UCAPAN TERIMAKASIH

Puji dan syukur kita panjatkan atas kehadiran allah SWT yang mana telah melimpahkan rahmat dan karunia-Nya kepada penulis sehingga dapat menyelesaikan tugas akhir ini yang berjudul **"EKSTRAKSI JALAN MENGGUNAKAN *DEEP LEARNING* DENGAN MODEL U-NET"** Tugas akhir ini disusun untuk memenuhi salah satu syarat mencapai gelar sarjana pada program studi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom.

Penulis menyadari bahwa tugas akhir ini tidak mungkin terselesaikan tanpa adanya dukungan, bantuan, bimbingan, dan nasehat dari berbagai pihak selama pengerjaan tugas akhir ini. Pada kesempatan ini penulis menyampaikan terima kasih setulus-tulusnya kepada:

1. Kedua Orang tua penulis, orang yang hebat yang selalu menjadi penyemangat saya sebagai sandaran terkuat dari kerasnya dunia. Yang tidak henti-hentinya memberikan kasih sayang dengan penuh cinta dan selalu memberikan motivasi. Terimakasih selalu berjuang untuk kehidupan saya, Terima kasih untuk semuanya berkat do'a dan dukungan mama dan papa saya berada di titik ini. sehat selalu dan hiduplah lebih lama lagi Mama & Papa harus selalu ada disetiap perjalanan & pencapaian hidup saya.
2. Ibu Dr. Sofia Naning Hertiana, S.T., M.T selaku pembimbing utama penulis yang selalu menyisihkan waktunya untuk membimbing dan memberikan masukan serta dorongan untuk membantu penulis menyempurnakan tugas akhir ini.
3. Ibu Sussi, S. Si., M. T. selaku pembimbing kedua dan wali dosen penulis yang selalu menyisihkan waktunya untuk membimbing dan memberikan masukan serta dorongan untuk membantu penulis menyempurnakan tugas akhir ini.
4. Azhary Abdullah Manab, S.T selaku saudara dari penulis yang selalu memberi motivasi dan dukungan kepada penulis.
5. Sri Rahayu selaku saudari dari penulis yang selalu memberi motivasi dan semangat kepada penulis.
6. Aulia Priyangga selaku kerabat dari penulis yang selalu memberi solusi, motivasi dan dukungan kepada penulis
7. Navri Yulenny, S.H., M.H selaku kerabat dari penulis yang selalu memberi semangat kepada penulis.
8. Bintang Septiawan selaku kerabat dari penulis yang telah memberi solusi kepada penulis dalam penulisan dokumen ini.

9. kepada teman-teman seperjuangan yang telah memberikan dukungan dan motivasi dengan cara mereka.

Semoga Allah SWT selalu memberikan rahmat serta karunia-Nya atas apa yang telah kalian berikan kepada penulis. Penulis berharap dengan adanya tugas akhir ini juga dapat memberikan bantuan ilmu bagi pembaca pada umumnya.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

CNN	: *Convolutional Neural Network*
FCN	: Fully Convolutional Network
GIS	: *Geographic Information System*
OBIA	: *Object Based Image Analysis*
RESNET	: *Residual Neural Network*

BAB 1
USULAN GAGASAN

1.1 Latar Belakang Masalah

Seiring dengan perkembangan zaman, ekstraksi jalan sangat diperlukan. Para peneliti berlomba-lomba melakukan ekstraksi jalan dengan berbagai cara dan model. Ekstraksi jalan adalah salah satu tugas yang mendasar di bidang penginderaan jauh. Ekstraksi jalan ini memiliki berbagai aplikasi seperti navigasi jalan otomatis, kendaraan tak berawak, perencanaan kota, dan pembaruan informasi geografis. Berbagai metode telah diusulkan untuk mengekstrak jalan dari citra penginderaan jauh dalam beberapa tahun terakhir. Sebagian besar metode ini dapat dibagi menjadi dua kategori: ekstraksi area jalan dan ekstraksi garis tengah jalan. Ekstraksi area jalan dapat menghasilkan pelabelan jalan tingkat piksel, sedangkan ekstraksi garis tengah jalan bertujuan untuk mendeteksi kerangka jalan.[1]

Ekstraksi jalan pada era sekarang masih memiliki beberapa kekurangan pada berbagai aspek tertentu. Ada beberapa faktor penghalang ekstraksi jalan otomatis seperti cakupan latar belakang, fitur di lingkungan jalan, komplikasi seperti kendaraan di jalan, jembatan, dan bayangannya. Apalagi masalah yang tercipta oleh bayangan, awan, kesalahan sensor, dll[2] Selain itu, pada ekstraksi jalan terdapat beberapa faktor yang menyebabkan ekstraksi jalan memiliki kekurangan seperti *update* informasi jalan yang dilakukan dengan selang waktu yang lama sehingga menyebabkan perubahan-perubahan yang ada pada jalanan menjadi tidak sesuai. Aplikasi *map* yang ada sekarang masih melakukan ekstraksi jalan dengan metode manual sehingga untuk memperbaharui informasi jalan membutuhkan banyak waktu dan biaya.

Ekstraksi jalan dengan menggunakan *Deep learning* dengan model U-Net diharapkan dapat menjadi solusi untuk mengatasi masalah ekstraksi jalan. Ekstraksi dengan metode semi-otomatis akan lebih mempersingkat waktu dan dapat membantu proses ekstraksi dengan menggunakan biaya yang lebih sedikit. Ekstraksi jalan dengan metode otomatis ini juga akan membantu dalam proses pembaharuan sehingga *update* informasi jalan dapat dilakukan dalam selang waktu yang lebih cepat sehingga informasi jalan bisa selalu *real*.

1.2 Informasi Pendukung Masalah

Mengekstrak jalan secara manual dari citra digital, meskipun memiliki akurasi yang tinggi, namun dari segi waktu dan biaya tidak hemat biaya, terutama ketika gambarnya sangat kompleks. Karena itu, sangat mendesak untuk mengembangkan jalan semi-otomatis/otomatis metode ekstraksi. Dalam literatur, metode otomatis menyiratkan proses yang sepenuhnya otomatis.[2]

Pada saat ini, perkembangan ekstraksi citra penginderaan jauh sudah sangat pesat. OBIA (*Object Based Image Analysis)* termasuk salah satu perkembangannya. Metode OBIA adalah pendekatan yang proses klasifikasinya mempertimbangkan aspek spektral dan aspek spasial objek. Selain metode OBIA, Digitasi juga merupakan metode pengubahan gambar analog menjadi gambar digital yang menggunakan alat digitasi. Dalam GIS (*Geographic Information System*) objek-objek seperti rumah, jalan dan lain-lain akan diubah menjadi bentuk garis digital. Dengan adanya digitasi ini dapat mempermudah proses ekstraksi dari segi waktu.

1.3 Analisis Umum

1.3.1 Aspek Ekonomi

Dari aspek ekonomi, dapat dilihat bahwa ekstraksi jalan dengan menggunakan metode *Deep learning* model U-Net lebih terjangkau dibandingkan menggunakan metode atau teknik *remote sensing* lainnya yang akan menambah biaya karena menggunakan metode manual dengan bantuan tenaga manusia.

1.3.2 Aspek *Manufacturability*

Dari aspek *Manufacturability*, dapat dilihat bahwa ekstraksi dengan menggunakan *Deep learning* model U-Net akan dilakukan pada aplikasi dan akan memproduksi hasil ekstraksi dengan mudah menggunakan dataset.

1.3.3 Aspek Teknologi

Dari aspek Teknologi, dapat dilihat bahwa dengan menggunakan teknologi pengambilan citra orthophoto yang diambil tegak lurus dari udara dengan menggunakan drone akan sangat mempermudah pengambilan citra orthophoto tanpa harus melihat langsung ke lokasi

1.4 Kebutuhan yang Harus Dipenuhi

Berdasarkan hasil analisis yang telah disebutkan di atas, didapatkan usulan untuk menggunakan *Deep learning* model U-Net untuk mengekstraksi jalan dengan

menggunakan dataset. Dataset yang digunakan merupakan dataset yang diproses dengan menggunakan teknik OBIA dan digitasi. Kedua dataset tersebut diharapkan dapat menghasilkan akurasi yang baik sehingga dapat memenuhi kebutuhan dari permasalahan waktu pada proses ekstraksi jalan. Selain itu, *software* yang digunakan juga harus mendukung untuk melakukan segmentasi dan klasifikasi dengan algoritma yang dipilih sehingga mendapatkan akurasi yang baik dan pada bagian *Deep learning* harus menggunakan Bahasa pemrograman python dan server yang memadai seperti github dan lain-lain.

1.5 Solusi Sistem yang Diusulkan

Solusi untuk tugas akhir ini, yaitu mendapatkan hasil skala dari citra orthophoto yang diambil tegak lurus dari permukaan bumi dengan resolusi yang sangat tinggi. Solusi lain yang akan penulis lakukan yaitu akan memperbanyak dataset sehingga hasil ekstraksi lebih akurat dengan model yang telah ditentukan. Dengan dataset yang baik, maka model dapat lebih mempelajari sistem dengan lebih detail sehingga akan menghasilkan hasil ekstraksi yang memiliki akurasi yang baik.

1.5.1 Karakteristik Produk

1.5.1.1 OBIA (*Object Based Image Analysis*)

OBIA (*Object Based Image Analysis)* adalah salah satu perkembangan ekstraksi citra penginderaan jauh. Metode OBIA adalah pendekatan yang proses klasifikasinya mempertimbangkan aspek-aspek pada citra seperti aspek spectral. Metode OBIA tidak hanya bergantung pada nilai spektral saja tapi juga mampu mengoptimasi aspek spasial dalam citra satelit sesuai dengan unsur interpretasi seperti bentuk, ukuran tekstur dan informasi kontekstual lainnya. OBIA mempunyai beberapa tahap pemrosesan yang melibatkan segmentasi, dan klasifikasi objek. Segmentasi adalah tahap pertama dalam OBIA, pada tahap ini citra akan dibagi menjadi segmen-segmen. setelah melakukan segmentasi selanjutnya segmen-segmen yang dihasilkan akan diklasifikasikan kedalam kelas yang relevan.

Dalam proses segemntasi menggunakan algoritma *multiresolution* dengan tiga parameter yaitu, *scale, shape,* dan *compactness.* Cara kerja algoritma *multiresolution* yaitu objek akan dibagi kedalam bentuk segmen-segmen berdasarkan bentuk, warna, dan tekstur sehingga di sebut dengan proses segmentasi. Pada proses klasifikasi digunakan algoritma *K-Nearest Neighbor* dengan input atribut *mean* dan *standard deviation. K-Nearest Neighbor* merupakan sebuah algoritma untuk mengklasifikasikan

objek berdasarkan data *training* yang mempunyai jarak yang paling dekat dengan objek tersebut. Pada proses klasifikasi citra akan di proses dengan mempertimbangkan nilai rata-rata dan simpangan baku pada citra.

1.5.1.2 *Deep learning* (Model U-Net)

Teknik ekstraksi jalan dengan cara otomatis ini sangat sering digunakan dengan menggunakan pendekatan otomatis. Teknik otomatis berguna sangat berguna dalam aplikasi waktu nyata dan tidak memerlukan campur tangan manusia tidak seperti semi-otomatis. Teknik otomatis ini lebih banyak menggunakan *Deep learning* dengan beberapa metode yang sering digunakan seperti metode U-Net, FCN dan Resnet.

U-Net merupakan salah satu arsitektur model *Deep learning* yang digunakan khusus untuk tugas segmentasi gambar. U-Net juga merupakan model CNN (Convolution Neural Network) yang sudah banyak dilakukan pada segmentasi gambar pada dunia medis. Dilihat dari arsitektur U-Net pada gambar 1.5.2.2, *input* dan *output*nya menggunakan struktur *encoder-decoder* yang simetris yaitu dengan menyatukan fitur-fitur spasial dari lapisan encoder dan decoder untuk mendapatkan hasil yang akurat. Model U-Net ini memiliki empat encoder yang akan digunakan untuk proses downsampling citra dan memiliki empat decoder yang akan digunakan untuk proses upsampling. Pada proses downsampling terdapat proses convolusi yang akan mengalikan dan menjumlahkan vector dan terdapat proses max pool untuk mencari nilai terbesar dari beberapa convolutional layer. Pada proses upsampling terdapat proses up conv yang akan mengembalikan citra yang telah di downsampling menjadi bentuk asli namun dalam bentuk segmentasi gambar.

Model U-Net memiliki beberapa keunggulan yaitu, arsitektur dirancang khusus untuk tugas segmentasi gambar, mempertahankan resolusi spasial yang sama pada *input* dan *output*, dan menggunakan jumlah parameter lebih sedikit dan efektif pada dataset pelatihan yang terbatas. Meskipun U-Net mempunyai banyak keunggulan, U-Net juga memiliki beberapa kekurangan yang harus diperhatikan yaitu, penggunaan sumber daya komputasi yang tinggi, memiliki masalah dengan klasifikasi piksel yang ambigu, sensitivitas terhadap variasi skala dalam gambar, interpretasi yang lebih sulit secara intuitif.

1. Fitur Utama

Model U-Net yang dapat menunjukkan hasil dari ekstraksi jalan dan bukan jalan. Selain itu dapat juga menunjukkan hasil keakuratan ekstraksi jalan dan bukan jalan dari citra yang sudah diterapkan.

2. Fitur Dasar

Hasil dari Teknik OBIA didapatkan ekstraksi jalan dan bukan jalan yang berbentuk *masking*. Hasil *masking* dari teknik OBIA adalah dataset untuk digunakan pada model U-Net.

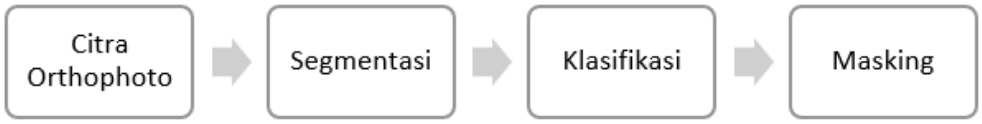
3. Fitur Tambahan

Hasil ekstraksi menggunakan OBIA yang dilakukan secara manual atau ada campur tangan manusia akan dibandingkan dengan hasil ekstraksi menggunakan *Deep learning* model U-Net yang dilakukan tanpa ada campur tangan manusia.

1.5.2 Skenario Penggunaan

1.5.2.1 OBIA (*Object Based Image Analysis)*

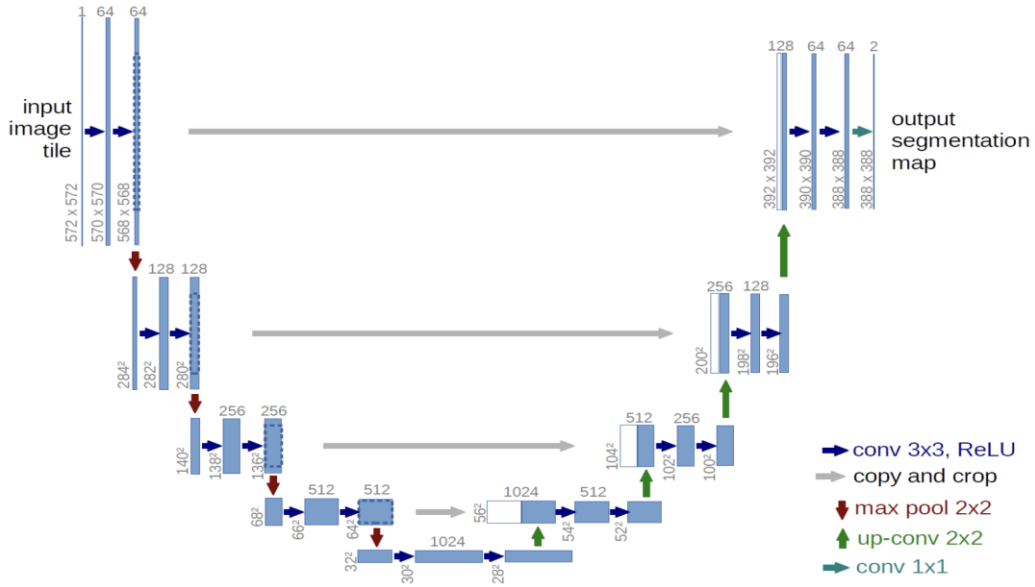
Berikut merupakan tahap pemrosesan citra dengan metode OBIA untuk menghasilkan *masking* sebagai dataset.



Gambar 1.5.2.1 Proses OBIA

1.5.2.2 U-Net

Dilihat dari arsitektur U-Net pada gambar 1.5.2.2, *input* dan *output*nya menggunakan struktur *encoder-decoder* yang simetris yaitu dengan menyatukan fitur-fitur spasial dari lapisan encoder dan decoder untuk mendapatkan hasil yang akurat. Model U-Net ini memiliki empat encoder yang akan digunakan untuk proses downsampling citra dan memiliki empat decoder yang akan digunakan untuk proses upsampling. Pada proses downsampling terdapat proses convolusi yang akan mengalikan dan menjumlahkan vector dan terdapat proses max pool untuk mencari nilai terbesar dari beberapa convolutional layer. Pada proses upsampling terdapat proses up conv yang akan mengembalikan citra yang telah di downsampling menjadi bentuk asli namun dalam bentuk segmentasi gambar.



Gambar 1.5.2.2 Arsitektur Model U-Net

Source: N. More, R. Lalla, R. Memon and V. Nikam, "Extraction of Road Network from Satellite images using Efficient Net," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 7, pp. 2-6, 2020.

1.6 Kesimpulan dan Ringkasan CD-1

Kesimpulan dari penulisan dokumen ini adalah karena adanya hambatan yang ada pada citra orthophoto, seperti adanya struktur kontekstual (bayangan, kendaraan, vegetasi, dan pepohonan) oleh karena itu, dilakukan ekstraksi jalan untuk mengatasi masalah tersebut. Dalam penulisan dokumen ini ekstraksi jalan dilakukan dengan metode *Deep learning* dengan model U-Net dan menggunakan metode OBIA untuk proses datasetnya. Metode OBIA dilakukan untuk mendapatkan *masking* kemudian *masking* tersebut diproses ke dalam metode *Deep learning* dengan model U-Net untuk mendapatkan hasil akurasi.

BAB 2
DESAIN KONSEP SOLUSI

2.1 Spesifikasi Produk

Produk yang akan dibuat akan menghasilkan hasil ekstraksi dengan menggunakan dataset dengan metode OBIA (*Object Based Image Analysis*) dan digitasi otomatis. Dengan adanya teknik OBIA dapat memudahkan untuk proses labelling secara otomatis dibanding dengan labelling secara manual sehingga efisiensi dalam segi waktu. Dataset dengan menggunakan OBIA dan digitasi otomatis tersebut dimasukkan ke dalam model *Deep learning* dengan model U-Net. Kedua dataset tersebut akan diekstraksi lalu diuji akurasi dan *loss*nya.

2.1.1 Spesifikasi #1

Pada produk ini dilakukan proses ekstraksi dengan menggunakan dataset dengan metode OBIA. Dataset yang dibuat dengan menggunakan OBIA diambil dengan proses segmentasi dan klasifikasi secara otomatis sehingga lebih mempersingkat waktu. Pemrosesan data tersebut disebut labelling. Dataset OBIA tersebut berbentuk *masking* yang berukuran 1024x1024 *piksel*. Dataset OBIA tersebut akan dimasukkan ke dalam *Deep learning* dengan model U-Net untuk menguji nilai akurasi dan *loss* dari hasil ekstraksi.

2.1.2 Spesifikasi #2

Pada produk ini dilakukan proses ekstraksi dengan menggunakan dataset digitasi. Dataset digitasi tersebut berbentuk *masking* yang nantinya akan diekstraksi dengan *Deep learning* dengan ukuran dataset 256x256 *piksel*. Dataset tersebut akan dimasukkan ke dalam codingan *Deep learning* dengan model U-Net untuk diekstraksi. Hasil ekstraksi akan diuji untuk mencari nilai akurasi dan *loss*nya.

Tabel 2.1.2 Spesifikasi 2

No	Hal	Rincian
1	Dataset OBIA	Dataset OBIA berasal dari citra orthophoto yang kemudian dilakukan proses segmentasi dan klasifikasi. Dataset ini memiliki bentuk hasil klasifikasi dan labelling yang dilakukan secara otomatis dengan ukuran 256x256 *piksel*.

2	Dataset digitasi	Dataset digitasi berasal dari citra orthophoto yang digitasi secara otomatis dengan bentuk hasil *masking* yang berukuran 256x256 *piksel*.

2.2 Verifikasi

Berdasarkan spesifikasi yang telah dijelaskan di atas akan dilakukan verifikasi dan dijelaskan proses dan cara kerjanya. Bagian spesifikasi tersebut kemudian akan dijelaskan pada tabel di bawah ini.

2.2.1 Verifikasi Spesifikasi 1

Tabel 2.2.1 Verifikasi Spesifikasi 1

Hal	Ekstraksi dengan menggunakan dataset OBIA
Rincian	Ekstraksi jalan akan dilakukan dengan menggunakan dataset yang diambil dari teknik segmentasi dan labelling OBIA
Metode Pengujian	Dataset akan dimasukkan ke dalam *Deep learning* model U-Net untuk mencari nilai akurasi dan *loss*
Prosedur pengujian	Dataset akan diresize dan akan dimasukkan ke dalam *Deep learning* untuk diuji keakuratannya.

2.2.2 Verifikasi spesifikasi 2

Tabel 2.3.2 Verifikasi Spesifikasi 2

Hal	Ekstraksi dengan menggunakan dataset digitasi
Rincian	Ekstraksi jalan akan dilakukan dengan menggunakan dataset digitasi
Metode Pengujian	Dataset akan dimasukkan ke dalam *Deep learning* untuk mencari nilai akurasi dan *loss*
Prosedur pengujian	Dataset akan diresize dan akan dimasukkan ke dalam *Deep learning* model U-Net untuk menunjukkan hasil ekstraksi

2.3 Kesimpulan dan Ringkasan CD-2

Ekstraksi jalan akan dilakukan dengan metode OBIA (*Object Based Image Analysis*) kemudian dimasukkan dataset yang nantinya akan menentukan hasil ekstraksi. Ekstraksi jalan juga akan dilakukan dengan menggunakan dataset digitasi. Ekstraksi jalan dengan kedua dataset tersebut akan dibandingkan dengan mengganti beberapa parameter. Hasil ekstraksi akan ditampilkan dengan menggunakan model U-net dengan tampilan yang nantinya akan membantu dalam berbagai bidang.

BAB 3
DESAIN RANCANGAN SOLUSI

3.1 Konsep Sistem

3.1.1 Pilihan Sistem

Pada penelitian ini penulis memiliki dua pilihan sistem yang akan nantinya akan dipilih satu sistem untuk dikembangkan dan diteliti. Pemilihan system dilakukan berdasarkan jurnal dan riset yang telah dilakukan.

3.1.1.1 U-Net

Seperti yang telah dijelaskan sebelumnya bahwa U-Net awalnya dikembangkan dalam kasus segmentasi semantik untuk tugas segmentasi gambar. Keunggulan utama U-Net adalah kemampuannya dalam tugas segmentasi gambar dengan baik, terutama memiliki *input* dan *output* dengan resolusi spasial yang sama. U-Net telah terbukti memberikan hasil yang akurat dalam aplikasi segmentasi semantik.

3.1.1.2 Resnet (*Residual Neural Network*)

Resnet merupakan jenis arsitektur *Convolution Neural Network* (CNN) dengan menggunakan model yang sudah dilatih. Prinsip kerja resnet ini adalah membangun jaringan yang lebih dalam dibandingkan dengan jaringan biasa lainnya dan secara bersamaan menemukan jumlah lapisan yang dioptimalkan untuk meniadakan masalah gradien yang hilang[3]. Keunggulan dari resnet ialah pada tugas klasifikasi dengan kemampuan dalam pelatihan jaringan yang dalam, mempelajari fitur representatif dan juga menunjukkan kinerja yang baik dalam melakukan tugas klasifikasi. meskipun resnet mempunyai beberapa keunggulan, ada beberapa kekurangan yang perlu diperhatikan juga seperti adanya kompleksitas model yang mengakibatkan penggunaan model yang lebih sulit dan diinterpretasikannya juga lebih sulit secara intuitif, resiko *overfitting*, penggunaan sumber daya yang lebih tinggi dan memerlukan dataset yang besar.

3.1.2 Analisis

3.1.2.1 Kriteria

Kriteria yang sesuai dengan penjelasan berdasarkan mekanisme dan spesifikasi produk yang akan dibuat, penulis memiliki kriteria sebagai berikut:

1. Pemilihan parameter

   Pemilihan parameter sangat penting dalam melakukan pengujian. Dipillih parameter seperti nilai *epoch*, *learning rate* dan *batch size* untuk mengolah dataset.

2. Efisiensi pengujian

   Dalam melakukan ekstraksi jalan dibutuhkan suatu model untuk melakukan pengujian pada dataset. Pada penelitian ini terdapat dua pilihan model yaitu, U-Net dan Resnet.

3. Mampu mengelolah dataset

   Dalam melakukan pengujian model, dapat dilihat kemampuan model dalam mengelolah dataset.

4. Mampu menampilkan hasil akurasi

   Model mampu dalam menampilkan hasil dari pemrosesan dataset yang telah dilakukan dengan *cofussion matrix* akurasi

3.1.2.2 Analisis konsep

Berdasarkan pilihan system dan analisis kriteria yang telah dilakukan maka analisis konsep secara kualitatif dijabarkan pada tabel berikut:

Tabel 3.1.2.2 Analisis Konsep

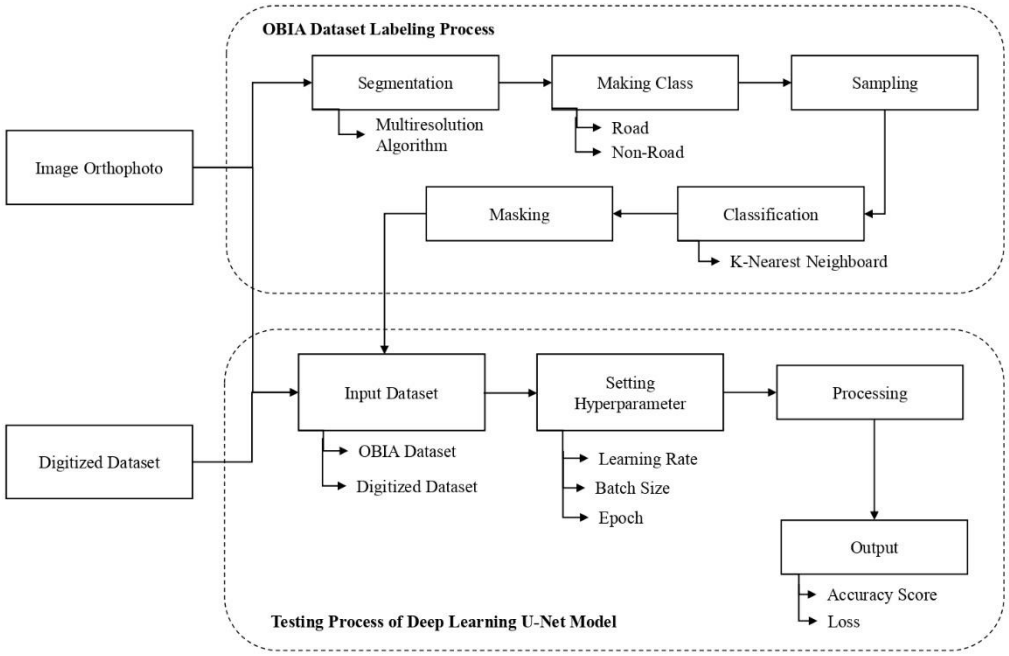
No	Kriteria	U-Net	Resnet
1	Pemilihann parameter	Terpenuhi	Kurang terpenuhi
2	Efisiensi pengujian	Terpenuhi	Terpenuhi
3	Mampu mengelolah dataset	Terpenuhi	Kurang terpenuhi
4	Mampu menampilkan hasil akurasi	Terpenuhi	Kurang terpenuhi

3.1.3 Sistem yang akan Dikembangkan

Pada penelitian ini penulis memilih sistem U-Net dari dua pilihan sistem yang tersedia. pemilihan sistem dilakukakn setelah dilakukan analisis kriteria dan analisis konsep yang dilakukan secara kualitatif. Sistem U-Net dipilih karena lebih relevan terhadap tugas yang akan diteliti, sumber daya yang tersedia dan dataset yang digunakan. selain itu, resiko yang akan timbul masih bisa diatasi dibandingkan Resnet seperti, penggunaan jumlah parameter yang relatif lebih sedikit dibandingkan Resnet dengan begitu akan mengurangi kompleksitas model, kebutuhan komputasi, dan resiko *overfitting*.

3.2 Rencana Desain Sistem

Berikut gambar rencana desain sistem yang akan dikembangkan pada penelitian ini.



Gambar 3. 1 Rencana Desain Sistem

3.3 Pengujian Komponen (Kalibrasi)

Pada pengujian ini akan dilakukan yaitu pengujian performansi dan juga akurasi pada model *Deep learning* U-Net. dengan menggunakan *confusion matrix* yaitu *accuracy.*

3.3.1 Accuracy

Accuracy yaitu untuk mengevaluasi seberapa sering model dengan benar dalam memprediksi kelas target secara keseluruhan data. adapun persamaan *accuracy* yaitu sebagai berikut:

$$Acccuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Dimana:

TP *(True Positive)*: Jumlah prediksi yang benar positif

TN *(True Negative):* Jumlah prediksi yang benar negative

FP *(False Positive)*: Jumlah prediksi yang salah positif

FN *(False Negative)*: Jumlah prediksi yang salah negative

3.4 Jadwal Pengerjaan

Tabel 3.2 Jadwal Pengerjaan

Aktivitas	Progres	2022	2023							
		Desember	Januari	Februari	Maret	April	Mei	Juni	Juli	Agustus
Pengumpulan Proposal Tugas Akhir	100%									
Pengerjaan Dataset dengan OBIA menggunakan Ecogniton	10%									
Pemotongan masking menggunakan Global Mapper	0%									
Pengerjaan Bagian Model Deep Learning	0%									
Penulisan dan Pengumpulan CD-4	0%									
Mengevaluasi dan pengujian Model Deep Learning	0%									
Penulisan dan Pengumpulan CD-5	0%									
Sidang Tugas Akhir	0%									

3.5 Kesimpulan dan Ringkasan CD-3

Kesimpulan dan ringkasan CD-3 ini yaitu melakukan perbandingan dan menentukan solusi yang akan digunakan pada penelitian ini. Dalam mengekstraksi jalan dengan *Deep learning* terdapat dua pilihan sistem yang tersedia yaitu sistem model U-Net dan Resnet. setelah melewati pertimbangan dan riset, pilihan sistem yang dipilih untuk dikembangkan adalah U-Net. Rencana desain sistem dimulai dengan melakukan labelling pada citra orthophoto menggunakan anotasi OBIA dengan *output* berbentuk *masking* yang selanjutnya akan diproses pada *Deep learning* sehingga menghasilkan keakuratan. jadwal pengerjaan tersedia pada tabel gantt chart dengan bentuk persentase dalam setiap bulannya.

BAB 4

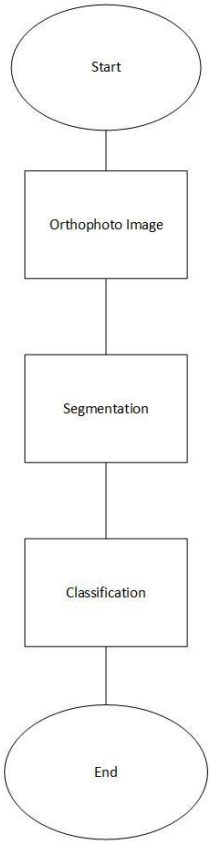
IMPLEMENTASI

4.1 Implementasi Sistem

4.1.1.1 OBIA (*Object Based Image Analysis*)

4.1.1.1 Cara Kerja

Cara kerja pada OBIA *(Object Based Image Analysis)* adalah dengan membedakan atau memisahkan objek berdasarkan kriteria objek yaitu bentuk, warna, tekstur dan ukuran. Proses OBIA ini dilakukan pada aplikasi Ecognition. Aplikasi Ecognition ini sendiri adalah aplikasi yang diciptakan khusus untuk menganalisis citra. Dalam proses OBIA akan ada 2 langkah yang akan dilakukan yaitu segmentasi dan klasifikasi. Proses segmentasi akan menggunakan algoritma *multiresolution segmentation* sedangkan untuk proses klasifikasi akan menggunakan algoritma *K-Nearest Neighbor*. Setelah melakukan kedua proses tersebut maka akan diperoleh *masking* yang selanjutnya akan diproses ke metode selanjutnya.

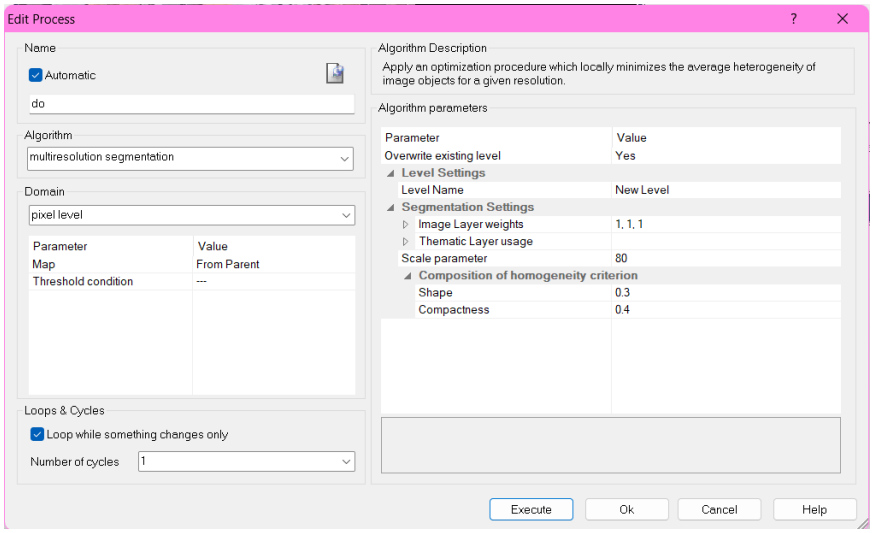


Gambar 4.1.1.1 Flowchart OBIA

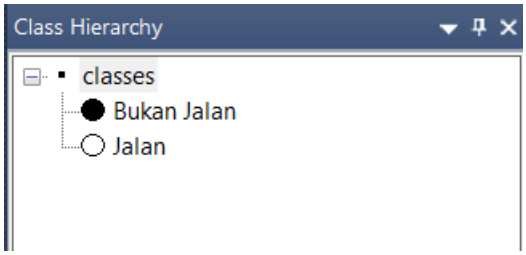
4.1.1.2 Implementasi

Proses segmentasi akan dilakukan menggunakan algoritma *multiresolution segmentation* dengan memasukkan skala-skala tertentu. Besar kecilnya nilai skala yang dimasukkan tentunya akan mempengaruhi hasil segmentasi. Selanjutnya, akan dilakukan proses klasifikasi dengan algoritma *K-Nearest Neighbor*. Proses ini akan membedakan antara jalan dan bukan jalan sehingga akan menghasilkan *masking*. Berikut beberapa tampilan langkah pemrosesan citra pada aplikasi eCognition.

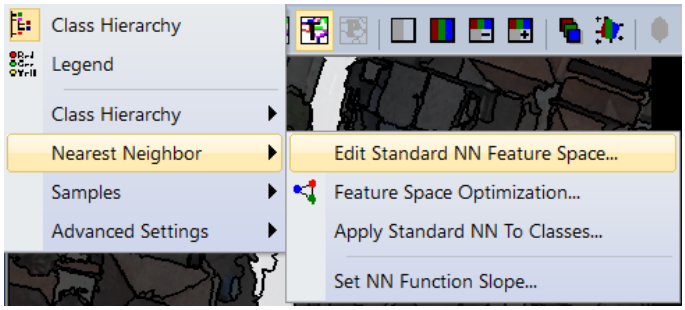
(a)



(b)



(c)



(d)

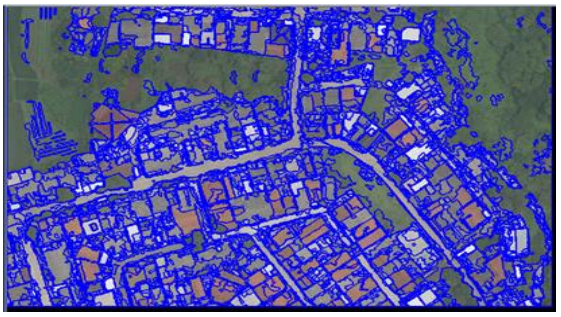
Gambar 4.1.1.2 (a) Algoritma & parameter OBIA, (b) Kelas, (c) Algoritma Klasifikasi, (d) Atribut Masukan

4.1.1.3 Pengujian

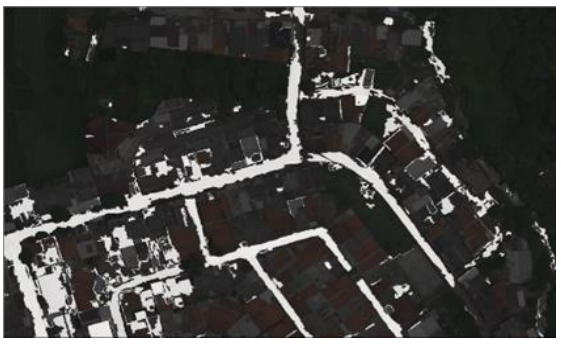
Pada proses pengujian segmentasi akan menggunakan algoritma *multiresolution segmentation.* Dalam algoritma tersebut akan dimasukkan nilai skala seperti skala parameter, *shape* dan *compactness* sedangkan pada proses pengujian klasifikasi akan menggunakan algoritma *K-Nearest Neighbor*, *classification* dan *merge region*. Ketiga algoritma tersebut digunakan untuk menghasilkan citra yang terklasifikasi. Berikut adalah gambar dari hasil pengujian yang melewati proses segmentasi dan klasifikasi. Warna putih menggambarkan jalan dan warna hitam menggambarkan bukan jalan.

(a)

(b)



(c)

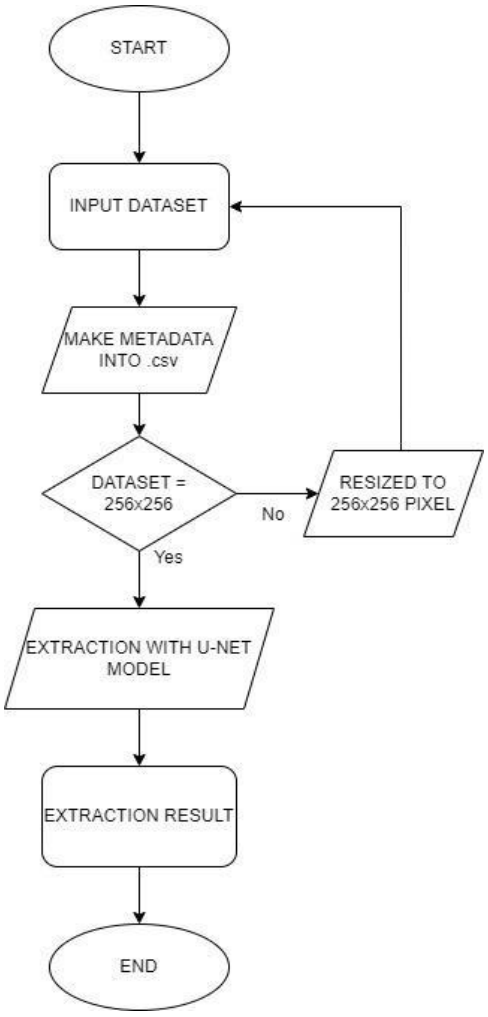


Gambar 4.1.1.3 (a) Citra Asli, (b) Proses Segmentasi, dan (c) hasil Klasifikasi

4.1.2 Sub-sistem 2

4.1.2.1 Cara Kerja Sub-Sistem

Pada dasarnya *Deep learning* model U-Net ini adalah subsistem ini bekerja menggunakan hasil dari subsistem sebelumnya yang berupa dataset dan akan diolah dengan menggunakan *Deep learning* model U-Net. pada proses ini juga penulis akan menggunakan pemrograman phyton. Dataset yang telah dihasilkan pada subsistem sebelumnya akan dimasukkan ke dalam proses pengujian dataset melalui *source code* yang sudah di buat. Dataset yang berupa citra dengan hasil ukuran *piksel* 256 x 256 dan akan diproses dengan model U-Net dan akan menghasilkan ekstraksi jalan yang memiliki resolusi tinggi dan akurat. U-Net merupakan salah satu jenis CNN yang biasa digunakan untuk segmentasi citra berjenis semantik dan U-Net terdiri dari *downsampler* dan *upsampler*. pada subsistem ini kami menggunakan beberapa parameter seperti *epoch*, *batch size* dan *learning rate* yang akan menjadi kombinasi terbaik untuk mendapatkan hasil akurasi yang tinggi.



Gambar 4.1.4.1 Flowchart *Deep learning* Model U-Net

4.1.2.2 Implementasi

Proses ekstraksi menggunakan *Deep learning* dengan model U-Net akan dilakukan dengan memasukkan dataset pada *source code* U-Net. Dataset tersebut akan dimasukkan ke dalam arsitektur U-Net dengan mengatur ukuran *piksel* nya yang awalnya ukuran 1024 x 1024 *piksel* akan diubah menjadi 256 x 256 pada *source* yang sudah dibuat pada pemrograman python yang digunakan. Setelah itu, dataset akan diproses dan dilakukan pengujian pada pemograman yang sudah dibuat. Sebelum melakukan pengujian penulis mengatur beberapa hyperparameter seperti *epoch*, *batch size*, *learning rate* untuk nantinya hasil yang didapatkan memiliki hasil akurasi yang tinggi.

4.1.2.3 Pengujian

Pada pengujian *Deep learning* dengan menggunakan model U-Net ini penulis menggunakan pemrograman python.

```
class Modeling:
    def __init__(self, train_data):
        self.train_data = train_data

    def _get_model(self):
        u = UNET()
        inputs = Input(shape=(256, 256, 3))
        model = u.unet_main(input_=inputs)
        return model

    def train_model(self, batch_size=4, model_name='unet_scratch', epochs=10, show_performance=True):
```

Gambar 4.1.5.3 Program Pengujian

Pada pengujian ini penulis melakukan dengan menggunakan sebuah program yang dibuat menggunakan bahasa pemrograman python. sebelum melakukan pengujian penulis mengatur beberapa hyperparameter seperti *epoch*, *batch size*, *learning rate* untuk nantinya hasil yang didapatkan memiliki hasil akurasi yang tinggi.

1. *Epoch*

   Epoch merupakan hyperparameter penting untuk algoritma, *epoch* juga menentukan jumlah lintasan lengkap dari seluruh dataset pelatihan yang melewati proses pelatihan atau pembelajaran algoritma. Semakin banyak *epoch* dapat membantu model untuk mempelajari pola yang lebih rumit dalam data pelatihan, namun terlalu banyak *epoch* juga bisa menyebabkan *overfitting*.

2. *Batch size*

   Batch size merupakan hyperparameter dalam pembelajaran sistem yang

menentukan jumlah sampel yang disebarkan melalui jaringan saraf selama setiap pelatihan. pada *batch size* juga merupakan parameter penting karena mempengaruhi kecepatan dan stabilitas proses pelatihan. pada *batch size* yang memiliki nilai lebih besar dapat menghasilkan waktu pelatihan yang lebih cepat, namun membutuhkan lebih banyak memori.

3. *Learning rate*

Learning rate merupakan hyperparameter kritis karena dapat mempengaruhi kecepatan dan stabilitas pada proses pelatihan *Deep learning*. *Learning rate* dengan nilai yang besar dapat menyebabkan model menyatu terlalu cepat dan menghasilkan kinerja yang buruk. sedangkan nilai *learning rate* yang kecil dapat menyebabkan model menyatu lebih lama, namun hasil yang didapatkan menghasilkan kinerja yang baik.

```
m = Modeling(train_data=train_data)
```

```
model = m.train_model()
```

Gambar 4.1.6.3 Program Training Model

Setelah menentukan hyperparameter penulis melakukan training model pada *source code* seperti gambar 4.1.2.3. setelah selesai melakukan training model makan akan keluar nilai *accuracy* dan *loss* dari setiap *epoch*. Berikut merupakan tampilan hasil setelah melakukan *training* pada dataset.

```
Epoch 1/10
224/224 [==============================] - 30s 69ms/step - loss: 0.6892 - accuracy: 0.0597 - val_loss: 0.6883 - val_accuracy: 0.0603
Epoch 2/10
224/224 [==============================] - 15s 66ms/step - loss: 0.6854 - accuracy: 0.0603 - val_loss: 0.6888 - val_accuracy: 0.0571
Epoch 3/10
224/224 [==============================] - 15s 66ms/step - loss: 0.6835 - accuracy: 0.0598 - val_loss: 0.6923 - val_accuracy: 0.0624
Epoch 4/10
224/224 [==============================] - 15s 66ms/step - loss: 0.6818 - accuracy: 0.0606 - val_loss: 0.6934 - val_accuracy: 0.0276
Epoch 5/10
224/224 [==============================] - 15s 66ms/step - loss: 0.6798 - accuracy: 0.0606 - val_loss: 0.6997 - val_accuracy: 0.0171
Epoch 6/10
224/224 [==============================] - 15s 66ms/step - loss: 0.6768 - accuracy: 0.0624 - val_loss: 0.6769 - val_accuracy: 0.0624
Epoch 7/10
224/224 [==============================] - 15s 67ms/step - loss: 0.6737 - accuracy: 0.0646 - val_loss: 0.6798 - val_accuracy: 0.0598
Epoch 8/10
224/224 [==============================] - 15s 67ms/step - loss: 0.6705 - accuracy: 0.0656 - val_loss: 0.6732 - val_accuracy: 0.0771
Epoch 9/10
224/224 [==============================] - 15s 67ms/step - loss: 0.6687 - accuracy: 0.0673 - val_loss: 0.6946 - val_accuracy: 0.0307
Epoch 10/10
224/224 [==============================] - 15s 67ms/step - loss: 0.6668 - accuracy: 0.0679 - val_loss: 0.6632 - val_accuracy: 0.0740
```

Gambar 4.1.7.4 Hasil *Deep learning*

4.2 Analisis Pengerjaan Implementasi Sistem

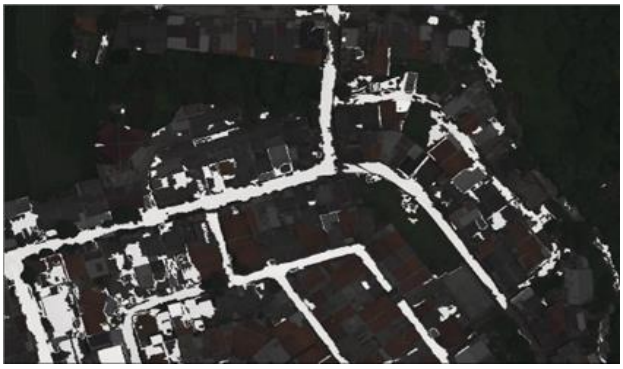
Tabel 4. 1 Analisis Pengerjaan Implementasi system

Timeline Pengerjaan Tugas Akhir Capston Desain Ekstraksi Jalan Menggunakan *Deep Learning* dengan Model U-Net															
Aktivitas	Minggu														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Mempelajari dan memproses dataset OBIA																
Mencari referensi dan mempelajari kodingan *deep learning* (U-Net)																
Melakukan pengujian pada dataset OBIA																
Melakukan pengujian pada dataset Digitasi																

4.3 Hasil Akhir Sistem

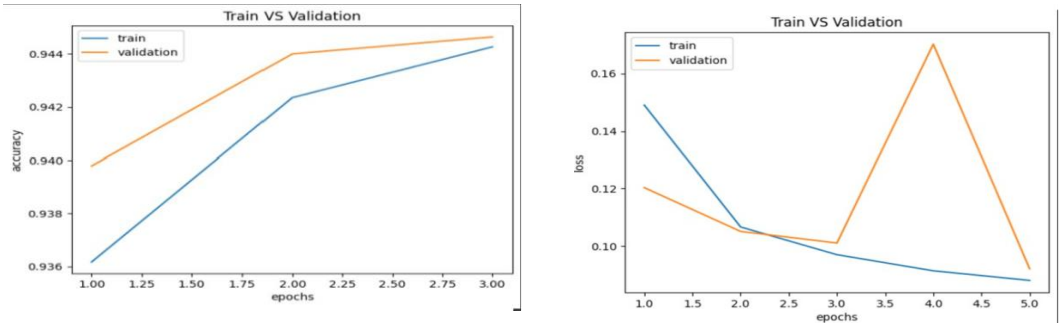
Hasil akhir dari setiap pengujian yang telah penulis lakukan pada setiap sub sistem yaitu seperti di bawah ini:

1. Hasil Klasifikasi OBIA



Gambar 4.8.1 Hasil Klasifikasi OBIA

2. Hasil Training *Deep learning* U-Net



Gambar 4.9.2 Grrafik Hasil Training *Deep Learning* Model U-Net

4.4 Kesimpulan dan Ringkasan CD-4

Pada ekstraksi dengan dataset OBIA memiliki dua proses utama yaitu, proses segmentasi dan klasifikasi yang nantinya akan menghasilkan *masking* dan akan dimasukkan dalam *Deep learning* model U-Net. Pada ekstraksi dengan metode digitasi juga akan menghasilkan *masking* digitasi yang nantinya akan dimasukkan ke *Deep learning* dengan model U-Net. Kedua dataset tersebut akan diekstraksi dengan *Deep learning* model U-Net dan akan menghasilkan hasil ekstraksi.

BAB 5
PENGUJIAN SISTEM

5.1 Skema Pengujian Sistem

Sistem akan diuji dengan menggunakan dataset OBIA dan digitasi. Kedua dataset tersebut akan diuji menggunakan *Deep learning* dengan model U-Net. Dataset tersebut akan diuji dengan dengan mengubah beberapa parameter dan akan menghasilkan hasil ekstraksi. Pengujian akan dilakukan berkali-kali dengan beberapa beberapa parameter yang berbeda. Dataset akan dimasukkan ke dalam *source code Deep learning* model U-Net. Hasil pengujian akan menghasilkan nilai akurasi yang tentunya berbeda pula.

5.2 Proses Pengujian

5.2.1 Proses Pengujian dengan Dataset OBIA

Pada pengujian pertama, dilakukan pada data set OBIA (*Object Based Image Analysis*) yang berbentuk citra orthophoto dan *masking*. Citra orthophoto dan *masking* tersebut berukuran 1024 x 1024 piksel akan berubah menjadi ukuran 256 x 256 piksel setelah diproses pada *Deep learning* model U-Net menggunakan program yang telah dibuat. Gambar 5.2.1 merupakan proses *Deep learning* sebelum melakukan pengujian yaitu mengubah parameter seperti nilai *epoch* dan *batch size* agar hasil keluaran *Deep learning* memiliki keakuratan yang baik.

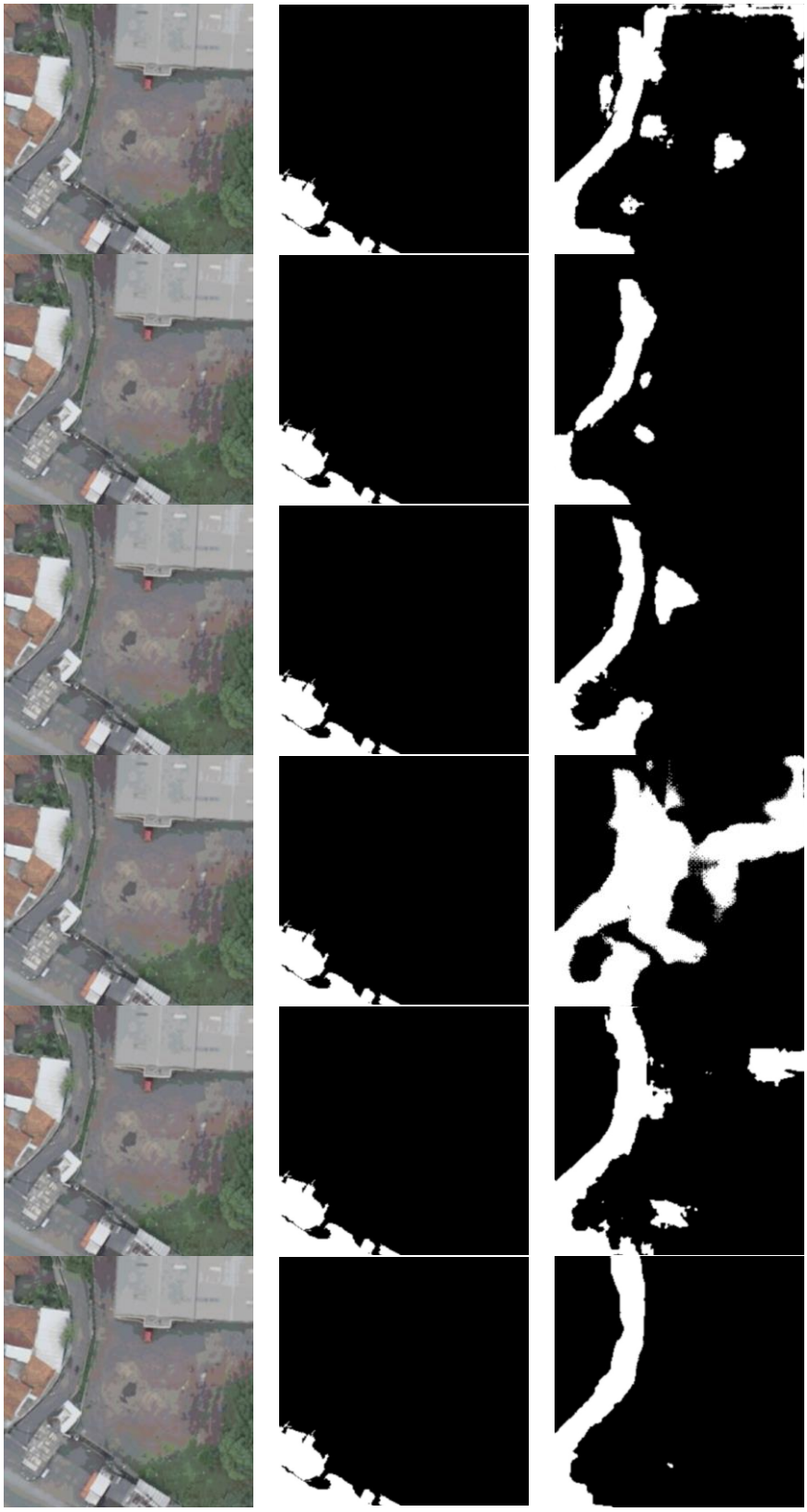
```
class Modeling:
    def __init__(self, train_data):
        self.train_data = train_data

    def _get_model(self):
        u = UNET()
        inputs = Input(shape=(256, 256, 3))
        model = u.unet_main(input_=inputs)
        return model

    def train_model(self, batch_size=4, model_name='unet_scratch', epochs=40, show_performance=True):
```

Gambar 5.2.1 Program Pengujian dengan Dataset OBIA

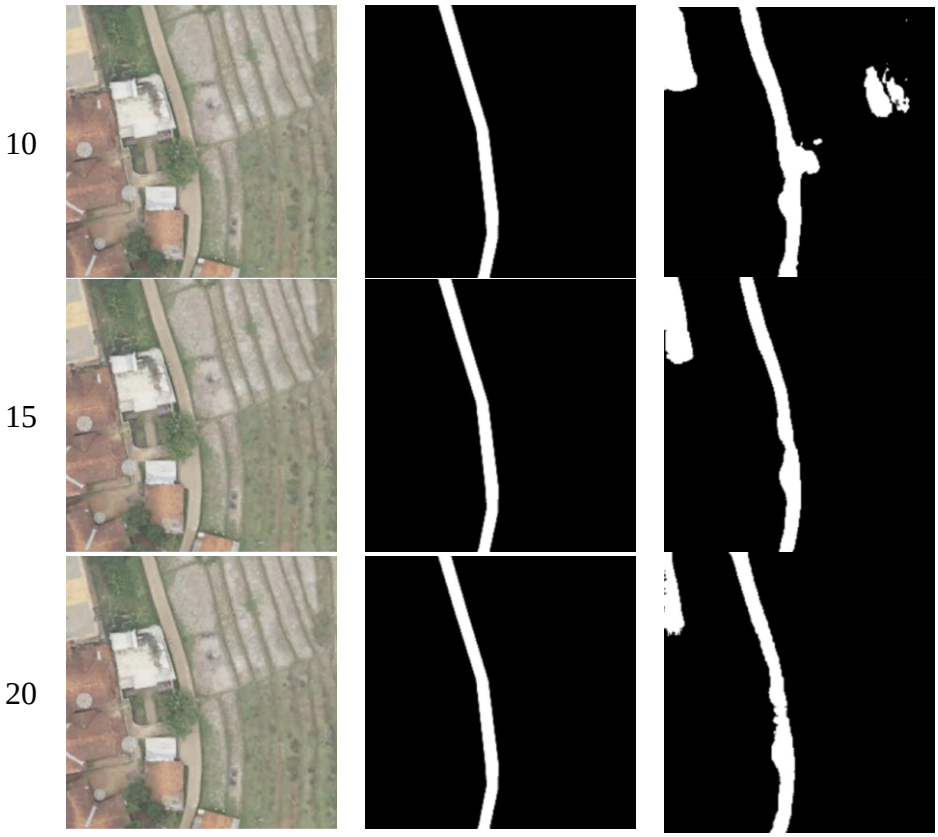
Pengujian pada dataset ini dilakukan sebanyak 7 kali pengujian dengan dengan nilai parameter *epoch* yang berbeda yaitu pada *epoch* 10, 15, 20, 25, 30, 35, dan 40. Berikut hasil pengujian pada dataset OBIA.

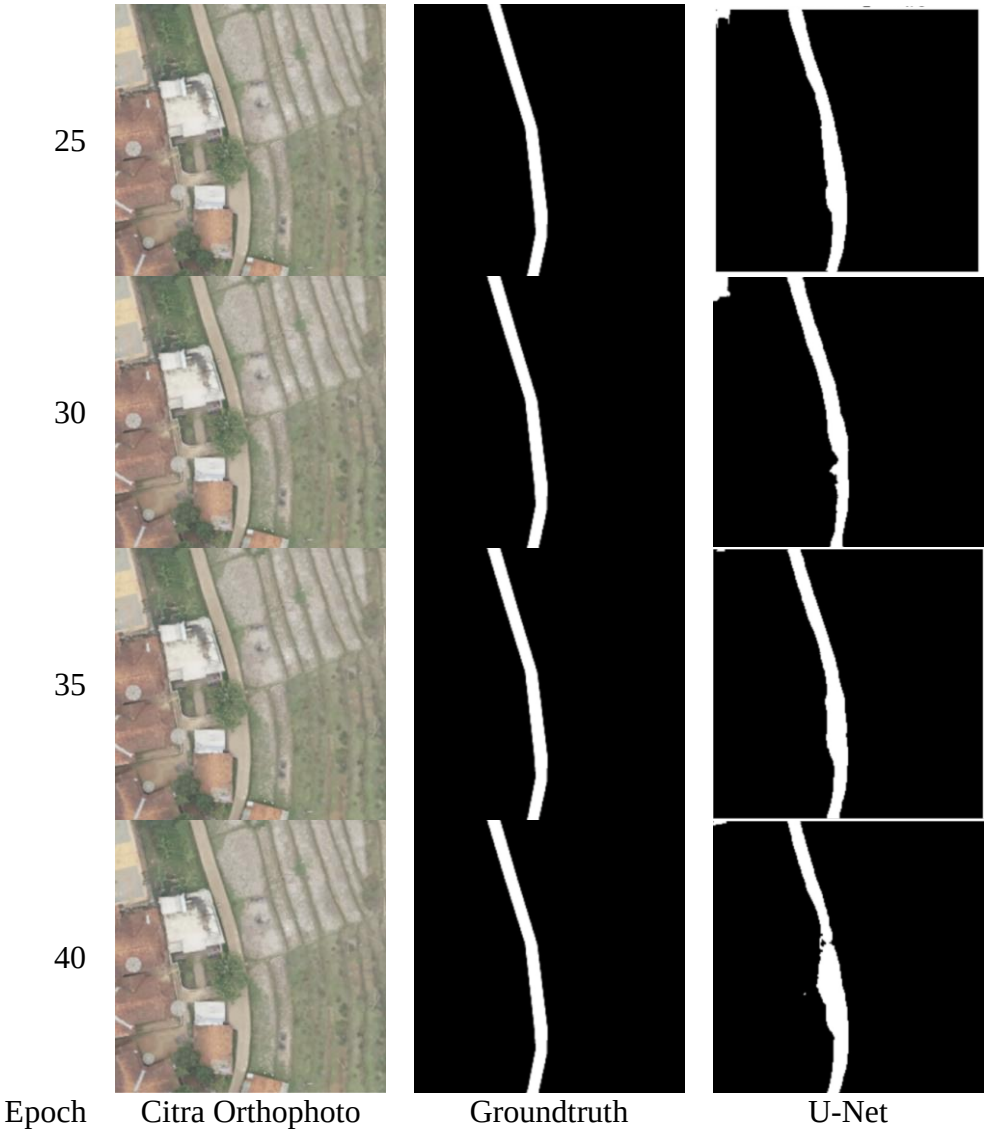


Gambar 5.2.1 Hasil Pengujian Dataset OBIA

5.2.2 Proses Pengujian 2

Pada pengujian kedua ini, dilakukan pada data set digitasi yang berbentuk *masking*. Data set tersebut berukuran 1024 x 1024 piksel akan berubah menjadi ukuran 256 x 256 piksel setelah di proses pada *Deep learning*. Langkah pada proses pengujian kedua ini dilakukan sama seperti proses pada pengujian pertama yaitu dengan mengatur parameter yang ingin digunakan pada pengujian, seperti *epoch, batch size* dan *learning rate* agar mendapatkan hasil prediksi yang baik. Pengujian pada dataset ini dilakukan sebanyak 7 kali pengujian dengan dengan nilai parameter *epoch* yang berbeda yaitu pada *epoch* 10, 15, 20, 25, 30, 35, dan 40 sama seperti pada pengujian dengan dataset OBIA. Berikut hasil pengujian pada dataset digitasi.



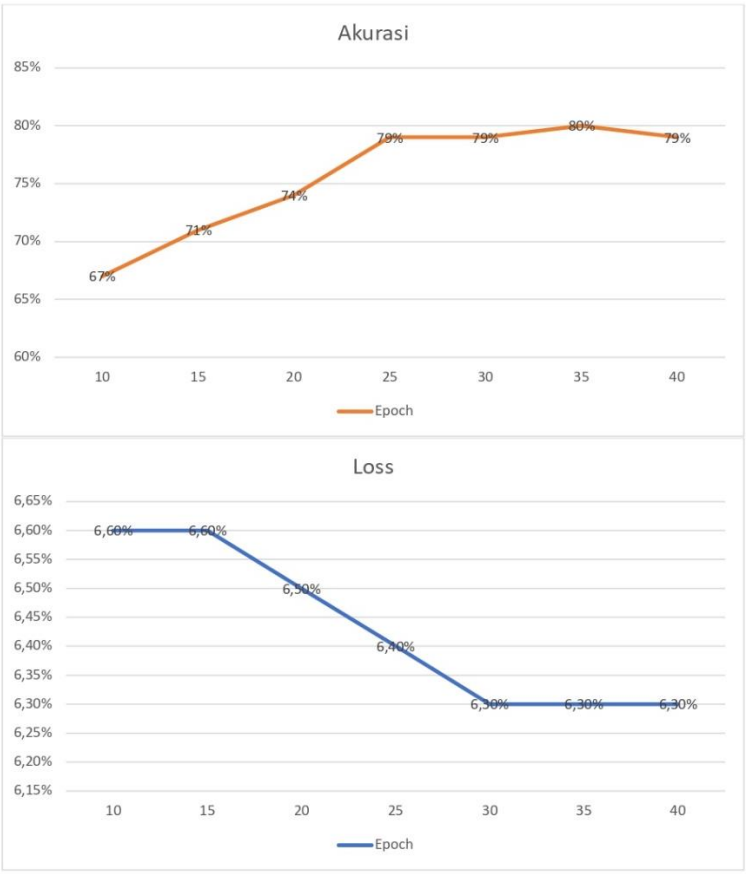


Gambar 5.3.2 Hasil Pengujian Dataset Digitasi

5.3 Analisis Hasil Pengujian

5.3.1 Analisis Hasil Pengujian 1

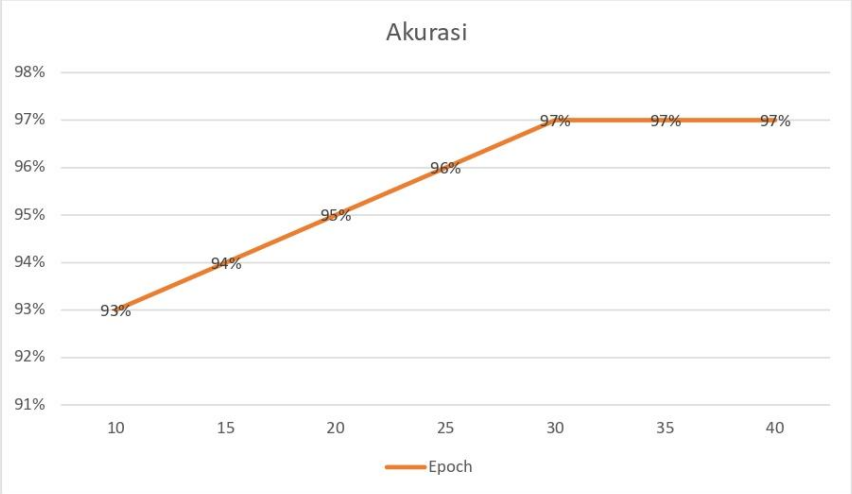
Pada pengujian 1 dengan menggunakan dataset OBIA menggunakan *Deep learning* dengan model U-Net dilakukan pengujian sebanyak 7 kali dengan nilai *epoch* yang berbeda. Pengujian dengan menggunakan dataset OBIA menghasilkan nilai akurasi yang terhitung lebih kecil. Hasil *prediction* didapatkan tidak terlalu akurat. Hal ini terjadi dikarenakan salah satunya jumlah dataset dan hasil *masking* OBIA yang kurang baik kualitasnya. Jika hasil *masking* OBIA dapat diperbaiki maka nilai akurasi akan bertambah dan hasil *prediction*nya akan lebih baik.



Gambar 5.1.1 Grafik Analisis Hasil Pengujian 1

5.3.2 Analisis Hasil Pengujian 2

Pada pengujian I dengan menggunakan dataset digitasi menggunakan *Deep learning* dengan model U-Net dilakukan pengujian sebanyak 7 kali dengan nilai *epoch* yang berbeda. Pengujian dengan menggunakan dataset digitasi menghasilkan nilai akurasi yang terhitung lebih besar. Hasil *prediction* didapatkan lebih bagus dibandingkan dengan hasil dengan OBIA. Hal ini terjadi karena dataset digitasi diambil dari citra orthophoto dengan kualitas tinggi yang di *masking* dengan otomatis. Dengan demikian hasil pengujian yang dilakukan menghasilkan hasil yang lebih optimal.



Gambar 5.2.2 Grafik Analisi Hasil Pengujian 2

5.4 Kesimpulan dan Ringkasan CD-5

Pada pengujian sistem dilakukan beberapa pengujian berkali-kali dengan nilai parameter yang berbeda. Pengujian dilakukan untuk mengetahui akurasi dari kedua dataset yang digunakan. Pengujian kedua dataset tersebut akan menghasilkan hasil yang berbeda juga. Pada hasil pengujian dengan dataset OBIA memiliki hasil yang kurang maksimal karenakan hasil *masking* OBIA memiliki kualitas yang kurang baik atau terdapat beberapa noise pada *masking* karena proses *labelling* dilakukan secara semi-otomatis yaitu masih menggunakan campur tangan manusia sedangkan pada pengujian dengan menggunakan dataset digitasi memiliki hasil yang lebih baik karena *masking* yang dihasilkan pada digitasi juga memiliki kualitas yang baik sehingga sistem lebih baik dalam mempelajari hasil *masking*nya.

DAFTAR PUSTAKA

[1] Z. Zhang, Q. Liu and Y. Wang, "Road Extraction by Deep Residual U-Net," *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS,* vol. 15, no. 5, p. 1, 5 May 2018.

[2] H. R. R. Bakhtiari, A. Abdollahi and H. Rezaeian, "Semi automatic road extraction from digital images," *The Egyptian Journal of Remote Sensing and Space Sciences,* vol. 20, no. 117-123, pp. 1-2, 2017.

[3] G. D and S. H, "Convolutional Neural Network dalam Analisis Citra Medis," *KONSTELASI: Konvergensi Teknologi dan Sistem Informasi,* vol. 2, no. 2, pp. 1-3, 2022.

[4] A. B. Pratama and R. E. A. H. M. A. Kadir, "Deteksi Ruang Kosong pada Jalan Menggunakan," *JURNAL TEKNIK ITS,* vol. 11, no. 1, pp. 1-6, 2022.

[5] A. Abdollahi, B. Pradhan, N. Shukla, S. Chakraborty and A. Alamri, "*Deep learning* Approaches Applied to Remote Sensing Datasets for Road Extraction: A State-Of-The-Art Review," *remote sensing,* vol. 1444, no. 12, pp. 1-22, 2020.

[6] X. Lu, Y. Zhong, Z. Zheng, Y. Liu, M. A. Zhao Ji and Y. Jie, "Multi-Scale and Multi-Task *Deep learning* Framework for Automatic Road Extraction," *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING,* vol. 57, no. 11, pp. 1-16, 2019.

[7] N. More, R. Lalla, R. Memon and V. Nikam, "Extraction of Road Network from Satellite images using Efficient Net," *International Journal of Engineering Research & Technology (IJERT),* vol. 9, no. 7, pp. 2-6, 2020.

[8] Q. Wu, F. Luo, P. Wu, B. Wang, H. Yang and Y. Wu, "Automatic Road Extraction from High-Resolution Remote Sensing Images Using a Method Based on Densely Connected Spatial Feature-Enhanced Pyramid," *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING,* vol. 14, pp. 1-15, 2021.

[9] J. Hormesea and C. Dr. Saravanan, "Automated Road Extraction From High Resolution Satellite Images," *International Conference on Emerging Trends in Engineering, Science and Technology,* vol. 24, pp. 1460-1467, 2016.

[10] W. Wang, N. Yang, Y. Zhang, F. Wang and T. Cao, "A review of road extraction from remote sensing images," *j o u rnal of traffic and t r a n s p o r t a t i on e n g in e e r i n g (e n g l i s h e d i t i o n),* vol. 3, no. 3, pp. 271-282, 2016.

[11] Z. Chen, C. Wang, J. Li, W. Fan, D. Jixiang and B. Zhong, "Adaboost-like End-to-End multiple lightweight U-nets for road extraction from optical remote sensing images," *International Journal of Applied Earth Observations and Geoinformation,* vol. 100, no. 102341, pp. 1-14, 2021.

[12] A. Wilujeng, D. K. Sunaryo and A. Noraini, "PEMANFAATAN METODE OBIA (OBJECT-BASED IMAGE-ANALYSIS) UNTUK ANALISIS KESESUAIAN PENGGUNAAN LAHAN AKTUAL TERHADAP RENCANA TATA RUANG WILAYAH (RTRW)".

[13] A. Pribadi and A. K. Adisusilo, "Pemanfaatan 3D U-Net untuk Segmentasi 3 Dimensi Gelembung Penyebab Kanker Paru-paru (Nodule) pada Lapisan Citra CT Scan," *JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTATION,* pp. 1-12.

[14] "PENERAPAN METODE RESIDUAL NETWORK (RESNET) DALAM KLASIFIKASI PENYAKIT PADA DAUN GANDUM," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika),* vol. 7, no. 1, pp. 1-5, 2022.

[15] A. N. Kasanah, Muladi and U. Pujianto, "Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN," *Rekayasa Sistem dan Teknologi Informasi),* vol. 3, no. 2, pp. 1-6, 2019.

[16] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical," *Computer Science Department and BIOSS Centre for Biological Signalling Studies,* vol. III, no. 9351, p. 235, 2015.

LAMPIRAN CD-4

1. Komponen dan perangkat yang digunkan pada penelitian.

LAMPIRAN CD-5

1. *Source code Deep learning*

```
[1] ! pip install patool --quiet
    ! pip install imgaug --quiet

        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.5/77.5 kB 9.0 MB/s eta 0:00:00
```

```
import os
import patoolib
import pandas as pd
import cv2 as cv
import numpy as np
import tensorflow as tf
import cv2
import imgaug.augmenters as iaa

from sklearn.model_selection import train_test_split
from skimage.filters import threshold_otsu
from matplotlib import pyplot as plt
from tensorflow.keras.layers import (
    Input,
    Conv2D,
    MaxPooling2D,
    concatenate,
    Conv2DTranspose,
    BatchNormalization,
    Dropout,
    Activation,
    Concatenate
)
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import (
    EarlyStopping,
    ModelCheckpoint
)
```

```
[7] project_path = '/content/train/'
```

```
[8] file_path = '/content/metadata.csv'
    df = pd.read_csv(filepath_or_buffer=file_path)
```

```
[9] train_data = df[df['split'] == 'train']
    valid_data = df[df['split'] == 'valid']
    test_data = df[df['split'] == 'test']
```

[10]
```python
class Dataset:
    def __init__(self, df):
        self.df = df
        self.sat_images = self.df['sat_image_path'].to_list()
        self.masks_fps = self.df['mask_path'].to_list()

    def __getitem__(self, i):
        # Read image directly into RGB format
        image = plt.imread('/content/' + self.sat_images[i])
        # Read image mask
        image_mask = plt.imread('/content/' + self.masks_fps[i])
        image_mask = image_mask[:, :, 0]

        # Resize image and mask
        image = cv2.resize(image, (256, 256))
        image_mask = cv2.resize(image_mask, (256, 256))

        # # Print resized image and mask shapes
        # print("Resized image shape:", image.shape)
        # print("Resized mask shape:", image_mask.shape)

        return image, image_mask

    def __len__(self):
        return len(self.df)
```

```python
class DataLoder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size=1, shuffle=False):
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(dataset))

    def __getitem__(self, i):
        # collect batch data
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size

        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.stack(samples, axis=0) for samples in zip(*data)]

        return tuple(batch)

    def __len__(self):
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        if self.shuffle:
            self.indexes = np.random.permutation(self.indexes)
```

```
class UNET:
    # convolution block
    def _convolve(self, input_, filters):
        x = Conv2D(filters=filters, kernel_size=(3, 3), kernel_initializer='he_normal', padding='same')(input_)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        x = Conv2D(filters=filters, kernel_size=(3, 3), kernel_initializer='he_normal', padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        return x

    # up-sampling and convolotion block
    def _convolve_by_upsampling(self, input_, skip_connector, filters, rate):
        x = Conv2DTranspose(filters=filters, kernel_size=(3, 3), strides=(2, 2), padding='same')(input_)
        x = concatenate([x, skip_connector])
        x = Dropout(rate)(x)
        x = self._convolve(input_=x, filters=filters)
        return x

    # UNET main model
    def unet_main(self, input_, filters=16, rate=0.001):
        # left encoder Path
        c1 = self._convolve(input_=input_, filters=filters)
        p1 = MaxPooling2D(pool_size=(2, 2))(c1)
        p1 = Dropout(rate)(p1)

        c2 = self._convolve(input_=p1, filters=filters * 2)
        p2 = MaxPooling2D(pool_size=(2, 2))(c2)
        p2 = Dropout(rate)(p2)

        c3 = self._convolve(input_=p2, filters=filters * 4)
        p3 = MaxPooling2D(pool_size=(2, 2))(c3)
        p3 = Dropout(rate)(p3)

        c4 = self._convolve(input_=p3, filters=filters * 8)
        p4 = MaxPooling2D(pool_size=(2, 2))(c4)
        p4 = Dropout(rate)(p4)

        # middle bridge
        c5 = self._convolve(input_=p4, filters=filters * 16)

        # right decoder path
        c6 = self._convolve_by_upsampling(input_=c5, skip_connector=c4, filters=filters * 8, rate=rate)
        c7 = self._convolve_by_upsampling(input_=c6, skip_connector=c3, filters=filters * 4, rate=rate)
        c8 = self._convolve_by_upsampling(input_=c7, skip_connector=c2, filters=filters * 2, rate=rate)
        c9 = self._convolve_by_upsampling(input_=c8, skip_connector=c1, filters=filters * 1, rate=rate)

        output_ = Conv2D(filters=1, kernel_size=(1, 1), activation='sigmoid')(c9)

        model = Model(inputs=[input_], outputs=[output_])

        return model
```

```python
class Modeling:
    def __init__(self, train_data):
        self.train_data = train_data

    def _get_model(self):
        u = UNET()
        inputs = Input(shape=(256, 256, 3))
        model = u.unet_main(input_=inputs)
        return model

    def train_model(self, batch_size=4, model_name='unet_scratch', epochs=30, show_performance=True):
        # since the original validation data does not have masks
        # i am considering `1%` of `train_data` as the `custom_validation`
        # in order to keep track of the model's performance
        self.custom_train, self.custom_valid = train_test_split(self.train_data, test_size=0.1, random_state=42)

        train_dataset = Dataset(df=self.custom_train)
        valid_dataset = Dataset(df=self.custom_valid)

        train_dataloader = DataLoder(train_dataset, batch_size=batch_size, shuffle=True)
        valid_dataloader = DataLoder(valid_dataset, batch_size=batch_size, shuffle=True)

        model = self._get_model()
        # callbacks
        model_path = project_path + 'models/{}.h5'.format(model_name)
        model_save_callback = ModelCheckpoint(
            filepath=model_path,
            save_weights_only=True,
            save_best_only=True,
            mode='max',
            monitor='val_accuracy'
        )
        callbacks = [model_save_callback]

        model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])

        if not os.path.isfile(path=model_path):
            tracker = model.fit(
                x=train_dataloader,
                steps_per_epoch=len(train_dataloader),
                epochs=epochs,
                validation_data=valid_dataloader,
                callbacks=callbacks
            )
            model.load_weights(model_path)

            if show_performance:
                self._plot_performance(tracker=tracker, epochs=len(tracker.history['accuracy']))

        else:
            print('Model is already trained and is present in directory.')
            model.load_weights(model_path)

        return model
```

```
def _plot_xy(self, x, y1, y2, yl, xl='epochs'):
    plt.plot(x, y1, label='train')
    plt.plot(x, y2, label='validation')
    plt.xlabel(xl)
    plt.ylabel(yl)
    plt.title('Train VS Validation')
    plt.legend()
    plt.show()
    return None

def _plot_performance(self, tracker, epochs):
    x = list(range(1, epochs + 1))

    train_iou = tracker.history['accuracy']
    valid_iou = tracker.history['val_accuracy']

    train_loss = tracker.history['loss']
    valid_loss = tracker.history['val_loss']

    self._plot_xy(x=x, y1=train_iou, y2=valid_iou, yl='accuracy')
    self._plot_xy(x=x, y1=train_loss, y2=valid_loss, yl='loss')

    return None

def plot_custom_valid_performance(self, model, n=10):
    vdata = self.custom_valid.head(n)

    # paths of images
    images_fps   = vdata['sat_image_path'].to_list()
    # paths of masked images
    masks_fps    = vdata['mask_path'].to_list()

    for oimg, mimg in zip(images_fps, masks_fps):
        # original image
        simage = plt.imread('/content/' + oimg)
        simage = cv2.resize(simage, (256, 256))
        # mask image
        mimage = plt.imread('/content/' + mimg)
        mimage = cv2.resize(mimage, (256, 256))
        mimage = mimage[:,:,0]

        # prediction
        predicted_image  = model.predict(simage[np.newaxis,:,:,:])
        predicted_mask = predicted_image.reshape(simage.shape[0], simage.shape[1])
        # automatic threshold identification
        pmask_thresh = threshold_otsu(predicted_mask)
        # binarizing the mask
        th, predicted_mask = cv.threshold(src=predicted_mask, thresh=pmask_thresh, maxval=255, type=cv.THRESH_BINARY)

        # titles
        image_title = oimg.split('/')[-1]
        mask_title = mimg.split('/')[-1]

        # plotting figure
        plt.figure(figsize=(15, 6))
```

```
            plt.subplot(131)
            plt.axis("off")
            plt.title(image_title)
            plt.imshow(simage)

            plt.subplot(132)
            plt.axis("off")
            plt.title(mask_title)
            plt.imshow(mimage, cmap='gray')

            plt.subplot(133)
            plt.axis("off")
            plt.title("Prediction - {}".format(image_title))
            plt.imshow(predicted_mask, cmap='gray')

            plt.show()

        return None

    def extract_road_path(self, x, model, n=10):
        x_ = x.head(n)

        # paths of images
        images_fps    = x_['sat_image_path'].to_list()
        # paths of masked images
        masks_fps     = x_['mask_path'].to_list()

        for oimg, mimg in zip(images_fps, masks_fps):
            # original image
            simage = plt.imread('/content/' + oimg)

            # prediction
            predicted_image  = model.predict(simage[np.newaxis,:,:,:])
            predicted_mask = predicted_image.reshape(simage.shape[0], simage.shape[1])
            # automatic threshold identification
            pmask_thresh = threshold_otsu(predicted_mask)
            # binarizing the mask
            th, predicted_mask = cv.threshold(src=predicted_mask, thresh=pmask_thresh, maxval=255, type=cv.THRESH_BINARY)
            image_title = oimg.split('/')[-1]

            plt.figure(figsize=(10, 6))

            plt.subplot(121)
            plt.axis("off")
            plt.title(image_title)
            plt.imshow(simage)

            plt.subplot(122)
            plt.axis("off")
            plt.title("Prediction - {}".format(image_title))
            plt.imshow(predicted_mask, cmap='gray')

            plt.show()

        return None
```

```
m = Modeling(train_data=train_data)

model = m.train_model()
```

```
m.plot_custom_valid_performance(model=model)
```

```
m.extract_road_path(x=train_data, model=model)
```