BAB III
METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi

Metode yang digunakan dalam tahap pengerjaan penelitian terdiri atas telaah literatur, analisis kebutuhan, pengerjaan program, uji coba dan evaluasi, dan konsultasi dan penulisan.

1. Telaah Literatur

   Pada tahap telaah literatur dilakukan pembelajaran mengenai keriput, segmentasi citra, pembelajaran mesin, jaringan syaraf tiruan, neuron (*artificial neuron*), *layer*, *fully convolutional network*, konvolusi, *subsampling*, *transposed convolution*, fungsi aktivasi, U-Net, augmentasi data, *Intersection over Union*. Materi pembelajaran didapat dari berbagai sumber seperti buku, referensi, situs *online*, data ataupun jurnal-jurnal penelitian yang terkait dengan penelitian yang akan dilakukan. Tahapan telaah literatur ini menjadi tahapan yang paling awal adalah proses penelitian yang akan dilakukan

2. Analisis Kebutuhan

   Pada tahap ini, dilakukan analisis terhadap metode, data dan proses apa saja yang akan digunakan pada penelitian ini. Proses analisis dimulai dari teori yang akan digunakan ke dalam program yang akan dibuat. Pada tahap ini juga akan menganalisis bentuk keluaran apa saja yang akan dihasilkan oleh program.

3. Pengerjaan Program

Pada tahap ini dilakukan proses implementasi dari rancangan yang sudah dilakukan pada tahap-tahap sebelumnya. Tahap pertama yaitu dilakukan anotasi gambar untuk memberikan informasi koordinat *ground truth* terhadap gambar. Tahap selanjutnya adalah membuat *pipeline machine learning*.

4. Uji Coba dan Evaluasi

Program yang telah diimplementasikan, diuji pada tahap ini untuk memastikan semua yang telah ditentukan dapat berjalan dengan baik. Hasil anotasi digunakan untuk menentukan nilai *Intersection over Union* (IoU). Evaluasi program dilakukan dengan menggunakan metrik evaluasi IoU.

5. Konsultasi dan Penulisan

Penulisan laporan dilakukan pada tahapan ini dengan tujuan mendokumentasikan segala bentuk proses penelitian serta menyimpulkan hasil akhir yang didapat dari pengerjaan tugas akhir ini.

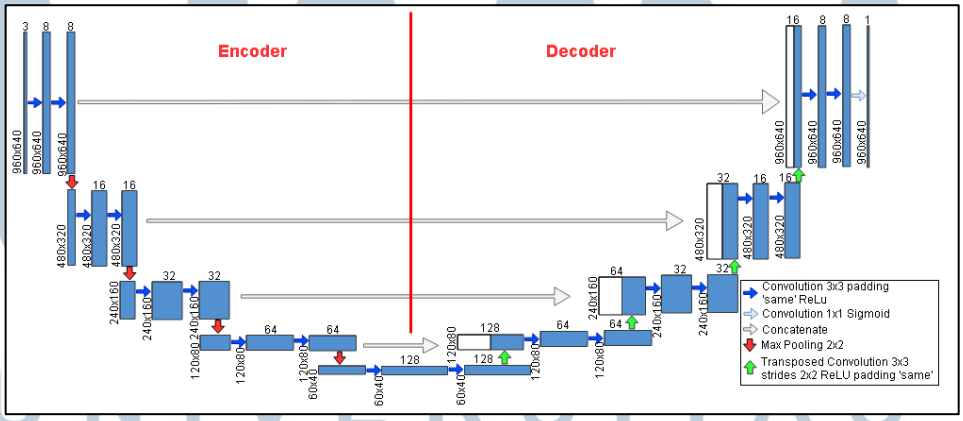
3.2 Perancangan

Perancangan implementasi arsitektur U-Net pada segmentasi citra wajah untuk deteksi keriput baik dari segi pelatihan dan *testing* akan dijabarkan dalam perancangan arsitektur U-Net dan *flowchart*.

3.2.1 Perancangan Arsitektur U-Net

Perancangan arsitektur U-Net yang akan dilakukan pada tahap pelatihan dan pengenalan terdiri dari tiga puluh dua layar yang terdiri dari *input layer*, *convolution layer*, *max pooling layer*, *transposed convolutiono layer*, dan *concatenate layer*. Alur dapat dibagi menjadi dua yaitu *contraction path encoder* dan *expansion path*

decoder. *Contraction path* (*encoder*) di mana diimplementasikan operasi konvolusi dan *max pooling layer*. Pada *encoder* ukuran gambar secara bertahap berkurang sementara kedalaman (*depth*) meningkat secara bertahap dimulai dari 960x640x3 menjadi 60x40x128. Pada tahap ini jaringan akan mempelajari informasi "APA" dalam gambar, tetapi kehilangan informasi "DI MANA". *Expansion path* (*decoder*) di mana diimplementasikan operasi *transposed convolution* dan konvolusi. Pada *decoder* ukuran gambar secara bertahap meningkat dan kedalamannya secara bertahap berkurang dimulai dari 60x40x128 menjadi 960x640x1. Secara intuitif, *decoder* memulihkan informasi "DI MANA". Untuk mendapatkan *precise locations* ketika *decoder* dilakukan penggabungan hasil dari *transposed convolution* dan *feature map* dari *encoder* yang berada pada level yang sama. Setelah digabung maka dilakukan operasi konvolusi sehingga model dapat belajar di mana lokasi yang tepat berada. Proses ini dapat disingkat menjadi Input (960x640x3) ➔ Encoder ➔ (60x40x128) ➔ Decoder ➔ Output (960x640x1). Gambar 3.1 menggambarkan perancangan arsitektur U-Net yang akan dibuat.



Gambar 3.1 Perancangan Arsitektur U-Net

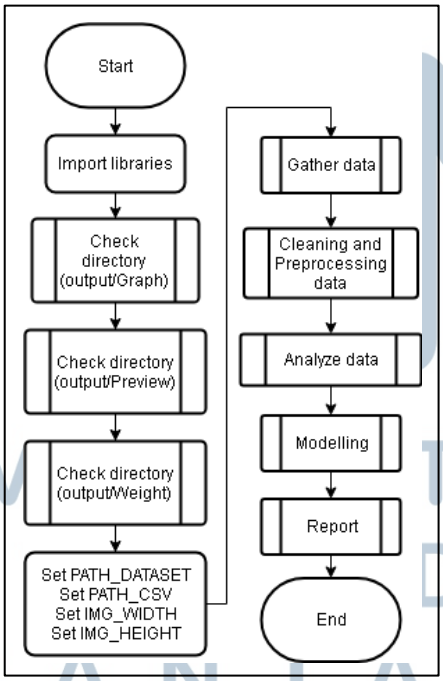
3.2.2 Flowchart

Alur dari program dapat digambarkan dalam *flowchart* seperti yang terlihat pada Gambar 3.2 hingga Gambar 3.23. *Flowchart* dibagi menjadi dua puluh dua

yaitu *flowchart* umum, *flowchart check directory*, *flowchart gather data*, *flowchart convert XML to CSV*, *flowchart cleaning and preprocessing data*, *flowchart get data*, *flowchart get image*, *flowchart get mask*, *flowchart analyze data*, *flowchart plot image and bounding box*, *flowchart plot line bounding box*, *flowchart plot image mask*, *flowchart make overlay mask*, *flowchart modelling*, *flowchart build model*, *flowchart mean iou*, *flowchart get train validation augmented*, *flowchart report*, *flowchart model validation*, *flowchart plot image predict*, *flowchart image with threshold*, dan *flowchart calculate Intersection over Union*.

A. Flowchart Umum

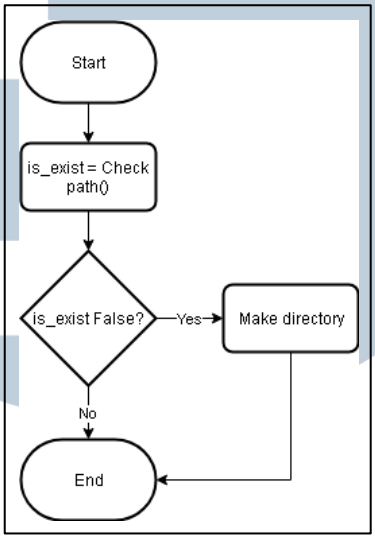
Pipeline machine learning digunakan dari awal pengambilan data sampai tes model. Proses dapat dibagi menjadi beberapa tahap yaitu persiapan program berjalan seperti cek direktori dan set variabel statik. Berikutnya adalah pengambilan dan pembersihan data (*gather and cleaning data*), *preprocessing* dan analisis data, modelling, dan laporan (*report*). *Flowchart* umum dijabarkan pada Gambar 3.2.



Gambar 3.2 *Flowchart* Umum

B. Flowchart Check Directory

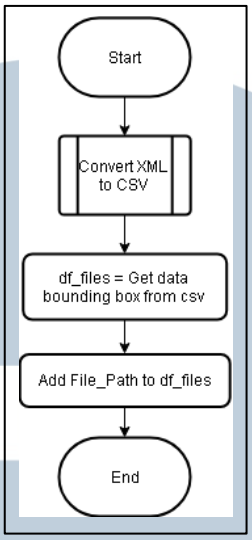
Pada proses *check directory* digunakan untuk memeriksa apakah direktori yang akan digunakan pada tahap selanjutnya seperti menyimpan hasil model, grafik, dan hasil gambaran (*preview*) sudah tersedia. Sehingga jika direktori belum tersedia maka akan membuat direktori tersebut. Pada Gambar 3.3 menjelaskan alur proses yaitu pertama dilakukan pengecekan apakah direktori sudah ada, jika belum maka akan membuat direktori tersebut.



Gambar 3.3 *Flowchart Check Directory*

C. Flowchart Gather Data

Pada tahap *gather data* digunakan untuk mengambil data input dan target yang akan diproses. Gambar 3.4 menjelaskan alur proses *gather data* yaitu pertama akan dilakukan proses mengubah bentuk anotasi dari XML ke CSV. Selanjutnya, dilakukan proses pengambilan data menjadi bentuk Pandas dataframe. Proses selanjutnya menambahkan informasi mengenai *path* dari data gambar ke dataframe.

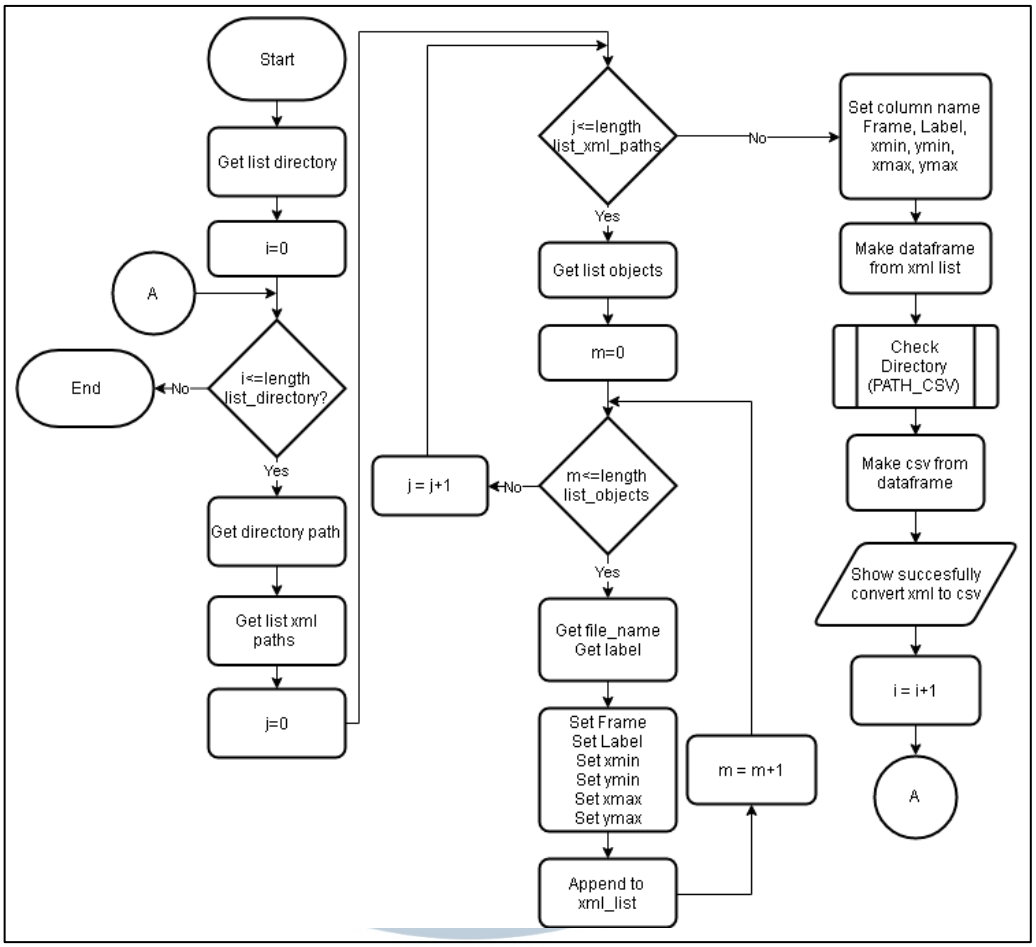


Gambar 3.4 *Flowchart Gather Data*

D. Flowchart Convert XML to CSV

Proses ini akan mengubah *file* informasi anotasi berekstensi XML menjadi CSV. Pada Gambar 3.5 menjelaskan alur proses yaitu pertama akan mendapatkan list direktori yang terdapat pada direktori *dataset*. Iterasi pertama dilakukan sebanyak list direktori, lalu mengambil *path* direktori *dataset* untuk mendapatkan list xml paths. Setelah itu iterasi kedua dilakukan sebanyak list xml paths. Proses selanjutnya mengambil list objek yang terdapat pada XML yang sudah dibaca dan dilakukan iterasi ketiga sebanyak jumlah objek. Selanjutnya mengambil nama *file*, label, dan set informasi *bounding box* xmin, ymin, xmax, ymax yang disimpan pada variabel xml_list.

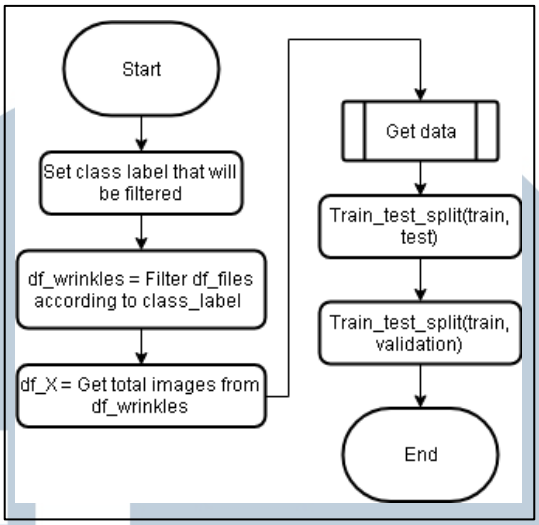
Setelah semua informasi pada setiap *file* XML didapatkan maka selanjutnya dilakukan proses membuat dataframe dari xml_list. Proses *check directory* dilakukan untuk memeriksa apakah direktori untuk menyimpan hasil konversi XML ke CSV sudah dibuat. Selanjutnya membuat *file* berekstensi CSV untuk dan disimpan pada *path* yang sudah ditentukan sebelumnya dan jika seluruh proses berhasil maka akan menampilkan pesan berhasil mengubah XML ke CSV.



Gambar 3.5 *Flowchart Convert XML to CSV*

E. Flowchart Cleaning and Preprocessing Data

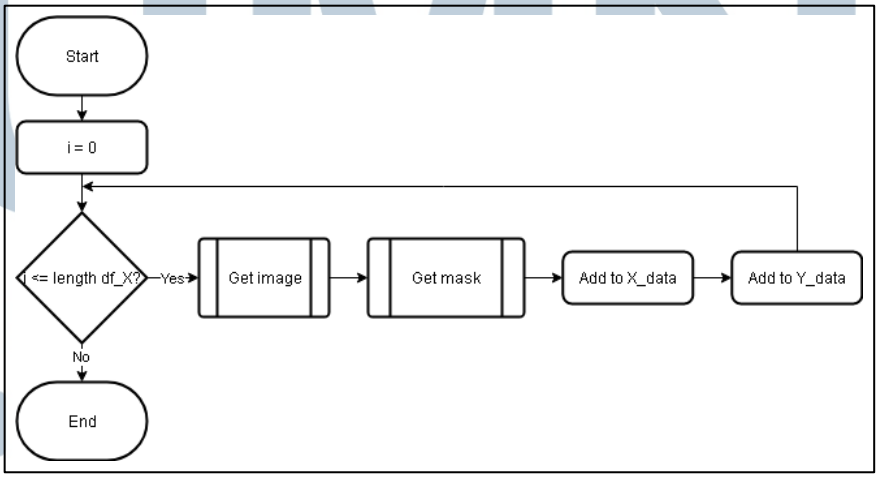
Gambar 3.6 menjabarkan alur proses untuk membersihkan dan *preprocessing* data pada dataframe yang telah dibuat sebelumnya. Tahap pertama akan set class_label sesuai dengan label pada dataframe yang akan diambil. Selanjutnya dilakukan filter pada dataframe sesuai dengan class_label yang disimpan ke dalam variabel df_wrinkles. Diambil total jumlah gambar yang terdapat pada dataframe yang disimpan ke dalam variabel df_X. Tahap selanjutnya dilakukan proses memuat data input dan anotasi menjadi bentuk gambar. Selanjutnya dilakukan pembagian jumlah data menjadi *training* dan *testing* data sebesar 20 persen dari total data dan pada data *training* dilakukan pembagian lagi menjadi *training* dan validasi data sebesar 10 persen dari jumlah data *training*.



Gambar 3.6 *Flowchart Cleaning and Preprocessing Data*

F. Flowchart Get Data

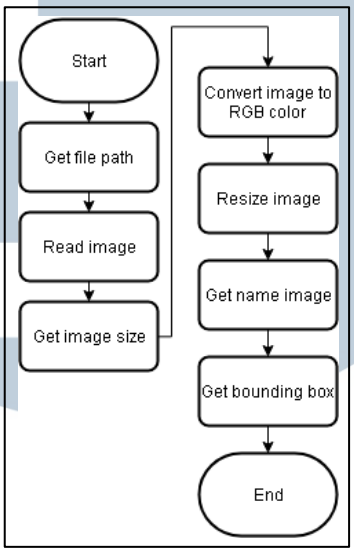
Pada proses *get data* dilakukan pengambilan data gambar input dan target dari informasi yang terdapat pada dataframe. Gambar 3.7 menjelaskan alur proses yaitu dilakukan iterasi sebanyak df_X. Selanjutnya dilakukan proses get image untuk mendapatkan nama, gambar, dan *bounding box*. Tahap berikutnya dilakukan proses get mask untuk mendapatkan gambar target sesuai informasi *bounding box* yang didapatkan sebelumnya. Data gambar yang telah didapatkan disimpan pada variabel X_data dan Y_data.



Gambar 3.7 *Flowchart Get Data*

G. Flowchart Get Image

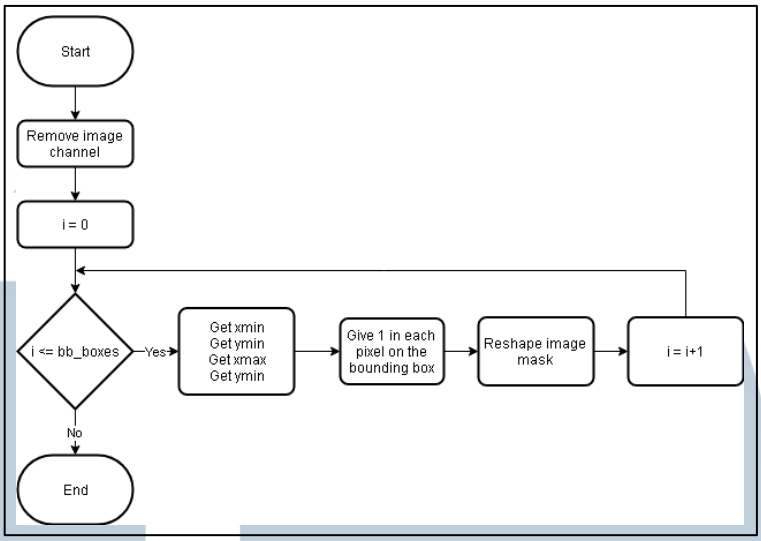
Proses *get image* dilakukan beberapa tahap dari informasi yang diterima. Gambar 3.8 menjelaskan alur proses yaitu pertama mengambil *file path* lalu dilakukan membaca gambar yang direpresentasikan ke dalam array. Selanjutnya dilakukan pengambilan informasi mengenai ukuran gambar. Dilakukan proses konversi gambar ke RGB dan *resize*. Tahap terakhir adalah mengambil informasi nama gambar dan *bounding box*.



Gambar 3.8 *Flowchart Get Image*

H. Flowchart Get Mask

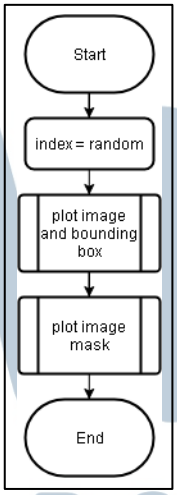
Proses *get mask* dilakukan untuk mendapatkan data target dalam bentuk gambar. Gambar 3.9 menjelaskan alur proses yaitu pertama mengambil ukuran gambar dan menghilangkan *channel* pada gambar. Dilakukan iterasi sebanyak data *bounding box* pada satu gambar, lalu mengambil nilai *bounding box* dan memberikan nilai 1 setiap piksel sesuai dengan informasi *bounding box* dan terakhir dilakukan *reshape* gambar untuk menambahkan satu *channel* pada dimensi ketiga dari array.



Gambar 3.9 *Flowchart Get Mask*

I. Flowchart Analyze Data

Proses *analyze data* digunakan untuk melihat apakah data gambar input dan target sudah sesuai. Gambar 3.10 menjelaskan alur proses yaitu pertama akan mencari indeks acak dengan nilai dari 0 sampai banyaknya data input. Selanjutnya, dilakukan dua proses yaitu *plot image and bounding box* dan *plot image mask*.

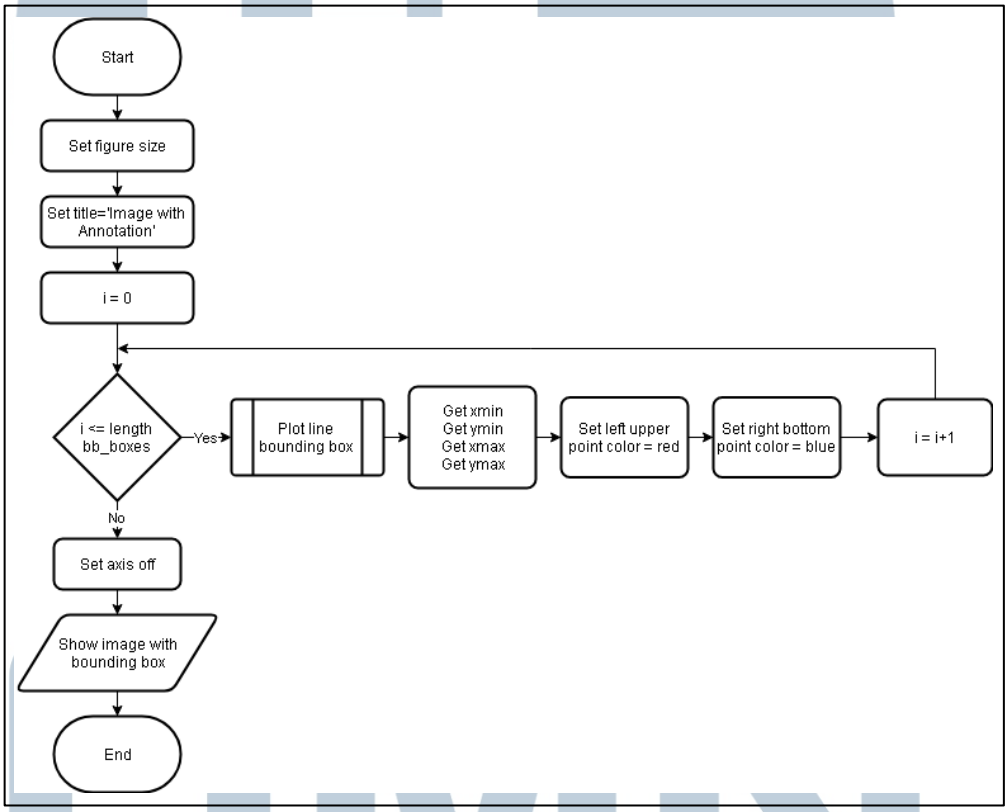


Gambar 3.10 *Flowchart Analyze Data*

J. Flowchart Plot Image and Bounding Box

Proses *plot image and bounding box* digunakan untuk menampilkan gambar dengan *bounding box* yang dimiliki pada dataframe. Gambar 3.11 menjelaskan alur proses yaitu set ukuran dan judul plot. Dilakukan iterasi sebanyak data pada

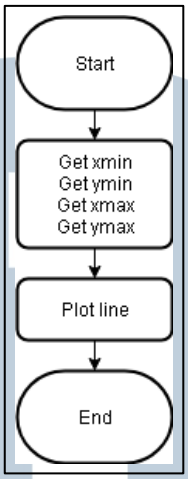
dataframe. Selanjutnya, dilakukan pembuatan garis untuk sisi *bounding box* dengan warna hijau dengan menggunakan proses *plot line bounding box*. Tahap berikutnya adalah mengambil informasi *bounding box* yaitu xmin, ymin, xmax, dan ymax. Kemudian, dilakukan pemberian warna merah pada titik kiri atas dan warna biru pada titik kanan bawah. Terakhir ditampilkan gambar input yang sudah ditambahkan informasi *bounding box*.



Gambar 3.11 *Flowchart Plot Image and Bounding Box*

K. Flowchart Plot Line Bounding Box

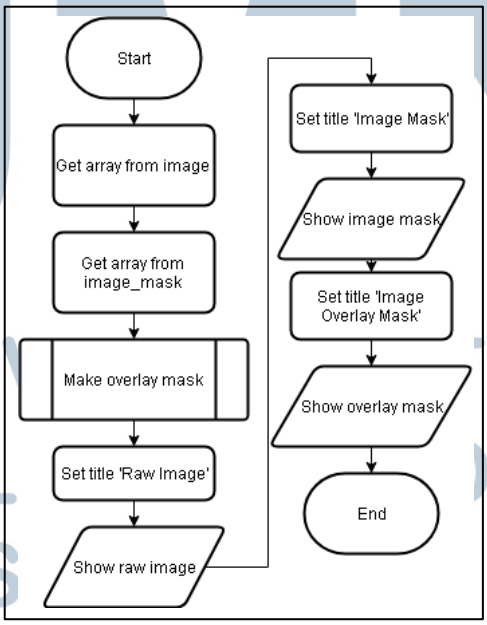
Proses *plot line bounding box* digunakan untuk membuat garis pada sisi *bounding box*. Gambar 3.12 menjelaskan alur proses pembuatan garis. Tahap pertama adalah mengambil informasi mengenai *bounding box* yaitu xmin, ymin, xmax, dan ymax. Selanjutnya ditarik garis antara titik sesuai informasi *bounding box* dengan warna yang telah ditentukan dan dengan lebar garis sebesar 2pt.



Gambar 3.12 *Flowchart Plot Line Bounding Box*

L. Flowchart Plot Image Mask

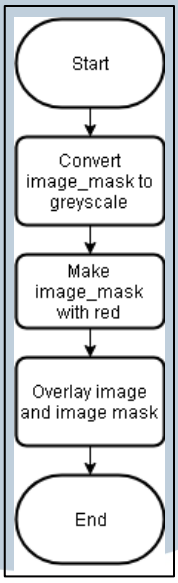
Proses ini dilakukan untuk menampilkan gambar menjadi tiga yaitu gambar input, gambar target, dan gabungan gambar input dengan target. Gambar 3.13 menjelaskan alur proses yaitu pertama dilakukan pengambilan array dari gambar input dan gambar target, kemudian dilakukan proses penggabungan gambar input dan gambar target dengan menggunakan proses make overlay mask. Selanjutnya, menampilkan gambar dengan urutan gambar input, gambar target, gambar input dengan target beserta judulnya.



Gambar 3.13 *Flowchart Plot Image Mask*

M. Flowchart Make Overlay Mask

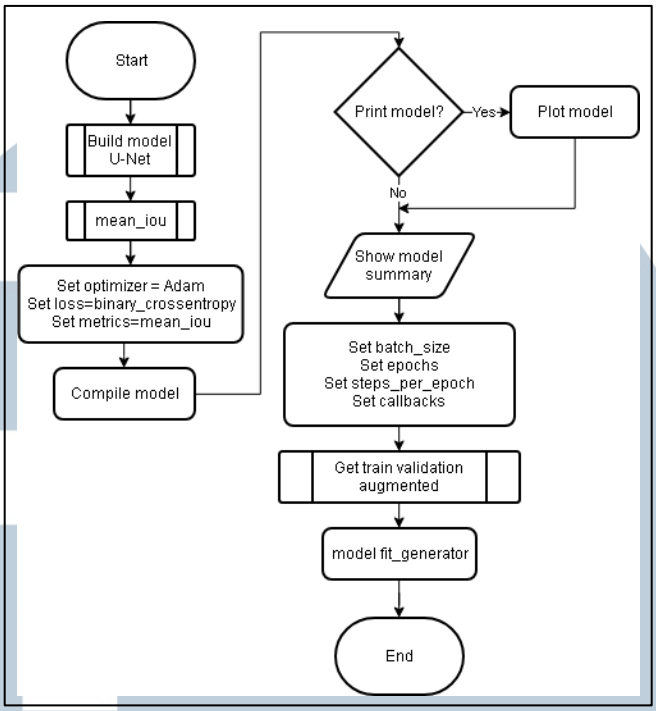
Proses ini digunakan untuk menggabungkan gambar input dengan gambar target sehingga menjadi satu gambar. Gambar 3.14 menjelaskan alur proses yaitu pertama dilakukan konversi gambar target menjadi *greyscale* lalu diganti menjadi warna merah. Setelah itu dilakukan penggabungan gambar input dan gambar target.



Gambar 3.14 *Flowchart Make Overlay Mask*

N. Flowchart Modelling

Proses modelling dimulai dengan membuat arsitektur U-Net. Selanjutnya dilakukan membuat IoU untuk metrik evaluasi. Kemudian dilakukan *compile* model dengan menggunakan *optimizer* Adam, *loss function* menggunakan Binary Crossentropy, dan metrik evaluasi menggunakan Mean_IoU. Jika memilih mengeluarkan gambar model maka akan dilakukan *generate* informasi model ke dalam bentuk Portable Network Graphic (.png). Berikutnya menampilkan kesimpulan dari model yang telah dibuat. Tahap selanjutnya yaitu menjalankan augmentasi data dengan proses get train validation generator. Tahap terakhir adalah dilakukan proses *training*. *Flowchart* dapat dilihat pada Gambar 3.15.



Gambar 3.15 *Flowchart Modelling*

O. Flowchart Build Model U-Net

Pada tahap ini digunakan tiga puluh dua *layer* untuk membuat arsitektur U-Net. Pada Gambar 3.16 menjelaskan alur proses dari pembuatan arsitektur U-Net. *Layer* pertama adalah *input layer* dengan parameternya adalah dimensi array gambar yang diterima yaitu ukuran tinggi gambar, ukuran lebar gambar, dan banyak *channel* sehingga menjadi image_height, image_width, 3. *Layer* kedua dan ketiga yaitu *layer* konvolusi dengan parameter filter = 8, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* keempat adalah *max pooling* dengan parameter ukuran pool = 2x2. *Layer* kelima dan keenam yaitu *layer* konvolusi dengan parameter filter = 16, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* ketujuh adalah *max pooling* dengan parameter ukuran pool = 2x2.

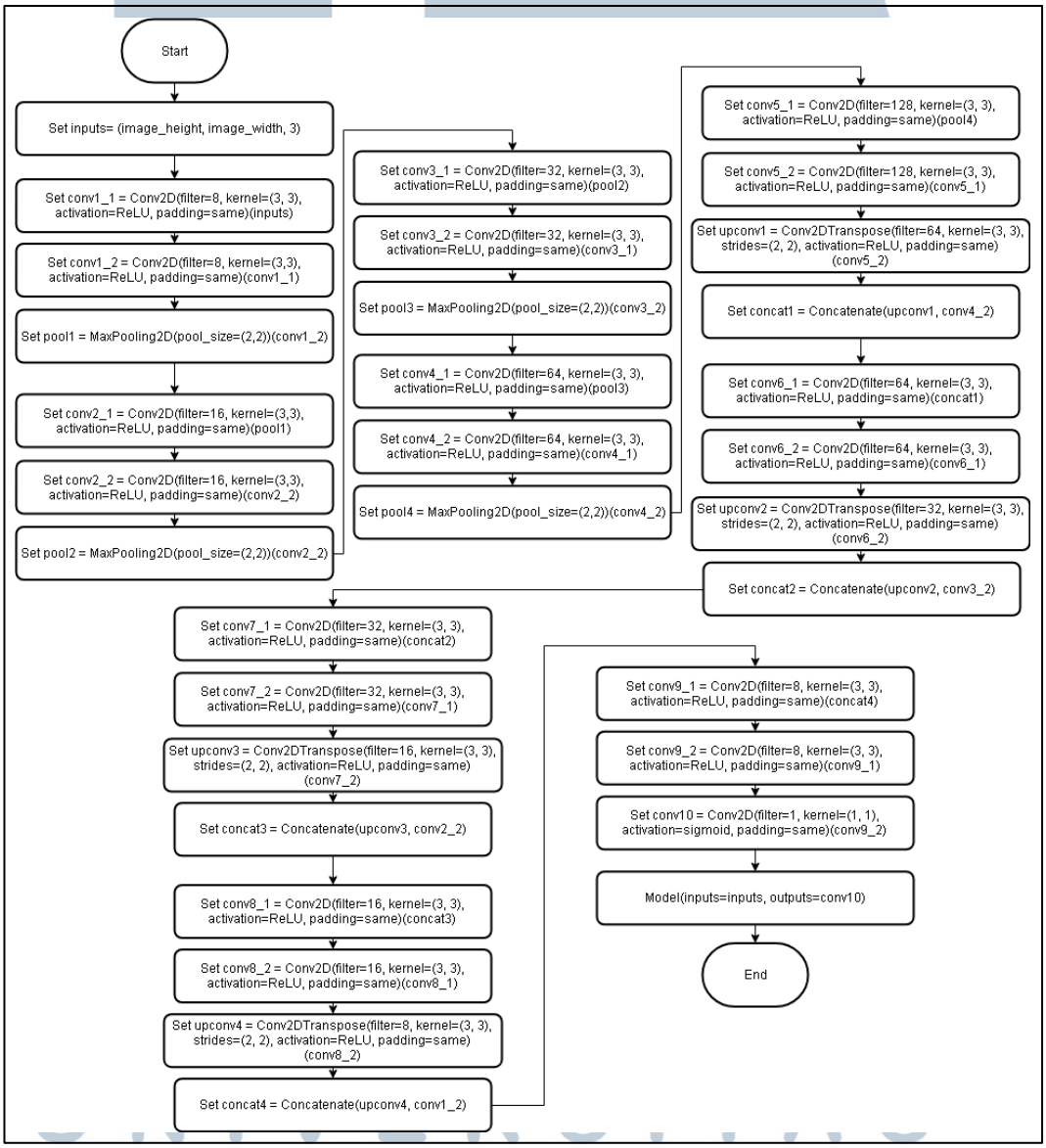
Layer kedelapan dan kesembilan yaitu *layer* konvolusi dengan parameter filter = 32, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* kesepuluh adalah *max pooling* dengan parameter ukuran pool = 2x2. *Layer* kesebelas dan

kedua belas yaitu *layer* konvolusi dengan parameter filter = 64, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* ketiga belas adalah *max pooling* dengan parameter ukuran pool = 2x2.

Layer keempat belas dan kelima belas yaitu *layer* konvolusi dengan parameter filter = 128, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* keenam belas adalah *transpose convolution* dengan parameter filter = 64, ukuran *kernel* = 3x3, *strides* = 2x2, aktivasi = ReLU, padding = same. *Layer* ketujuh belas adalah menggabungkan antara *layer* keenam belas dan kedua belas. *Layer* kedelapan belas dan kesembilan belas yaitu *layer* konvolusi dengan parameter filter = 64, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* kedua puluh adalah *transpose convolution* dengan parameter filter = 32, ukuran *kernel* = 3x3, *strides* = 2x2, aktivasi = ReLU, padding = same. *Layer* kedua puluh satu adalah menggabungkan antara *layer* kedua puluh dan kesembilan.

Layer kedua puluh dua dan kedua puluh tiga yaitu *layer* konvolusi dengan parameter filter = 32, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* kedua puluh empat adalah *transpose convolution* dengan parameter filter = 16, ukuran *kernel* = 3x3, *strides* = 2x2, aktivasi = ReLU, padding = same. *Layer* kedua puluh lima adalah menggabungkan antara *layer* kedua puluh empat dan keenam. *Layer* dua puluh enam dan kedua puluh tujuh yaitu *layer* konvolusi dengan parameter filter = 16, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* kedua puluh delapan adalah *transpose convolution* dengan parameter filter = 8, ukuran *kernel* = 3x3, *strides* = 2x2, aktivasi = ReLU, padding = same. *Layer* kedua puluh sembilan adalah menggabungkan antara *layer* kedua puluh delapan dan ketiga.

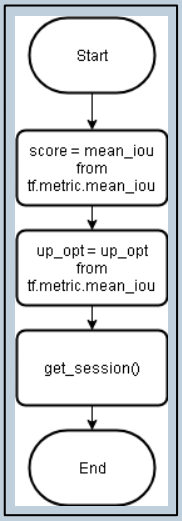
Layer ketiga puluh dan ketiga puluh satu yaitu *layer* konvolusi dengan parameter filter = 8, ukuran *kernel* = 3x3, aktivasi = ReLU, padding = same. *Layer* ketiga puluh dua yaitu *layer* konvolusi dengan parameter filter = 1, ukuran *kernel* = 1x1, aktivasi = Sigmoid, padding = same. Terakhir menjalankan model dengan input yang sudah didefinisikan dan output dari *layer* ketiga puluh dua.



Gambar 3.16 *Flowchart Build Model*

P. Flowchart Mean IoU

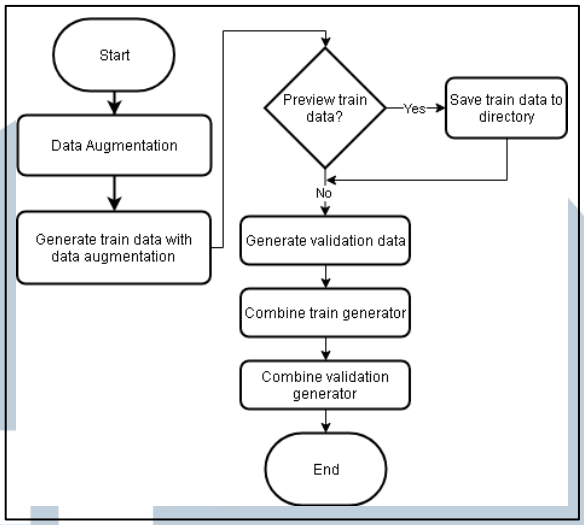
Proses *mean IoU* digunakan untuk metrik evaluasi saat melakukan pelatihan. Pada Gambar 3.17 menjelaskan alur proses yaitu mengambil skor mean_iou dan update_op dari tf.metrics.mean_iou. Setelah melakukan mendapatkan hasil, selanjutnya adalah menjalankan session karena menggunakan Tensorflow.



Gambar 3.17 *Flowchart Mean IoU*

Q. Flowchart Get Train Validation Augmented

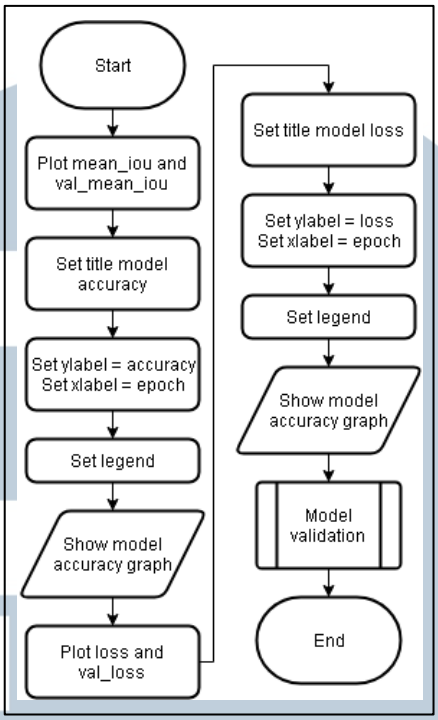
Proses *get train validation augmented* untuk mendapatkan augmentasi data. Pada Gambar 3.18 menjelaskan alur proses yaitu, pertama dijalankan persiapan augmentasi data yaitu dengan kombinasi dari rotasi sebesar 10, pergeseran lebar sebesar 0.1, pergeseran tinggi sebesar 0.1, memotong sebesar 0.2, memperbesar sebesar 0.2, membalik secara horizontal, dan dengan diisi warna konstan. Setelah itu *generate train* data dengan kombinasi augmentasi data. Terdapat pilihan, jika *preview* bernilai benar maka hasil *train* data akan disimpan dalam direktori. Selanjutnya dilakukan proses *generate* validasi data. Berikutnya menggabungkan *train* data dan target data serta menggabungkan validasi data dan target validasi data.



Gambar 3.18 *Flowchart Get Train Validation Augmented*

R. Flowchart Report

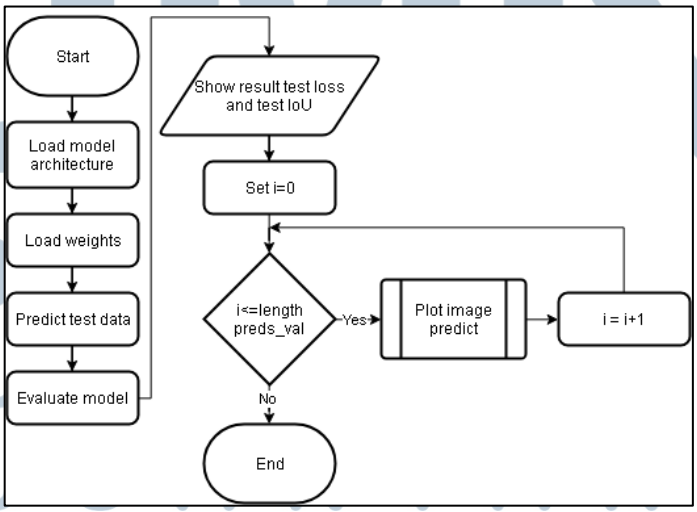
Proses ini dilakukan untuk membuat grafik antara *loss* dan validasi *loss* serta IoU dan validasi IoU. Pada Gambar 3.19 menjelaskan alur proses yaitu tahap pertama membuat IoU dan validasi IoU lalu membuat judul menjadi model *learning rate* setelah itu set y label dengan Intersection over Union dan x label dengan Epoch serta menambahkan legenda. Selanjutnya menampilkan dalam bentuk grafik. Tahap berikutnya membuat untuk *loss* dan validasi *loss* dengan membuat judul model *loss* dan set y label menjadi *loss* dan x label menjadi *epoch* serta menambahkan legenda. kemudian menampilkan dalam bentuk grafik. Tahap terakhir adalah menjalankan proses model validasi.



Gambar 3.19 *Flowchart Report*

S. Flowchart Model Validation

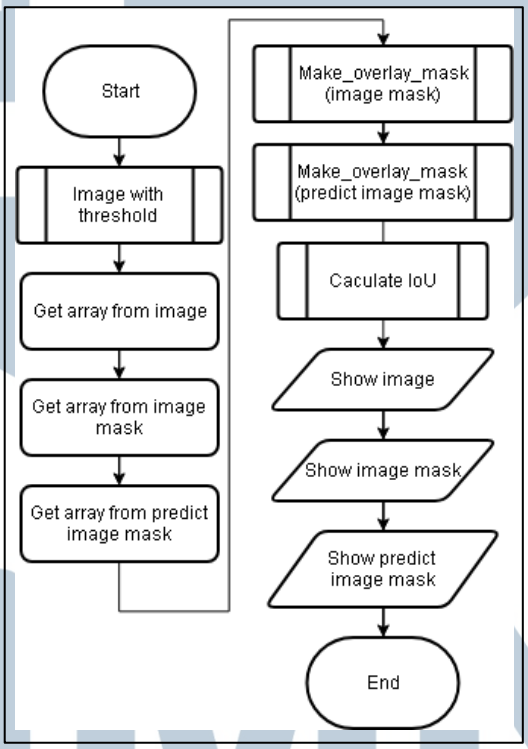
Proses ini dilakukan memuat arsitektur model serta memuat bobot. Pada Gambar 3.20 menjelaskan alur proses yaitu, dilakukan prediksi dari *testing* data. Tahap berikutnya melakukan evaluasi untuk mendapatkan hasil *loss* dan IoU dari target yang ada. Selanjutnya, menampilkan hasil *loss* dan IoU. Kemudian iterasi dilakukan sebanyak data *testing* untuk menampilkan hasil dari prediksi.



Gambar 3.20 *Flowchart Model Validation*

T. Flowchart Plot Image Predict

Proses ini digunakan untuk menampilkan hasil dari prediksi. Pada Gambar 3.21 menjelaskan alur proses yaitu, tahap pertama yaitu mengambil gambar input, gambar target, dan gambar prediksi. Selanjutnya dilakukan proses penggabungan gambar input dengan gambar target dan gambar input dengan gambar prediksi. Ditampilkan gambar input, hasil gambar input dengan target, dan gambar input dan prediksi.

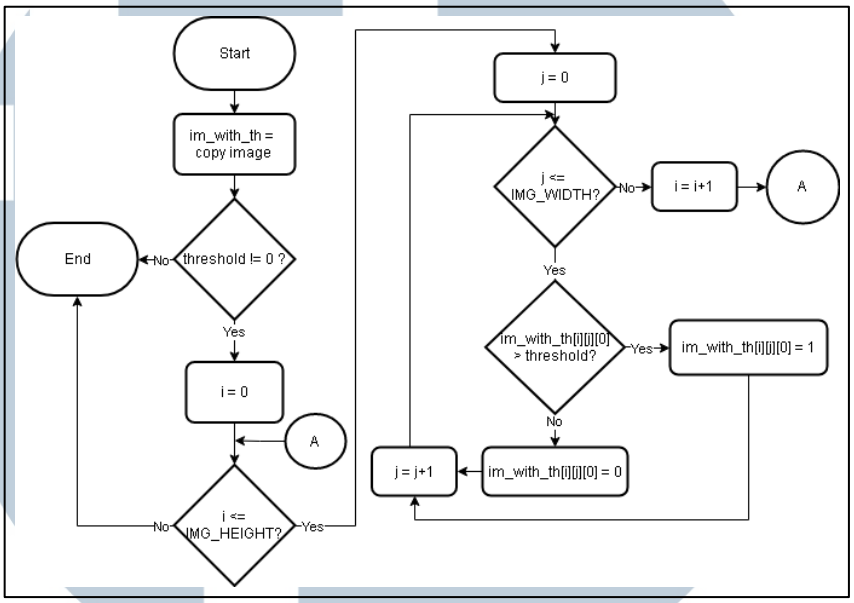


Gambar 3.21 *Flowchart Plot Image Predict*

U. Flowchart Image with Threshold

Gambar 3.22 menunjukkan alur proses gambar target dengan *threshold* yang ditentukan. Tahap pertama adalah menyalin gambar yang disimpan ke dalam variabel im_with_th. Tahap kedua melakukan cek apakah nilai *threshold* bernilai bukan 0. Jika nilai *threshold* 0 selanjutnya adalah melakukan iterasi sebanyak ukuran tinggi gambar. Berikutnya adalah iterasi sebanyak ukuran lebar gambar.

Sehingga dapat dilakukan pengecekan nilai pada piksel yang akan dibandingkan dengan nilai *threshold*. Jika nilai piksel lebih besar dari *threshold*, maka akan diberikan nilai 1, jika lebih kecil diberikan nilai 0.

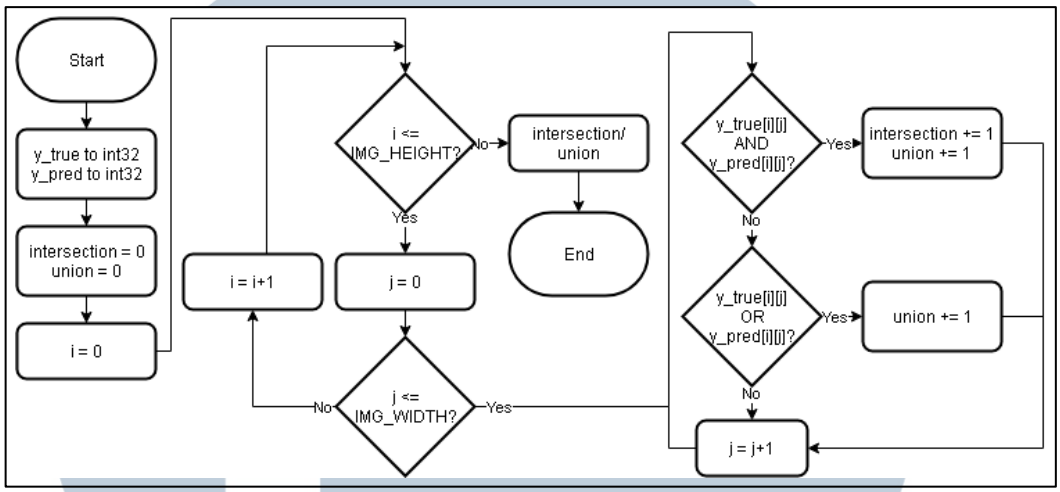


Gambar 3.22 *Flowchart Image with Threshold*

V. Flowchart Calculate Intersion over Union

Gambar 3.23 menunjukkan alur proses untuk menghitung secara manual nilai *Intersection over Union*. Tahap pertama adalah menyamakan tipe data parameter yang diterima yaitu y_true dan y_pred menjadi int32. Tahap kedua adalah melakukan deklarasi dan inisialisasi dengan nilai 0 variabel intersection dan union. Selanjutnya dilakukan iterasi sebanyak ukuran tinggi gambar. Berikutnya melakukan iterasi sebanyak ukuran lebar gambar. Sehingga mendapatkan nilai piksel yang akan dibandingkan yaitu nilai piksel y_pred dan y_true dengan menggunakan operator AND. Jika kedua nilai 1, maka variabel intersection dan union akan bertambah 1. Selanjutnya dilakukan pengecekan nilai y_true dan y_pred dengan menggunakan operator OR. Jika benar, maka variabel union akan

bertambah 1. Terakhir adalah membagi nilai intersection dengan union, sehingga didapatkan nilai IoU.



Gambar 3.23 *Flowchart Calculate Intersection over Union*