BAB III
METODE PENELITIAN

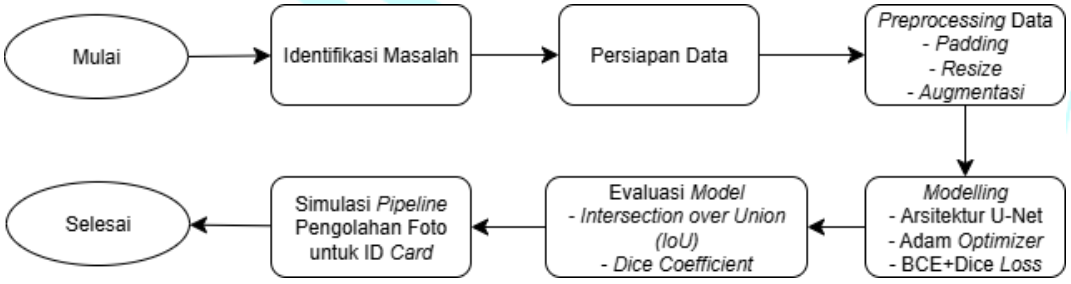
3.1 Objek Penelitian

Objek penelitian dalam studi ini adalah segmentasi gambar pada foto potret karyawan. Segmentasi dilakukan untuk memisahkan objek utama dari latar belakang secara otomatis, sehingga foto dapat diproses lebih lanjut sesuai standar pencetakan *ID Card* . Penelitian ini berfokus pada penerapan arsitektur U-Net untuk tugas segmentasi citra dalam penghapusan latar belakang foto untuk *ID Card* secara otomatis. Model U-Net yang telah dilatih kemudian digunakan dalam rangkaian proses otomatisasi (*pipeline*) pengolahan foto untuk *ID Card* berbasis *Python*. *Pipeline* ini disusun untuk mengevaluasi kemampuan model dalam mendukung penghapusan latar belakang serta alur pengolahan lanjutan.

Lokasi penelitian dilaksanakan di Perum Peruri, pada bagian Seksi Pengamanan Elektronik yang bertanggung jawab dalam pencetakan dan pengelolaan *ID Card* . Adapun waktu pelaksanaan penelitian berlangsung selama enam bulan, dari bulan November 2024 hingga April 2025.

1.2 Prosedur Penelitian

Prosedur penelitian ini disajikan pada Gambar 3.1.



Gambar 3. 1 Alur Penelitian

3.2.1 Identifikasi Masalah

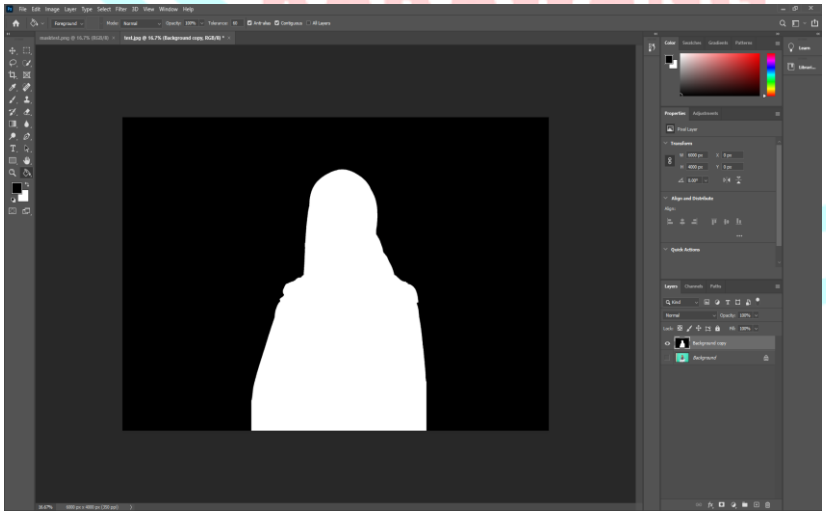
Tahap identifikasi masalah bertujuan untuk memahami kendala dalam proses pengolahan foto untuk *ID Card* yang dilakukan, seperti ketidakkonsitenan hasil, ketidak efisienan waktu dan rentan risiko kesalahan. Oleh karena itu diperlukan pengembangan model segmentasi berbasis U-Net untuk menghapus

latar belakang foto secara otomatis dan mendukung *pipeline* otomatisasi guna meningkatkan efisiensi dan konsistensi hasil.

3.2.2 Persiapan Data

Data yang digunakan dalam penelitian ini terdiri dari citra manusia dengan latar belakang beragam. Pengumpulan data dilakukan dari dua sumber, yaitu *database* internal perusahaan dan *platform* Pexels (https://www.pexels.com/). Data dari perusahaan mencakup foto-foto karyawan dalam berbagai pose formal, sedangkan dari Pexels diambil citra-citra dengan lisensi bebas pakai yang merepresentasikan variasi latar, pencahayaan, dan pose.

Setiap citra yang dikumpulkan kemudian diproses melalui tahap anotasi untuk menghasilkan label segmentasi. Proses anotasi dilakukan secara manual dengan menggunakan perangkat lunak Adobe Photoshop, dengan membuat mask biner yang membedakan antara objek dan latar belakang. Pada tahap ini, area objek (*foreground*) ditandai dengan warna putih, sedangkan latar belakang (*background*) diberi warna hitam. *Mask* biner yang dihasilkan akan digunakan sebagai *ground truth* untuk melatih model U-Net dalam mempelajari pola pemisahan antara objek dan latar belakang. Ilustrasi proses pembuatan *mask* biner ditampilkan pada Gambar 3.2.

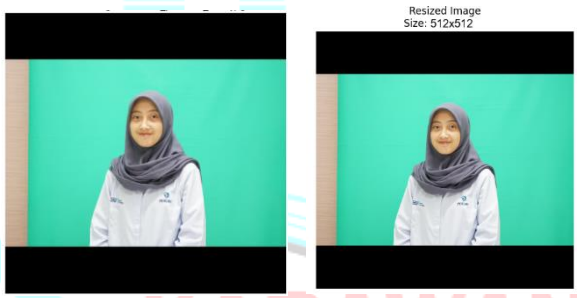


Gambar 3. 2 Proses Pembuatan *Mask* Biner Menggunakan Adobe Photoshop

3.2.3 *Preprocessing* Data

Sebelum ke tahap pemodelan, *dataset* yang telah dikumpulkan diproses melalui tahap *preprocessing* untuk memastikan data memiliki format yang seragam dan sesuai dengan kebutuhan arsitektur model U-Net. Pertama adalah penambahan *padding* untuk mengubah dimensi gambar menjadi persegi tanpa mengubah rasio aspek. *Padding* berwarna hitam (nilai piksel 0) ditambahkan pada sisi terpendek dengan menghitung selisih terhadap sisi terpanjang. Proses ini bertujuan untuk menjaga proporsi gambar saat dilakukan *resize* tanpa mendistorsi bentuk objek di dalamnya.

Selanjutnya dilakukan *resize*, yaitu penyesuaian ukuran gambar dan *mask* menjadi 512x512 piksel untuk menjaga keseragaman *input* dan kompabilitas dengan arsitektur model. Ilustrasi dari proses penambahan *padding* dan *resize* dapat dilihat pada Gambar 3.3.



(a) (b)

Gambar 3. 3 llustrasi Penambahan (a) *Padding* dan (b) *Resize*

Selanjutnya, augmentasi data diterapkan untuk meningkatkan variasi dalam *dataset* dan mengurangi risiko *overfitting* selama pelatihan model. Augmentasi mencakup rotasi acak, *flipping* horizontal, penyesuaian tingkat kecerahan (*brightness adjustment*), serta perbesaran (*zoom*) secara acak. Teknik-teknik ini diterapkan untuk memastikan model dapat mengenali pola dalam berbagai orientasi dan simetri. Ilustrasi dari proses augmentasi dapat dilihat pada Gambar 3.4, yang menunjukkan ilustrasi gambar hasil augmentasi.

(a) (b) (c) (d)

Gambar 3. 4 Ilustrasi Augmentasi Data:

(a) Gambar hasil rotasi acak; (b) Gambar hasil *flipping horizontal*, (c) Gambar hasil *brightness adjustment*, (d) Gambar hasil *zoom*

3.2.4 Modelling

Setelah tahap *preprocessing* selesai, model segmentasi berbasis arsitektur U-Net dibangun dan dilatih menggunakan *dataset* yang telah diproses. *Dataset* dibagi menjadi tiga bagian yaitu data latih (80%), data validasi (10%), dan data uji (10%). Data latih digunakan untuk melatih model, sedangkan data validasi digunakan untuk memantau kinerja model selama proses pelatihan. Data uji hanya digunakan setelah model selesai dilatih untuk mengevaluasi performanya. Tahapan *modelling* U-Net dapat dilihat pada *Algorithm 1 :*

Algorithm 1 : *Modelling* Segmentasi U-Net

Input:
- X_*train*, Y_*train* - *dataset* latih (gambar dan *mask*)
- X_*val*, Y_*val* - *dataset* validasi
- *Hyperparameters*: *learning rate* (lr), jumlah *epoch* (E), *batch size* (B)

Output: - *Model* U-Net terlatih dengan bobot terbaik

Initialize:
- *Model* U-Net dengan struktur *encoder-decoder*
- *Loss function = Binary Cross-Entropy + Dice Loss*
- *Optimizer* = Adam
- *Early stopping* berdasarkan *validasi loss*

1 ***for*** setiap blok di encoder ***do***
2 Terapkan: 2× *Conv2D → BatchNorm → ReLU*
3 Simpan *output* untuk *skip connection*
4 Terapkan *MaxPooling* (*downsampling*)
5 ***end for***
6 Terapkan *bottleneck* (2× *Conv2D + BatchNorm + ReLU*)
7 ***for*** setiap blok di *decoder* ***do***
8 *Upsampling* dengan *Conv2DTranspose*
9 Gabungkan dengan *output* dari *encoder* (*skip connection*)
10 Terapkan: 2× *Conv2D → BatchNorm → ReLU*

```
11 end for
12 Terapkan 1×1 Conv2D + sigmoid untuk menghasilkan mask output
13 for epoch = 1 hingga E do
14    for setiap batch (X, Y) dalam (X_train, Y_train) do
15       Prediksi output
16       Hitung loss (BCE + Dice)
17       Update bobot dengan Adam
18    end for
19    Hitung validasi loss
20    Simpan bobot jika val loss membaik
21    Jika val loss tidak membaik selama N epoch → aktifkan early stopping
22    Jika val loss stagnan selama M epoch → turunkan learning rate
23 end for
24 Simpan model U-Net dengan bobot terbaik
25 End
```

Model U-Net memiliki struktur *encoder-decoder*, di mana bagian *encoder* mengekstraksi fitur penting dari gambar input melalui lapisan konvolusi dan *pooling*. Bagian *decoder* merekonstruksi peta segmentasi ke resolusi asli menggunakan *transpose convolution*, dengan bantuan *skip connections* untuk mempertahankan informasi spasial yang hilang selama proses *downsampling*.

Pelatihan model dilakukan menggunakan pustaka *TensorFlow*. Fungsi *loss Binary Cross-Entropy* dan *Dice Loss*, yang berperan dalam menghitung kesalahan prediksi pada tingkat piksel sekaligus mempertimbangkan kesesuaian bentuk objek secara keseluruhan. Optimisasi bobot dilakukan secara iteratif menggunakan algoritma *Adam Optimizer* untuk meminimalkan nilai fungsi *loss*. Selain itu, diterapkan teknik *early stopping* yang menghentikan proses pelatihan jika performa model tidak mengalami peningkatan dalam beberapa *epoch* berturut-turut. elain itu, diterapkan pula *callback ReduceLROnPlateau* yang menurunkan nilai *learning rate* ketika tidak terjadi peningkatan performa, sehingga membantu proses konvergensi berjalan lebih stabil. Setelah proses pelatihan selesai, bobot model terbaik disimpan dan digunakan pada tahap pengujian.

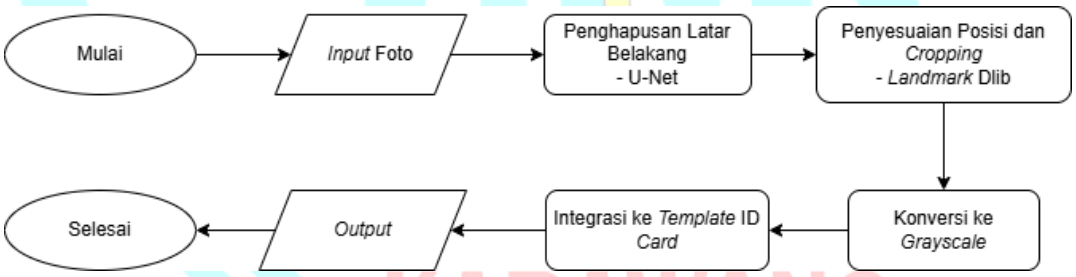
3.2.5 Evaluasi Model

Model yang telah selesai dilatih selanjutnya dievaluasi menggunakan data pengujian untuk mengukur performa segmentasi terhadap citra yang belum pernah diproses sebelumnya. Evaluasi dilakukan dengan menerapkan dua metrik utama,

yaitu *Intersection over Union* (IoU) dan *Dice Coefficient*, yang berfungsi untuk menilai sejauh mana hasil segmentasi mendekati *ground truth*.

3.2.6 Simulasi *Pipeline* Pengolahan Foto untuk *ID Card*

Pada tahap akhir, model U-Net yang telah dilatih digunakan dalam *pipeline* untuk mendemonstrasikan pengolahan foto untuk *ID Card* secara menyeluruh. *Pipeline* ini disusun untuk mensimulasikan alur pengolahan foto *ID Card* sebagai bagian dari pengujian kemampuan model segmentasi U-Net. Tujuan dari simulasi ini adalah untuk mengevaluasi apakah hasil segmentasi yang dihasilkan model dapat digunakan dalam proses pengolahan lanjutan, seperti penyesuaian posisi wajah dan integrasi ke dalam *template* desain *ID Card* . Simulasi dilakukan dalam lingkungan pemrograman *Python* untuk menunjukkan potensi penerapan model segmentasi dalam pengolahan gambar *ID Card* yang lebih efisien dan konsisten. *Pipeline* mencakup beberapa tahap seperti ditunjukan pada Gambar 3.5.



Gambar 3. 5 *Pipeline* Pengolahan Foto untuk *ID Card*

a. *Input* Foto

   Foto karyawan dimasukkan sebagai *input* untuk diproses secara dalam *pipeline*.

b. Penghapusan Latar Belakang

   Citra *input* diproses oleh model U-Net untuk memisahkan objek utama dari latar. Hasil segmentasi berupa *mask* biner yang digabungkan ke citra asli menjadi citra RGBA dengan latar belakang transparan. Proses penghapusan latar belakang ditunjukkan pada *Algorithm* 2 berikut.

Algorithm 2 : *Penghapusan Latar Belakang*

Input : Citra RGB
Output : Citra RGBA

1 *Resize* citra ke 512×512 piksel
2 Prediksi *mask* segmentasi dengan model U-Net
3 *Resize mask* ke ukuran asli
4 Haluskan *mask*
5 Normalisasi nilai *mask* dengan *threshold*
6 Bentuk *alpha channel* dari *mask*
7 Gabungkan *alpha* dengan citra asli menjadi gambar RGBA

c. Penyesuaian Posisi dan *Cropping* Citra

Citra hasil segmentasi diperiksa orientasinya dengan mendeteksi *landmark* wajah yaitu koordinat mata kiri dan kanan, menggunakan pustaka Dlib. Dlib memanfaatkan *file shape_predictor_68_face_landmarks.dat* untuk mendeteksi 68 titik *landmark* wajah secara otomatis, seperti ditunjukkan pada Gambar 3.6.

Gambar 3. 6 68 Titik *Landmark* Wajah Dlib (Das et al., 2021)

Penyesuaian dilakukan agar arah wajah seragam menghadap ke kanan, sesuai standar *template ID Card* . Jika hasil deteksi menunjukkan sudut kemiringan garis mata melebihi ambang yang ditentukan, maka citra akan dicerminkan secara horizontal agar posisi kepala lurus ke arah kanan. Tahapan penyesuaian posisi ini dijelaskan pada *Algorithm* 4.

Algorithm 4 : Penyesuaian Posisi

Input: Citra RGB
Output : Citra

1 Deteksi wajah → deteksi *landmark*
2 Ambil koordinat mata kiri dan mata kanan
3 Hitung sudut kemiringan garis mata
4 Jika sudut kemiringan di luar ambang, lakukan *flip* horizontal.

Setelah orientasi posisi kepala disesuaikan, citra dipotong otomatis agar proporsi wajah sesuai standar. Proses *cropping* menggunakan koordinat *landmark* wajah Dlib (dagu, alis, hidung) untuk menghitung area potong yang seragam pada setiap gambar. Tahapan proses *cropping* ditunjukkan pada *Algorithm* 5.

Algorithm 5: *Cropping*

Input: Citra RGBA
Output : Citra hasil cropping

1 Deteksi *landmark* wajah
2 Hitung tinggi wajah = dagu - alis
3 Tentukan tinggi *crop* = faktor perbesaran × tinggi wajah
4 Hitung batas kiri-kanan: hidung ± 0.45 × tinggi *crop*
5 Hitung batas atas-bawah: hidung ± 30% & 70% tinggi *crop*
6 Potong citra sesuai koordinat

d. Konversi ke *Grayscale*

Citra selanjutnya dikonversi ke *grayscale* untuk memastikan konsistensi warna dan kontras sesuai standar *template ID Card* . Tahapan *grayscale* ditunjukan pada *Algorithm 3.*

Algorithm 3 : *Konversi ke Grayscale*

Input : Citra hasil cropping
Output : Citra grayscale + alpha channel

1 Konversi RGB ke *grayscale*
2 Gabungkan *channel grayscale* dengan *alpha*

e. Integrasi ke *Template*

Gambar yang telah diproses di-*overlay* ke *template ID Card* menggunakan teknik *layering* untuk memastikan skala dan posisi sesuai. *Template ID Card* yang digunakan berukuran 638 piksel × 1004 piksel dengan area *bounding box* berkoordinat titik kiri atas di (312, 383) dan kanan bawah di (544, 999). Area inilah yang menjadi acuan penempatan hasil *overlay* agar proporsi wajah tetap seragam. Proses integrasi gambar ke *template* ditunjukkan pada *Algorithm 6.*

Algorithm 6 : *Integrasi ke Template*

Input: Citra RGBA hasil cropping & grayscale
Output : Foto final di template

1 Deteksi piksel terluar kepala (*mask alpha*)
2 Hitung lebar kepala → tentukan faktor skala
3 *Resize overlay* → samakan dengan lebar *bounding box*
4 Hitung *offset* pusat kepala ke *bounding box*
5 Tempel *overlay* ke *template* di posisi *bounding box*
6 Potong kelebihan area jika ada
7 Gabungkan dengan kanal *alpha*

---

f. Output

Hasil akhir berupa foto yang telah terintegrasi ke dalam *template ID Card* disimpan dalam format *file digital* yang siap dicetak.