

Improvisasi Feature Extractor dan Penskalaan pada Model YOLOv7-Tiny untuk Meningkatkan Performa Deteksi Gulma pada Tanaman Jagung

Pendahuluan

1. Latar Belakang

Gulma merupakan tanaman liar yang tumbuh pada suatu area tanah yang tidak dikehendaki. Gulma dapat tumbuh di lahan pertanian, kebun, taman, dan lingkungan lainnya (Umiyati dan Widayat 2017). Pada tanaman Jagung, gulma dapat memberikan dampak negatif pada tanaman jagung karena dapat bersaing untuk mendapatkan sumber daya yang diperlukan untuk pertumbuhan, menjadi inang bagi hama dan penyakit yang dapat menyerang Jagung, menurunkan kualitas Jagung, dan meningkatkan biaya produksi Jagung (Redaksi Agromedia 2007). Eksperimen lapangan pernah dilakukan selama musim tanam pada tahun 2005-2007 di Stasiun Penelitian Pertanian Regional, Lam, Guntur, Andhra Pradesh untuk menemukan praktik pengendalian gulma yang paling efektif dalam budidaya jagung tanpa olah tanah setelah tanaman padi. Pertumbuhan gulma yang tidak terkendali sepanjang periode pertumbuhan tanaman menyebabkan penurunan pertumbuhan tanaman yang dimulai dari biji sebesar 43% (Rao et al. 2009). Oleh karena itu pentingnya melakukan deteksi Gulma pada lahan Jagung dengan baik agar dapat dilakukan pengendalian secara efektif sehingga dapat meningkatkan kualitas Jagung yang dihasilkan.

Deteksi gulma dapat dilakukan dengan berbagai cara. Pertama dengan cara melakukan observasi mandiri dan berkonsultasi dengan pakar yang dapat memvalidasi apakah objek tersebut gulma atau bukan berdasarkan morfologinya (Wang et al. 2019). Cara ini membutuhkan waktu dan tenaga dalam melakukan observasi dan berkonsultasi dengan pakar terkait. Oleh karena itu, perlu pendekatan *computer vision* agar dapat mengidentifikasi gulma secara efektif dan efisien. Pendekatan *computer vision* untuk deteksi objek dapat dilakukan secara tradisional dan berbasis *deep learning* (Wang et al. 2019).

Pendekatan secara tradisional dilakukan dengan mengekstrak karakteristik informasi dari gambar baik itu berupa warna, tekstur maupun bentuk. Berbagai penelitian mengenai deteksi objek pada bidang pertanian telah dilakukan menggunakan pendekatan secara tradisional. Penelitian terkait pertama dilakukan oleh (Murawwat et al. 2018) mengenai deteksi gulma menggunakan algoritme SVM (*Support Vector Machine*). Penelitian ini mendapatkan hasil akurasi berkisar dari 90%. Kendati demikian penelitian ini belum mampu mendeteksi gulma pada lingkungan asli yang memiliki keberagaman objek sehingga diperlukannya teknik praproses dan augmentasi citra yang lebih bervariasi untuk mengatasi hal tersebut. Penelitian selanjutnya dilakukan oleh Sinlae et al. (2022) mengenai klasifikasi gulma *broadleaf* menggunakan kombinasi dari KNN (*K-Nearest Neighbor*) dan PCA (*Principal Component Analysis*). Hasilnya akurasi rata-rata pada pengujian diperoleh sebesar 90%. Kendati demikian, masih terdapat eror pada model disaat menguji citra pada kondisi asli di lapangan karena data yang digunakan terlalu sedikit dan belum ada teknik praproses citra untuk menangani *background* yang terlalu bervariasi pada citra. Permasalahan yang muncul dari metode deteksi objek tradisional yaitu dari segi waktu deteksi dan akurasi yang dihasilkan pada saat pengujian berlangsung. Metode deteksi objek tradisional tidak dapat memenuhi persyaratan tersebut karena mereka perlu menyediakan fitur artifisial terlebih dahulu. Deteksi objek berdasarkan pendekatan *deep learning* dapat memainkan peran yang baik dalam situasi ini karena mampu

mendeteksi objek dengan lebih cepat dan akurasi yang dapat dipertimbangkan (Zou et al. 2019; Liu et al. 2020).

Berbagai penelitian terkait telah dilakukan menggunakan metode *deep learning*. Penelitian terkait pertama dilakukan oleh (Sunil et al. 2022) mengenai klasifikasi biji gulma menggunakan SVM dan VGG 16 yang berbasis *deep learning*. Hasil penelitian ini menunjukkan bahwa akurasi pengujian yang diperoleh pada model SVM berkisar 77,7-92,7%, sedangkan pada model VGG berkisar 82,9-100%. Hal ini menunjukkan bahwa terdapat peningkatan performa ketika menggunakan model yang berbasis *deep learning* juga karena perlakuan praproses pada citra sebelum melakukan klasifikasi. Namun, terdapat beberapa hal yang perlu ditingkatkan dalam penelitian ini yaitu dengan melakukan investigasi terkait *classifier* yang digunakan agar dalam meningkatkan performa model. Penelitian terkait selanjutnya dilakukan oleh (Luo et al. 2023) mengenai klasifikasi biji gulma menggunakan arsitektur *deep learning* yaitu CNN (*Convolutional Neural Network*). Hasil penelitian ini menunjukkan bahwa akurasi pengujian yang diperoleh pada model yang berbasis *deep learning* diatas 90%. Kendati demikian, penelitian ini perlu ditingkatkan dalam hal praproses yang terdiri dari menyamakan ukuran input pada dataset, dan memperbanyak data yang memiliki fitur bentuk dan warna yang serupa namun berbeda ukuran objek.

Penelitian selanjutnya yang dilakukan oleh (Subeesh et al. 2022) mengenai deteksi gulma pada *polyhouse* paprika menggunakan *deep learning*. Penelitian ini mendapatkan hasil pengujian dengan akurasi rata-rata berkisar 94-98%. Pekerjaan potensial selanjutnya yang dapat dilakukan yaitu termasuk deteksi tanaman dan gulma secara *real-time* menggunakan perangkat tertentu dan pengendaliannya. Metode *deep learning* menggunakan CNN tradisional masih belum dapat digunakan secara *real-time*. Untuk itu perlu metode *deep learning* lain agar dapat melakukan deteksi secara *real-time*. Metode deteksi objek yang saat ini masih menjadi *state of the art* dari penelitian deteksi objek (Zou et al. 2019).

Metode *deep learning* saat ini terbagi menjadi 2 yaitu *one stage detector* dan *two stage detector* yang memiliki kelebihan masing-masing (Zou et al. 2019; Liu et al. 2020). *One stage detector* unggul dalam kecepatan deteksi contohnya YOLO sedangkan *two stage detector* unggul dalam akurasi deteksi sedangkan *two stage detector* unggul dalam akurasi contohnya Mask R-CNN (Zou et al. 2019; Liu et al. 2020). Pengendalian gulma sebaiknya dilakukan dengan cepat agar tanaman jagung dapat tumbuh dengan baik. Pendekatan deteksi objek berbasis *one-stage detector* dapat digunakan dan diharapkan dapat mendeteksi gulma dengan tepat dengan akurasi yang dapat dipertimbangkan. Penelitian ini akan menggunakan metode YOLO yang menjadi *state of the art* dari penelitian deteksi objek berbasis *one-stage detector*. YOLO tepat untuk digunakan dalam deteksi objek pada data stream, di mana objek-objek yang muncul dalam citra diproses secara langsung dan hasilnya dapat ditampilkan secara *real-time*.

Berbagai penelitian terkait metode YOLO telah dilakukan oleh banyak peneliti. Penelitian terkait pertama dilakukan oleh Liu et al. (2022) mengenai deteksi gulma pada bibit tanaman Jagung menggunakan *deep learning* berbasis YOLOv4-tiny secara *real-time*. Penelitian ini menghasilkan nilai mAP dari model yang diusulkan adalah 86,69%, dengan ukuran dan kecepatan deteksi yang diperoleh yaitu 34,08 MB & 33 f/s. Kendati demikian, penelitian ini terdapat kekurangan pada tahapan *feature extraction* yang terdapat pada model. Fitur-fitur dari beberapa target tidak diekstraksi dengan cukup baik untuk menangani data yang tidak seimbang.

Penelitian selanjutnya dilakukan oleh Zhao et al. (2022) mengenai deteksi jenis gulma pada lahan tanaman Kentang menggunakan Improved-YOLOv4. Algoritme yang diusulkan

menggantikan jaringan utama CSPDarknet53 dalam struktur jaringan YOLOv4 dengan jaringan MobileNetV3 yang ringan dan memperkenalkan konvolusi *Depthwise separable* sebagai pengganti konvolusi tradisional sebagian pada *Path Aggregation Network* (PANet), yang mengurangi biaya komputasi model dan mempercepat deteksinya. MAP yang diperoleh dari data uji yaitu 98,52%. Meskipun penggunaan model jaringan yang kompleks untuk deteksi gulma pada tanaman meningkatkan akurasi pengenalan, kecepatan deteksi tidak dapat memenuhi persyaratan *real-time* karena ukuran besar dari model jaringannya sehingga dibutuhkan arsitektur yang ringan yang juga mampu mengekstrak fitur-fitur pada citra dengan baik sehingga akurasi deteksi dapat dipertahankan.

Penelitian terkait selanjutnya dilakukan oleh Chen et al. (2022) yang melakukan deteksi gulma pada lahan tanaman wijen menggunakan YOLO yang telah ditambahkan *attention mechanism* dan *feature fusion*. Hasilnya cukup memuaskan penambahan teknik tersebut mampu mendeteksi gulma secara akurat dengan MAP sebesar 96,16% dan FPS 36,8 detik yang dapat diterapkan secara *real-time*. Penelitian ini menyarankan agar menerapkan arsitektur yang ringan pada model yang juga mampu mengekstrak fitur-fitur pada citra dengan baik sehingga akurasi deteksi dapat dipertahankan sehingga dapat diterapkan pada *embedded device*.

Penelitian selanjutnya dilakukan oleh (Ying et al. 2021) mengenai deteksi jenis gulma pada lahan tanaman Wortel menggunakan Improved-YOLOv4. Jaringan utama (*backbone network*) dari YOLOv4 diganti dengan jaringan saraf ringan yaitu MobileNetV3-Small. Eksperimen perbandingan menunjukkan bahwa model yang diusulkan mencapai kinerja keseluruhan yang lebih baik daripada YOLOv4, YOLOv4-tiny, YOLOv3, dan YOLOv3-tiny, seperti yang terbukti dari mAP (mean average precision) sebesar 88,46%, waktu deteksi rata-rata sebesar 12,65 ms, dan ukuran model sebesar 159,0 MB. Kendati demikian, penelitian ini perlu ditingkatkan dalam hal mengekstraksi fitur-fitur yang benar-benar berpengaruh dalam mendeteksi Gulma tanpa mengurangi kecepatan model.

Feature extraction merupakan permasalahan penting yang harus diperhatikan dalam model deteksi berbasis *deep learning* seperti YOLO. *Feature extraction* pada seri YOLO sebelumnya telah disempurnakan pada YOLOv7 yang merupakan seri YOLO yang menjadi *state of the art* saat ini. YOLOv7 telah menunjukkan kinerja yang lebih baik dibandingkan detektor objek lain seperti YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B, dan beberapa lainnya dalam hal kecepatan dan akurasi (Wang et al. 2022).

Berbagai penelitian terkait metode YOLOv7 telah dilakukan oleh banyak peneliti. Narayana and Ramana (2023) dalam penelitiannya mendeteksi jenis gulma menggunakan YOLOv7 secara *real-time*. Penelitian ini menghasilkan mAP@0.5 sebesar 80-99% untuk data yang relatif kecil. Model YOLOv7 memiliki keterbatasan dalam mendeteksi gulma yang kecil atau terhalang, maupun kesalahan mengklasifikasikan objek non-gulma sebagai gulma. Studi ini menggunakan dataset terbatas untuk melatih dan menguji model YOLOv7, yang dapat memengaruhi akurasi dan generalisabilitas hasilnya. Penelitian yang mungkin dilakukan di masa depan yaitu mencari strategi metode pelatihan model dan augmentasi data, meningkatkan kualitas dataset, dan meningkatkan arsitektur model.

Penelitian selanjutnya dilakukan oleh (Gallo et al. 2023) mengenai deteksi gulma dengan citra UAV menggunakan metode YOLOv7. Hasilnya, YOLOv7 menghasilkan hasil yang lebih baik dibandingkan varian YOLO lainnya untuk dataset CP (Chicory Plant) sebesar 56,6% untuk skor mAP@0,5, 62,1% untuk recall, dan 61,3% untuk presisi. Selain itu, model YOLOv7 yang diterapkan pada dataset LB (Lincoln Beet) berhasil melampaui hasil yang telah

dipublikasikan sebelumnya. Ketinggian pengambilan gambar UAV dapat mempengaruhi perspektif dan kualitas gambar, yang nantinya dapat mempengaruhi akurasi deteksi gulma. Dengan melakukan improvisasi *feature extractor* dan penskalaan gambar pada YOLOv7 diharapkan dapat membantu model dalam mendapatkan fitur-fitur penciri terpenting dan mendeteksi gambar dengan skala berbeda dengan tepat, waktu deteksi dan komputasi yang cepat. Selain itu model yang dirancang diharapkan dapat menghasilkan ukuran yang kecil agar harapannya dapat dimanfaatkan untuk diterapkan pada perangkat komputer yang dapat mendeteksi objek secara real-time.

Berdasarkan penelitian diatas, *feature extractor* berperan disini dalam meningkatkan ketahanan model dengan menyediakan fitur-fitur yang tepat dari objek yang ingin dideteksi. Kemudian penskalaan gambar dibutuhkan agar model mampu mendeteksi gambar dengan baik untuk berbagai skala yang berbeda. Saya ingin melakukan deteksi gulma pada lahan jagung dengan menyediakan *feature extractor* dan teknik penskalaan yang mampu meningkatkan ketahanan model dalam mendeteksi objek pada lingkungan yang bervariasi dan memiliki ukuran yang ringan agar dapat diterapkan pada *embedded device* baik *software* maupun *hardware* dalam kasus monitoring dan pengendalian Gulma khususnya pada tanaman Jagung.

2. Rumusan Masalah

Gulma dan Jagung masing-masing mempunyai fitur-fitur utama yang dapat membedakan keduanya. Dalam pemodelan *deep learning*, fitur-fitur utama akan diekstrak sehingga dapat menghasilkan akurasi yang baik. Selain itu faktor penskalaan yang berbeda dapat mempengaruhi hasil deteksi. Proses tersebut terkadang membuat model berjalan lambat dan memiliki ukuran yang besar. Permasalahannya jika mengesampingkan tahapan *feature extraction* dan penskalaan nantinya akurasi yang diperoleh akan menurun. Kontribusi Saya dalam penelitian ini yaitu membuat *feature extractor* dan teknik penskalaan optimal pada YOLOv7 yang mampu mengekstrak fitur-fitur utama pada citra tanpa mengurangi akurasi deteksi dan ukuran model serta mempercepat kinerja model deteksi. Hal ini dilakukan agar dapat diterapkan pada perangkat lunak. Sehingga dapat dirumuskan masalah yaitu bagaimana membangun *feature-extractor* dan teknik penskalaan pada model YOLOv7-tiny yang dapat meningkatkan performa model sebelumnya ?

3. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membangun teknik *feature-extraction* pada model YOLOv7-tiny yang dapat meningkatkan performa model sebelumnya.

4. Ruang Lingkup Penelitian

1. Data sekunder diperoleh dari <https://data.mendeley.com/datasets/jjbfckcrsp>.
2. Algoritma yang digunakan berbasis *one-stage detector* YOLOv7-tiny.

Tinjauan Pustaka

1. Gulma

Gulma merupakan tanaman liar yang tumbuh pada suatu area tanah yang tidak dikehendaki. Gulma dapat tumbuh di lahan pertanian, kebun, taman, dan lingkungan lainnya (Umiyati dan Widayat 2017). Gulma tidak mematikan tanaman budi daya namun menghambat pertumbuhan dan menurunkan hasil panen. Oleh karena itu, membuat manusia senantiasa

berusaha mengurangi atau memberantasnya. Gulma yang dihilangkan selama periode tumbuh pertanaman berlangsung disebut pemberantasan gulma (Winarsih 2008). Berdasarkan klasifikasi botani, tumbuhan pengganggu dibagi menjadi tiga kategori yaitu rumput, teki, dan tumbuhan daun lebar.

Gulma jenis rumput termasuk dalam keluarga Gramineae atau Poaceae. Gulma jenis ini mempunyai ciri-ciri batang yang berbentuk bulat atau sedikit pipih, sebagian besar berongga, daun dengan tulang sejajar, dan terdiri dari pelepah dan helaian daun. Beberapa contoh Gulma jenis rumput yang sering ditemukan adalah *Imperata cylindrica*, *Echinochloa crusgalli*, *Cynodon dactylon*, dan *Digitaria sanguinalis*. Gulma jenis teki termasuk dalam famili Cyperaceae. Ciri-ciri umum Gulma teki meliputi batang berbentuk segitiga atau bulat yang tidak berongga, serta daun berwarna ungu tua. Salah satu sifat yang mencolok adalah kemampuannya untuk membentuk umbi baru dengan cepat, yang dapat memasuki fase dorman dalam lingkungan tertentu. Beberapa contohnya antara lain *Cyperus difformis*, *Cyperus iria*, *C. esculentus*, dan *C. rotundus*. Gulma jenis daun lebar umumnya masuk ke dalam kelompok Dicotyledoneae dan Pteridophyta. Ciri-cirinya meliputi daun yang lebar dengan tulang daun berbentuk jala. Beberapa contoh gulma dari kelompok ini adalah *Monochoria vaginalis*, *Limncharis flava*, *Portulaca oleracea*, dan *Amaranthus spinosus* (Winarsih 2008). Pada tanaman Jagung, gulma dapat memberikan dampak negatif pada tanaman jagung karena dapat bersaing untuk mendapatkan sumber daya yang diperlukan untuk pertumbuhan, menjadi inang bagi hama dan penyakit yang dapat menyerang Jagung, menurunkan kualitas Jagung, dan meningkatkan biaya produksi Jagung (Budi Daya Jagung Hibrida 2007).

Eksperimen lapangan pernah dilakukan selama musim tanam pada tahun 2005-2007 di Stasiun Penelitian Pertanian Regional, Lam, Guntur, Andhra Pradesh untuk menemukan praktik pengendalian gulma yang paling efektif dalam budidaya jagung tanpa olah tanah setelah tanaman padi. Pertumbuhan gulma yang tidak terkendali sepanjang periode pertumbuhan tanaman menyebabkan penurunan pertumbuhan tanaman yang dimulai dari biji sebesar 43% (Rao et al. 2009).

Sebelum melakukan tindakan pengendalian gulma, sangat penting bagi kita mengetahui cara-cara pengendalian gulma sehingga dapat kita pilih yang paling tepat untuk suatu jenis tanaman budi daya dan gulma yang tumbuh di daerah tersebut. Beberapa teknik pengendalian gulma antara lain sebagai berikut (Winarsih 2008).

1. Pengendalian dengan upaya pencegahan (preventif), misalnya dengan membuat peraturan atau perundangan, karantina, sanitasi, dan peniadaan sumber invasi.
2. Pengendalian secara mekanis atau fisik, misalnya dengan pengolahan tanah, menyaingi, pencabutan, pembabatan, penggenangan, dan pembakaran.
3. Pengendalian secara kultur teknis, misalnya dengan penggunaan bibit jenis unggul, pemilihan waktu tanam yang tepat, pemilihan cara penanaman, penggunaan tanaman sela, rotasi tanaman, dan penggunaan mulsa.
4. Pengendalian secara hayati, misalnya dengan penggunaan musuh-musuh alami dari gulma.
5. Pengendalian secara kimiawi, misalnya dengan penggunaan herbisida, surfaktan, alat aplikasi, dan sebagainya.
6. Pengendalian secara terpadu, yaitu pengendalian gulma yang melibatkan beberapa cara pengendalian secara bersamaan dengan tujuan untuk mendapatkan hasil yang optimal. Hal tersebut dilakukan terutama jika gulma tidak dapat dikendalikan secara tuntas hanya dengan satu cara saja. Contohnya, pengendalian gulma secara mekanis dipadukan dengan cara kimiawi dan pengaturan jarak tanam.

7. Pengendalian dengan upaya memanfaatkannya untuk berbagai keperluan misalnya dijadikan sayur, makanan ternak, obat, biogas, pupuk, bahan kerajinan, dan sebagainya.

2. Persiapan Data untuk Model *Deep Learning*

Persiapan data merupakan proses mengubah data mentah menjadi format yang bisa digunakan untuk melatih model cerdas yang efektif. Persiapan data untuk model deteksi objek berbasis *deep learning* terbagi menjadi beberapa bagian yaitu praproses citra, augmentasi citra, anotasi citra, dan pembagian data (Bochkovskiy et al. 2020). Setiap tahapan harus diperhatikan dengan baik agar dapat menghasilkan performa model yang bagus.

a. Praproses Citra

Praproses citra dilakukan sesuai dengan apa yang ingin kita dapatkan. Dalam kasus citra untuk bidang pertanian, hal yang harus diperhatikan adalah seringnya terdapat blur dan noise pada citra dan rentang warna yang serupa.

b. Augmentasi Citra

Augmentasi citra adalah proses meningkatkan ukuran set data pelatihan dengan membuat versi yang dimodifikasi dari gambar asli. Teknik ini umumnya digunakan dalam aplikasi *deep learning* dan computer vision untuk meningkatkan performa klasifikasi gambar, deteksi objek, dan tugas-tugas lainnya.

c. Anotasi Citra

Anotasi citra adalah proses menambahkan metadata atau label ke sebuah citra. Metadata tersebut bisa berisi informasi seperti kelas objek, lokasi objek, dan masker segmentasi semantik. Peng-antaran gambar adalah tugas penting dalam aplikasi penglihatan komputer seperti deteksi objek, klasifikasi gambar, dan segmentasi semantik. Ada beberapa jenis anotasi citra, termasuk anotasi kotak pembatas (bounding box annotation), anotasi polygon (polygon annotation), segmentasi semantik (semantic segmentation), dan anotasi landmark (landmark annotation). Pemilihan jenis peng-antaran tergantung pada tugas spesifik dan tingkat detail yang diperlukan (Sager et al. 2021).

d. Pembagian Data

Pembagian data merujuk pada pembagian dataset menjadi subset yang terpisah untuk tujuan pelatihan, validasi, dan pengujian. Set pelatihan digunakan untuk melatih model, set validasi digunakan untuk menyetel hiperparameter model, dan set pengujian digunakan untuk mengevaluasi performa model yang dilatih (Joseph 2022).

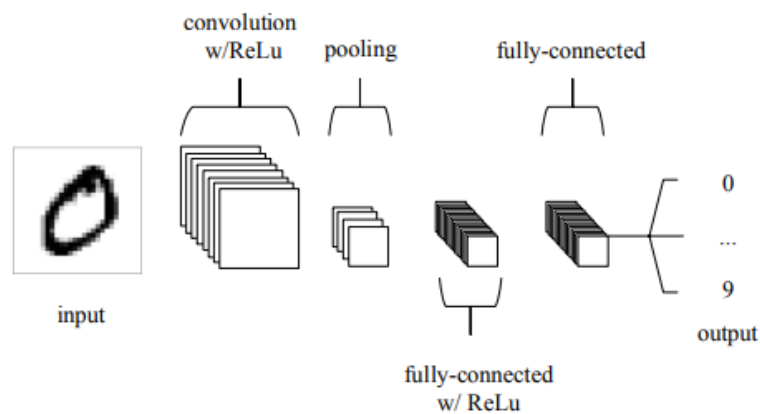
Pemilihan proporsi data yang hati-hati diperlukan untuk memastikan bahwa model dilatih dan dievaluasi pada sampel data yang representatif. Pembagian yang dipilih dengan buruk dapat menyebabkan overfitting, di mana model berperforma baik pada data pelatihan tetapi buruk pada data baru yang belum pernah dilihat.

Secara umum, proporsi data yang dibagi ke dalam set pelatihan, validasi, dan pengujian dapat bervariasi tergantung pada ukuran dataset dan kompleksitas tugasnya. Pembagian umum adalah 60% untuk pelatihan, 20% untuk validasi, dan 20% untuk pengujian, namun pembagian lain seperti 70-15-15 atau 80-10-10 juga dapat digunakan. Penting untuk memastikan bahwa pembagian data mewakili dataset secara keseluruhan dan bahwa tidak ada overlap antara pembagian untuk menghindari bias dalam evaluasi kinerja model (Joseph 2022).

3. Deteksi Objek Menggunakan Deep Learning

Deteksi objek merupakan bidang ilmu turunan dari *computer vision* yang bertujuan untuk mendeteksi dan menemukan objek yang menarik dalam gambar atau video. Tugasnya melibatkan mengidentifikasi posisi dan batas objek dalam sebuah gambar, dan mengklasifikasikan objek ke dalam kategori yang berbeda. Saat ini deteksi objek menggunakan *deep learning* terbagi menjadi 2 bagian yaitu berbasis deteksi satu tahap dan dua tahap. Metode satu tahap memprioritaskan kecepatan inferensi, dan model contoh termasuk YOLO dan SSD. Metode dua tahap memprioritaskan akurasi pendeteksian, dan model contoh termasuk Faster R-CNN, Mask R-CNN, dan Cascade R-CNN (Zou et al. 2023). Penelitian ini terfokus pada deteksi satu tahap karena memprioritaskan kecepatan inferensi. Namun sebelum membahas lebih jauh, saya akan membahas CNN yang merupakan dasar dari arsitektur *deep learning*.

CNN adalah metode *deep learning* yang digunakan untuk menyelesaikan tugas-tugas pengenalan pola berbasis gambar yang sulit dan dengan arsitekturnya yang tepat namun sederhana. CNN dibuat menggunakan pendekatan metode Jaringan Syaraf Tiruan yang telah dimodifikasi. Algoritme ini dibuat agar dapat menangani overfitting yang terjadi pada metode Jaringan Syaraf Tiruan (Saxena 2022). Arsitektur CNN dapat dilihat pada Gambar 2.



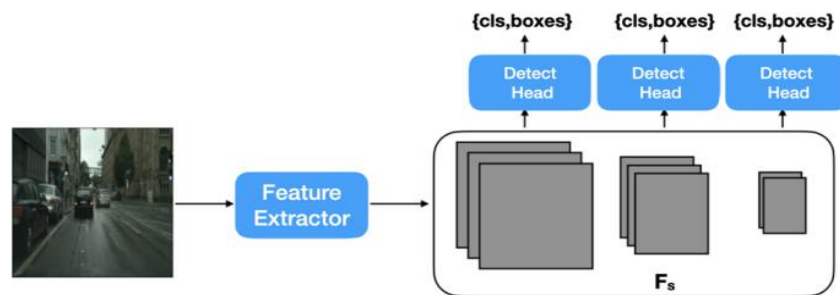
Gambar 1. Arsitektur CNN

CNN dapat dipecah menjadi empat bidang utama, yaitu:

1. Seperti yang ditemukan dalam bentuk JST lainnya, lapisan input akan menyimpan nilai piksel dari gambar.
2. Lapisan convolutional akan menentukan output dari neuron yang terhubung ke daerah lokal input melalui perhitungan produk skalar antara bobotnya dan wilayah yang terhubung ke volume input. Unit linier yang diperbaiki (biasanya disingkat ReLu) bertujuan untuk menerapkan Pengenalan Jaringan Syaraf Konvolusional 5 fungsi aktivasi 'elementwise' seperti sigmoid ke output aktivasi yang dihasilkan oleh lapisan sebelumnya.
3. Lapisan pooling kemudian hanya akan melakukan downsampling sepanjang dimensi spasial dari input yang diberikan, selanjutnya mengurangi jumlah parameter dalam aktivasi tersebut.
4. Lapisan fully-connected kemudian akan melakukan tugas yang sama seperti yang ditemukan di JST standar dan berupaya menghasilkan skor kelas dari aktivasi, yang akan digunakan untuk klasifikasi.

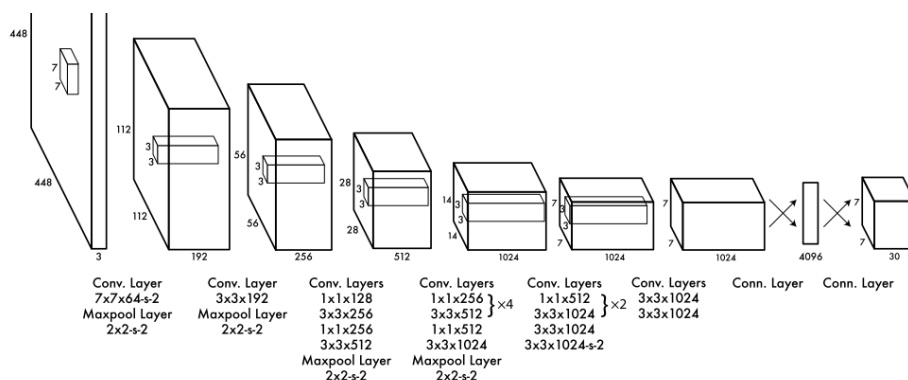
Pendekatan deteksi objek dengan deep learning dapat dibagi menjadi 2 yaitu one stage detector dan two stage detector. One stage detector mengandalkan kecepatan inferensi, sedangkan two stage detector lebih mengandalkan keakuratan (Soviany and Ionescu 2018), (Zou et al. 2019).

One stage detector memiliki arsitektur umum yang digambarkan pada Gambar 2. Arsitekturnya terdiri dari feature extractor diikuti oleh beberapa detection heads. Detection heads ini terdiri dari beberapa input fitur dengan skala yang berbeda sehingga memungkinkan detektor untuk secara efektif menangani objek dengan ukuran yang berbeda (Vidit and Salzmann 2022). One stage detector pada object detection layaknya regresi sederhana dengan mengambil gambar input dan mempelajari probabilitas kelas dan koordinat kotak pembatas pada gambar yang diambil (Soviany and Ionescu 2018).

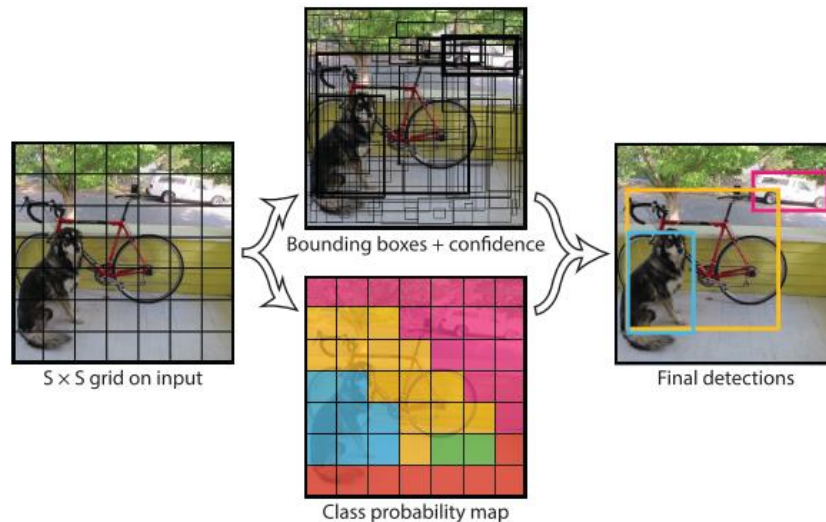


Gambar 2. Arsitektur Umum One-Stage Detector (Vidit and Salzmann 2022)

YOLO (*You Only Look Once*) adalah metode *one-stage detector* yang dipublish pada tahun 2016 yang mampu mendeteksi objek secara *realtime* dengan kecepatan 45 FPS. YOLO terbukti memiliki akurasi dan kecepatan deteksi yang lebih baik dibandingkan dengan metode sebelumnya yaitu R-CNN, DPM dan Poselets yang dievaluasi menggunakan dataset VOC 2007, Picasso, and People-Art Datasets (Redmon et al. 2016). YOLO memiliki arsitektur yang berdasarkan jaringan konvolusional. YOLO memiliki 24 lapisan konvolusi diikuti oleh 2 lapisan yang terhubung sepenuhnya pada jaringan deteksinya. Lapisan convolutional 1×1 bergantian mengurangi ruang fitur dari lapisan sebelumnya. Lapisan konvolusional telah dilatih pada tugas klasifikasi ImageNet dengan resolusi gambar sebesar 224×224 (Redmon et al. 2016). Arsitektur tersebut dapat dilihat pada Gambar 5.



Gambar 5. Arsitektur YOLO (Redmon et al. 2016)



Gambar 6. Model Deteksi YOLO (Redmon et al. 2016)

Gambar 6 menunjukkan gambaran model deteksi YOLO yang merupakan permasalahan regresi. Model tersebut membagi gambar menjadi kisi $S \times S$ dan untuk setiap sel kisi memprediksi *bounding box* B , tingkat kepercayaan untuk kotak tersebut, dan probabilitas kelas C . Prediksi ini dikodekan sebagai tensor $S \times S \times (B * 5 + C)$. Setiap kotak pembatas terdiri dari 5 prediksi: x , y , w , h , dan kepercayaan. Koordinat (x, y) mewakili pusat kotak relatif terhadap batas sel kisi, koordinat (w, h) mewakili ukuran relatif terhadap keseluruhan gambar dan koordinat h mewakili tingkat kepercayaan yang mencerminkan seberapa yakin model bahwa kotak itu berisi objek dan juga seberapa akurat prediksi objek didalam kotak tersebut (Redmon et al. 2016).

YOLOv2 memiliki beberapa peningkatan, antara lain penggunaan kotak jangkar untuk mendeteksi objek dengan ukuran berbeda, teknik kombinatorial multiskala untuk meningkatkan akurasi deteksi, dan proses pelatihan yang lebih kompleks dengan kumpulan data yang lebih besar dan beragam memberikan beberapa peningkatan yang signifikan (Redmon and Farhadi 2017). YOLOv3 menggunakan tulang punggung Darknet-53 yang lebih kuat, melakukan deteksi pada tiga skala berbeda untuk meningkatkan deteksi objek kecil, menggunakan fitur seperti link skipping dan upsampling untuk meningkatkan representasi fitur, YOLOv3 meningkatkan kinerja dibandingkan YOLOv2 dengan mendukung deteksi kumpulan data yang sangat besar (Redmon and Farhadi 2018). Lalu datanglah YOLOv4, menggabungkan berbagai teknik mutakhir seperti menggunakan *backbone* CSPDarknet53, teknik pengoptimalan aktivasi Mish, dan teknik augmentasi data tingkat lanjut. YOLOv4 juga memperkenalkan fitur seperti *Panoptic Feature Pyramid Networks* (PFPN) dan *Path Aggregation Network* (PAN) untuk meningkatkan deteksi dan lokalitas objek (Bochkovskiy et al. 2020). YOLOv5, yang semakin meningkatkan performa model dan menambahkan fitur baru seperti dukungan untuk segmentasi panoptik dan pelacakan objek (Jocher 2020). YOLOv6 menghadirkan berbagai peningkatan seperti menggunakan *backbone* RepVGG atau CSPStackRep, topologi neck PAN, dan head dengan strategi hybrid-channel serta penerapan teknik kuantisasi yang ditingkatkan juga membuatnya lebih cepat dan akurat dalam mendeteksi objek (Li et al. 2022). YOLOv7 mengusulkan beberapa perubahan arsitektur dan serangkaian teknik bag-of-freebies, yang meningkatkan akurasi tanpa mempengaruhi kecepatan inferensi, hanya mempengaruhi waktu pelatihan (Wang et al. 2022).

precision dan *recall* tinggi, dan rendah ketika salah satunya rendah di bawah rentang nilai ambang kepercayaan (Anwar A 2022). Nilai ini bertujuan untuk mengukur sejauh mana sistem deteksi objek mampu menghasilkan hasil yang akurat dan konsisten. Rumus AP dihitung dengan membandingkan hasil deteksi dengan *ground truth* (label yang benar). Setelah nilai AP didapatkan, maka nilai MAP dapat diperoleh.

$$\text{Average Precision (AP)} = \int_{r=0}^1 p(r) dr \quad (1)$$

Di mana:

- AP adalah Average Precision
- r adalah *recall*, yaitu proporsi objek yang berhasil dideteksi terhadap total objek yang ada ($0 \leq r \leq 1$)
- $p(r)$ adalah presisi pada nilai *recall* r , yaitu proporsi objek yang benar terdeteksi terhadap total objek yang terdeteksi pada tingkat *recall* r

$$\text{Mean Average Precision (MAP)} = \frac{1}{k} \sum_i^k AP_i \quad (2)$$

Di mana:

- MAP adalah Mean Average Precision
- k adalah jumlah kelas atau kategori objek yang ada dalam dataset
- AP_i adalah Average Precision untuk kelas atau kategori objek ke- i

5. *Frame per Second (FPS)*

FPS (Frame Per Second) adalah ukuran yang digunakan untuk mengukur seberapa banyak frame atau gambar yang dapat ditampilkan atau diproses dalam satu detik. Dalam konteks permainan video atau pengolahan gambar dan video, FPS menunjukkan seberapa cepat gambar-gambar tersebut diperbarui atau diproses untuk menciptakan ilusi gerakan yang halus. Rumus FPS adalah sebagai berikut:

$$FPS = \frac{1}{T}$$

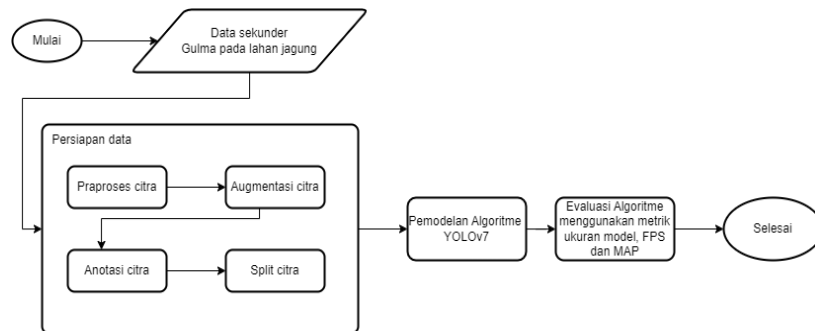
Di mana:

- FPS adalah Frame Per Second
- T adalah waktu yang dibutuhkan untuk memperbarui atau memproses satu frame

Metode

Metode penelitian yang dilakukan dapat dilihat pada Gambar 1. Penelitian dimulai dengan pencarian data sekunder yang akan digunakan dalam penelitian. Data sekunder yang digunakan diperoleh dari mendeley pada link berikut <https://data.mendeley.com/datasets/jjbfcckrsp>. Setelah memperoleh data sekunder, langkah selanjutnya adalah melakukan persiapan data. Persiapan data dilakukan untuk memperoleh data yang mempunyai kualitas dan kuatitas yang cukup baik untuk dimodelkan menggunakan algoritme *deep learning*. Kemudian, tahapan selanjutnya adalah pemodelan algoritme YOLOv4. Kedua algoritme ini akan dimodifikasi arsitekturnya guna mendapatkan model yang mempunyai ukuran yang kecil, kecepatan deteksi yang cepat, dan akurasi yang dapat dipertimbangkan. Tahap selanjutnya adalah mengevaluasi algoritme terkait menggunakan

metrik pengujian ukuran model, FPS (*Frame per Second*) dan MAP (*Mean Average Precision*). Langkah terakhir adalah implementasi dan pengujian model berbasis android.



Gambar 9. Metode penelitian

1. Data

Data sekunder diperoleh dari mendeley dengan link berikut: <https://data.mendeley.com/datasets/jjbfckrsp>. Dataset berisi gambar tanaman jagung dan spesies gulma. Dataset ini berisi 36874 gambar secara total dan disimpan dalam empat folder yaitu Gambar Jagung-Weed Beranotasi, Deskripsi Data dan Kuesioner, Gambar Gulma Jagung Musim Kemarau, dan Gambar Gulma Jagung Musim Hujan. Musim Kemarau berisi 18187 gambar yang diambil selama survei pertanian musim kemarau, Musim Hujan berisi 18187 gambar yang diambil selama survei pertanian musim hujan dan Anotasi berisi 500 gambar beranotasi yang dipilih dari survei Musim Kemarau yang disimpan dalam format JSON, XML, dan txt. Anotasi dicapai dengan menggunakan suite Labelmg. Gambar mentah musim hujan dan kemarau telah ditangkap menggunakan kamera digital beresolusi tinggi selama survei gulma, sedangkan anotasi gambar beranotasi dilakukan menggunakan suite Labelmg. Data perlu ditingkatkan karena data anotasi sebanyak 500 masih perlu dilakukan praproses dan augmentasi sehingga akan dilakukan proses persiapan data.

2. Persiapan data

Praproses citra

Tahapan ini berguna untuk memproses data agar mempunyai resolusi citra yang sama. Teknik *resize* diaplikasikan dengan mengubah ukuran citra menjadi input yang dibutuhkan oleh model yaitu 416 x 416 piksel. Praproses dilakukan menggunakan Python.

Augmentasi citra

Tahapan ini dilakukan guna untuk memperbanyak kuantitas data yang digunakan agar model dapat bertahan untuk beragam kondisi lingkungan pengujian. Teknik yang digunakan adalah (*flip*, *rotate*, *shearing* dan *cropping*). Tabel 1 menunjukkan pengaturan untuk setiap teknik augmentasi yang dilakukan. Augmentasi dilakukan menggunakan Roboflow (Roboflow, 2023).

Tabel 1. Pengaturan Teknik Augmentasi Citra

No	Teknik Augmentasi	Pengaturan
1	<i>Flip</i>	Horizontal
2	<i>Rotation</i>	-15° & 15°
3	<i>Shear</i>	-15° vertikal & 15° horizontal
4	<i>Hue</i>	-15° & 15°

5	<i>Brightness</i>	Diantara -25 & 25%
---	-------------------	--------------------

Anotasi citra

Tahapan ini dilakukan dengan memberikan kotak pembatas untuk setiap kelas. Kelas yang digunakan yaitu gulma dan jagung. Gambar 10 menunjukkan contoh anotasi pada citra. Anotasi dilakukan menggunakan Roboflow (Roboflow, 2023). Semua gambar yang telah dianotasi akan diekspor kedalam format YOLOv7 Pytorch. Deskripsi format file anotasi tersebut dapat dilihat pada Tabel 2.



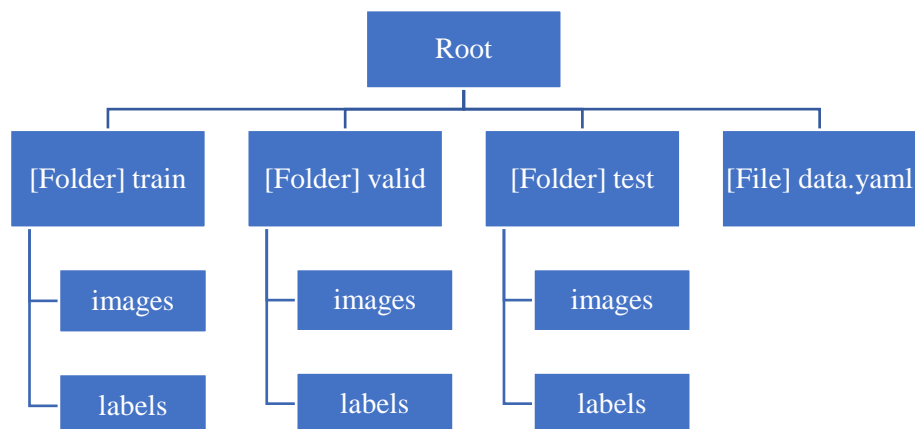
Gambar 10. Anotasi Citra

Tabel 2. Deskripsi Format File Anotasi YOLOv7 Pytorch

No	Nama	Keterangan
1	<i>Class_id</i>	Id kelas yang akan dilatih
2	<i>Center_x</i>	Titik tengah <i>bounding box</i> pada sumbu x
3	<i>Center_y</i>	Titik tengah <i>bounding box</i> pada sumbu y
4	<i>Width</i>	Lebar dari <i>bounding box</i>
5	<i>Height</i>	Tinggi dari <i>bounding box</i>

Split citra

Tahapan ini akan membagi citra kedalam 3 jenis data yaitu data latih, validasi dan uji dengan proporsi 70, 30 dan 10%. Struktur folder dari dataset YOLOv7 Pytorch dapat dilihat pada Gambar 11. Folder *images* berisi file citra dan folder *labels* berisi file anotasi. Kedua file ini harus mempunyai nama file yang sama. File data dengan ekstensi yaml merupakan berisi total kelas, nama kelas, dan lokasi folder *train*, *valid* dan *test*.



Gambar 11. Struktur Folder Dataset YOLOv7 Pytorch

3. Pemodelan algoritme YOLO

Penentuan *hyperparameter*

Tahapan pemodelan dimulai dengan menentukan *hyperparameter* yang akan digunakan. *Hyperparameter* bawaan akan digunakan dalam melatih model YOLOv7 yang ditunjukkan pada Tabel 3.

Table 3. *Hyperparameter default* pada YOLOv7

Attribute Names	Values
lr0	0.01
lrf	0.1
momentum	0.937
weight_decay	0.0005
warmup_epochs	3.0
warmup_momentum	0.8
warmup_bias_lr	0.1
box	0.05
cls	0.3
cls_pw	1.0
obj	0.7
obj_pw	1.0
iou_t	0.20
anchor_t	4.0
fl_gamma	0.0
hsv_h	0.015
hsv_s	0.7
hsv_v	0.4
degrees	0.0
translate	0.2
scale	0.9
shear	0.0
perspective	0.0
flipud	0.0
fliplr	0.5
mosaic	1.0
mixup	0.15
copy_paste	0.0
paste_in	0.15
loss_ota	1

Kustomisasi *arsitektur*

Selanjutnya adalah melakukan kustomisasi arsitektur pada model YOLOv7. Saya gunakan model YOLOv7-tiny karena model ini memiliki parameter yang terkecil yaitu berjumlah 6,2 juta dengan FPS terbesar yang mencapai 286 FPS pada GPU V100 dibandingkan arsitektur lainnya yang ada pada YOLOv7 (Wang et al. 2022). Model YOLOv7-tiny cocok diterapkan pada perangkat tertentu yang dapat berguna misalnya pada Robot pembasmi Gulma otomatis. Arsitektur model YOLOv7-tiny dapat dilihat pada Gambar 12.

Backbone modifikasi kedua dapat dilihat pada Gambar 14. Saya menambahkan blok residual dan modul SPP (*Spatial Pooling Layer*) pada *backbone*. Layer awal pertama [ReOrg] dibuat untuk menggantikan konvolusional pada YOLOv7-tiny untuk memperkaya representasi spasial dalam arsitektur jaringan. Kemudian dilanjutkan dengan penggunaan blok residual nilai 2x dari layer sebelumnya dan modul SPP dengan nilai 2x dari blok residual untuk blok pertama dan ketiga.

```
backbone:
  # [from, number, module, args] c2, k=1, s=1, p=None, g=1, act=True
  [[[-1, 1, ReOrg, []],
    [-1, 1, Conv, [32, 3, 2, None, 1, nn.LeakyReLU(0.1)]]],

    [-1, 1, Res, [64]],
    [-1, 1, DownC, [128]],

    [-1, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, MP, []], # 8-P3/8
    [-1, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [64, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [64, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, Res, [256]],
    [-1, 1, DownC, [512]],
    [-1, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [128, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [128, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, MP, []], # 8-P3/8
    [-1, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [256, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [256, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [512, 1, 1, None, 1, nn.LeakyReLU(0.1)]]]
```

Gambar 14. Modifikasi 2 *Backbone* dengan Residual Block dan SPP

Backbone modifikasi ketiga dapat dilihat pada Gambar 15. Saya menambahkan blok residual dan modul SPP (*Spatial Pooling Layer*) pada *backbone*. Layer ReOrg ditambahkan pada YOLOv7-tiny setelah layer konvolusional untuk memperkaya representasi spasial dalam arsitektur jaringan. Kemudian dilanjutkan dengan penggunaan blok residual nilai 2x dari layer sebelumnya untuk blok pertama dan terakhir. Modul SPP juga ditambahkan dengan nilai 2x dari blok residual untuk blok kedua dan ketiga.

```
backbone:
  # [from, number, module, args] c2, k=1, s=1, p=None, g=1, act=True
  [[[-1, 1, Conv, [32, 3, 2, None, 1, nn.LeakyReLU(0.1)]]], # 0-P1/2
    [-1, 1, ReOrg, []],

    [-1, 1, Res, [64]],
    [-1, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, DownC, [128]],
    [-1, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [64, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [64, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, DownC, [256]],
    [-1, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [128, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [128, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [128, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],

    [-1, 1, MP, []], # 8-P3/8
    [-1, 1, Res, [512]],
    [-1, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-2, 1, Conv, [256, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [256, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [-1, 1, Conv, [256, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
    [[-1, -2, -3, -4], 1, Concat, [1]],
    [-1, 1, Conv, [512, 1, 1, None, 1, nn.LeakyReLU(0.1)]]]
```

Gambar 15. Modifikasi 3 *Backbone* dengan Residual Block dan SPP

4. Evaluasi algoritme menggunakan metrik ukuran model, FPS dan MAP
Hal yang akan dievaluasi yaitu dari segi ukuran model, kecepatan deteksi dengan satuan FPS dan MAP untuk akurasi model yang dihasilkan.

Hasil dan Pembahasan

A. Data

Data yang digunakan sebanyak 1.968 gambar yang diambil dari data sekunder dengan musim yang berbeda yaitu musim kemarau dan musim hujan. Citra keduanya ditunjukkan pada Gambar 16.



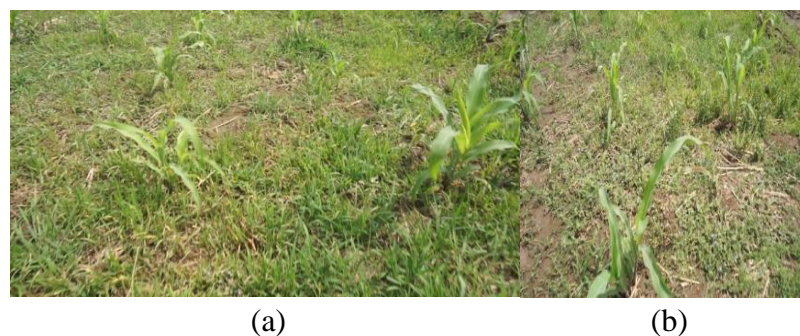
Gambar 16. (a) Musim kemarau, (b) Musim hujan

B. Persiapan Data

1. Praproses:

Resize

Data yang berukuran 768 x 432 piksel di ubah ukurannya menjadi 416 x 416 piksel sesuai dengan format dataset YOLOv7 Pytorch. Gambar 17 menunjukkan perbedaan dari keduanya. Praproses hanya melakukan *resize* karena pada YOLOv7 teknik praproses sudah tersedia dalam parameter. Namun untuk melihat hasil nyata citra yang telah diproses secara otomatis dari model itu belum tersedia dari repository YOLOv7 itu sendiri.



Gambar 17. (a) Citra ukuran 768 x 432 piksel, (b) Citra ukuran 416 x 416 piksel

Augmentasi

Hasil dari proses *resizing* citra kemudian akan diaugmentasi agar dapat memperbanyak kuantitas dataset sehingga memberikan kemampuan pada model untuk mendeteksi citra dalam berbagai kondisi. Teknik augmentasi yang digunakan dapat dilihat pada Tabel 1 diatas. Sebelum mendapatkan data latih, validasi dan uji, Saya lakukan pembagian sementara guna mendapatkan proporsi yang sesuai dengan yang

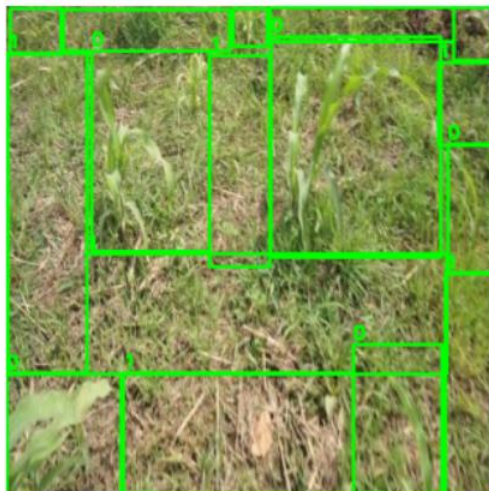
diinginkan yaitu 70:20:10. Gambar 18 menunjukkan pembagian sementara tersebut. Sebanyak 855 gambar akan dilakukan augmentasi hingga berjumlah 2.775. Data yang dilakukan augmentasi hanya pada data latih karena data latih sangat berperan penting dalam keberhasilan deteksi objek.



Gambar 18. Pembagian Sementara

Anotasi

Anotasi dilakukan menggunakan tools Roboflow. Data sekunder hanya menyediakan 500 file yang telah teranotasi. 1000 citra yang belum teranotasi akan dianotasi sendiri menggunakan tools Roboflow. Gambar 19 menunjukkan contoh hasil anotasi pada 1 citra. Label 0 merepresentasikan Jagung dan label 1 merepresentasikan Gulma.



Gambar 19. Contoh Hasil Anotasi

Split data

Setelah data dianotasi, langkah selanjutnya adalah pembagian data menjadi 3 bagian yaitu data latih, validasi dan uji. Tabel 4 menunjukkan proporsi data. Tabel 5 menunjukkan pembagian data latih dan data uji.

Tabel 4. Total citra untuk Setiap Jenis Data

No	Jenis Data	Jumlah
1	Train	2.565
2	Valid	745
3	Test	368
Total		3.678

Tabel 5. Total Objek perkelas untuk Setiap Jenis Data

No	Jenis Data	Objek	Jumlah
1	Train	Maize	14.733
2	Train	Weed	14.814
3	Valid	Maize	4.263
4	Valid	Weed	4.299
5	Test	Maize	2.125
6	Test	Weed	2.151

C. Pemodelan YOLOv7-Tiny

Kustomisasi Backbone YOLOv7-Tiny

Modifikasi pertama dilakukan dengan menambahkan blok residual dan modul SPP (Spatial Pooling Layer) pada backbone. Layer ReOrg ditambahkan pada YOLOv7-tiny setelah layer konvolusional untuk memperkaya representasi spasial dalam arsitektur jaringan. Kemudian dilanjutkan dengan penggunaan blok residual nilai 2x dari layer sebelumnya untuk blok pertama dan terakhir. Modul SPP juga ditambahkan dengan nilai 2x dari blok residual untuk blok kedua dan ketiga. Hasilnya parameter awal dari YOLOv7-tiny yang mempunyai layer sebanyak 263, 6.023.106 parameter, dan 6.023.106 gradien berhasil dikurangi parameter dan gradien nya menjadi masing-masing 5.701.730 parameter, 5.701.730 gradien meskipun layer bertambah menjadi 299 layer.

Modifikasi kedua dilakukan menambahkan blok residual dan modul SPP (Spatial Pooling Layer) pada backbone. Layer awal pertama [ReOrg] dibuat untuk menggantikan konvolusional pada YOLOv7-tiny untuk memperkaya representasi spasial dalam arsitektur jaringan. Kemudian dilanjutkan dengan penggunaan blok residual nilai 2x dari layer sebelumnya dan modul SPP dengan nilai 2x dari blok residual untuk blok pertama dan ketiga. Hasilnya parameter awal dari YOLOv7-tiny yang mempunyai layer sebanyak 263, 6.023.106 parameter, dan 6.023.106 gradien meningkat sebanyak 2 juta parameter gradien nya menjadi masing-masing 6.481.378 parameter, 6.481.378 gradien dan layer bertambah menjadi 312 layer.

Modifikasi ketiga dilakukan menambahkan blok residual dan modul SPP (Spatial Pooling Layer) pada backbone. Layer awal pertama [ReOrg] dibuat untuk menggantikan konvolusional pada YOLOv7-tiny untuk memperkaya representasi spasial dalam arsitektur jaringan. Kemudian dilanjutkan dengan penggunaan blok residual nilai 2x dari layer sebelumnya dan modul SPP dengan nilai 2x dari blok residual untuk blok pertama dan ketiga. Hasilnya parameter awal dari YOLOv7-tiny yang mempunyai layer sebanyak 263, 6.023.106 parameter, dan 6.023.106 gradien meningkat sebanyak 2 juta parameter gradien nya menjadi masing-masing 6.465.730 parameter, 6.465.730 gradien dan layer bertambah menjadi 310 layer.

Pelatihan Model

Model dilatih menggunakan *script* yang ada pada Gambar 20. Nilai pada {} yang digunakan adalah 64 untuk batch size, konfigurasi file dari model yolov7-tiny, epoch berjumlah 50, ukuran gambar 416, data yang digunakan yaitu file dari dataset berformat yaml, *weight* nya dari model YOLOv7-tiny dan device menggunakan GPU Tesla T4.

```
!python train.py --batch {batch size} --cfg {config.yaml} --epochs {num of epoch} --img-size {image size}
--data {data.yaml} --weights {yolov7-tiny.yaml} --device 0
```

Gambar 20. *Script* pelatihan

D. Evaluasi Model YOLOv4-Tiny

Evaluasi menggunakan Data Uji

1. Original YOLOv7-Tiny

```
test: Scanning 'Weed-Maize-Detection-2-10/test/labels' images and labels... 368 found, 0 missing, 0 empty, 0 corrupted: 100% 368/368
test: New cache created: Weed-Maize-Detection-2-10/test/labels.cache
      Class      Images      Labels      P      R      mAP@.5      mAP@.5:.95:      0% 0/12 [00:00<?, ?it/s]/usr/local/li
return _VF.meshgrid(tensors, **kwargs) # type: ignore[attr-defined]
      Class      Images      Labels      P      R      mAP@.5      mAP@.5:.95: 100% 12/12 [02:07<00:00, 10.64s/it]
      all         368         4276      0.758      0.72      0.755      0.4
      0           368         2125      0.87       0.873     0.906      0.51
      1           368         2151      0.646      0.566     0.604      0.289
Speed: 335.4/1.0/336.4 ms inference/NMS/total per 640x640 image at batch-size 32
```

Gambar 21. Evaluasi Model YOLOv7-Tiny Original

MAP dengan IoU 0.5 yang diperoleh pada model ini adalah 0,755 dan MAP dengan IoU 0.95 sebesar 0,4. MAP pada objek Jagung dengan IoU 0.5 yang diperoleh pada model ini adalah 0,906 dan MAP dengan IoU 0.95 sebesar 0,51. MAP pada objek Gulma dengan IoU 0.5 yang diperoleh pada model ini adalah 0,604 dan MAP dengan IoU 0.95 sebesar 0,289. Waktu yang tercatat adalah 335.4 milidetik untuk tahap inferensi, 1.0 milidetik untuk tahap operasi NMS, dan total waktu 336.4 milidetik untuk keseluruhan proses. Ukuran model yang terbaik dihasilkan sebesar 12,3 MB.

Dalam kasus ini, total waktu adalah 336,4 milidetik dan batch-size adalah 32 gambar. Maka dapat dihitung FPS (*Frame per Second*) dalam mendeteksi 1 set data uji.

$$\text{FPS} = 1 / (336,4 / 32) = 1 / 10,5125 = 0,1\text{fps}$$

2. Modifikasi pertama

```
test: Scanning 'Weed-Maize-Detection-2-10/test/labels.cache' images and labels... 368 found, 0 missing, 0 empty, 0 corrupted: 100% 368/368
      Class      Images      Labels      P      R      mAP@.5      mAP@.5:.95: 100% 12/12 [00:12<00:00, 1.00s/it]
      all         368         4276      0.698      0.685     0.696      0.3
      0           368         2125      0.804      0.858     0.868      0.38
      1           368         2151      0.591      0.511     0.523      0.22
Speed: 6.1/2.2/8.4 ms inference/NMS/total per 640x640 image at batch-size 32
```

Gambar 22. Evaluasi Model YOLOv7-Tiny Modifikasi 1

MAP dengan IoU 0.5 yang diperoleh pada model ini adalah 0,696 dan MAP dengan IoU 0.95 sebesar 0,3. MAP pada objek Jagung dengan IoU 0.5 yang diperoleh pada model ini adalah 0,868 dan MAP dengan IoU 0.95 sebesar 0,38. MAP pada objek Gulma dengan IoU 0.5 yang diperoleh pada model ini adalah 0,523 dan MAP dengan IoU 0.95 sebesar 0,22. Waktu yang tercatat adalah 6.1 milidetik untuk tahap inferensi, 2.2 milidetik untuk tahap operasi NMS, dan total waktu 8.4 milidetik untuk keseluruhan proses. Ukuran model yang terbaik dihasilkan sebesar 11,6 MB.

Dalam kasus ini, total waktu adalah 8,4 milidetik dan batch-size adalah 32 gambar. Maka dapat dihitung FPS (*Frame per Second*) dalam mendeteksi 1 set data uji.

$$\text{FPS} = 1 / (8,4 / 32) = 1 / 0,2625 = 3,81\text{fps}$$

3. Modifikasi kedua

```
test: Scanning 'Weed-Maize-Detection-2-10/test/labels.cache' images and labels... 368 found, 0 missing, 0 empty, 0 corrupted: 100% 368/368
      Class      Images      Labels      P      R      mAP@.5      mAP@.5:.95: 100% 12/12 [00:10<00:00, 1.13it/s]
      all         368         4276      0.71       0.608     0.647      0.263
      0           368         2125      0.778      0.762     0.786      0.334
      1           368         2151      0.642      0.454     0.508      0.193
Speed: 2.9/2.7/5.7 ms inference/NMS/total per 640x640 image at batch-size 32
```

Gambar 23. Evaluasi Model YOLOv7-Tiny Modifikasi 2

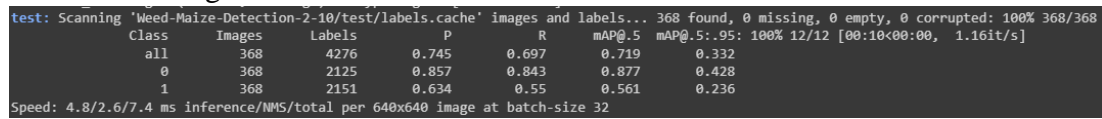
MAP dengan IoU 0.5 yang diperoleh pada model ini adalah 0,647 dan MAP dengan IoU 0.95 sebesar 0,263. MAP pada objek Jagung dengan IoU 0.5 yang diperoleh pada model ini adalah 0,786 dan MAP dengan IoU 0.95 sebesar 0,334. MAP pada objek Gulma dengan IoU 0.5 yang diperoleh pada model ini adalah 0,508 dan MAP dengan IoU 0.95 sebesar 0,193. Waktu yang tercatat adalah 2.9 milidetik untuk tahap inferensi,

2.7 milidetik untuk tahap operasi NMS, dan total waktu 5.7 milidetik untuk keseluruhan proses. Ukuran model yang terbaik dihasilkan sebesar 12,6 MB.

Dalam kasus ini, total waktu adalah 5,7 milidetik dan batch-size adalah 32 gambar. Maka dapat dihitung FPS (*Frame per Second*) dalam mendeteksi 1 set data uji.

$$\text{FPS} = 1 / (5,7 / 32) = 1 / 0,178125 = 5,62 \text{ fps.}$$

4. Modifikasi ketiga



Class	Images	Labels	P	R	mAP@0.5	mAP@0.5:0.95
all	368	4276	0.745	0.697	0.719	0.332
0	368	2125	0.857	0.843	0.877	0.428
1	368	2151	0.634	0.55	0.561	0.236

Speed: 4.8/2.6/7.4 ms inference/NMS/total per 640x640 image at batch-size 32

Gambar 24. Evaluasi Model YOLOv7-Tiny Modifikasi 3

MAP dengan IoU 0.5 yang diperoleh pada model ini adalah 0,719 dan MAP dengan IoU 0.95 sebesar 0,332. MAP pada objek Jagung dengan IoU 0.5 yang diperoleh pada model ini adalah 0,877 dan MAP dengan IoU 0.95 sebesar 0,428. MAP pada objek Gulma dengan IoU 0.5 yang diperoleh pada model ini adalah 0,561 dan MAP dengan IoU 0.95 sebesar 0,236. Waktu yang tercatat adalah 4.8 milidetik untuk tahap inferensi, 2.6 milidetik untuk tahap operasi NMS, dan total waktu 7.4 milidetik untuk keseluruhan proses. Ukuran model yang terbaik dihasilkan sebesar 13,2 MB.

Dalam kasus ini, total waktu adalah 7,4 milidetik dan batch-size adalah 32 gambar. Maka dapat dihitung FPS (*Frame per Second*) dalam mendeteksi 1 set data uji.

$$\text{FPS} = 1 / (7,4 / 32) = 1 / 0,23125 = 4,32 \text{ fps.}$$

Kesimpulan

Penelitian ini bertujuan untuk membangun teknik feature-extraction pada model YOLOv7-tiny yang dapat meningkatkan performa model sebelumnya. Modifikasi terdapat pada layer *backbone* dengan menambahkan blok residual dan modul SPP. Hasil menunjukkan bahwa modifikasi pertama menghasilkan peningkatan yang signifikan dalam kecepatan deteksi dan efisiensi, meskipun ada penurunan sedikit pada akurasi deteksi dengan IoU 0.95. Modifikasi kedua memberikan peningkatan kecepatan deteksi yang lebih rendah, sementara modifikasi ketiga memberikan keseimbangan antara akurasi deteksi dan kecepatan.

Daftar Pustaka

- Anwar A. 2022. What is Average Precision in Object Detection & Localization Algorithms and how to calculate it? Towar Data Sci.:1–11. <https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b>.
- Aristo Jansen Sinlae A, Alamsyah D, Suhery L, Fatmayati F. 2022. Classification of Broadleaf Weeds Using a Combination of K-Nearest Neighbor (KNN) and Principal Component Analysis (PCA). *Sinkron*. 7(1):93–100. doi:10.33395/sinkron.v7i1.11237.
- Bochkovskiy A, Wang C-Y, Liao H-YM. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. <http://arxiv.org/abs/2004.10934>.
- Budi Daya Jagung Hibrida. 2007. AgroMedia.
- Chen J, Wang H, Zhang H, Luo T, Wei D, Long T, Wang Z. 2022. Weed detection in sesame fields using a YOLO model with an enhanced attention mechanism and feature fusion. *Comput Electron Agric*. 202(October):107412. doi:10.1016/j.compag.2022.107412. <https://doi.org/10.1016/j.compag.2022.107412>.
- Gallo I, Rehman AU, Dehkordi RH, Landro N, La Grassa R, Boschetti M. 2023. Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images. *Remote Sens*. 15(2):1–17. doi:10.3390/rs15020539.
- He K, Zhang X, Ren S, Sun J. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)*. 8691 LNCS(PART 3):346–361. doi:10.1007/978-3-319-10578-9_23.
- He K, Zhang X, Ren S, Sun J. 2016. Deep residual learning for image recognition. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 2016-Decem:770–778. doi:10.1109/CVPR.2016.90.
- Joseph VR. 2022. Optimal ratio for data splitting. *Stat Anal Data Min*. 15(4):531–538. doi:10.1002/sam.11583.
- Li C, Li Lulu, Jiang H, Weng K, Geng Y, Li Liang, Ke Z, Li Q, Cheng M, Nie W, et al. 2022. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. <http://arxiv.org/abs/2209.02976>.
- Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M. 2020. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis*. 128(2):261–318. doi:10.1007/s11263-019-01247-4.
- Liu S, Jin Y, Ruan Z, Ma Z, Gao R, Su Z. 2022. Real-Time Detection of Seedling Maize Weeds in Sustainable Agriculture. *Sustain*. 14(22). doi:10.3390/su142215088.
- Luo T, Zhao J, Gu Y, Zhang S, Qiao X, Tian W, Han Y. 2023. Classification of weed seeds based on visual images and deep learning. *Inf Process Agric*. 10(1):40–51. doi:10.1016/j.inpa.2021.10.002. <https://doi.org/10.1016/j.inpa.2021.10.002>.
- Murawwat S, Qureshi A, Ahmad S, Shahid Y. 2018. Weed Detection Using SVMs. *Eng Technol Appl Sci Res*. 8(1):2412–2416. doi:10.48084/etasr.1647.
- Pal SK, Pramanik A, Maiti J, Mitra P. 2021. Deep learning in multi-object detection and tracking: state of the art. *Appl Intell*. 51(9):6400–6429. doi:10.1007/s10489-021-02293-7.

- Rao AS, Ratnam M, Reddy TY. 2009. Weed management in zero till sown maize. *Indian J Weed Sci.* 41(1):46–49.
- Redmon J, Divvala S, Girshick R, Farhadi A. 2016. You only look once: Unified, real-time object detection. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit.* 2016-Decem:779–788. doi:10.1109/CVPR.2016.91.
- Redmon J, Farhadi A. 2017. YOLO9000: Better, faster, stronger. *Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017.* 2017-Janua:6517–6525. doi:10.1109/CVPR.2017.690.
- Redmon J, Farhadi A. 2018. YOLOv3: An Incremental Improvement. <http://arxiv.org/abs/1804.02767>.
- Sager C, Janiesch C, Zschech P. 2021. A survey of image labelling for computer vision applications. *J Bus Anal.* 4(2):91–110. doi:10.1080/2573234X.2021.1908861. <https://doi.org/10.1080/2573234X.2021.1908861>.
- Subeesh A, Bhole S, Singh K, Chandel NS, Rajwade YA, Rao KVR, Kumar SP, Jat D. 2022. Deep convolutional neural network models for weed detection in polyhouse grown bell peppers. *Artif Intell Agric.* 6:47–54. doi:10.1016/j.aiia.2022.01.002. <https://doi.org/10.1016/j.aiia.2022.01.002>.
- Sunil, Zhang Y, Koparan C, Ahmed MR, Howatt K, Sun X. 2022. Weed and crop species classification using computer vision and deep learning technologies in greenhouse conditions. *J Agric Food Res.* 9(June):100325. doi:10.1016/j.jafr.2022.100325. <https://doi.org/10.1016/j.jafr.2022.100325>.
- Umiyati U, Widayat D. 2017. *Gulma dan Pengendaliannya*. Bandung: CV Budi Utama.
- Wang A, Zhang W, Wei X. 2019. A review on weed detection using ground-based machine vision and image processing techniques. *Comput Electron Agric.* 158(January):226–240. doi:10.1016/j.compag.2019.02.005. <https://doi.org/10.1016/j.compag.2019.02.005>.
- Wang C-Y, Bochkovskiy A, Liao H-YM. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. :1–15. <http://arxiv.org/abs/2207.02696>.
- Winarsih S. 2008. Mengenal Gulma. :1–2.
- Ying B, Xu Y, Zhang S, Shi Y, Liu L. 2021. Weed detection in images of carrot fields based on improved YOLO v4. *Trait du Signal.* 38(2):341–348. doi:10.18280/TS.380211.
- Zhao J, Tian G, Qiu C, Gu B, Zheng K, Liu Q. 2022. Weed Detection in Potato Fields Based on Improved YOLOv4: Optimal Speed and Accuracy of Weed Detection in Potato Fields. *Electron.* 11(22). doi:10.3390/electronics11223709.
- Zou Z, Chen K, Shi Z, Guo Y, Ye J. 2023. Object Detection in 20 Years: A Survey. *Proc IEEE.*:1–22. doi:10.1109/JPROC.2023.3238524.
- Zou Z, Shi Z, Guo Y, Ye J. 2019. Object Detection in 20 Years: A Survey. :1–39. <http://arxiv.org/abs/1905.05055>.

LAMPIRAN

Lampiran 1. Arsitektur Model YOLOv7-Tiny