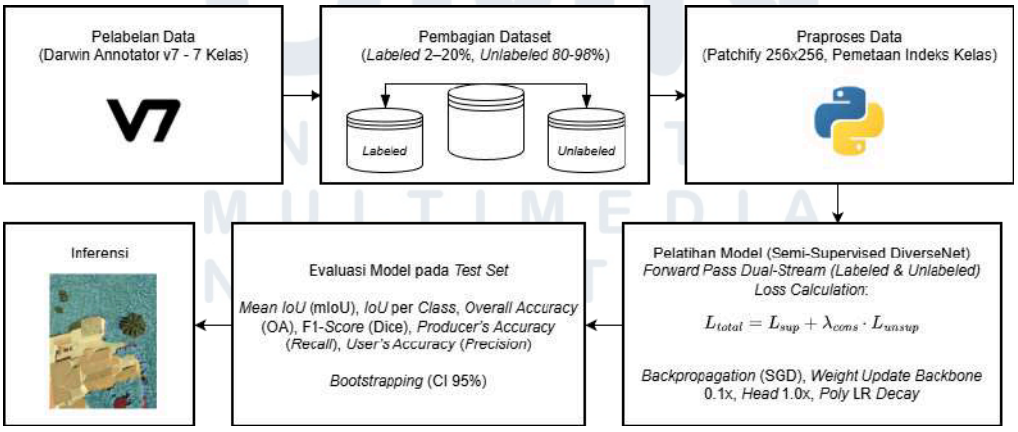
BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Metode penelitian yang diterapkan dalam studi ini dirancang untuk menyelesaikan permasalahan segmentasi semantik *multi-class* pada citra *drone* kebun salak dengan kendala utama berupa keterbatasan data berlabel. Penelitian ini menggunakan *framework* DiverseNet, khususnya varian DiverseHead untuk memanfaatkan data tidak berlabel yang melimpah. Seluruh rangkaian tahapan penelitian, sebagaimana divisualisasikan secara sistematis pada Gambar 3.1, dimulai dari pelabelan ulang tujuh kelas pada citra, kemudian dilanjutkan ke tahap pemrosesan awal (*preprocessing*) yang mencakup teknik *patching* dan pemetaan indeks kelas. Alur penelitian kemudian memasuki tahap berupa skenario pemisahan data dengan proporsi label berbeda (2%, 5%, 10, dan 20%) yang disertai analisis distribusi kelas pasca-*patching* untuk memastikan keterwakilan fitur pada setiap skenario, sebelum akhirnya dilakukan pelatihan model, evaluasi kinerja berbasis metrik *mean Intersection over Union* (mIoU) untuk kualitas rata-rata, tetapi juga mencakup IoU per *class*, *Overall Accuracy* (OA), *Producer's Accuracy* (PA), *User's Accuracy* (UA), dan F1-Score yang didukung dengan analisis statistik *Bootstrapping* untuk mengestimasi *confidence interval* dan stabilitas performa model, serta modul inferensi untuk menghasilkan visualisasi segmentasi peta.



Gambar 3.1 Alur penelitian

3.2 Perancangan Modul

3.2.1 Pengumpulan Dataset

Penelitian ini menggunakan dataset primer yang telah diakuisisi pada bulan Juli 2024 dari kawasan perkebunan salak Kelompok Tani Sedyo Makmur dan Muda Jaya, Turi, Sleman. Dataset terdiri dari 1.000 citra udara dalam format .png yang terorganisasi dalam enam blok penyimpanan (*salak-1-1* hingga *salak-1-6*). Secara teknis, dataset ini memiliki dua variasi resolusi tinggi. Mayoritas data sebanyak 914 citra (91,4%) memiliki dimensi 5472 x 3648 piksel, sedangkan sisanya sebanyak 86 citra (8,6%) memiliki dimensi 4000 x 3000 piksel. Lalu, untuk menjamin validitas pengujian yang objektif, penelitian ini juga mengalokasikan 20 citra tambahan di luar dataset utama tersebut yang difungsikan sebagai data uji atau *Ground Truth* mutlak. Secara keseluruhan, total 1.020 citra tersebut mencakup variasi lanskap campuran kawasan kebun salak, meliputi pohon salak, pohon inang alternatif, bangunan, jalan dan halaman, tanah terbuka dan semak, badan air, dan lain-lain, yang menjamin model diuji pada kondisi lingkungan yang heterogen sesuai dengan kondisi nyata di lapangan. Penggunaan dataset pelatihan sama dengan penelitian terdahulu bertujuan untuk memastikan konsistensi *baseline* data, namun dikembangkan lebih lanjut melalui pendekatan anotasi *multi-class* yang berbeda [7], [10].

3.2.2 Pelabelan Ulang Data

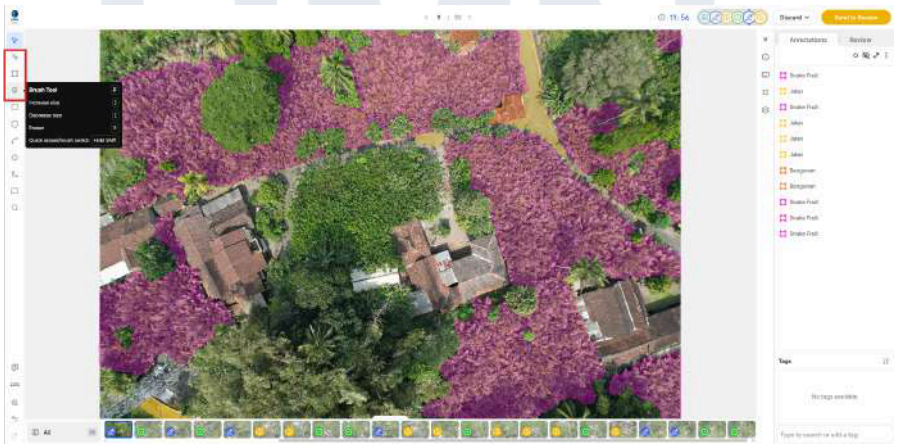
Mengingat penelitian terdahulu hanya menggunakan label biner, tahap ini berfokus pada pelabelan ulang dataset menggunakan platform Darwin V7 Labs [8]. Proses ini bertujuan untuk memetakan tujuh kelas semantik yang merepresentasikan faktor risiko ekologis pada kawasan kebun salak Turi, Sleman.

Anotasi dilakukan dengan menggambar poligon presisi pada objek-objek berikut dan dapat dilihat pada Gambar 3.2 untuk contoh-contohnya:

1. Kelas 0 (Lain-lain/*Background*): Seluruh objek yang tidak termasuk ke dalam enam kategori utama.
2. Kelas 1 (Badan Air): Sungai, kanal, atau kolam yang berperan sebagai sumber kelembapan, zona transisi, dan habitat predator alami hama.
3. Kelas 2 (Bangunan): Rumah warga, gudang, dan gubuk petani di sekitar kebun.
4. Kelas 3 (Jalan dan Halaman): Jalan beraspal maupun jalan tanah sebagai jalur akses.
5. Kelas 4 (Pohon Inang Alternatif): Kebun cabai yang dapat menjadi inang sekunder hama lalat buah.
6. Kelas 5 (Pohon Salak): Tanaman utama kebun salak sebagai objek target segmentasi.
7. Kelas 6 (Tanah Terbuka dan Semak): Lahan kosong, semak belukar, dan area berpotensi menjadi tempat penumpukan limbah organik.

Gambar 3.2. Contoh potongan gambar dari setiap kelas terlabel.

Platform Darwin V7 Labs dipilih karena menyediakan ekosistem anotasi yang kompatibel dengan kebutuhan segmentasi semantik multi-kelas pada citra *drone*. Pada penelitian ini, beberapa fitur inti Darwin V7 digunakan sesuai fungsi teknis masing-masing. Pertama, **A**uto-Annotate & AI-Assisted Tool (SAM), yang letaknya di atas Polygon Tool, dimanfaatkan sebagdai tahap awal untuk menghasilkan *initial mask* berbasis Segment Anything Model, terutama pada objek berukuran besar atau kompleks seperti kanopi pohon atau vegetasi rapat. *Mask* awal ini kemudian diperhalus secara manual menggunakan dua alat utama. Polygon Tool, yang letaknya di atas Brush Tool, digunakan ketika objek memiliki kontur geometris yang relatif tegas dan dapat diikuti melalui klik berurutan, seperti bangunan atau lahan terbuka dengan batas tepi jelas. Sementara itu, Brush Tool dan Eraser Tool digunakan untuk objek dengan bentuk organik dan tepi tidak beraturan, misalnya badan air, semak, dan lain-lain. Letak *tools* dapat dilihat pada Gambar 3.3. Pemilihan alat ini bertujuan untuk memastikan bahwa setiap mask kelas memiliki batas spasial yang akurat sebelum diolah dalam pipeline *machine learning*.



Gambar 3.3 Contoh *tools* yang digunakan saat anotasi

Darwin V7 Labs juga menerapkan *workflow* berjenjang yang digunakan dalam penelitian ini, terdiri dari empat status utama: *New*, *Being Annotated*, *In Review*, dan *Complete*. Pada tahap *New*, citra belum tersentuh dan menunggu penugasan anotator. Tahap *Being Annotated* menandai proses pembuatan mask oleh anotator utama. Setelah selesai, citra dipindahkan ke status *In Review*, di mana anotasi diperiksa ulang untuk konsistensi kelas, ketepatan batas, dan kesesuaian definisi. Hanya setelah lolos validasi, citra ditetapkan sebagai *Complete* dan siap digunakan dalam tahap preprocessing serta pelatihan model [8].

3.2.3 *Preprocessing* Data

Tahap *preprocessing* bertujuan untuk mengubah data mentah berupa citra asli dan *mask* RGB menjadi format tensor yang kompatibel dengan arsitektur jaringan saraf tiruan. Proses ini terdiri dari dua langkah utama, yaitu pemotongan citra (*patching*) dan pemetaan indeks kelas.

a. *Patching*

Citra *drone* yang diakuisisi memiliki resolusi spasial yang sangat besar (sebagai contoh, 5472 x 3078 piksel). Memasukkan citra dengan dimensi tersebut secara langsung ke dalam model *Deep Learning* akan menyebabkan kegagalan alokasi memori pada GPU. Hal ini disebabkan oleh mekanisme *backpropagation* yang tidak hanya menyimpan data citra input, tetapi juga harus menampung jutaan parameter bobot, peta fitur dari setiap lapisan konvolusi, serta gradien untuk kalkulasi *loss*. Oleh karena itu, diterapkan teknik *patching* menggunakan pustaka Python patchify [24] untuk memecah beban komputasi.

Teknik ini memecah citra besar menjadi potongan-potongan kecil (*patches*) berukuran 256 x 256 piksel. Penelitian ini menerapkan mekanisme tumpang tindih (*overlap*) dengan menentukan nilai langkah (*step*) sebesar 220 piksel. Penggunaan *overlap* bertujuan untuk menjaga konsistensi konteks spasial antar-potongan dan mencegah hilangnya informasi objek yang berada tepat di garis potong. Setiap *patch* yang dihasilkan kemudian disimpan secara berurutan untuk memudahkan proses rekonstruksi kembali pada tahap inferensi.

Penggunaan *overlap* ini berfungsi menjaga integritas objek di tepian, di mana objek visual seperti pohon salak yang terpotong di satu *patch* akan muncul secara utuh di bagian tengah patch berikutnya, sehingga fitur pembeda utama tetap terjaga. Lalu, setiap patch yang dihasilkan kemudian disimpan secara berurutan untuk memudahkan proses rekonstruksi kembali pada tahap inferensi.

b. Pemetaan Indeks Kelas ke Indeks *Grayscale*

Langkah selanjutnya adalah konversi format mask. Platform Darwin V7 mengekspor hasil anotasi dalam format citra RGB (3 *channel*), di mana setiap kelas direpresentasikan oleh kombinasi warna unik. Namun, fungsi *loss* standar untuk segmentasi semantik multi-class pada *framework* PyTorch, yaitu CrossEntropyLoss, mensyaratkan target label berupa LongTensor yang berisi indeks kelas bilangan bulat [0, C-1], bukan nilai intensitas warna RGB maupun *one-hot encoding* [25].

Oleh karena itu, dilakukan proses *remapping* di mana setiap piksel RGB dikonversi menjadi indeks skalar 0-6. Proses ini mengubah dimensi data target dari format (H x W x 3) menjadi (H x W x 1), menghilangkan dimensi channel warna yang tidak diperlukan. Selain memenuhi persyaratan mutlak fungsi *loss*, pendekatan ini menyederhanakan struktur tensor target dengan mereduksi redundansi

informasi warna menjadi representasi kategori yang lebih ringkas. Peta konversi warna ke indeks kelas disajikan pada Tabel 3.1.

Tabel 3.1 Peta Konversi RGB ke Indeks Kelas

Nama Kelas	Warna RGB	Indeks Kelas
Background	(0, 0, 0)	**0**
Badan Air	(255, 50, 50)	**1**
Bangunan	(255, 225, 50)	**2**
Jalan dan Halaman	(109, 255, 50)	**3**
Pohon Inang Alternatif	(50, 255, 167)	**4**
Pohon Salak	(50, 167, 255)	**5**
Tanah Terbuka	(109, 50, 255)	**6**

Hasil akhir dari proses *remapping* ini disimpan dalam format PNG. Format ini dipilih karena mendukung kompresi *lossless*, sehingga integritas nilai indeks kelas pada setiap piksel tetap terjaga dan tidak mengalami kompresi sebagaimana yang terjadi pada format JPG.

3.2.4 Pemisahan Data dan Analisis Distribusi Kelas

Untuk menguji efektivitas metode *Semi-Supervised Learning* (DiverseNet) dalam mengatasi kelangkaan label, dirancang strategi pemisahan data. Dataset dibagi menjadi tiga himpunan utama:

1. Himpunan Latih Berlabel
2. Himpunan Latih Tidak Berlabel
3. Himpunan Uji (*Ground Truth*)

a. Skenario Pemisahan Bertingkat

Proses pemisahan dilakukan pada tingkat citra asli sebelum proses *patching*. Hal ini dilakukan untuk mencegah kebocoran data, di mana potongan gambar yang saling bertetangga (berasal dari satu citra asli yang sama) terpisah ke dalam himpunan latih dan uji, yang dapat menyebabkan bias evaluasi. Dataset latih dibagi ke dalam empat

skenario proporsi data berlabel yang juga mewakili *checkpoint* dari ketersediaan data berlabel dalam periode penelitian:

1. Skenario 2% (20 data terlabel)
2. Skenario 5% (50 data terlabel)
3. Skenario 10% (100 data terlabel)
4. Skenario 20% (200 data terlabel)

Sisa data dari masing-masing skenario (misalnya 98%, 95%, 90%, dan 80%) dialokasikan sebagai *Unlabeled Set* yang berfungsi sebagai domain pembelajaran *semi-supervised* bagi model DiverseNet, di mana informasi diekstraksi melalui mekanisme *pseudo-labeling*.

b. Analisis Distribusi Kelas Pasca-Patching

Mengingat citra *drone* memiliki variasi lanskap yang tinggi, pengambilan sampel acak sebesar 2% berisiko menghasilkan dataset yang tidak representatif (misalnya, hanya berisi pohon salak tanpa adanya kelas minoritas seperti sampah atau badan air). Oleh karena itu, dilakukan analisis distribusi kelas setelah proses *patching*.

Perhitungan distribusi dilakukan dengan mengakumulasi total piksel (N_{pixel}) untuk setiap kelas (c) pada seluruh *patch* dalam *labeled set*, kemudian menghitung persentasenya terhadap total resolusi, sebagaimana dirumuskan dalam persamaan di bawah ini.

$$P_C = \frac{\sum_{i=1}^{K} count(C)_i}{\sum_{i=1}^{K} total\ pixels_i} \cdot 100\%$$

(Dimana K adalah jumlah total patch dalam subset).

Analisis distribusi kelas ini diterapkan secara menyeluruh mencakup seluruh variasi himpunan latih (2%, 5%, 10%, dan 20%) serta Himpunan Uji yang terdiri dari 20 citra *Ground Truth*. Analisis ini tidak ditujukan untuk melakukan penyeimbangan kelas secara artifisial, melainkan sebagai langkah verifikasi dalam skema

pencarian titik efisiensi data. Fokus utamanya terbagi menjadi dua aspek teknis. Pertama, memvalidasi keterwakilan sampel guna menjamin bahwa teknik pengambilan acak, bahkan pada skenario ekstrem 2%, tetap berhasil menangkap keberadaan kelas minoritas seperti *Badan Air* dan *Inang Alternatif*, sehingga mencegah kegagalan pembelajaran fitur akibat absennya data. Kedua, dokumentasi distribusi ini berfungsi sebagai basis data pembanding untuk interpretasi hasil di Bab 4, yang digunakan untuk mengukur sensitivitas model terhadap penambahan volume data dan menjelaskan korelasi antara kelangkaan fitur alami dengan capaian metrik performa pada setiap skenario pelatihan.

3.2.5 Implementasi Model dan Konfigurasi Pelatihan

Tahap ini mencakup implementasi teknis arsitektur jaringan saraf tiruan serta strategi pelatihan yang dioptimasi untuk menangani skenario keterbatasan label. Sebagai fondasi utama, penelitian ini menggunakan *framework* DiverseNet, khususnya arsitektur DiverseHead, yang menerapkan arsitektur *multiple decision heads within a single network* dalam skema *semi-supervised learning*.

DiverseNet menginovasi penggunaan *single network* yang berbagi *backbone* namun bercabang menjadi beberapa *decision heads* ringan. Dalam implementasinya, arsitektur tersebut dimodifikasi menggunakan strategi perturbasi berupa *dynamic freezing* dan *dropout* pada level *head*. Strategi ini bertujuan menciptakan diversitas keputusan antar-*head* tanpa perlu menambahkan *noise* buatan pada data input yang sering kali berisiko merusak fitur citra penginderaan jauh. Melalui mekanisme pemungutan suara berupa *voting* dari berbagai *head* tersebut, sistem dapat menghasilkan *pseudo-labels* dari data tak berlabel, memungkinkan model belajar secara mandiri dan konsisten.

Sebelum memasuki fase pelatihan, manajemen dataset diatur untuk menjaga integritas eksperimen. Dataset yang masif dan didapat

dari proses *patching* ini akan dipartisi dengan skema 90% untuk keperluan pelatihan (*training*) dan 10% untuk validasi internal (*validation*), sementara 20 citra *Ground Truth* utuh dialokasikan secara terpisah sepenuhnya sebagai Himpunan Uji guna menjamin objektivitas evaluasi akhir.

Fokus utama manajemen data dalam penelitian ini ditekankan pada pencarian titik efisiensi data yang optimal melalui skema pengujian yang bertahap. Skema ini dirancang untuk mensimulasikan kelangkaan, untuk mengukur secara presisi sensitivitas dan ketangguhan model terhadap variasi volume data. Dengan demikian, volume data minimum yang diperlukan untuk mencapai performa maksimal dapat ditemukan dan dianalisis. Implementasinya dilakukan melalui mekanisme pengindeksan berbasis berkas teks (labeled_files_20.txt, labeled_files_50.txt, dst.) yang mengatur proporsi data berlabel secara bertahap mulai dari 2% (20 citra), 5% (50 citra), 10% (100 citra), hingga 20% (200 citra). Daftar ini disusun secara inklusif dan progresif, di mana himpunan data yang lebih besar secara otomatis mencakup himpunan data yang lebih kecil. Pendekatan ini menjamin konsistensi komparasi, memastikan bahwa setiap kenaikan performa model yang terukur adalah murni dampak dari penambahan informasi baru, dan bukan akibat bias variabilitas sampel. Untuk itu, berikut adalah langkah-langkah dan konfigurasi untuk pelatihan model:

a. Modifikasi Arsitektur dan Lingkungan Pengembangan

Meskipun menggunakan konsep *multi-head* dari DiverseNet, implementasi dalam penelitian ini melakukan sejumlah adaptasi teknis:

1. Menggantikan sistem *logging* konvensional dengan WandB untuk pemantauan eksperimen berbasis *cloud* secara *real-time*, memungkinkan pendokumentasian metrik per kelas dan dimanfaatkan untuk deteksi dini *overfitting* [26].

2. Kode sumber asli DiverseNet yang sebelumnya terikat secara spesifik pada arsitektur DeepLabV3+ telah direstrukturisasi menjadi bersifat modular guna mengakomodasi integrasi arsitektur UNet++ dengan *backbone* MobileNetV2 atau DeepLabV3+ *backbone* Xception. Keputusan perubahan desain arsitektur ini dilandaskan pada studi terdahulu dalam ekosistem MySalak oleh Zaini dan Tjandra [7], [10], yang telah membuktikan efektivitas modelnya masing-masing dalam menangani segmentasi objek vegetasi salak.
3. Implementasi pustaka tqdm untuk visualisasi progres iterasi yang lebih informatif [27].

b. Mekanisme Pelatihan *Semi-Supervised*

Proses pelatihan dirancang dalam dua aliran yang berjalan secara simultan dalam setiap iterasi:

1. Aliran *Supervised*: Menggunakan data berlabel (2%, 5%, 10%, atau 20%) untuk menghitung *Supervised Loss* (L_{sup}). Prediksi dari seluruh *head* dibandingkan langsung dengan *ground truth* manual menggunakan fungsi kerugian *CrossEntropyLoss*.
2. Aliran *Unsupervised*: Menggunakan data tidak berlabel untuk menghitung *Unsupervised Loss* (L_{unsup}). Pada aliran ini, diterapkan strategi perturbasi keputusan berupa *dynamic freezing* dan *dropout* pada level *head*. Prediksi dari berbagai *head* yang terperturbasi digabungkan melalui mekanisme *dual voting* untuk menghasilkan *pseudo-label*. Konsistensi antara prediksi individu *head* dan *pseudo-label* inilah yang dijadikan sinyal pembelajaran.

Total kerugian (L_{total}) yang dioptimalkan oleh model merupakan penjumlahan berbobot dari kedua komponen tersebut:

$$L_{total} = L_{sup} + \lambda \cdot L_{unsup}$$

Dimana λ adalah bobot *unsupervised loss ramp-up* yang mengatur seberapa besar pengaruh data tidak berlabel terhadap pembaruan bobot model.

c. Strategi Pelatihan *Baseline* v2

Untuk mengatasi tantangan distribusi data yang sangat tidak seimbang, di mana kelas minoritas seperti *Badan Air* memiliki representasi piksel yang jauh lebih sedikit dibandingkan *Salak*, penelitian ini menerapkan penambahan metode. Sebelum menerapkan strategi intervensi, konfigurasi standar (*baseline* DiverseHead_DF) tanpa mekanisme penyeimbangan kelas akan diuji terlebih dahulu sebagai tolak ukur performa awal. Selanjutnya, strategi ditingkatkan melalui pendekatan Baseline v2 yang mengintegrasikan dua inovasi utama guna memperbaiki kelemahan model standar tersebut:

1. *Sqrt-Damped Class-Balanced Loss*

   Fungsi *loss* Cross Entropy ditambahkan dengan varian berbobot dari tiap kelas. Bobot setiap kelas (W_c) dihitung menggunakan rumus akar kuadrat invers frekuensi:

   $$W_c = \sqrt{\frac{N_{max}}{N_c}}$$

   Keterangan:

   - W(c): bobot untuk kelas ke-*c*
   - max□ N□: jumlah piksel maksimum di antara seluruh kelas
   - k ∈ {0, 1, …, C−1}: himpunan indeks seluruh kelas
   - N(c): jumlah piksel pada kelas ke-*c*

   Strategi ini memberikan penalti gradien yang lebih besar pada kelas minoritas (misalnya: ≈ 22.5x untuk kelas *Badan*

Air) tanpa menyebabkan ledakan gradien atau pembaruan bobot setiap selesai iterasi.

2. *Class Specific Confidence Thresholds*

   Dalam menghitung komponen *unsupervised loss*, *confidence threshold* untuk *pseudo-labeling* tidak lagi statis, melainkan diubah dan disesuaikan berdasarkan tingkat kesulitan kelas. Misalnya, Kelas Mudah (*Background*, Salak, Bangunan) diberikan ambang batas tinggi (τ = 0.95) untuk menjaga kemurnian label. Lalu, Kelas Sulit (Badan Air, Jalan), *treshold* diturunkan (τ = 0.80) agar model tetap dapat belajar dari fitur-fitur langka meskipun dengan *confidence treshold* yang lebih moderat.

d. Konfigurasi *Hyperparameter* dan Lingkungan Komputasi

Tahap ini menjelaskan pengaturan parameter teknis yang digunakan untuk melatih model DiverseNet pada berbagai skenario ketersediaan data. Seluruh eksperimen dijalankan menggunakan *optimizer* Stochastic Gradient Descent (SGD) dengan *momentum* 0.9 dan *weight decay* sebesar 1 x 10^{-4} untuk menjaga regularisasi model. Tabel 4.1 merangkum detail *hyperparameter* yang diterapkan pada setiap skenario pembagian data (*split*).

Tabel 3.2 Konfigurasi *Hyperparameter* Pelatihan Model

Parameter Konfigurasi	**Skenario 2% (20 Data)**	**Skenario 5% (50 Data)**	**Skenario 10% (100 Data)**	**Skenario 20% (200 Data)**
Total Iterations	40,000	60,000	60,000	60,000
Base Learning Rate	$3e^{-3}$ (0,003)	$4e^{-3}$ (0,004)	$4e^{-3}$ (0,004)	$5e^{-3}$ (0,005)
Batch Size	16			
Optimizer	SGD (Momentum 0.9)			

Parameter Konfigurasi	Skenario 2% (20 Data)	Skenario 5% (50 Data)	Skenario 10% (100 Data)	Skenario 20% (200 Data)
Weight Decay	$1e^{-4}$ (0,0001)			
Loss Function	*Weighted CE*			
CPS Weight	7.5			

Penentuan nilai *Base Learning Rate* (LR) dalam penelitian ini dilakukan secara dinamis dengan mempertimbangkan volume data berlabel pada masing-masing skenario. Berdasarkan hasil eksperimen awal, skenario dengan data yang lebih melimpah (20% atau 200 citra) menunjukkan performa yang lebih optimal dan konvergensi yang lebih cepat saat menggunakan LR 0,005 dibandingkan nilai yang lebih rendah. Namun, pada data yang sangat terbatas (2% dan 5%), nilai LR diturunkan secara bertahap hingga 0,003 untuk menjaga stabilitas pelatihan dan mencegah model untuk mengalami fluktuasi performa akibat minimnya *supervision signal* dari *ground truth*. Pengaturan LR yang proporsional ini divalidasi melalui pemantauan grafik pada *platform* Weights & Biases (WandB), di mana jarak antara *train loss* dan *validation loss* terjaga secara stabil dengan selisih yang moderat (berada pada rentang > 1.5).

Selama setiap *epoch*, variabel-variabel kritis seperti *Total Loss* (gabungan *supervised* dan *unsupervised*), *Validation mIoU*, dan metrik-metrik evaluasi lainnya dicatat secara berkala setiap 500 iterasi. Untuk mengamankan progres pelatihan, diterapkan mekanisme penyimpanan model otomatis (*checkpoint*) dalam format .pth (PyTorch State Dictionary). Sistem secara otomatis menyimpan *checkpoint* best_model.pth setiap kali model mencapai nilai mIoU validasi tertinggi baru, serta last_model.pth pada setiap 500 iterasi dan

akhir pelatihan untuk memungkinkan kelanjutan proses pelatihan apabila terjadi gangguan teknis.

3.2.6 Evaluasi Model

Setelah proses pelatihan selesai, kinerja model dievaluasi menggunakan himpunan uji (*test set*) yang terpisah. Himpunan ini terdiri dari 20 citra resolusi penuh yang telah dianotasi secara manual dan menyeluruh. Ke-20 citra ini difungsikan sebagai *Ground Truth* mutlak yang terpisah sepenuhnya dari data latih. Tujuannya adalah untuk memverifikasi metrik kuantitatif secara objektif dan membandingkan prediksi model terhadap kondisi riil lapangan tanpa bias, sekaligus mengukur kemampuan model dalam memetakan tujuh kelas risiko ekologis secara akurat. Pengukuran dilakukan berdasarkan *confusion matrix* yang menghitung elemen *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN) untuk setiap kelas. Metrik-metrik kuantitatif yang digunakan dalam penelitian ini meliputi:

a. *Intersection over Union* (IoU) per Kelas

IoU mengukur persentase soal seberapa tepat area prediksi model berimpit dengan area *ground truth* untuk setiap kelas. Metrik ini menjadi indikator utama hasil segmentasi pada setiap kelas, di mana nilai yang mendekati 1 menunjukkan bahwa segmentasi kelas yang dihasilkan model memiliki kesesuaian posisi yang tinggi dengan kondisi sebenarnya pada citra. Dalam konteks aplikasi, metrik ini juga metrik utama yang menjadi standar *stakeholder* dalam permasalahan ini. Untuk mengetahui kesuksesan metrik, misalnya pada kelas *Salak*, IoU menghitung seberapa besar area hasil segmentasi pohon yang digambar oleh model beririsan dengan anotasi pohon sebenarnya pada citra.

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}$$

b. *Mean Intersection over Union* (mIoU)

mIoU merupakan nilai rata-rata dari IoU seluruh kelas yang ada. Metrik ini merepresentasikan kemampuan generalisasi model dalam menangani ketujuh kelas sekaligus. Nilai mIoU yang tinggi mengindikasikan bahwa model memiliki kinerja yang stabil dan mampu memetakan berbagai jenis objek dengan kualitas yang setara, baik pada objek berukuran besar maupun kecil. Sebaliknya, nilai mIoU yang rendah mengindikasikan ketimpangan performa, di mana model cenderung hanya mengenali kelas yang dominan namun gagal mendeteksi kelas-kelas minoritas dengan benar.

$$mIoU_c = \frac{1}{C} \sum_{i=1}^{C} IoU_i$$

c. *Overall Accuracy* (OA)

OA menghitung persentase total piksel dalam citra yang diklasifikasikan dengan benar oleh model. Metrik ini memberikan gambaran kesesuaian setiap piksel citra global antara prediksi dan citra asli. Nilai OA yang tinggi menunjukkan bahwa secara visual, peta hasil segmentasi sebagian besar sudah sesuai dengan realitas lapangan. Namun, metrik ini harus diinterpretasikan dengan hati-hati pada data tidak seimbang, karena nilai tinggi bisa saja tercapai meskipun model gagal mendeteksi kelas-kelas kecil lainnya yang cukup langka.

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

d. *Producer's Accuracy* (PA) / *Recall*

PA atau *Recall* mengukur tingkat kelengkapan deteksi model terhadap suatu kelas target dengan meminimalkan *False Negative*. Nilai PA yang tinggi menunjukkan bahwa model mampu menemukan sebagian besar area dari objek yang dicari. Sebagai contoh, pada kelas

Bangunan, PA tinggi berarti hampir seluruh rumah yang ada di citra berhasil dikenali oleh model dan tidak ada yang terlewat atau dianggap sebagai kelas lainnya.

$$PA = \frac{TP}{TP + FN}$$

e. *User's Accuracy* (UA) / *Precision*

UA atau *Precision* mengukur tingkat reliabilitas dari prediksi yang dihasilkan dengan meminimalkan *False Positive*. Nilai UA yang tinggi menandakan bahwa model selektif dalam memberikan label. Misalnya pada kelas *Bangunan*, UA tinggi berarti area yang ditandai model sebagai bangunan memiliki probabilitas tinggi memang merupakan atap rumah, bukan objek lain yang memiliki kemiripan warna atau tekstur.

$$UA = \frac{TP}{TP + FP}$$

f. F1-Score

F1-Score merupakan rata-rata harmonik yang menyeimbangkan nilai *Precision* (UA) dan *Recall* (PA). Metrik ini digunakan untuk mendapatkan penilaian performa yang objektif pada dataset dengan distribusi kelas yang tidak seimbang. Nilai F1-Score yang tinggi menunjukkan bahwa model mampu mendeteksi objek dengan lengkap tanpa menghasilkan banyak deteksi palsu yang tidak perlu.

$$F1 = 2 \cdot \frac{PA \cdot UA}{PA + UA}$$

Lalu, untuk menjamin reliabilitas statistik dari pengukuran metrik-metrik tersebut, khususnya di tengah keterbatasan jumlah sampel uji (20 citra), penelitian ini menerapkan metode *Bootstrapping non-parametrik*. Prosedur ini melibatkan proses pengambilan sampel ulang terhadap himpunan uji sebanyak 1.000 iterasi (*R=1000*) dengan mekanisme pengembalian. Pendekatan ini mengubah hasil evaluasi

dari sekadar estimasi titik tunggal menjadi rentang estimasi *confidence interval* (CI) sebesar 95%. Dengan demikian, performa model dilaporkan dalam bentuk rentang nilai (*Mean* ± *margin of error*), yang memberikan gambaran lebih objektif mengenai stabilitas dan konsistensi kinerja model terhadap variasi data, sekaligus meminimalisir potensi bias kebetulan yang mungkin timbul dari pengujian statis. Perhitungan *margin of error* adalah sebagai berikut:

$$margin\ of\ error = \frac{batas\ atas\ (CI) - batas\ bawah\ (CI)}{2}$$

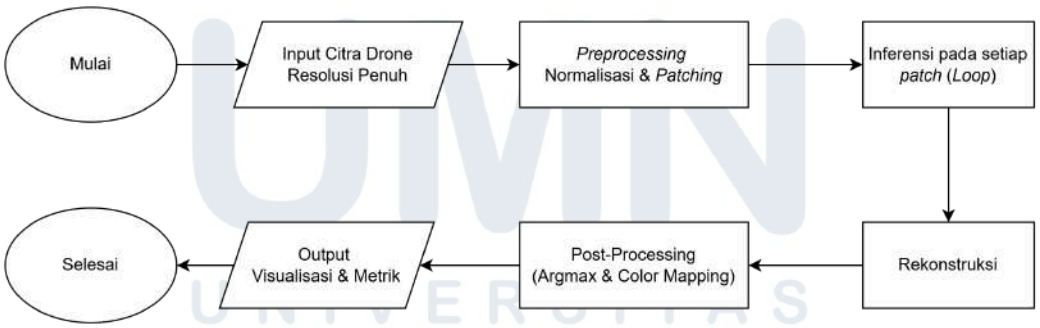
Guna kemudahan analisis komparatif antar-skenario pelatihan (2% hingga 20%), penelitian ini merancang sistem pelaporan hasil evaluasi yang terotomasi sebagai solusi atas keterbatasan pencatatan manual yang rentan terhadap kesalahan manusia. Sistem ini dibangun dengan memanfaatkan pustaka Pandas untuk manajemen struktur data tabular dan XlsxWriter untuk pengaturan visual, menghasilkan dokumen rekapitulasi performa dalam format *spreadsheet*. Mekanisme penyimpanan inkremental diterapkan pada *spreadsheet* yang memungkinkan sistem membaca berkas laporan yang sudah ada dan menambahkan hasil eksperimen model baru tanpa menimpa data, sehingga memungkinkan filter model yang terus konsisten berkembang seiring berjalannya eksperimen. Kedua, visualisasi kondisional yang menerapkan format warna (*conditional formatting*) secara otomatis pada sel-sel metrik, dengan gradasi hijau untuk nilai tinggi dan merah untuk nilai rendah, guna memfasilitasi analisis visual yang cepat terhadap performa model pada setiap kelas.

3.3 Pengujian Inferensi dan Visualisasi Segmentasi

Tahap pengujian inferensi merupakan fase validasi akhir yang bertujuan untuk menguji generalisasi model terbaik (*best checkpoint*) yang telah disimpan pada proses pelatihan. Pada tahap ini, model dengan performa terbaik dimuat kembali dan diatur ke mode evaluasi. Tahap ini bertujuan untuk menerjemahkan matriks probabilitas keluaran model menjadi representasi

visual yang dapat diinterpretasikan secara kualitatif dan dilaporkan metriknya secara kuantitatif, seperti peta sebaran hama atau identifikasi lokasi inang alternatif pada citra drone resolusi penuh (misalnya: 5472 x 3078 piksel). Mekanisme pra-pemrosesan data yang diterapkan pada tahap ini, meliputi normalisasi statistik ImageNet dan strategi *patching*, mengikuti prosedur standar yang identik dengan yang telah diterapkan pada modul evaluasi model di sub-bab sebelumnya. Konsistensi teknik ini dijaga untuk memastikan bahwa performa model pada tahap inferensi visual tetap selaras dengan metrik akurasi yang telah terverifikasi sebelumnya.

Penerapan model pada citra resolusi penuh secara tidak langsung memberikan beban komputasi tersendiri karena dimensi citra yang jauh melebihi kapasitas input model. Untuk mengatasi hal tersebut tanpa mengorbankan detail resolusi, diterapkan strategi *Sliding Window Inference*. Strategi ini memecah citra besar menjadi serangkaian jendela proses yang saling tumpang tindih, melakukan prediksi secara parsial, dan menyatukannya kembali melalui mekanisme rekonstruksi probabilistik. Seluruh rangkaian proses inferensi citra utuh ini, yang terdiri dari lima tahapan utama mulai dari pra-pemrosesan hingga visualisasi *overlay*, diilustrasikan secara skematis pada Gambar 3.3.



Gambar 3.3 Alir Inferensi

3.3.1 Metodologi Pra-pemrosesan Data Uji

Sebelum dimasukkan ke dalam model, citra uji diproses menggunakan pustaka Albumentations untuk menjamin konsistensi format dengan data latih [28]. Tahapan ini mencakup:

1. Normalisasi Citra

Setiap saluran warna dinormalisasi menggunakan standar statistik dataset ImageNet, yaitu dengan nilai Mean: [0.485, 0.456, 0.406] dan Standard Deviation: [0.229, 0.224, 0.225]. Langkah ini dilakukan karena *backbone* model yang digunakan telah dilatih sebelumnya (*pre-trained*) pada domain distribusi data tersebut. Penerapan normalisasi yang identik memastikan fitur visual diekstraksi dengan bobot yang presisi.

2. Penyiapan Tensor

Citra dalam bentuk *patches* dikonversi menjadi format tensor PyTorch dengan dimensi input target 256 x 256 piksel.

3.3.2 Konfigurasi Arsitektur Inferensi

Sistem inferensi dibangun di atas *framework* PyTorch, secara spesifik menggunakan konfigurasi model terbaik yang telah divalidasi pada tahap pelatihan. Pemilihan konfigurasi tetap ini menjamin efisiensi komputasi saat diterapkan pada citra resolusi tinggi sekaligus mempertahankan presisi deteksi batas objek.

Fitur yang membedakan sistem ini dari inferensi standar adalah implementasi mekanisme *Multi-Head Output* DiverseNet. Meskipun menggunakan satu *backbone* terpusat, model tidak bergantung pada satu keluaran tunggal, melainkan melakukan perhitungan *logits* dari seluruh *decision heads* yang tersedia. Proses akumulasi ini diformulasikan sebagai penjumlahan *element-wise* dari skor logits mentah sebelum fungsi aktivasi Softmax. Pendekatan ini dipilih untuk mengakumulasi intensitas keyakinan dari setiap *head* secara murni tanpa distorsi normalisasi, sehingga keputusan akhir lebih merepresentasikan konsensus model yang sebenarnya. Proses ini diformulasikan sebagai penjumlahan *element-wise* dari probabilitas mentah sebelum fungsi aktivasi:

$$Logits_{final} = \sum_{i=1}^{N} Head_i(x)$$

Mekanisme penjumlahan ini dirancang untuk berfungsi sebagai *ensemble* internal yang secara teoritis dapat meminimalisir variansi prediksi antar-*head*.

3.3.3 Strategi Inferensi *Sliding Window*

Untuk mengatasi kendala memori GPU pada citra resolusi tinggi, diterapkan strategi *Sliding Window* dengan spesifikasi teknis sebagai berikut:

1. *Patching* dengan *Overlap*

   Citra utuh dipecah menjadi *patches* berukuran 256 x 256 piksel.

2. *Stride* & *Overlap*

   Pemotongan dilakukan dengan nilai langkah (*stride*) atau *steps* sebesar 220 piksel. Konfigurasi ini menghasilkan area tumpang tindih (*overlap*) sebesar 36 piksel (sekitar 14% dari ukuran *patch*). Area *overlap* ini berfungsi mereduksi artefak batas yang sering muncul pada metode pemotongan *grid* biasa.

3. *Reflective Padding*

   Pada bagian tepi citra yang ukurannya tidak memenuhi 256 x 256 piksel, diterapkan teknik *Reflective Padding*, yaitu mencerminkan piksel terdekat untuk mengisi kekosongan tanpa memperkenalkan nilai nol alias warna hitam yang dapat mengganggu prediksi tepian.

3.3.4 Rekonstruksi

Tahap ini merupakan proses penggabungan kembali hasil prediksi dari ratusan potongan gambar menjadi satu kesatuan peta risiko utuh. Hal ini karena inferensi dilakukan menggunakan strategi *sliding window* dengan *overlap*, satu piksel pada citra asli dapat diprediksi beberapa kali oleh *patch* yang berbeda. Oleh karena itu, rekonstruksi tidak dapat dilakukan dengan sekadar menempelkan *patch*, melainkan

memerlukan mekanisme akumulasi probabilistik untuk menjamin konsistensi spasial.

1. Inisialisasi Variabel Kanvas Akumulasi Kosong

   Sistem mengalokasikan kanvas probabilitas kosong berukuran (H x W x C) untuk menampung prediksi, serta peta hitungan untuk mencatat frekuensi tumpang tindih di setiap koordinat piksel.

2. *Averaging*

   Setiap kali sebuah *patch* diprediksi, nilai probabilitasnya ditambahkan ke lokasi koordinat yang sesuai di kanvas utama. Pada area *overlap*, nilai probabilitas dari berbagai *patch* yang bertumpuk akan terakumulasi. Nilai akhir didapatkan dengan membagi total probabilitas dengan jumlah tumpukan di setiap piksel.

$$P_{pixel} = \frac{\sum Patch}{N_{overlap}}$$

3. *Final Mask Generation*

   Setelah seluruh area citra terproses, fungsi Argmax diterapkan pada dimensi kelas (*channel*) untuk menentukan label kelas final dengan probabilitas tertinggi pada setiap piksel.

3.3.5 Visualisasi Luaran

Tahap akhir dari alur inferensi adalah transformasi data matriks menjadi informasi visual yang dapat diinterpretasikan. Hasil rekonstruksi indeks kelas dikonversi kembali menjadi citra RGB menggunakan tabel pemetaan warna yang telah ditetapkan sebelumnya. Sistem menghasilkan tiga jenis luaran utama:

1. Peta Prediksi Kelas: Merupakan visualisasi utama hasil segmentasi model, di mana setiap piksel telah diklasifikasikan dan dikonversi menjadi warna tematik sesuai indeks kelas. Peta ini berfungsi untuk menampilkan zonasi risiko (misal: membedakan area salak dan inang alternatif) tanpa interferensi visual dari tekstur citra asli.

2. Peta *Overlay* Transparan (α = 0,35): Visualisasi tumpang tindih antara peta prediksi dengan citra asli drone. Mode ini berfungsi untuk memetakan posisi risiko secara kontekstual di atas kondisi lapangan, memudahkan pengguna dalam mengidentifikasi lokasi spesifik inang alternatif atau titik rawan lainnya.
3. Peta *Difference*: Khusus untuk citra uji yang memiliki *masks* (*Ground Truth*), sistem menghasilkan visualisasi *side-by-side* yang menyandingkan Label Manual dengan Prediksi Model, serta menonjolkan area selisihnya. Visualisasi ini memetakan piksel *False Positive* dan *False Negative* dengan warna kontras (misalnya merah), yang berfungsi untuk diagnosis kesalahan model secara visual.

Selain luaran visual, sistem juga secara otomatis menghitung dan menampilkan kembali nilai metrik evaluasi kuantitatif pasca-rekonstruksi yang sama dengan parameter pengujian pada Sub-bab 3.2.6, meliputi mIoU, IoU per kelas, *Overall Accuracy* (OA), F1-Score, serta *Producer's* dan *User's Accuracy* (PA/UA) per kelas.