of regression trees (Kazemi & Sullivan, 2014)	Makalah ini membahas masalah *face alignment* untuk satu gambar	Data: HELEN Algoritma: Ensemble of regression trees (ERT)	algoritma *ensemble of regression trees* dapat digunakan untuk meregresi lokasi landmark wajah dari nilai intensitas subset piksel acak diekstraksi dari gambar input
Real-Time Eye Blink Detection using Facial Landmarks (Cech & Soukupova, 2016)	algoritma *real-time* untuk mendeteksi kedipan mata pada video dari kamera standar	Data: in the wild (300-VW) Algoritma: Viola-Jones, SVM	Deteksi *landmark* wajah berbasis regresi, sudah cukup tepat untuk memperkirakan tingkat keterbukaan mata secara *reliable*.
Penerapan *Facial Landmark Point* Untuk Klasifikasi Jenis Kelamin Berdasarkan Citra Wajah (Rosiani et al., 2020)	Klasifikasi jenis kelamin dari titik (*landmark point*) wajah	Data: iBUG 300W	Akurasi belum bagus (68%) karena fitur yang digunakan memiliki nilai yang dekat satu sama lain sehingga sulit untuk diklasifikasikan.
A virtual makeup augmented reality	sistem *augmented reality* yang	Data & Algoritma:	prototipe mampu mendeteksi

system (Borges & Morimoto, 2019)	memungkinkan pengguna untuk menerapkan virtual *makeup* langsung di wajah menggunakan aplikator fisik, mensimulasikan pengalaman cermin virtual.	Realsense SDK	sentuhan dengan sangat akurat (sekitar 2,2 mm), dan yang mencapai kinerja interaktif *real-time* (sekitar 15 fps)

Dari kajian literatur diatas, terdapat informasi bahwa FPS tercepat dengan 24FPS ada pada jurnal "*BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs*", akantetapi titik wajah yang digunakan hanya empat, sehingga tidak cukup untuk digunakan pada aplikasi virtual *makeup*. Pada jurnal "*A virtual makeup augmented reality system*" mampu meng-hasilkan kecepatan deteksi 15 fps dengan menggunakan *tools* Realsense SDK, kemduain pada jurnal "*Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs*" juga menghasilkan kecepatan deteksi yang sama tetapi dengan menggunakan menggunakan algoritma *neural network*.

Pada jurnal yang berjudul "*Virtual Makeup Application Using Image Processing Methods*" dapat mendeteksi dan *masking* bibir dengan menggunakan *tools Matlab Functions*, tetapi disebutkan bahwa masih terdapat kekurangan karena masih terjadi delay.

Untuk melakukan training deteksi wajah menggunakan dataset iBUG 300W pada jurnal-jurnal: "Penerapan *Facial Landmark Point* Untuk Klasifikasi Jenis Kelamin Berdasarkan Citra Wajah", "*Real-time face alignment: evaluation methods, training strategies and implementation optimization.",* "*Semantic Alignment: Finding Semantically Consistent Ground-truth for Facial Landmark Detection*", "*Real-Time Eye Blink Detection using Facial Landmarks*".

Kemudian pada jurnal "*One millisecond face alignment with an ensemble of regression trees*" telah dapat mendeteksi landmark wajah dengan subset piksel acak pada gambar. Akan tetapi tidak disebutkan menghasilkan kecepatan deteksi berapa FPS.

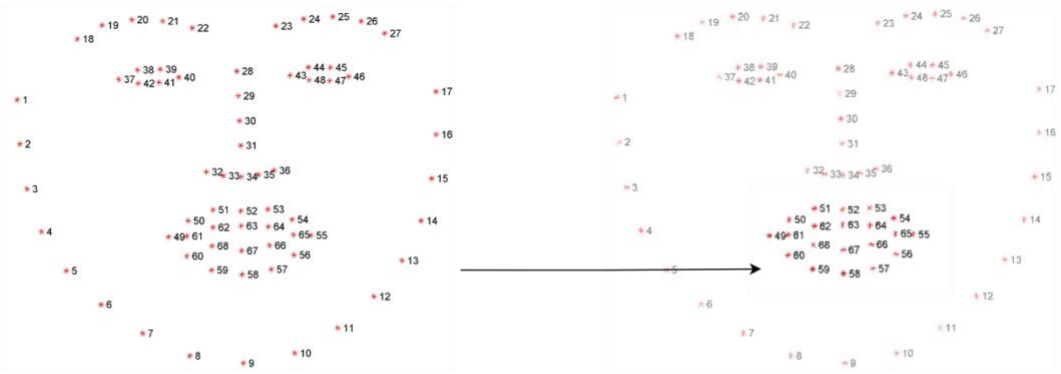
2.3. Tinjauan Objek Penelitian

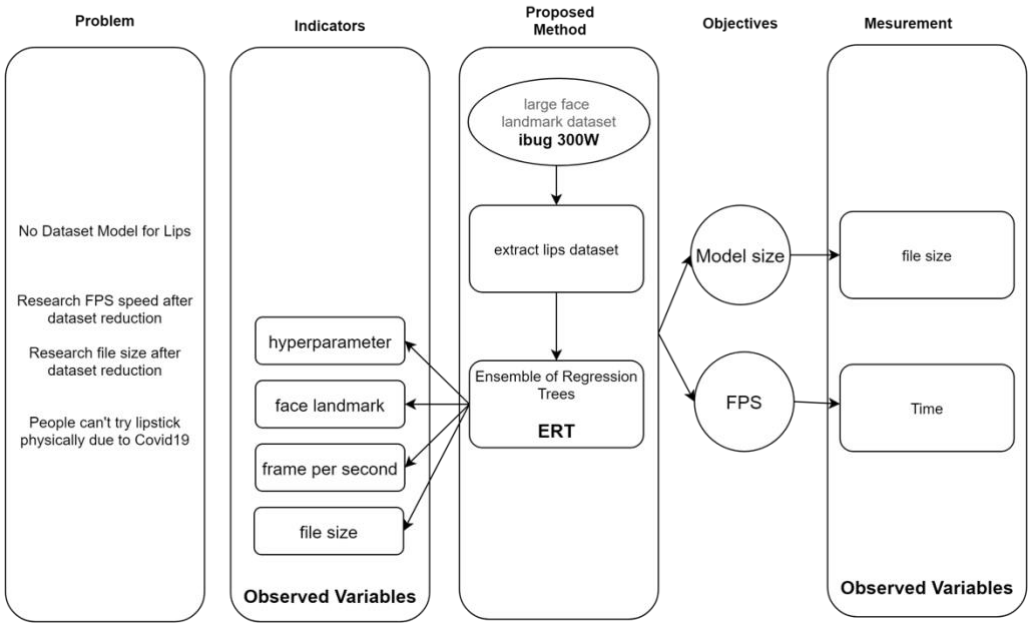
Objek utama penelitian adalah menyediakan *dataset* dan *prediction model* baru yaitu spesifik area bibir, karena sepengetahuan penulis belum ada *dataset* dan *prediction model* spesifik area bibir saja. Penelitian yang dilakukan oleh penulis adalah berdasarkan informasi yang didapat dari jurnal terdahulu untuk menguji hipotesis dan menyediakan *dataset* baru, yang dapat digunakan untuk penelitian selanjutnya. Dan *prediction model* baru, yang dapat digunakan untuk pengembang-an aplikasi serupa atau lebih baik lagi.

Aplikasi pendeteksian wajah dapat diaplikasikan pada berbagai bidang, seperti security, absensi, aplikasi *Augment Reality* atau aplikasi virtual *makeup*.

2.4. Kerangka Konsep Pemikiran

Berdasarkan identifikasi masalah, tujuan penelitian, kajian teori, studi dari penelitian sebelumnya, dan tinjauan obyek penelitian, maka dapat dibangun kerangka konsep penelitian tentang "Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*".





Gambar 2. 13 Kerangka Pemikiran

Berdasarkan kerangka konsep pemikiran tersebut dapat dijelaskan tahapan kerangka konsep pemikiran sebagai berikut:

Permasalah utama yang dihadapi adalah tidak adanya *dataset* atau *prediction model* khusus untuk spesifik area bibir saja, yang diharapkan akan dapat meringankan beban komputasi jika titik / point yang dideteksi cukup sepesifik (bukan wajah secara keseluruhan).

Selanjutnya untuk mengimplementasikan "Deteksi Bibir Menggunakan Metode *Ensemble of Regression Trees* untuk Prototipe Virtual *Makeup*" digunakan *dataset* yang sudah tersedia yakni iBug 300W, tetapi dengan kebaruan yakni mengekstraksi dari 68 titik wajah menjadi 20 titik bibir. Kemudian dengan *DLib*

library dan metode *Ensemble of Regression Trees* akan digunakan sebagai metode untuk pendeteksian area titik / *landmark* bibir.

2.5. Hipotesis

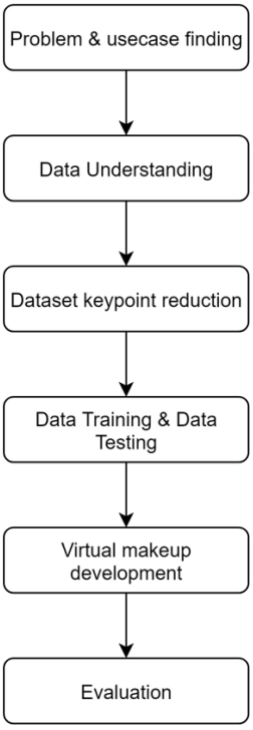
- Diduga dengan mengekstraksi jumlah deteksi titik wajah spesifik pada area bibir saja, alih-alih pada seluruh bagian wajah dapat mengurangi penggunaan *resource*.
- Diduga dari model yang dikembangkan dapat mensimulasikan penggunaan virtual *makeup* lipstik.
- Diduga nilai *cascade* = 10 adalah nilai terbaik untuk salah satu *hyperparameter* mengacu kepada penelitian terdahulu.

BAB III: METODOLOGI DAN RANCANGAN/DESAIN PENELITIAN

Metodologi penelitian dan rangangan/desain penelitian adalah berkaitan dengan langkah-langkah kerja dalam melakukan penelitian. Bab ini akan membahas tahapan-tahapan dalam melakukan penelitian. Penelitian yang dilakukan berkaitan dengan aplikasi pendeteksian bibir dan simulasi virtual *makeup*. Secara umum aplikasi ini terdiri dari beberapa tahap diantaranya pengambilan *input* data citra, training dan evaluasi, serta pengembangan aplikasi virtual *makeup*. Maka dari itu pada metodologi penelitian ini adalah tahapan sistematis yang akan dilakukan pada penelitian.

3.1. Tahapan Penelitian

Penelitian ini dilakukan untuk mendapatkan *dataset* baru yang spesifik pada area bibir. Kemudian, *dataset* area bibir / *lips landmarks* akan digunakan untuk mengembangkan aplikasi *prototype* virtual *makeup* (virtual *try-on*) untuk simulasi penggunaan lipstik.

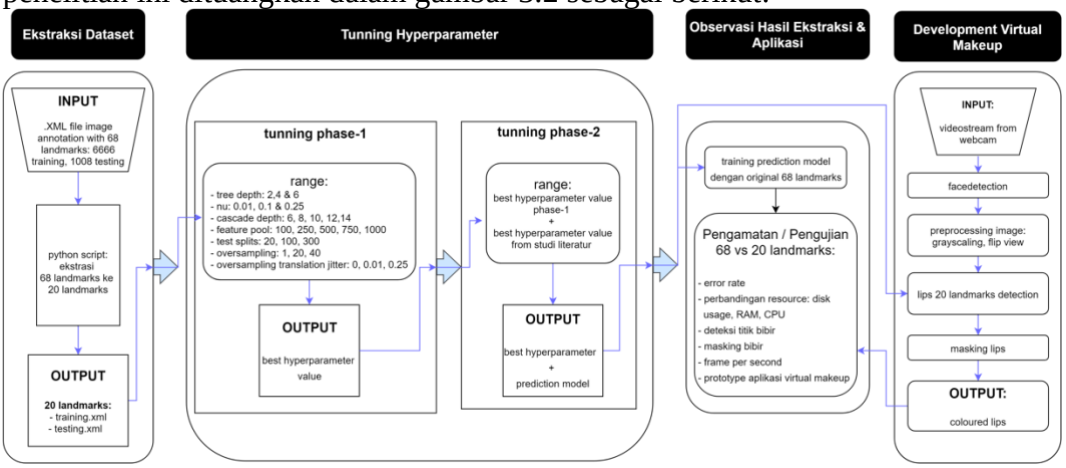


Gambar 3. 1 Tahapan Penelitian

3.2. Metode Penelitian

Dalam penelitian ini deteksi *face landmarks* menggunakan metode yang digunakan adalah *Machine Learning Ensemble of Regression Trees (ERT)* untuk melakukan deteksi wajah dan pengenalan bibir. *Tools* / *framework* menggunakan *Dlib* untuk proses *training* dan *testing*, kemudian untuk aplikasi *prototype* virtual

makeup library tambahan OpenCV. Adapun gambaran alur sistematika proses dari penelitian ini dituangkan dalam gambar 3.2 sebagai berikut:



Gambar 3. 2 Alur aplikasi

Berdasarkan alur aplikasi dan pengembangan aplikasi di atas dapat dijelaskan tahapan-tahapannya sebagai berikut:

- data *training* dan data *testing* berupa gambar dan anotasi lokasi *landmark* wajah didapatkan dari website resmi iBug-300W.
- anotasi lokasi *landmark* yang awalnya berjumlah 68 titik (*keypoints*) akan diekstraksi menjadi 20 titik bibir saja
- untuk proses *training* pendeteksian bibir menggunakan metode *Ensemble of Regression Tree (ERT)* akan menggunakan *DLib library* yang telah mengimplementasikan metode ERT pada *library*-nya
- untuk *testing* juga akan menggunakan DLib *library*
- kemudian untuk proses *masking* bibir dan *prototyping* aplikasi virtual *makeup* akan menggunakan *OpenCV library*

3.2.1. *Dataset*

Dataset dari https://ibug.doc.ic.ac.uk/resources/300-W digunakan sebagai data utama. Pada *dataset* tersebut terdapat beberapa folder seperti: AFW, Helen, iBug dan FLPW dengan total file size 1,89 Gb. (Sagonas et al., 2013a, 2013b, 2016), kemudian akan dilakukan ekstraksi (pengurangan) *keypoint landmark* yang awalnya mendeteksi seluruh wajah dengan total 68 titik, menjadi hanya *landmark* bibir dengan 20 titik.

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

Tabel 3. 1 Data anotasi gambar wajah

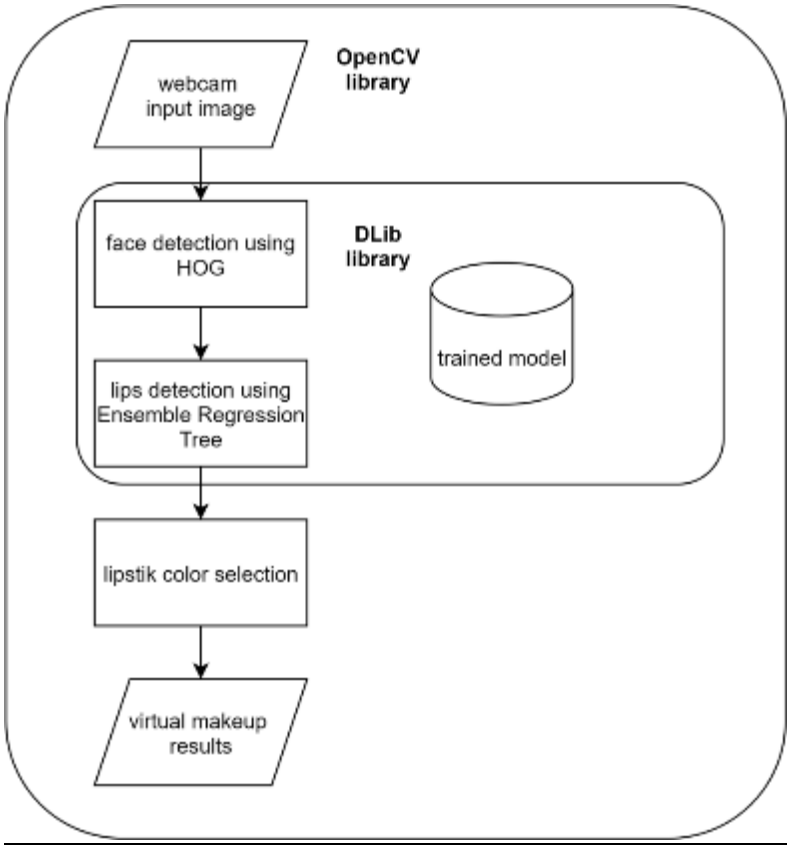
Jenis gambar	Jumlah data Training	Jumlah data testing
iBug 300W	6666	1008

3.3. Teknik Analisis, Rancangan dan Pengujian

Untuk mencapai hasil yang diinginkan penelitian ini menggunakan teknik penelitian terdahulu mengenai pendeteksian wajah dengan *tools DLib*.

Rancangan aplikasi yaitu melakukan ekstraksi terhadap titik / *keypoint landmark* sehingga diharapkan akan meningkatkan kecepatan pendeteksian wajah dalam *frame per second (FPS)*.

Pengujian akan dilakukan dengan dua cara: pertama akan dilakukan pengecekan terhadap akurasi *dataset*. Kedua akan dilakukan pengujian manual untuk melihat kecepatan pendeteksian wajah dalam FPS.



Gambar 3. 3 Tahapan pengujian penelitian

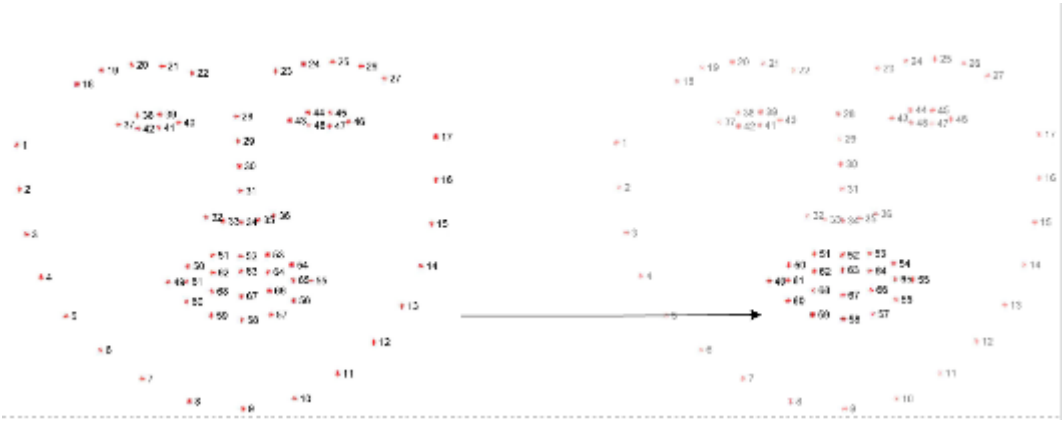
3.4.1. Ekstraksi *keypoint landmark*

Dataset dari iBUG-300W dapat memprediksi lokasi dari masing-masing 68 (x, y) pasangan koordinat *face landmark* yang dapat mendeteksi 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* adalah:

Tabel 3. 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 – 40	4
Mata kiri (*left eye*)	43 – 46	4
Mulut / bibir (*lips*)	49 – 68	20

Maka, dari data yang diperoleh dari website iBug, akan dilakukan modifikasi / ekstraksi jumlah titik menjadi 20 titik *lips landmark*. Yaitu dengan cara *parsing* file anotasi 68 titik wajah yang berformat XML, kemudian yaitu dari 6666 data *training* dan 1008 data *testing* dengan 68 titik wajah pada file yang berformat XML dilakukan *parsing* menggunakan *Python script*, sehingga pada file XML akan tersisa 20 titik bibir saja.



Gambar 3. 4 Ekstraksi keypoint dataset

Untuk memahami bagaimana kita bisa 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** 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>
...
```

Semua data *training* dalam *dataset* iBUG-300W dituliskan dalam XML. Setiap gambar memiliki penanda ***image***. Di dalam tag ***image*** memiliki atribut ***file*** yang menunjuk lokasi penyimpanan gambar. Selain itu, setiap ***image*** memiliki elemen ***box*** yang terkait dengannya. Elemen ***box*** mewakili koordinat kotak pembatas (*bounding box*) wajah pada gambar. Untuk memahami bagaimana elemen ***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:
1. *Name* : Indeks/nama *landmark* wajah.
2. x : Koordinat x dari *landmark*.
3. y : Koordinat y dari *landmark*.

Maka dengan *Python script* nantinya hanya akan menggunakan titik dari mulai part 47 sampai 67 yang merupakan titik bibir. Dan kurang lebih hasilnya seperti berikut (tidak ada titik wajah 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>
...
```

3.4.2. Pendeteksian bibir dengan DLib ERT

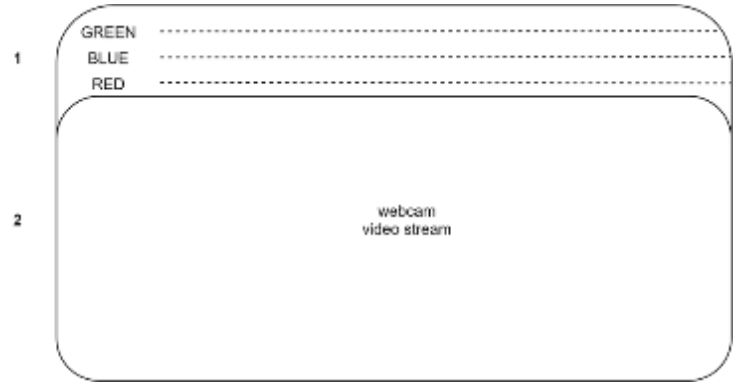
Pada tahapan ini, dataset yang sebelumnya telah didapat akan digunakan untuk *training*. *Training* dilakukan dengan *dataset* yang sebelumnya sudah disiapkan. Pada tahapan ini juga dilakukan konfigurasi *hyperparameter* yang digunakan dalam *training*. (Omar et al., 2021)

Dari *dataset* yang telah diekstraksi dari 68 *keypoint face landmark* menjadi 20 *keypoint lips landmark*. Selanjutnya akan digunakan *tools / framework* DLib untuk melakukan *training dan testing* model pendeteksian area bibir.

3.4.3. Perancangan aplikasi virtual makeup

Dari data *prediction* model yang berhasil di-*generate* oleh *DLib* selanjutnya akan diuji cobakan pendeteksian bibir menggunakan video *webcam*. Kemudian akan dikembangkan aplikasi *prototype* (purwarupa) virtual *makeup*, yang mana pada aplikasi tersebut akan diberikan pilihan warna *green, blue* dan *red*. Pada saat user memilih warna, maka bibir akan terwarnai dengan warna tersebut, yang seolah-olah seperti mengaplikasikan virtual *makeup* lipstik.

Perancangan antarmuka memiliki penjelasan mengenai desain dan implementasi aplikasi yang digunakan, dalam aplikasi yang dibuat dan diwujudkan dalam tampilan antarmuka digunakan untuk menghubungkan *user* dengan aplikasi.

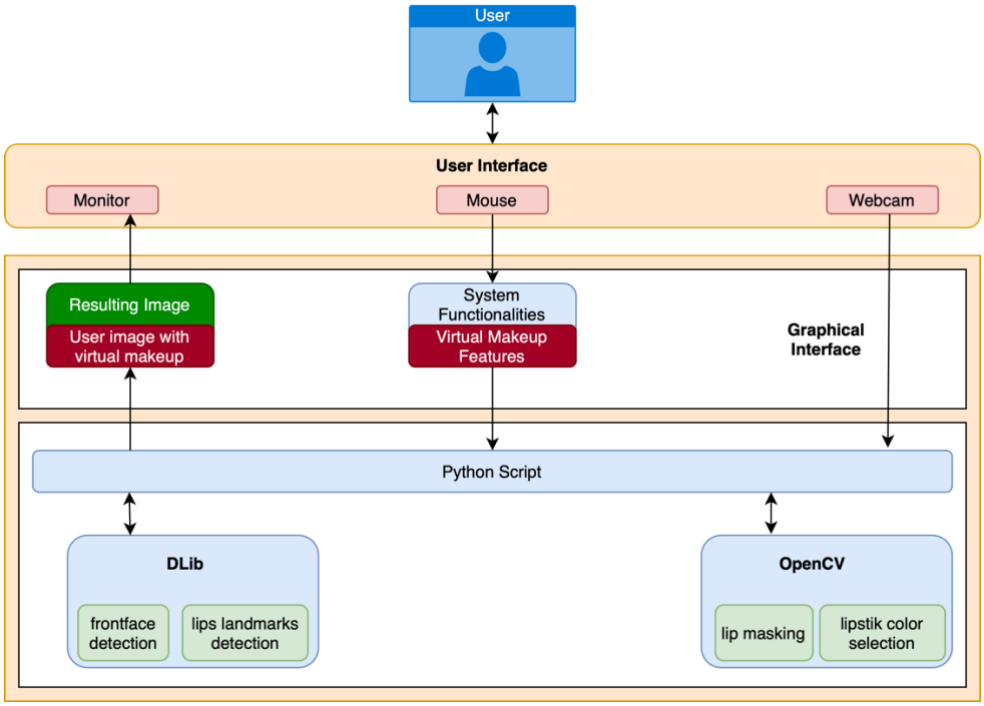


Gambar 3. 5 Perancangan antarmuka

Keterangan gambar:

1. Pilihan warna adalah fitur berbentuk *slider* yang dapat digeser ke kanan dan ke kiri oleh *user*. Yang mana user dapat melakukan perubahan warna antara merah, hijau dan biru dengan cara meng-klik atau men-*drag* pilihan warna yang tersedia.
2. *Webcam* video *stream*, adalah tampilan langsung dari user dengan menggunakan *webcam* yang tersedia, kemudian dilakukan *mirroring* posisi sehingga seolah-olah sedang melihat wajahnya pada sebuah cermin.

3.4.3.1. Blockdiagram aplikasi *virtual makeup*



Gambar 3. 6 *Blockdiagram* aplikasi virtual *makeup*

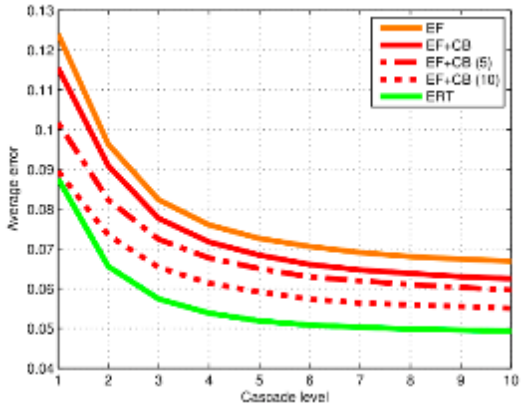
User dapat berinteraksi dengan aplikasi virtual makeup melalui bantuan perangkat layar monitor, mouse dan *webcam*. *Webcam* berfungsi untuk menangkap objek yang ada di depannya, kemudian objek akan di deteksi sebagai wajah atau non-wajah, jika terdeteksi wajah maka selanjutnya akan diproses *face aligment* dan deteksi *landmark* bibir. Dan dengan menggunakan *mouse*, user dapat mengganti warna bibir sesuai keinginannya.

3.4.4. Pengujian aplikasi

Dari latar belakang masalah ada dua masalah yang akan dicoba teliti yakni:

1. *model* yang dikembangkan apakah dapat mengurangi penggunaan *resource*?
2. *model* yang dikembangkan memiliki kecepatan berapa *frame per second* (FPS)?

Pada tahapan satu, dilakukan evaluasi terhadap *pretrained model* pada metriks akurasi, jika metriks akurasi dirasa sudah mencukupi maka *training* selesai dan *model* akan disimpan untuk digunakan pada tahapan selanjutnya, jika akurasi dari *model* dirasa tidak mencukupi, maka dapat dilakukan modifikasi terhadap *hyperparameter model* agar didapatkan akurasi yang lebih baik (Omar et al., 2021). Dari hasil data *training* akan dilakukan pengujian *error rate*.



Gambar 3. 7 Contoh hasil pengujian average error (Kazemi & Sullivan, 2014)

Kemudian pada tahap dua, untuk pengujian kecepatan FPS akan dilakukan pengujian manual seberapa cepat proses pendeteksian *lips keypoint landmark* dengan berbagai kondisi percobaan.

BAB IV: PEMBAHASAN DAN HASIL PENELITIAN

Pengembangan *prototype* aplikasi *virtual makeup* menggunakan ekstraksi *dataset* untuk mengetahui apakah model yang dikembangkan dapat mensimulasikan lipstik sebagai virtual *makeup*, mencari tahu *hyperparameter* terbaik dan mencari tahu variable apa saja yang berubah setelah dilakukan ekstraksi *dataset*.

4.1. Persiapan Pengujian, Analisis dan Temuan-Temuan

Persiapan untuk melakukan pengujian hasil penelitian adalah dengan menentukan parameter / poin-poin pengujian, skenario pengujian dan *script* untuk melakukan pengujian. Analisis data dilakukan dari temuan hasil penelitian.

4.1.1. Analisis Bahan dan Pengujian Penelitian

Hasil studi pustaka/literatur pada penelitian dengan metode *Ensemble of Regression Tress* (ERT) bahwa data hasil *training* akan menghasilkan *prediction model* yang akan mendeteksi titik-titik pada wajah sesuai dengan data anotasi pada gambar. Dan untuk mengetahui performa dari *prediction model* tersebut dilakukan pengujian *error loss rate* yaitu *Sum of square error* yang berarti performa *model* lebih baik jika nilainya lebih kecil atau mendekati angka nol. *Error rate* akan dilakukan pada dua data yaitu *training* dan *testing*, hal ini dilakukan guna untuk mengetahui apakah *prediction model* yang dikembangkan *overfitting* atau tidak.

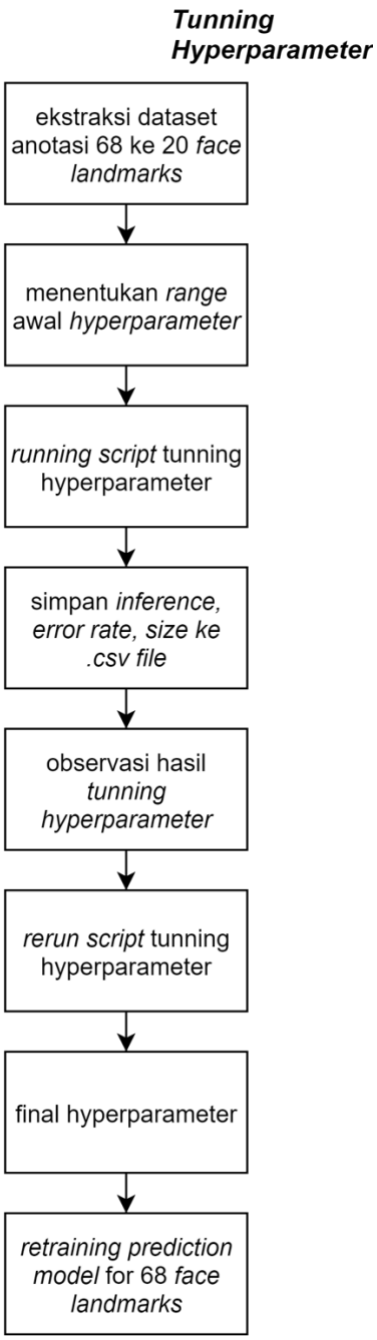
Kemudian untuk menguji aplikasi dari sudut pandang *software developer* dan *user*, penulis melakukan pengujian tambahan untuk penggunaan *resource* dan *frame per second* (FPS) dari aplikasi *prototype* virtual *makeup*. Maka *prediction model* di-*training* sebanyak dua kali untuk *hyperparameter* yang sama, menggunakan dataset anotasi hasil ekstraksi menjadi 20 *landmarks* bibir, sedangkan satu lagi menggunakan *dataset* anotasi aslinya yaitu 68 *landmarks* wajah.

4.1.2. Analisis Temuan-Temuan

Analisis temuan-temuan yang akan dikerjakan sebagai orientasi untuk pengujian dalam penelitian metode *Ensemble of Regression Trees* dan pengembangan *prototype* aplikasi virtual *makeup* adalah sebagai berikut:

1. Dipersiapkan data file anotasi *dataset* wajah dengan format xml.
2. Dipersiapkan file untuk *tunning hyperparameter*
3. Dipersiapkan *Python script* untuk mendeteksi bibir, *masking bibir* dan *prototype* virtual *makeup*
4. Dipersiapkan skenario pengujian untuk pengamatan *frame per second*

Untuk persiapan *tunning hyperparameter*, mula-mula ditentukan *range awal* sebagai landasan untuk mencari tahu *hyperparameter* terbaik. Kemudian, dilakukan percobaan *tunning* ke-2 untuk mendapatkan hasil *hyperparameter* paling optimal dengan menggabungkan hasil *hyperparameter* terbaik pada tunning pertama dan *hyperparameter* terbaik dari hasil studi literatur. Berikut alur proses *tunning hyperparameter*:



Gambar 4. 1 proses *tunning hyperparameter*

Setelah *hyperparameter* terbaik menurut penulis didapatkan, kemudian dilakukan *training* satu kali lagi untuk mendapatkan *prediction model* 68 *face landmarks* yang nantinya akan digunakan untuk pengamatan dan perbandingan dengan hasil ekstraksi *20 lips landmarks*. Maka selanjutnya akan dilakukan pengujian untuk *error rate* terhadap data *training* dan data *testing* untuk melihat apakah *error rate* berhasil berkurang dan mendekati angka nol?

Kemudian untuk pengujian dari sudut pandang *developer* dan *user* akan dilakukan pengujian terhadap *frame per second* (FPS), yang artinya semakin tinggi nilai FPS maka aplikasi akan terlihat dan terasa semakin *smooth*.

4.1.3. Analisis Kebutuhan Lingkungan Pengembangan dan Pengujian

Lingkungan pengujian aplikasi pada penelitian ini memberikan gambaran tentang kebutuhan apa saja yang digunakan dalam proses pengujian aplikasi. Untuk pengembangan *prediction model* dan aplikasi *prototype* virtual makeup menggunakan *hardware* yang berbeda. Berikut daftar kebutuhan yang digunakan untuk penelitian:

1. *Hardware* untuk *training prediction* model

   Untuk *training prediction* model menggunakan *hardware* dengan dua spesifikasi yaitu:

   - Laptop

     Untuk *training* pada *trials* pertama menggunakan laptop dengan spesifikasi sebagai berikut:

     - prosesor yang digunakan: 2,4 GHz Quad-Core Intel Core i5
     - memori RAM: 16 GB 2133 MHz LPDDR3

   - VPS AWS Server

     Untuk *training* pada *trials* kedua dan *training prediction* model untuk 68 titik wajah menggunakan hardware dengan spesifikasi sebagai berikut:

     - tipe *instance*: r5a.large
     - jumlah *virtual* CPU: 2
     - *random access memory*: 16 GB
     - *instance storage* / *hardisk*: 28 GB

2. *Hardware* untuk *development* dan *running* aplikasi *prototype* virtual *makeup*

   Spesifikasi hardware yang digunakan untuk *development* / pengembangan aplikasi dan menjalankan *prototype* aplikasi pada penelitian ini adalah:

   - komputer / laptop
   - spesifikasi prosesor yang digunakan: 2,4 GHz Quad-Core Intel Core i5
   - memori RAM: 16 GB 2133 MHz LPDDR3

   Spesifik laptop yang digunakan adalah MacBook Pro (13-inch, 2019, *Four Thunderbolt* 3 *ports*) karena laptop ini yang digunakan sehari-hari oleh penulis. Untuk menjalankan aplikasi *prototype* ini tidak memerlukan (tidak mendukung) penggunaan GPU, sehingga laptop ini dirasa sudah cukup untuk pengembangan aplikasi maupun menjalankan *prototype*-nya.

3. *Software* untuk *tunning hyperparameter* dan *training prediction* model

   Untuk proses *tunning hyperparameter* (pencarian parameter terbaik untuk *prediction* model) dan *training prediction* model, menggunakan list *software* sebagai berikut:

   - Sistem operasi: Ubuntu 18.04 dan MacOS Big Sur
   - *Library training prediction model:* DLib

- *Scripting tunning hyperparameter:* Python

Pada penelitian ini membutuhkan dua buah sistem operasi, karena seperti dijelaskan pada kebutuhan *hardware* untuk *training prediction model* di atas, bahwa laptop penulis mengalami kendala teknis. Sehingga untuk proses awal *tunning dan training prediction model* untuk laptop dengan sistem operasi MacOS kemudian untuk selanjutnya menyewa sebuah *instance* VPS dari AWS dengan sistem operasi Ubuntu versi 18.04.

4. *Software* untuk *development* dan *running* aplikasi *prototype* virtual *makeup*

Daftar *software* yang digunakan untuk menjalankan *prototype* aplikasi pada penelitian ini adalah:

- sistem operasi: macOS Big Sur
- *library load prediction model*: DLib
- *library prototyping virtual makeup*: Python dan OpenCV

Sistem operasi yang digunakan adalah macOS, karena mengingat laptop yang digunakan yaitu Macbook yangmana *default* sistem operasinya adalah macOS. Merujuk pada website resmi DLib.net, *library* ini dapat berjalan pada berbagai sistem operasi (*cross platform*) baik pada Microsoft Windows, Linux dan Mac OS X sistem. Namun, DLib seharusnya juga dapat bekerja pada sistem operasi POSIX apa pun dan telah digunakan pada Solaris, HPUX, dan BSD. Kemudian untuk *library* kedua yaitu OpenCV merujuk pada website resmi opencv.org/about juga bersifat *cross platform*, yang dapat berjalan pada sistem operasi Windows, Linux, Android dan Mac OS.

4.2. Pengembangan Aplikasi

Prosedur pengembangan aplikasi secara umum untuk *prototype* virtual *makeup* terdiri atas beberapa tahap, antara lain meliputi:

4.2.1. *Tunning 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: berapa banyak *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 / dangkal yang lebih cepat, tetapi berpotensi kurang akurat. Nilai tree_depth yang lebih besar akan membuat *tree* yang lebih dalam yang lebih lambat, tetapi berpotensi lebih akurat.
- nu: *regularization* parameter untuk membantu model melaukan *generalize* sebuah *shape*. Nilai yang mendekati 1 akan membuat model lebih *fit* dengan data pelatihan, tetapi berpotensi menyebabkan *overfitting*. Nilai mendekati 0 akan membantu model menggeneralisasi; namun, semakin dekat nu ke 0 semakin banyak data pelatihan yang diperlukan.

- cascade_depth: jumlah *cacsade* untuk *refine and tune* (menyempurnakan) prediksi awal. Parameter ini akan memiliki dampak dramatis pada akurasi dan ukuran *output* file model. Semakin banyak *cascade* yang digunakan, semakin besar ukuran model (dan berpotensi lebih akurat). Semakin sedikit cascase yang digunakan, semakin kecil model (tetapi juga dapat menghasilkan akurasi yang lebih rendah).
- feature_pool_size: mengontrol jumlah piksel acak yang digunakan untuk menghasilkan fitur di dalam *cascade*. Semakin banyak piksel yang digunakan, semakin lambat model akan berjalan (tetapi juga dapat menghasilkan prediktor *shape* / bentuk yang lebih akurat). Semakin sedikit piksel yang diperhitungkan, semakin cepat model akan berjalan (tetapi dapat juga berdampak kurang akurat).
- num_test_splits: jumlah pemisahan tes mempengaruhi waktu pelatihan dan akurasi model. Semakin banyak num_test_splits, 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*. Dari penulis, sejauh ini masih belum mengetahui apakah *jitter* yang dimaksud adalah menambahkan *noise*, menghilangkan anotasi atau *data augmentation* (rotasi gambar).
- oversampling_translation_jitter: Mengontrol jumlah *translation* "jitter"/augmentasi yang diterapkan ke kumpulan data.

Kemudian penjelasan dari kolom untuk *output* percobaan *tunning 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 *tunning hyperparameter* dilakukan pada file *Python script* "tune_predictor_hyperparams.py". Inference_speed didapatkan dari *prediction model* yang melakukan deteksi / prediksi pada satu buah gambar yang dalam penelitian ini adalah file "image_0237_mirror.jpg" yang dilakukan pada sebuah kode fungsi bernama "evaluate_model_speed". Prediksi *inference speed* dilakukan secara default yaitu sebanyak 10x. 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 di rata-rata.

Untuk pencarian *hyperparameter* terbaik, pada *trials* pertama sebagai dasar *tunning hyperparameter* yang digunakan adalah sebagai berikut:

- 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* di atas, didapatkan hasil *tunning* sebagai berikut:

Tabel 4. 1 Hasil pertama *tunning 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
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

Notes, 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* atau percobaan dengan *hyperparameter* di atas, dilakukan *sorting* terhadap nilai *training error* dari terkecil ke terbesar, pertama untuk *tunning hyperparameter* didapatkan hasil *training error* terkecil pada percobaan ke-37 dengan nilai **2,8049668939726**. Nilai tersebut sangat baik karna nilai *Sum of Square Error* semakin mendekati 0, maka akan semakin baik. Akantetapi, model ini terlihat *overfitting* karna pada saat dievaluasi pada data testing memiliki nilai *testing error*-nya yaitu **86,16431928**, yang berarti jika nilai *testing error* dibagi dengan *training error* menghasilkan nilai **31x**. 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*: 2,804969763; 2,865275815; 2,865899087. Kemudian nilai *testing error*: 83,39018955; 31,80795445; 30,33404961. Dengan perbandingan *training* dan *testing error*: **30x**; **11x**; **11x**.

Nilai perbandingan *training* dan *testing error*, di atas pada percobaan ke-29 dan ke-9 turun cukup drastis dari angka puluhan ke belasan, akan tetapi jauh terlihat lebih kecil pada percobaan ke-10 dengan nilai **2x**, dengan nilai *training error* **4,707032812** dan *testing error* **8,917515415**. Akan tetapi, untuk *file size* dari *prediction model* masih terlihat cukup besar yakni 57,18 MB. Selanjutnya akan dilakukan pengamatan terhadap *file size* juga untuk mendapatkan *hyperparameter* yang terbaik menurut penulis.

	trials	tree_depth	nu	cascade_depth	feature_pool_si	num_test_splits	oversampling_ai	oversampling_tri	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
15	49	6	0,1	14	250	100	40	0,25	0,001829099	21726,44662	7,570128869	10,31579369	76859997	1
16	43	4	0,1	14	250	100	40	0	0,00125606	18669,77317	7,811741392	9,505495547	23828548	1

Gambar 4. 2 Hasil *tunning* pertama *hyperparameter sort by training error ascending*

Masih pada urutan / *sorting* berdasarkan *training error* terkecil, ukuran file mulai menyusut pada percobaan ke- 48 dengan *file size* dalam MB yaitu 9.34 seperti pada gambar di atas.

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 di bawah ini:

	tria	tre	nu	cascade_	feature_	num_test_s	oversa	oversampli	inference_speed	training_time	training_error	testing_error	model_size	testing / training
2	10	6	0,25	10	1000	300	40	0,1	0,001360893	25611,48332	4,707032812	8,917515415	57188350	2
3	16	4	0,25	8	500	300	40	0	0,000879955	19965,74196	7,52887763	9,043093333	16331771	1
4	28	4	0,1	12	750	100	40	0	0,001047682	35849,05467	7,987491006	9,460420688	21501596	1
5	43	4	0,1	14	250	100	40	0	0,00125606	18669,77317	7,811741392	9,505495547	23828548	1
6	8	6	0,25	6	250	100	40	0,1	0,000686645	16801,24467	7,209111893	10,12585976	36436306	1

Gambar 4. 2 Hasil *tunning* pertama *hyperparameter sort by testing error ascending*

Percobaan kedua dengan memperhatikan dan menggabungkan *hyperparameter* pada *batch-trials* pertama, hasil *research* dari *paper* dan website resmi DLib yaitu: Penggunaan dan pengaturan *hyperparameter* untuk proses *training* merujuk pada halaman 4 section 3 dari jurnal (Kazemi & Sullivan, 2014), yaitu: *cascade* T (*cascade_depth*) = 10, K = 500, *tree depth* = 5, P (feature_pool_size) = 400 piksel, S (num_test_splits) = 20. Dan dari website resmi DLib (http://dlib.net/train_shape_predictor_ex.cpp.html), yaitu nu: 0,05, *oversampling_amount* = 300, *oversampling_translation_jitter* = 0.1.

Hasil *hyperparameter* terbaik pada *batch-trials* pertama ada pada *trial* urutan ke-10, 53, 41, 48 dan 16. Akan tetapi, *trial* urutan ke-16 menjadi yang terbaik menurut penulis, karena memiliki perbandingan train_test dan error_test rendah yaitu 1:1, ukuran file yang cukup kecil 16.33MB, nilai terendah urutan ke-2 untuk *error_test* dengan angka 9.04 dan *inference_speed* dengan angka 0,000879954933471676. 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

Hasil studi literatur pada *paper* dan website DLib yaitu penggunaan dan pengaturan *hyperparameter* untuk proses *training* jika di sajikan dalam bentuk list, yaitu:

- *tree_depth* = 5
- nu = 0,05
- *cascade_depth* = 10
- *feature_pool_size* = 400
- *num_test_splits* = 20
- *oversampling_amount* = 300
- *oversampling_translation_jitter* = 0,1

Maka untuk mendapatkan hasil *hyperparameter* terbaik dilakukan penggabungan dari kedua hasil *trials* dan *research*, guna digunakan kembali untuk *tunning hyperparameter* dengan 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

Untuk hasil akhir dari *tunning hyperparameter* kedua adalah sebagai berikut:

Tabel 4. 2 Hasil kedua tunning 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

Dari tabel di atas, penulis mengambil kesimpulan bahwa *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. Maka *hyperparameter* terbaik menurut penulis adalah:

- tree_depth = 4
- nu = 0,25
- cascade_depth = 8
- feature_pool_size = 500
- num_test_splits = 300
- oversampling_amount = 40
- oversampling_translation_jitter = 0

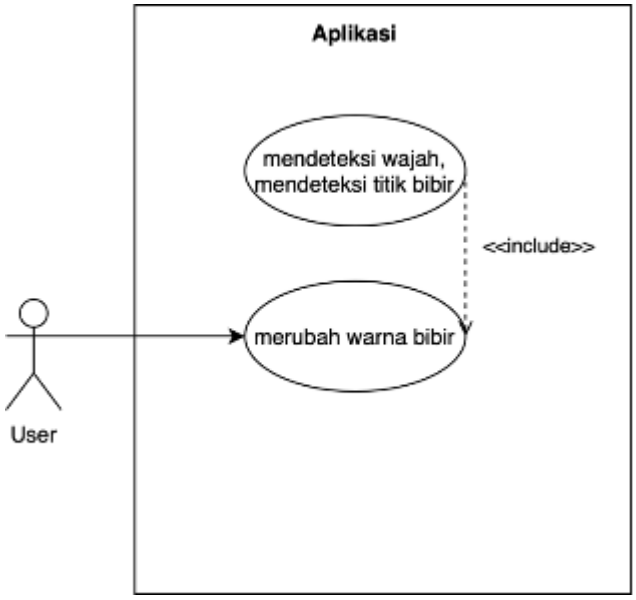
Selanjutnya agar dapat melakukan pengamatan dan perbandingan antara *prediction model* hasil ekstraksi *dataset* dan *original dataset*, maka dilakukan *training* dengan *hyperparameter* yang sama terhadap *training dataset*. Hasil dari *training* prediction model untuk original *dataset* (68 *face landmarks*) adalah sebagai berikut:

Tabel 4. 3 Hasil training pada *original dataset*

NO	TD	NU	CD	FPZ	NTS	OA	OTJ	IS	TT	TRE	TEE	MS	VSE
1	4	0,25	8	500	300	40ß	0	0,0015	32868	6,8	9,602	53165277	1

4.2.2. *Use Case* Diagram

Proses yang terpenting dalam pengembangan aplikasi adalah pendeteksian titik-titik bibir, kemudian dengan aksi dari *user* dengan memilih warna dapat mensimulasikan penggunaan lipstik / pewarnaan pada bibir.

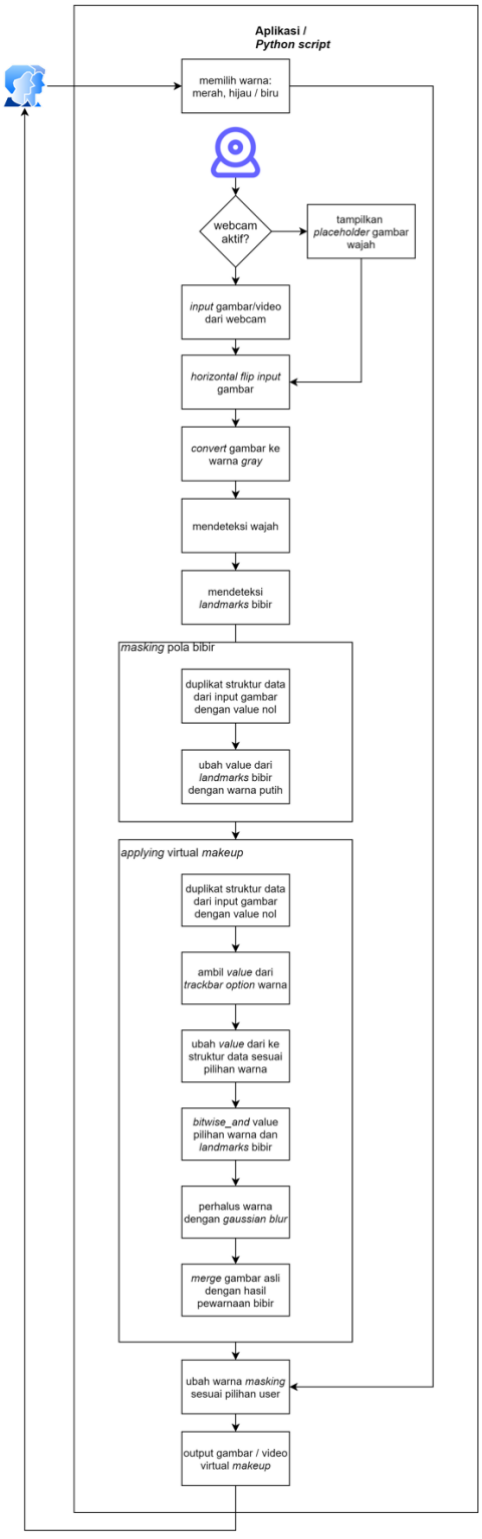


Gambar 4. 3 *Use Case* diagram aplikasi

Interaksi *user* terhadap sistem yaitu ketika user melakukan *action* dengan mengganti warna pada *option slider* untuk warna *red, green* dan blue, maka warna bibir akan mengikuti warna yang telah dipilih oleh *user*. Di dalam aplikasi di belakang layar melakukan proses pendeteksian wajah dan kemudian proses pendeteksian titik-titik pada bibir. Selanjutnya titik bibir tersebut dikoneksikan sehingga terbentuk pola bibir, yang mana pola bibir tersebut akan menjadi *masking* dari bibir pada *video stream* dan *masking* akan diwarnai sesuai pilihan *user.*

4.2.3. Diagram Proses Sistem

Diagram proses yang dimaksudkan adalah bagaimana sistem akan bekerja, proses-proses yang digunakan, mulai dari user melakukan aksi mengubah warna merah, hijau atau biru kemudian hingga aplikasi mengeluarkan output berupa gambar/video hasil pengaplikasian virtual *makeup*.



Gambar 4. 4 Proses aplikasi *prototype* virtual *makeup*

Aksi dari *user* hanya dapat mengubah / memilih warna bibir, selanjutnya aplikasi akan mengambil *input* gambar dari *webcam* dengan bantuan *library OpenCV*, pendeteksian wajah dan *face landmarks* menggunakan *library DLib*. Selanjutnya dengan *python script* akan dilakukan proses *masking* bibir, memberi

warna pada *masking* sesuai pilihan *user* dan terakhir *blending* / penggabungan antara *masking* bibir yang sudah terwarnai dengan gambar asli dari video *webcam*.

4.2.4. *Flowchart* Diagram

4.2.4.1. Flowchart overview aplikasi virtual *makeup*

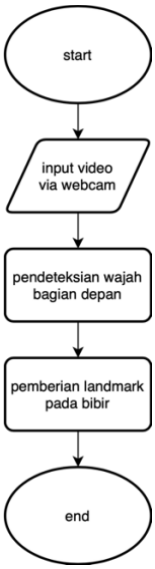
Pada bagian ini akan digambarkan mengenai *overview* dari rancangan aplikasi virtual *makeup* secara keseluruhan, yaitu:

Gambar 4. 5 *Flowcart* keseluruhan aplikasi virtual *makeup*

Aplikasi virtual *makeup* bekerja dengan cara yang awalnya kamera *webcam* mengambil *input* berupa video. Kemudian aplikasi akan mendeteksi apakah terdapat wajah pada 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 virtual *makeup* yang sudah diaplikasikan.

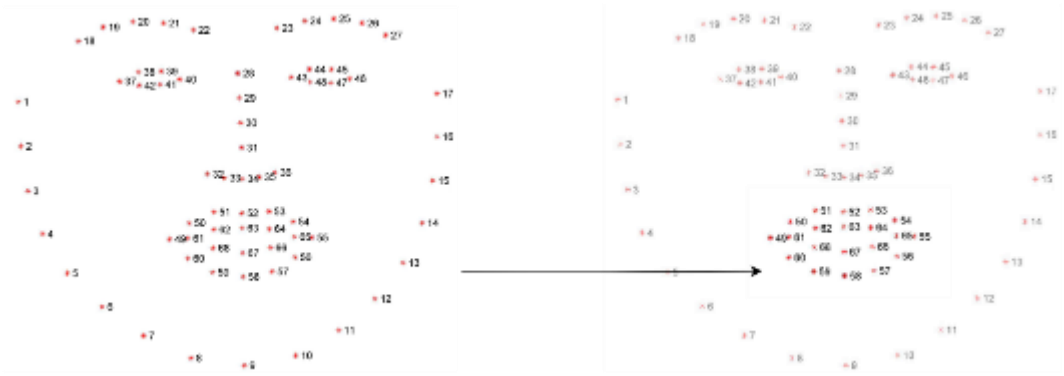
4.2.4.2. Pendeteksian wajah dan *landmarks* bibir

Pada bagian ini akan dijelaskan tentang pendeteksian wajah dan *landmarks* bibir:



Gambar 4. 6 *Flowchart* pendeteksian wajah dan *landmarks* bibir

Untuk pendeteksian wajah dan menentukan titik landmark dari wajah pengguna, peneliti menggunakan *library* yang sudah ada yakni DLib, akan tetapi peneliti mengekstraksi *dataset* anotasi wajah dari awalnya 68 titik wajah menjadi 20 titik bibir. Untuk mendeteksi titik *landmark* wajah, program diawali dengan mendeteksi bagian wajah depan pengguna dari data video, setelah itu wajah pengguna akan dideteksi titik *landmark* bibir, sesuai dengan *ground truth* / anotasi bibir dari gambar wajah. Koordinat titik bibir disimpan pada array *landmark* dalam bentuk x, y sesuai gambar berikut:

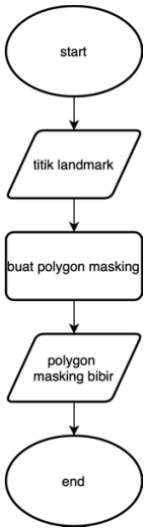


Gambar 4. 7 Hasil ekstraksi *landmark* bibir

Setelah wajah terdeteksi dan *landmark* bibir telah didapatkan, maka langkah berikutnya yaitu membuat *masking* untuk bibir yang selanjutnya dapat diaplikasi warna pada bibir yang mengsimulasikan penggunakan virtual *makeup*.

4.2.4.3. Pembuatan *Masking*

Pada bagian ini akan dijelaskan proses pembuatan *masking* untuk area titik *landmarks* bibir, yaitu:



Gambar 4. 8 Flowchart pembuatan *masking*

Langkah pembuatan masking bibir yaitu dengan menggunakan 20 *landmark* pada indeks ke 1-20. Jika pada 68 *landmarks*, maka titik bibir ada pada indeks ke 49 – 68, yang berupa bibir atas untuk indeks 49 - 55, 65, 64, 63, 62, dan 61; dan bibir bawah untuk indeks 55 - 60, 49, 61, 68, 67, 66, dan 65.

4.2.4.4. Pembuatan *virtual* makeup

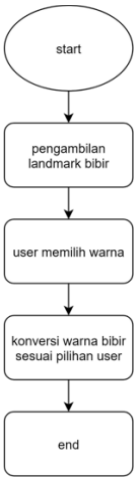
Pada bagian ini akan dijelaskan tentang proses pembuatan virtual *makeup lipstick*:

Gambar 4. 9 *Flowcart* pembuatan virtual *makeup*

Input gambar bibir yang sudah di-*masking* akan diubah warnanya sesuai dengan keinginan pengguna.

4.2.4.5. Penambahan virtual *makeup* pada bibir

Kemudian untuk *flowchart* / proses pembuatan virtual *makeup lipstick* adalah:

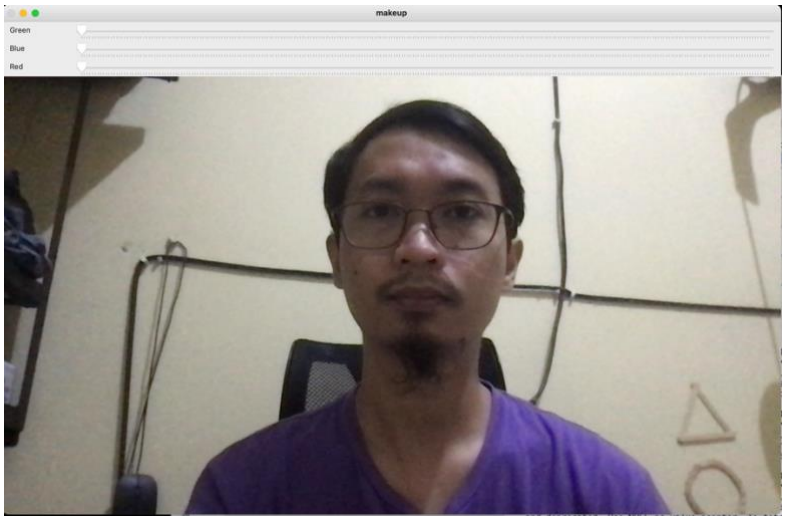


Gambar 4. 10 *Flowchart* pembuatan virtual *makeup*

Gambar di atas menunjukkan proses perubahan warna bibir sesuai dengan keinginan pengguna.

4.2.5. Konstruksi Antarmuka

Konstruksi model yang di bangun mengacu sesuai dengan rancangan desain pada Bab 3 yang di dalamnya terdapat dua blok yaitu blok atas pilihan warna kepada user berupa hijau, biru, merah atau gabungan dari ketiganya. Blok sebelah bawah untuk *output* dari pengolahan gambar. Adapun *interface* aplikasi yang dibangun adalah sebagai berikut:



Gambar 4. 11 konstruksi antarmuka saat dijalankan

4.3. Pengujian Aplikasi

Pengujian atau *testing* merupakan bagian dari penelitian yang tidak kalah penting, pengujian digunakan untuk memastikan hasil dari penelitian memenuhi tujuan yang dicapai atau tidak. Pengujian ini dilakukan berdasarkan parameter atau kriteria yang sudah ditentukan berdasarkan *output* dari *error rate loss*. Juga pengujian penggunakan *resource* dan *frame per second*.

4.3.1. Pengujian *error rate* dari *prediction* model

Untuk melihat dan menguji seberapa akurat *prediction* model yang telah di-*training*, maka dilakukan evaluasi terhadapnya 85% *training* dan 15% *training*.

Dari hasil evaluasi diketahui bahwa *error rate* dari *prediction model* yaitu **5,419** pada *dataset training* dan menghasilkan *error rate* sebesar **11,073** pada data *testing*. Dengan parameter pada saat training yaitu:

- *tree_depth* = 4
- nu / *regularization* = 0,1
- *cacscade depth* = 15
- *feature pool size* = 400
- *number of test split* = 50
- *oversampling amount* = 5
- *oversampling translation jitter* = 0.1

Secara lebih detail, disajikan perbandingan antara *prediction model* yang penulis lakukan dengan *prediction model default* dari *library DLib*, yaitu:

Tabel 4. 4 Pengujian Error Rate

Model name	*Evaluate error rate on*	
	Training dataset	*Test 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 verfikasi bahwa model yang kita kembangkan tidak *overfitting* dan idealnya *prediction* model dapat mendeteksi gambar di luar set *training*. Nilai evaluasi dari *training loss* akan lebih kecil daripada *testing loss*. Hal ini tidak berarti bahwa model yang dikembangkan memiliki performa yang buruk, akantetapi secara sederhana berarti bahwa model tersebut melakukan pekerjaan prediksi yang lebih baik pada *training* data daripada *testing* data.

Dari hasil pengujian dapat disimpulkan bahwa untuk *error rate prediction model* 20 *face landmark* yang dikembangkan oleh penulis pada *training dataset* memiliki *error rate* yang lebih baik jika dibandingkan dengan *prediction model default* dari *DLib library.* Akan tetapi, pada *test dataset* menunjukkan sebaliknya, 68 *face landmark* lebih baik dari pada 20 *face landmark*.

4.3.2. Pengamatan ukuran file dari *prediction model*

Pada *device* dengan *resource* terbatas seperti *embedded system* atau *IoT* seperti *Raspberry Pi*, ukuran file dari *prediction* model dapat berpengaruh terhadap kinerja sistem. Semakin kecil ukuran file akan lebih menghemat *resource* terutama media penyimpanan file. Berikut ukuran dari file *prediction model* dari *default DLib library* dengan *prediction model* yang dirancang oleh penulis:

Tabel 4. 5 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)

Dari tabel di atas dapat ditarik kesimpulan bahwa, pengurangan *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.

4.3.3. Pengamatan penggunaan CPU & *memory usage*

Untuk melihat pengaruh ekstraksi *dataset*, maka dilakukan pengamatan terhadap penggunaan *resource* pada komputer untuk melihat dan membandingkan penggunaan *resource* secara detail, disajikan dalam bentuk tabel sebagai berikut:

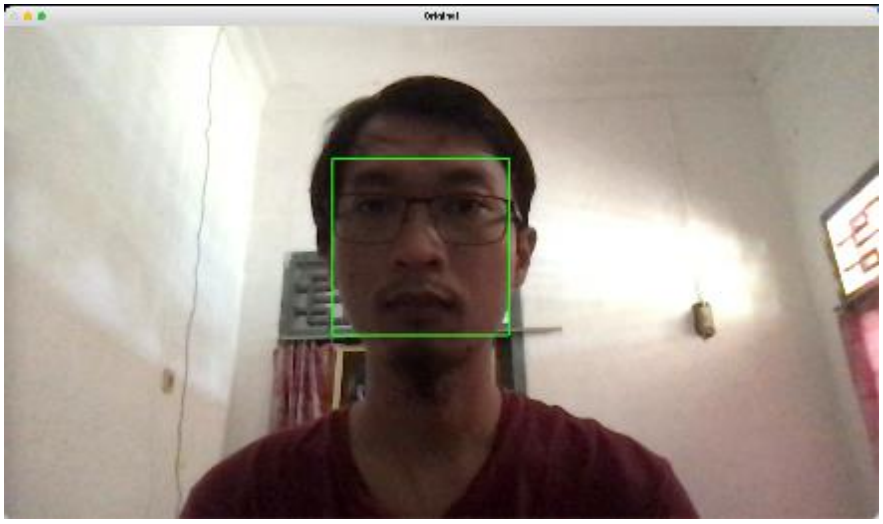
Tabel 4. 6 Perbandingan penggunaan *resource*

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%

Dari table di atas diketahui bawah 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%** untuk penggunaan *resouce* CPU.

4.3.4. Pengujian deteksi wajah (*face detection*)

Pertama-tama akan dilakukan deteksi wajah pengguna melalui video *webcam* menggunakan *function default* dari Dlib yaitu get_frontal_face_detector. Input *video stream* didapatkan dari *function VideoCapture* dari *library OpenCV* yang kemudian akan di *flip* posisinya secara *horizontal* agar pergerakan pada *video stream* mengikuti pergerakan *user.* Aplikasi dapat mendeteksi wajah dengan memberikan penanda kotak (*bounding box*) seperti pada gambar berikut:



Gambar 4. 12 Pengujian deteksi wajah

4.3.5. Pengujian deteksi titik bibir (*face localization*)

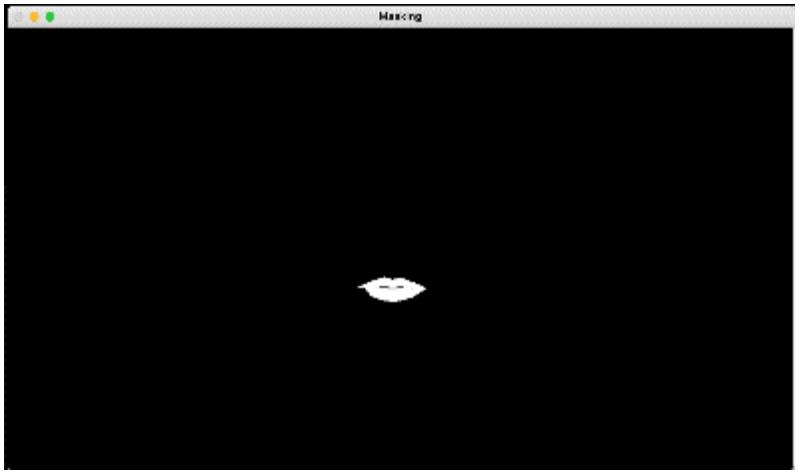
Setelah dilakukan *face detection* maka gambar wajah akan di *convert* ke warna *grayscale*, selanjutnya dilakukan *face localization* yang dalam penelitian ini mendeteksi titik-titik pada area bibir. Dan aplikasi dapat mendeteksi titik-titik bibir seperti pada gambar berikut:

Gambar 4. 13 Pengujian pendeteksian titik bibir

Titik-titik bibir tersebut berjumlah 20 titik yang didapatkan dari hasil *training* yang kemudian digunakan pada *function shape_predictor* dari DLib *library*. Kemudian pada saat pengujian diberikan titik lingkaran berwarna putih 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*).

4.3.6. Pengujian *masking* bibir

Untuk mengaplikasikan virtual *makeup lipstik* pada bibir, titik-titik bibir dikoneksikan satu sama lain sehingga membentuk pola bibir. Hasil dari pola bibir / *masking* dapat dilihat sesuai pada gambar berikut:



Gambar 4. 14 Masking bibir

Masking bentuk bibir berwarna putih seperti gambar di atas, melalui proses sebagai berikut:

1. gambar wajah akan ditranslasi kedalam bentuk *matrix array*
2. setiap *value* dari *matrix array* akan diganti dengan 0 agar berwarna hitam
3. *masking* titik-titik bibir hasil dari *prediction model* dengan warna putih
4. kemudian hasil *masking* dan gambar wajah aslinya akan digabungkan sehingga menghasilkan 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. Karna pewarnaan tidak langsung dilakukan pada *input videostream webcam*, akantetapi pewarnaan *lipstik* pada bibir akan diberikan pada bagian / *layer masking* ini.

4.3.7. Pengamatan *Frame Per Second* (FPS)

Untuk mengetahui kinerja *prediction model* dari sisi *frame per second* (*FPS)*--jumlah frame yang ditampilkan per detik, akan dilakukan beberapa skenario pengujian. Penghitungan FPS adalah berdasarkan satu detik dibagi dengan waktu akhir setelah pemrosesan *face landmark detection* dikurangi dengan waktu awal *script* program dijalankan.

```
start_time = time.time()
#... line of code for face landmark detection
fps = 1 / (time.time() - start_time)
```

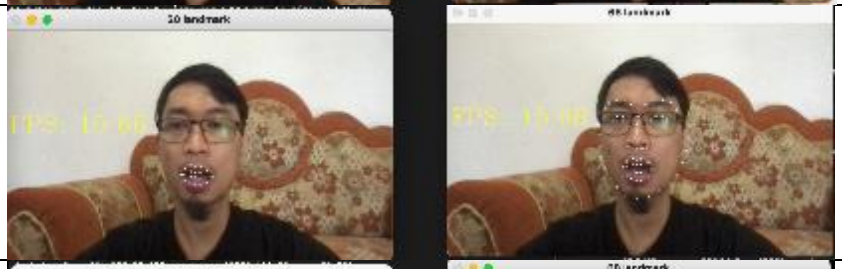
Tetapi tidak berjalan baik karena awalnya pengujian dilakukan satu persatu. Yaitu *Python script* untuk pengamatan 68 *landmarks* diamati dan disimpan

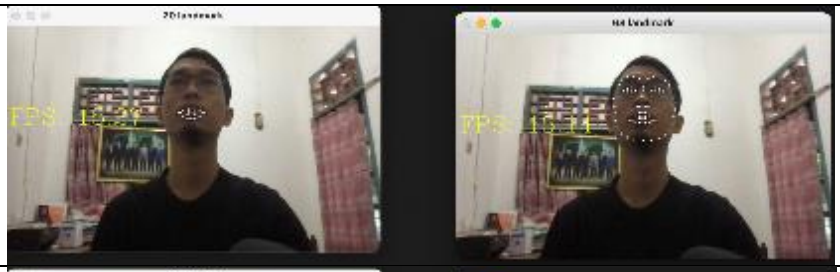
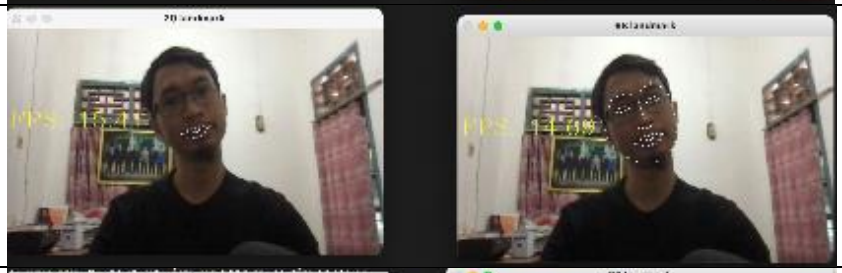
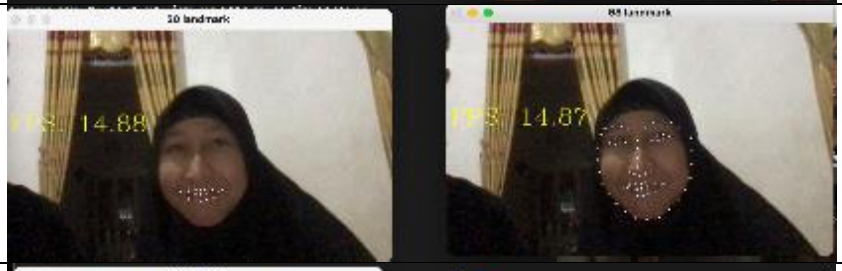
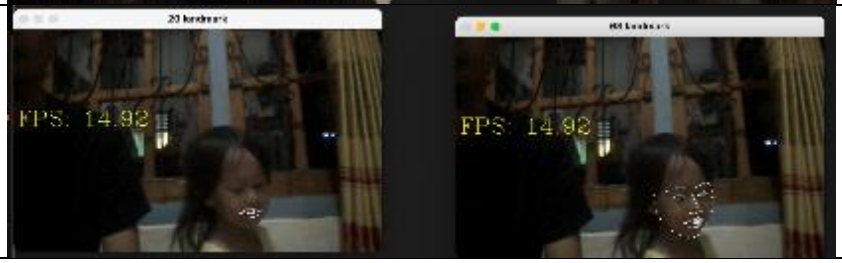
beberapa kali percobaan untuk kemudian di rata-rata, baru kemudian selanjutnya hal yang sama dilakukan pada *script* untuk pengamatan 20 *landmarks*. Akantetapi menurut penulis pengamatannya menjadi kurang baik, karena pergerakan / kondisi wajah akan berbeda pada saat pengujian 68 dan 20 *landmarks*.

Kemudian penulis sudah mencoba untuk menggabungkan *prediction model* untuk 68 *landmarks* dan 20 *landmarks* ke dalam satu buah *Python script*. Akan tetapi tidak memberikan hasil, karena FPS yang dihasilkan selalu identik (hasil FPS untuk 68 dan 20 *landmarks* hasilnya selalu sama).

Akhirnya, dua *Python script* dijalankan secara bersamaan dan kemudian *output* darinya disandingkan untuk diamati FPS-nya. Pengujian akan dilakukan beberapa skenario yaitu: pencahayaan yang berbeda, kondisi background yang berbeda, kondisi rotasi kepala yang berbeda, kondisi mulut yang berbeda.

Tabel 4. 7 Pengujian FPS pada berbagai kondisi

Kondisi / objek	Screenshot
Pencahayaan cukup	
Mulut terbuka	
Kepala miring	
Kurang pencahayaan	

Menghadap atas	
Kepala miring kiri	
Wanita usia sedang	
Wanita usia anak	

Pada pengujian pada kondisi pertama yaitu "pencahayaan cukup" terlihat terdapat peningkatan hampir dua FPS yaitu sebesar 1,89, yang awalanya 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 disebab-kan oleh banyak hal terutama seperti: objek yang terlalu jauh dari kamera *webcam*, kurangnya jumlah data wajah yang digunakan pada proses *training* untuk kondisi tersebut, pergerakan yang terlalu signifikan, pencahayaan yang kurang baik. Berikut tabel ringkasan dari hasil pengujian FPS:

Tabel 4. 8 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%)
9	Wanita usia tua	14,80	12,85	> 1,95 (13%)

Dari hasil pengujian dan pengamatan, dapat disimpulkan bahwa dengan ekstraksi dataset meningkatkan kecepatan *frame per second* (FPS) atau paling minimal sama. Akantetapi, hasilnya terlihat ada penurunan pada kondisi mulut terbuka, penulis sarankan 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*.

4.3.8. Pengujian aplikasi virtual *makeup*

Sebagai langkah terakhir untuk aplikasi virtual *makeup*, maka dari hasil *masking* bibir dapat diberikan warna tertentu sesuai keinginan user, sehingga seolah-olah seperti mengaplikasikan *lipstik* pada bibir. aplikasi *prototype* virtual *makeup* digunakan untuk proses pengujian, penulis terlebih dahulu melakukan pengujian terhadap jalannya aplikasi. Metode yang digunakan untuk pengujian aplikasi adalah metode kotak hitam (*Black-Box Testing*) untuk memastikan apakah fungsi, masukan dan keluaran dari aplikasi sesuai dengan spesifikasi yang dibutuhkan. Berikut tabel pengujian aplikasi yang meliputi proses awal menjalankan aplikasi *prototype* aplikasi virtual *makeup* menggunakan metode *Ensemble of Regression Trees*.

Tabel 4. 9 Pengujian aplikasi virtual makeup

Kondisi pengujian	screenshot
lipstik warna merah	
Lipstik warna biru	

Minim cahaya, lipstik warna merah		
Tersenyum, lipstik warna merah		
Mulut terbuka lebar, lipstik warna merah		
Wajah menghadap kanan, lipstik warna merah		
Wajah menghadap kiri, lipstik warna merah		

Wajah menghadap atas, lipstik warna merah	
Wajah menghadap bawah, lipstik warna merah	

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

4.4. Pembahasan Hasil Penelitian

Setelah melalui beberapa pengujian untuk kriteria dan scenario, maka hasil yang diperolah sebagai berikut:

Tabel 4. 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%. Akantetapi, terbukti lebih baik pada *testing dataset* sebesar 0,058%.
2	Ukuran file	Berhasil. Berkurang 69,36%
3	*Resource* CPU & RAM	Berhasil. CPU berkurang 3,8% dan RAM 30,8%
4	*Face detection*	Berhasil
5	*Face landmarks localization*	Berhasil. *Prediction model* dapat mendeteksi titik-titik (*landmarks*) bibir saja
6	*Masking* bibir	Berhasil. Aplikasi dapat 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	Virtual *makeup*	Berhasil sebagian. Aplikasi dapat mendeteksi bibir dan mewarnai *masking* bibir, tetapi tidak akurat secara visual untuk kondisi tertentu.

Dari tabel diatas 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 *landmark*s menjadi 20 *landmarks* berhasil 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 menurut penulis tidak masalah, karena peningkatan terjadi pada perbandingan 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 saat 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.

4.5. Implikasi Penelitian

Implikasi yang dihasilkan pada penelitian ini dibagi menjadi tiga aspek yaitu aspek penelitian lanjut dan aspek pengembangan aplikasi lanjutan. Implikasi dari aspek penelitian lanjut berkaitan dengan penelitian lanjutan yang diperlukan untuk meningkatkan kualitas penelitian sebelumnya, termasuk diantaranya memperluas *scope* atau ruang lingkup, memperbanyak variable *tunning hyperparameter*, memperbanyak *sample* pengujian, dengan menjelaskan secara rinci apa saja yang diperluas/diperbanyak dan maksud/tujuan/sasaran. Aspek pengembangan aplikasi lanjutan dapat berkaitan dengan pengimplementasian metode pada bahasa pemrograman yang lain, pada *environment* aplikasi web atau mobile maupun pada studi kasus selain *makeup* virtual.

Adapun pembahasan satu persatu aspek adalah sebagai berikut:

4.4.1. Aspek penelitian lanjutan

Hasil dari penelitian dapat digunakan oleh para peneliti selanjutnya sebagai bahan rujukan atau referensi untuk penelitian sejenis dan dapat menjadi model awal yang dapat dikembangkan untuk menghasilkan penelitian lanjut yang lebih baik.

Upaya untuk meningkatkan penelitian berkaitan dengan *tunning hyperparameter* dan pendeteksian area bibir saja alih-alih mendeteksi seluruh wajah. Pada teknik yang dikembangkan dalam penelitian ini, aplikasi dapat dikembangkan kearah lebih besar seperti pengembangan virtual *makeup* untuk *production ready model*.

4.4.2. Aspek pengembangan lanjutan

Hasil dari penelitian dapat digunakan oleh para *software developer* atau *company* untuk dikembangkan atau diimplementasikan pada bahasa pemgrograman seperti bahasa C karena seharusnya akan memiliki performa yang lebih baik, karena lebih *low level* daripada bahasa Python.

Pengembang aplikasi mobile juga dapat mengembangkan aplikasi yang mirip dengan MSQRD atau Instagram *filter*, karena penulis melihat sudah terdapat *porting* atau implementasi *non official* dari DLib pada bahasa pemrograman android di website Github.

Sayangnya penulis adalah pengembang aplikasi web, tetapi belum menemukan implementasi algoritma *Ensemble of Regression Trees* pada aplikasi web. Namun, sangat mungkin melakukan *porting* ke bahasa JavaScript atau agar dapat berjalan pada *environment* web browser dengan memanfaatkan *WebAssembly* (Wasm).

BAB V: PENUTUP

5.1. Kesimpulan

Berdasarkan pembahasan hasil penelitian yang sudah dibahas maka dapat ditarik kesimpulan yaitu:

1. Metode ERT dan ekstraksi *dataset* menghasilkan *prediction* model yang berpengaruh terhadap *file size* dan penggunaan resource lebih kecil; serta peningkatan secara *frame per second* (FPS) pada beberapa kondisi.
2. Aplikasi *prototype* virtual *makeup* lipstik dapat mensimulasikan penggunaan lipstik secara *realtime* melalui video *webcam*. Akan tetapi, perlu dilakukan *improvement* untuk dua kondisi yaitu wajah menghadap bawah dan mulut terbuka lebar, karena gagal mendeteksi posisi area bibir.
3. Dengan *Sum of square error:* 9,043093333 pada *testing dataset*, *inference speed* 0,001464176178 dan ukuran file 16,3 MB, maka *Hyperparameter* terbaik untuk menurut penulis adalah:
   - *tree_depth* = 4
   - nu = 0,25
   - *cascade_depth* = 8
   - *feature_pool_size* = 500
   - *num_test_splits* = 300
   - *oversampling_amount* = 40
   - *oversampling_translation_jitter* = 0

5.2. Saran

Untuk 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 pada awalnya langsung mendeteksi bibir, alih-alih mendeteksi seluruh area wajah. Karena pada penelitian ini penulis pertama-tama mendeteksi area wajah, *konversi* gambar ke warna *gray* dan baru kemudian deteksi titik-titik bibir, diharapkan dengan langsung mendeteksi bibir bisa menurunkan penggunakan *hyperparameter* sehingga FPS juga akan naik. Disarankan juga untuk melakukan percobaan dan penelitian untuk 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.

Dataset yang digunakan oleh penulis adalah dari website iBug yang diperbolehkan untuk digunakan dalam hal non-komersial dan penelitian, tetapi tidak diijinkan untuk selain kedua hal tersebut. Maka untuk kebutuhan perusahaan yang akan menjual produknya atau menggunakan *prediction model* untuk kebutuhan virtual *makeup* di lingkungan *production* dan komersial, maka perlu mencari dan menggunakan *dataset* lain. Kemudian untuk data *training* dan *testing* penulis menggunakan *dataset* yang telah tersedia di website resmi iBUG tanpa ada proses pembagian dengan metode khusus seperti K-Fold, diharapkan jika menggunakan salah satu metode seperti K-Fold dapat menurunkan *error rate* dari *prediction model*.

Untuk saat ini *user* lebih banyak bersama perangkat atau gawai *mobile*, maka aplikasi dapat dikembangkan lebih lanjut agar bisa dijalankan pada platform *mobile android* atau *iOS*, maupun aplikasi berbasis web agar dapat diakses melalui web *browser*. Untuk perusahaan atau bidang industri komersial, *Augmented Reality* seperti virtual *makeup* dapat memudahkan pengguna yang mayoritas adalah kaum wanita, sehingga untuk wanita yang berhijab dapat mencoba dengan leluasa di rumah atau ruangan pribadi. Hal ini bisa menjadi langkah pertama akuisisi pelanggan, karena jika pengguna telah mecoba berbagai macam warna lipstik, kemudian merasa cocok dengan satu atau beberapa warna. Maka dapat meningkatkan keinginan untuk langsung membeli produk lipstik atau mengingkatkan keingingan untuk nantinya dapat mencoba secara langsung produk lipstik pada gerai toko fisik.

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 atau pengembangan aplikasi untuk pendeteksi bibir atau virtual *makeup* dengan metode CNN. Terakhir, terdapat beberapa *hyperparameter* yang tidak tereksplorasi karena resource RAM yang digunakan penulis masih kurang besar yakni 16GB, diharapkan dengan mencoba memperbesar resource mendapatkan percobaan untuk mendapatkan *hyperparameter* yang lebih baik.

Daftar Pustaka

[1] Abbas, H. H., Altameemi, A. A., & Farhan, H. R. (2019). Biological landmark vs quasi-landmarks for 3D face recognition and gender classification. *International Journal of Electrical and Computer Engineering*, *9*(5), 4069–4076. https://doi.org/10.11591/ijece.v9i5.pp4069-4076

[2] Albiol, A., Monzo, D., Martin, A., Sastre, J., & Albiol, A. (2008). Face recognition using HOG-EBGM. *Pattern Recognition Letters*, *29*(10), 1537–1543. https://doi.org/10.1016/j.patrec.2008.03.017

[3] Anzai, Y. (2012). *Pattern Recognition and Machine Learning*. Elsevier Science. https://books.google.co.id/books?id=3RxQmtJnH5YC

[4] Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). *BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs*.

[5] Borges, A. D. F. S., & Morimoto, C. H. (2019). A virtual makeup augmented reality system. *Proceedings - 2019 21st Symposium on Virtual and Augmented Reality, SVR 2019*, 34–42. https://doi.org/10.1109/SVR.2019.00022

[6] Carmigniani, J., & Furht, B. (2011). Handbook of Augmented Reality. In *Handbook of Augmented Reality*. https://doi.org/10.1007/978-1-4614-0064-6

[7] Cech, J., & Soukupova, T. (2016). Real-Time Eye Blink Detection using Facial Landmarks. *Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague*, 1–8.

[8] Déniz, O., Bueno, G., Salido, J., & De La Torre, F. (2011). Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*, *32*(12), 1598–1603. https://doi.org/10.1016/j.patrec.2011.01.004

[9] Fu, K. S., & Rosenfeld, A. (1976). Pattern Recognition and Image Processing. *IEEE Transactions on Computers*, *C–25*(12), 1336–1346. https://doi.org/10.1109/TC.1976.1674602

[10] Graña, M. (2010). *Face Recognition Algorithms Proyecto Fin de Carrera Ion Marqués*.

[11] Hsu, C. F., Lin, C. C., Hung, T. Y., Lei, C. L., & Chen, K. T. (2020). A Detailed look at cnn-based approaches in facial landmark detection. *ArXiv*, 15–21.

[12] Jamshed, M., Parvin, S., & Akter, S. (2015). Significant HOG-Histogram of Oriented Gradient Feature Selection for Human Detection. *International Journal of Computer Applications*, *132*(17), 20–24. https://doi.org/10.5120/ijca2015907704

[13] Johnston, B., & Chazal, P. de. (2018). A review of image-based automatic facial landmark identification techniques. *Eurasip Journal on Image and Video Processing*, *2018*(1). https://doi.org/10.1186/s13640-018-0324-4

[14] Kartynnik, Y., & Ablavatski, A. (2019). *Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs*. 2–5. http://arxiv.org/abs/1907.06724

[15] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an

ensemble of regression trees. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1867–1874. https://doi.org/10.1109/CVPR.2014.241
[16] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758.
[17] Lazaro, A., Buliali, J. L., & Amaliah, B. (2017). Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV. *Jurnal Teknik ITS*, *6*(2). https://doi.org/10.12962/j23373539.v6i2.23175
[18] Liu, Z., Zhu, X., Hu, G., Guo, H., Tang, M., Lei, Z., Robertson, N. M., & Wang, J. (2019). Semantic alignment: Finding semantically consistent ground-truth for facial landmark detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 3462–3471. https://doi.org/10.1109/CVPR.2019.00358
[19] Mjolsness, E., & DeCoste, D. (2001). Machine learning for science: State of the art and future prospects. *Science*, *293*(5537), 2051–2055. https://doi.org/10.1126/science.293.5537.2051
[20] Mukhopadhyay, S. (2018). Deep Learning and Neural Networks. In *Advanced Data Analytics Using Python*. https://doi.org/10.1007/978-1-4842-3450-1_5
[21] Omar, J., Shabrina, N. H., Bhakti, A. N., & Patria, A. (2021). *Emotion Recognition using Convolutional Neural Network on Virtual Meeting Image*. *13*(1).
[22] Putra, J. W. G. (2019). Pengenalan Konsep Pembelajaran Mesin dan Deep Learning. In *Computational Linguistics and Natural Language Processing Laboratory* (Vol. 4). https://www.researchgate.net/publication/323700644
[23] Rao, B. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*, *7*(3), 1174–1179. https://doi.org/10.21275/ART20203995
[24] Rosalina, R., & Wijaya, A. (2020). Pendeteksian Penyakit pada Daun Cabai dengan Menggunakan Metode Deep Learning. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(3), 452–461. https://doi.org/10.28932/jutisi.v6i3.2857
[25] Rosiani, U. D., Asmara, R. A., & Laely, N. (2020). Penerapan Facial Landmark Point Untuk Klasifikasi Jenis Kelamin Berdasarkan Citra Wajah. *Jurnal Informatika Polinema*, *6*(1), 55–60. https://doi.org/10.33795/jip.v6i1.328
[26] Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2016). 300 Faces In-The-Wild Challenge: database and results. *Image and Vision Computing*, *47*, 3–18. https://doi.org/10.1016/j.imavis.2016.01.002
[27] Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013a). 300 faces in-the-wild challenge: The first facial landmark Localization Challenge. *Proceedings of the IEEE International Conference on Computer Vision*, 397–403. https://doi.org/10.1109/ICCVW.2013.59
[28] Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013b). A semi-automatic methodology for facial landmark annotation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 896–903. https://doi.org/10.1109/CVPRW.2013.132
[29] Suresh, H., & Guttag, J. (2021). Understanding Potential Sources of Harm

throughout the Machine Learning Life Cycle. *MIT Case Studies in Social and Ethical Responsibilities of Computing*, *1*(1), 1–13. https://doi.org/10.21428/2c646de5.c16a07bb

[30] Wäldchen, J., & Mäder, P. (2018). Machine learning for image based species identification. *Methods in Ecology and Evolution*, *9*(11), 2216–2225. https://doi.org/10.1111/2041-210X.13075

[31] Wang, J., Li, P., Ran, R., Che, Y., & Zhou, Y. (2018). A short-term photovoltaic power prediction model based on the Gradient Boost Decision Tree. *Applied Sciences (Switzerland)*, *8*(5). https://doi.org/10.3390/app8050689

[32] Yolcu Oztel, G., & Kazan, S. (2015). Virtual Makeup Application Using Image Processing Methods. *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, *1*, 401–404. www.ijseas.com