BAB I
PENDAHULUAN

1.1 Latar Belakang

Jagung merupakan salah satu tanaman yang paling banyak dibudidayakan di dunia dan memiliki peran penting dalam perekonomian global. Selain sebagai bahan pangan utama, jagung juga berkontribusi besar dalam industri pakan ternak dan berbagai sektor industri lainnya. Pengembangan teknologi produksi jagung memiliki potensi besar untuk meningkatkan kualitas pertanian serta mendorong pertumbuhan ekonomi nasional. Secara khusus, biji jagung menjadi elemen penting dalam sistem pertanian, terutama dalam proses produksi dan pengendalian mutu, sehingga diperlukan metode yang akurat untuk identifikasi dan klasifikasinya (Bi et al. 2022).

Menurut FAO (*Food and Agriculture Organization*), jagung merupakan salah satu komoditas utama di pasar sereal dunia. Dalam perekonomian global, sereal memiliki peran strategis, sehingga diperlukan standar mutu yang konsisten dalam proses komersial, termasuk dalam analisis kualitas produk. Penerapan teknologi klasifikasi biji jagung berbasis kecerdasan buatan dapat mendukung proses standarisasi kualitas secara objektif dan efisien (Suárez et al. 2023).

Penilaian otomatis kualitas biji jagung telah menarik perhatian besar dari komunitas penelitian. Evaluasi kualitas umumnya mencakup parameter seperti kadar air, protein kasar, serat, dan karakteristik warna biji jagung (Velesaca et al. 2020). Proses pengolahan jagung seperti pemanenan, penyimpanan dan pengeringan sering kali mengalami kerusakan yang dapat mempengaruhi kualitas biji jagung (Kruszelnicka et al. 2024). Selain itu, biji jagung rentan terinfeksi oleh jamur seperti *Fusarium*, *Aspergillus*, terutama pada kondisi lingkungan yang lembab dan panas Kerentanan ini disebabkan oleh kadar air yang tinggi serta kandungan nutrisi biji jagung yang mendukung pertumbuhan jamur, sehingga dapat menyebabkan penurunan kualitas dan keamanan bahan pangan (Yang et al. 2022).

Kualitas biji jagung yang memenuhi standar perusahaan sangat penting dalam proses penerimaan bahan baku. Salah satu tahap penting dalam proses deteksi dan penilaian kualitas biji jagung, termasuk identifikasi biji yang berjamur, normal dan pecah. Ini menjadi aspek penting dalam memastikan kualitas akhir produk pertanian yang diterima oleh industri pasar (Ferdyanto 2020).

Kemajuan dalam teknologi *computer vision* dan *deep learning* telah memungkinkan otomatisasi proses inspeksi visual melalui deteksi, segmentasi, dan klasifikasi gambar (Tian et al. 2020). Teknologi ini mampu memberikan evaluasi cepat dan akurasi terhadap kualitas biji jagung seperti mendeteksi keretakan dan kerusakan fisik lainnya. Penerapan *computer vision* ini juga membantu mengurangi ketergantungan pada inspeksi manual yang rawan terhadap ketidakakuratan dan memakan waktu dalam penelitian kualitas jagung (Yumang et al. 2020).

Tantangan utama deteksi kualitas biji jagung menggunakan *computer vision* adalah kemampuan sistem untuk membedakan antara kelas yang, mirip secara visual seperti biji normal, biji pecah, dan biji berjamur terutama dalam kondisi gerakan cepat pada konveyor yang dapat menyebabkan blur. (M. Liu et al. 2024) menunjukkan bahwa

model deteksi berbaris *deep learning* mampu meningkatkan akurasi deteksi kualitas biji jagung bijij pecah dan biji berjamur hingga mencapai nilai mAP sekitar 96.1% memulai pengembangan struktur jaringan dan penerapan mekanisme *attention*, sehingga mampu mempertahankan performa deteksi yang tinggi pada lingkungan produksi dengan laju pemrosesan yang tinggi.

Ekstraksi fitur visual merupakan tahap penting dalam sistem klasifikasi kualitas biji jagung karena fitur warna dan tekstur mengandung informasi penting yang berkaitan dengan kondisi fisik. (Fan et al. 2024) menunjukkan bahwa pendekatan berbasis *machine vision* mampu menghasilkan model prediksi tingkat kerusakan biji jagung dengan akurasi tinggi melalui pemanfaatan karakteristik visual biji.

Dalam praktik industri, proses inspeksi visual pada sistem konveyor sering kali mengalami *motion blur* karena gerakan relatif antara kamera dan objek yang diamati, yang secara signifikan menurunkan kualitas citra dan akurasi ekstraksi fitur visual. Studi pada sistem sortasi buah menunjukkan bahwa peningkatan kecepatan konveyor secara nyata menyebabkan degradasi citra yang berdampak negatif pada akurasi klasifikasi objek (*motion blur*), dan diperlukan teknik perbaikan citra untuk meminimalkan efek tersebut (Diva. 2024). Selain itu, efek *motion blur* juga terbukti mengurangi akurasi pemisahan objek dalam aplikasi konveyor industri seperti pemisahan logam, menunjukkan bahwa *blur* perlu ditangani untuk meningkatkan keandalan sistem klasifikasi berbasis visi komputer (Yifeng Li. 2024).

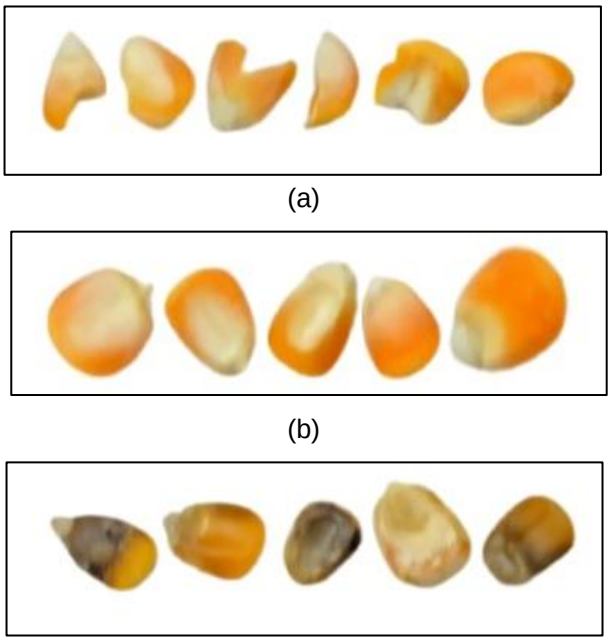
Berdasarkan permasalahan tersebut, penelitian ini mengusulkan meintegrasikan metode *Total Variation Deblurring* berbasis adaptif dengan algoritma deteksi objek YOLOv11 untuk meningkatkan akurasi deteksi dan klasifikasi kualitas biji jagung pada sistem konveyor. Pada penelitian ini, citra biji jagung yang diperoleh dari sistem konveyor terlebih dahulu diproses menggunakan *Adaptive Total Variation Deblurring,* dimana parameter deblurring disesuaikan secara otomatis berdasarkan karakteristik Tingkat blur pada citra guna mengurangi efek blur akibat kecepatan konveyor, kemudian hasil citra yang telah ditingkatkan kualitasnya digunakan sebagai masukan pada model YOLOv11 untuk melakukan deteksi dan klasifikasi biji jagung.

1.2 Landasan Teori

1.2.1 Jagung

Jagung merupakan salah satu komoditas pangan utama yang berperan penting sebagai sumber bahan pangan, pakan ternak, dan bahan baku industri. Kualitas biji jagung menjadi faktor kunci yang mempengaruhi nilai ekonomi, daya simpan, serta keamanan pangan. Kerusakan fisik pada biji jagung dapat menurunkan mutu dan meningkatkan risiko kontaminasi jamur penghasil aflatoksin. (Indonesia 2020). Biji jagung hasil panen memiliki variasi fisik yang dapat digunakan untuk menilai kualitasnya. Biji jagung dapat diklasifikasikan menjadi tiga kategori, yaitu normal, pecah, dan berjamur. Biji normal merupakan biji utuh tanpa kerusakan fisik maupun pertumbuhan jamur. Biji pecah mengalami kerusakan fisik akibat proses panen atau mekanis, seperti retakan pada permukaan, pecah pada bagian mahkota, pangkal, bahkan hingga hancur. Sementara itu, biji berjamur menunjukkan pertumbuhan jamur pada permukaannya, baik secara

menyeluruh maupun berbintik. Klasifikasi ini menjadi dasar dalam penelitian untuk mendeteksi kualitas biji jagung menggunakan sistem komputer vision dengan tiga kelas utama tersebut (M. Liu et al. 2024).



(a) (b) (c)

Gambar 1. Sampel Biji Jagung (a) Biji Jagung Pecah (b) Biji Jagung Normal (d) Biji Jagung Berjamur

1.2.2 You Only Look Once Version 11 (YOLOv11)

Algoritma *You Only Look Once* (YOLO) merupakan metode objek deteksi berbasis *Convolutional Neural Network* yang termasuk dalam kategori *single stage detector*. Algoritma ini melakukan proses deteksi objek dengan cara memprediksi lokasi pembatas *bounding box* dan kelas objek bersamaan dalam satu jaringan saraf, sehingga proses deteksi dapat dilakukan dengan efisien (Plastiras. 2018). YOLOv11 pengembagan dari keluarga YOLO yang di perkenalkan oleh ultralytics. Secara umum arsitektur YOLOv11 terdapat tiga bagian utama yaitu *backbone*, *neck* dan *head*. Bagian *backbone* berfungsi untuk mengekstraksi fitur visual dari citra masukan melalui lapisan konvolusi. Bagian *neck* menggabungkan fitur – fitur tersebut dari berbagai skala menggunakan *multi level feature* map, sehingga informasi objek kecil dan besar dapat dipertahankan. Bagian *head* bertugas untuk menghasilkan prediksi berupa koordinat kotak pembatas, nilai kepercayaan objek, dan probabilitas kelas objek (Rahima. 2024). YOLOv11 melakukan proses deteksi beberapa *feature map* dengan resolusi yang berbeda. Setiap lapisan *grid* di *feature map* model memprediksi jumlah *bounding box* beserta ukuran, posisi dan kelas objek. YOLOv11 mendeteksi objek dengan variasi ukuran secara efektif termasuk objek kecil.

Pada pelatihan YOLOv11 mencocokkan *bounding box* hasil prediksi dengan *bounding box* dasar atau *ground truth* berdasarkan nilai *Intersection Over Union* (IoU).

Kotak yang *bounding box* yang memiliki IoU tertinggi dengan *ground truth* diperlakukan sebagai sampel positif, sedangkan kotak lainya dianggap sebagai sampel negatif (Rahima 2024). Pada tahap pengujian menghasilkan *bounding box* beserta nilai kepercayaan atau *confidence score* untuk setiap kelas objek*,* diterapkan metode *Non-Maximum Suppression* (NMS) guna menghilangkan kotak pembatas yang saling tumpang tindih dan mempertahankan kotak dengan skor kepercayaan tertinggi sebagai hasil akhir deteksi objek (Plastiras. 2018).

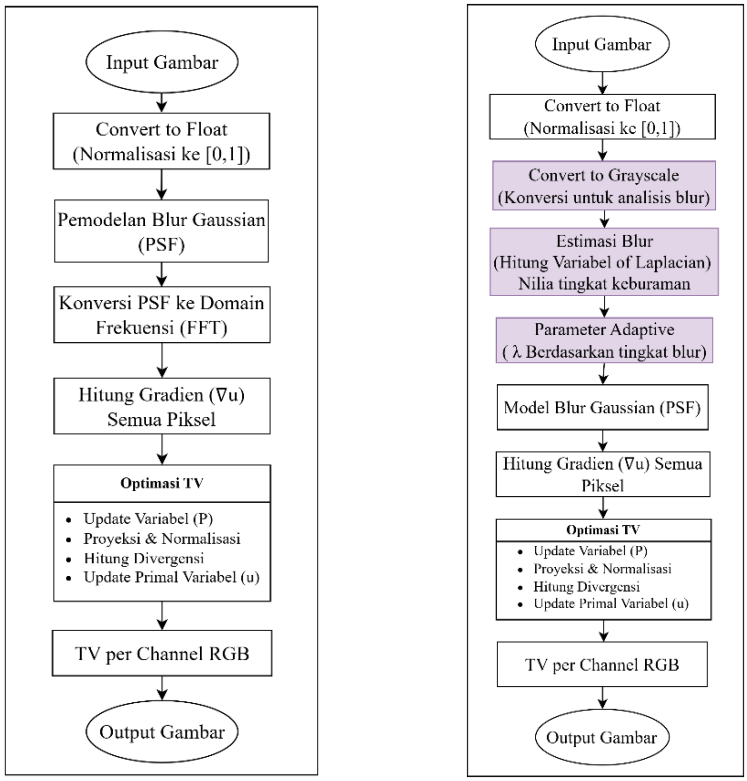
1.2.3 Image Processing

Pengolahan citra merupakan bidang yang mempelajari pemrosesan citra yang sudah dipresentasikan secara digital menggunakan algoritma komputer. Secara konsep, gambar digital tersusun dari banyak titik kecil disebut piksel. Setiap piksel memiliki nilai tertentu. Nilai piksel ini di proses dengan operasi komputasi untuk menghasilkan keluaran gambar yang yang sudah diperbaiki atau menghasilkan informasi fitur dari gambar seperti tepi objek, tekstur, dan pola (Wetzstein 2015). Pengolahan citra dilakukan agar informasi di dalam citra terlihat lebih jelas dan siap digunakan pada tahap analisis lanjutan. Dalam kondisi nyata, citra sering mengalami degradasi seperti *noise*, tepi menjadi tegas, kontras kurang sesuai dan *blur*, sehingga interpretasi menjadi menurun. Karena itu, pengolahan citra biasanya dipakai untuk mengurangi *noise*, memperbaiki kontras, menajamkan gambar, atau memperjelas bagian tertentu agar hasilnya lebih baik (Tomasi 1998).

Gambar tersusun dari banyak piksel, yaitu elemen kecil yang membentuk tampilan gambar. Setiap piksel memiliki nilai intensitas yang menunjukkan warna atau tingkat kecerahan pada posisi tertentu. Warna pada citra dapat dijelaskan dengan model warna seperti RGB, HSI, dan HSV. Pada model RGB, warna dibentuk dari gabungan intensitas merah (R), hijau (G), dan biru (B). Jika ketiganya bernilai maksimal maka menghasilkan warna putih, sedangkan jika tidak ada intensitasnya maka menghasilkan warna hitam (Manakitsa, 2024)

1.2.4 Total Variation

Total Variation Regularisasi adalah teknik yang banyak digunakan dalam pemrosesan citra untuk menghilangkan *blur* sambil mempertahankan tepi dan struktur utama dalam gambar. Metode ini bekerja berdasarkan prinsip bahwa perubahan intensitas yang signifikan dalam gambar. TV Regularisasi metode yang banyak digunakan dalam berbagai aplikasi, termasuk restorasi gambar medis, pemrosesan foto, dan pengawasan digital (Sanogo, Navasca, and Kindermann 2021). Keunggulan utama dari TV Regularisasi adalah kemampuannya dalam mempertahankan detail penting dalam gambar tanpa menyebabkan *over – smoothing*. Secara selektif menghilangkan *blur* di area yang homogen sementara tetap mempertahankan ketajaman pada tepi objek (J. Liu et al. 2021).



(a) (b)

Gambar 2. Diagram Perbandingan (a) Total Variation *(Non Adapative)* (b) Total Variation *(Adapative)*

1. Pada proses *deblurring* berbasis intensitas, citra RGB dikonversi menjadi *grayscale* agar setiap piksel hanya memiliki satu nilai kecerahan. Konversi ini dilakukan untuk menyederhanakan data dari tiga *channel* warna menjadi satu *channel* intensitas, sehingga mempermudah proses analisis. Menggunakan persamaan:

$$I(x,y) = 0.299R(x,y) + 0.587G(x,y) + 0.114B(x,y) \quad (1)$$

Keterangan:
$\mathrm{R,G,B}$: Nilai *channel* merah, hijau, dan biru pada piksel (x, y)
$\mathrm{I(x, y)}$: Intensitas *grayscale*
Angka koefisien (0.299, 0.587, 0.114) bobot kontribusi tiap kanal warna

2. Laplacian dan Blur Score digunakan untuk mengukur perubahan intensitas sehingga mampu mendeteksi tepi atau tekstur.
Kernel Laplacian 4 – *neighbor*:

$$\mathrm{K}=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

Dengan kernel tersebut, nilai laplacian pada piksel dapat dituliskan sebagai:

$$L(x,y) = I(x-1,y) + I(x+1,y) + I(x,y-1) + I(x,y+1) - 4I(x,y) \quad (2)$$

Keterangan:
$L(x,y)$: nilai Laplacian pada piksel (x, y)
$I(x,y)$: intensitas grayscale
$I(x \pm 1, y)$, $I(x, y \pm 1)$: intensitas piksel tetangga
Nilai $\mathrm{L}(\mathrm{x},\mathrm{y})$ dapat bernilai positif maupun negatif, karena laplacian mengukur perbedaan intensitas antara piksel pusat dan tetangganya. Pada area yang homogen, nilai laplacian cenderung mendekati nol, sedangkan pada area tepi atau tekstur nilai laplacian cenderung memiliki nilai lebih besar.
Tingkat blur diestimasi menggunakan varians laplacian:

$$\mathrm{Var}(L) = \frac{1}{N}\sum_{i=1}^{N}(L_i - \mu)^2 \quad (3)$$

Keterangan:
$\mathrm{Var}(L)$: Varians dari nilai laplacian.
N : Jumlah total piksel yang dihitung.
L_i : Nilai laplacian pada piksel ke-i.
μ : Rata-rata (*mean*) dari semua nilai laplacian.
Secara praktis, semakin kecil nilai varians laplacian, semakin rendah detail (*edge*) yang terdeteksi, sehingga citra cenderung lebih *blur*. Sebaliknya, nilai varians yang besar mengindikasikan citra lebih tajam.

3. Model degradasi citra digunakan untuk menggambarkan bagaimana citra asli mengalami proses blur. Model degradasi citra pada proses deblurring dinyatakan sebagai:

$$f = Ku \quad (4)$$

Keterangan:
f : Citra blur (input)
u : Citra asli (yang ingin dipulihkan)
K : Point Spread Function (PSF)
Berdasarkan model tersebut, tujuan utama proses deblurring adalah untuk memperoleh kembali citra asli u dari citra blur f. Untuk itu digunakan pendekatan optimasi dengan metode Total Variation (TV) yang dirumuskan sebagai:

$$\min_u(\| \nabla u \| + \lambda(x) \| Ku - f \|^2) \quad (5)$$

Keterangan:
∇u: Gradien citra (perubahan intensitas)
$\| \nabla u \|$: Total Variation (menjaga tepi citra)
λ: Parameter regularisasi
Ku: Hasil konvolusi citra dengan kernel PSF
f: Citra blur

4. PSF merepresentasikan bagaimana satu piksel menyebar ke piksel lain akibat blur, sehingga menjadi komponen penting dalam model degradasi citra. PSF Padding ini menunjukkan bahwa kernel *blur* (*Point Spread Function / PSF*) yang awalnya berukuran kecil misalnya gaussian 13 x 13 diperluas (*Padding*) agar sesuai dengan ukuran citra yang akan diproses misalnya 640 x 640. Transformasi kernel:

$$F_K(u,v) = \mathcal{F}\{K_{\mathrm{pad}}(x,y)\} \quad (6)$$

Keterangan:

$F_K(u,v)$: transformasi Fourier dari kernel

$K_{pad}(x,y)$: kernel PSF yang telah di-padding

5. Primal-Dual TV, menghitung gradien citra untuk mengetahui perubahan intensitas pada arah horizontal dan vertikal. Gradien ini menjadi dasar dalam proses TV regularisasi karena digunakan untuk mengidentifikasi lokasi tepi. Dihitung gradien citra untuk mengetahui perubahan intensitas pada arah horizontal dan vertikal, yang dinyatakan sebagai:
   Gradien dari citra estimasi dihitung sebagai berikut:

$$u_x(i,j) = \bar{u}(i,j+1) - \bar{u}(i,j) \quad (7)$$
$$u_y(i,j) = \bar{u}(i+1,j) - \bar{u}(i,j) \quad (8)$$

   Keterangan:

   $u_x(i,j)$: gradien arah horizontal

   $u_y(i,j)$: gradien arah vertikal

   $u(i,j)$: nilai citra pada piksel (i, j)

   Dalam metode primal-dual, digunakan variabel bantu (variabel dual) $p = (p_x, p_y)$untuk merepresentasikan arah gradien yang telah diregularisasi. Variabel ini diinisialisasi dengan nol:

$$p_x^{(0)} = 0, p_y^{(0)} = 0 \quad (9)$$

   Keterangan:

   $p^{(0)}$: nilai awal variabel dual

   Nilai nol menunjukkan belum ada informasi gradien

6. Proyeksi (Normalisasi) Untuk menjaga stabilitas algoritma, dilakukan normalisasi terhadap variabel dual agar memenuhi batas $\| p \| \leq 1$. Proses ini dirumuskan sebagai:

$$m = \sqrt{p_x^2 + p_y^2}, norm = \max(1, m) \quad (10)$$
$$p_x \leftarrow \frac{p_x}{norm}, p_y \leftarrow \frac{p_y}{norm} \quad (11)$$

   Keterangan:

   $norm$: nilai normalisasi

   $\sqrt{p_x^2 + p_y^2}$: panjang vektor gradien

   Fungsi max menjaga nilai ≤ 1

   Jika $m \leq 1$, maka nilai tidak berubah. Jika $m > 1$, maka vektor dinormalisasi agar panjangnya kembali ≤ 1.

7. Setelah seluruh proses selesai, jika citra awal berupa RGB, maka setiap kanal warna yang telah diproses akan digabung kembali untuk menghasilkan citra berwarna. Hasil akhir dari metode ini adalah citra dengan tingkat blur yang berkurang, kontras yang meningkat, serta tepi objek yang tetap tajam. Dengan demikian, Total Variation Regularisasi menjadi metode yang efektif dalam meningkatkan kualitas citra, khususnya pada proses deblurring.

1.2.7 Konveyor

Sistem konveyor merupakan salah satu komponen utama dalam otomasi industri yang berfungsi untuk memindahkan material secara kontinu dengan kecepatan dan arah yang terkontrol. Dalam dalam perkembanganya, konveyor tidak hanya berperan sebagai alat transportasi mekanis, tetapi juga sebagai platfrom utama untuk proses monitoring. (Kim, Yi, and Suh 2025) menjelaskan bahwa belt konveyor terdiri dari berbagai komponen mekanis seperti motor, *roller*, *pulley* dan sabuk yang bekerja secara simultan.

Selain pemantauan kondisi mekanis, konveyor juga berperan penting dalam sistem inspeksi visual otomatis. (Miao et al. 2023) menyatakan bahwa permukaan belt konveyor yang bergerak secara kontinu sangat sesuai digunakan sebagai media akuisisi citra untuk mendeteksi objek tertentu, seperti benda asing yang berpotensi merusak sistem atau menurunkan kualitas material. Dengan memanfaatkan kamera industri dan algoritma deteksi objek berbasis *deep learning*, konveyor dapat berfungsi sebagai platform cerdas yang terintegrasi dengan teknologi visi komputer untuk meningkatkan efisiensi, akurasi dan keandalan proses industri. Penerapan sistem deteksi kualitas biji jagung berbasis konveyor, proses sortir dapat dilakukan secara lebih cepat dan objektif dibandingkan metode manual, sehingga mampu mendukung otomasi pada industri pertanian.

1.2.8 Object Detection

Object detection adalah teknologi dalam *computer vision* yang digunakan untuk mendeteksi keberadaan objek, seperti benda atau manusia di dalam gambar maupun video. Teknologi ini membantu komputer memahami isi visual secara otomatis, sehingga bisa mengetahui apakah ada suatu objek pada sebuah tampilan. Dalam prosesnya *object detection* bisa memberikan informasi yang berbeda. Sistem tidak hanya mengetahui objeknya ada, tetapi juga menunjukkan lokasi atau area objek tersebut di dalam gambar. Umumnya, sistem membedakan antara bagian yang termasuk objek target yang dicari dan *non-object* bagian lain seperti latar belakang. Secara sederhana, deteksi objek dapat dibagi menjadi dua: *soft detection* yang hanya memastikan objek ada atau tidak, dan *hard detection* yang memastikan objek ada sekaligus menunjukkan letaknya (Wang et al. n.d. 2024.).

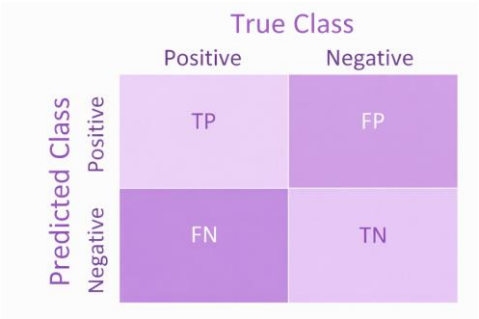
Dalam penerapannya, *object detection* menghadapi tantangan karena kondisi gambar di dunia nyata sangat beragam, misalnya ukuran objek bisa kecil/besar, objek saling menutupi (*occlusion*), pencahayaan berubah, latar belakang ramai, serta kebutuhan pemrosesan cepat (*real-time*). Oleh sebab itu, riset *object detection* terus berkembang untuk meningkatkan ketepatan deteksi dan ketahanan terhadap kondisi-kondisi tersebut (Li et al. 2024).

1.2.9 Confusion Matrix

Confusion matrix mengevaluasi kinerja model dalam mendeteksi dan mengklasifikasi objek pada tiap kelas. Pada evaluasi deteksi objek, setiap prediksi dibandingkan dengan label sebenarnya. *Confusion matrix* memberikan gambaran mengenai sejauh mana

model mampu mengklasifikasikan objek ke kelas yang tepat, sekaligus menunjukkan pola kekeliruan yang terjadi antar kelas (Niu et al. 2024). *Confusion matrix* merupakan metrik yang menunjukkan jumlah:

1. *True Positive* (TP) : label "*True*", prediksi "*True*".
2. *False Positive* (FP) : label "*False*", prediksi "*True*".
3. *True Negative* (TN) : label "*True*", prediksi "*False*".
4. *False Negative* (FN) : label "*False*", prediksi "*False*".



Gambar 3*. Confusion Matrix*

Metrik evaluasi digunakan adalah *precision*, *recall*, dan mAP.

1. *Precision* menggambarkan prediksi model, dari semua yang diprediksi sebagai suatu kelas berapa banyak yang benar. Precision tinggi bererti model jarang menghasilkan FP (*False Positive*). *Precision* penting untuk mengurangi kesalahan klasifikasi yang membuat objek masuk kategori kelas yang tidak sesuai. Nilai *precision* dapat diperoleh dengan persamaan (12).

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

2. *Recall* menunjukkan kemampuan deteksi model dari semua objek yang seharusnya ada pada suatu kelas, berapa banyak yang berhasil ditemukan model. *Recall* tinggi berarti model jarang melewatkan objek FN (*False Negative*). *Recall* rendah dapat menyebabkan objek lolos terdeteksi. Nilai *recall* dapat diperoleh dengan persamaan (13).

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

3. mAP (*Mean Average Precision*) adalah metrik evaluasi yang digunakan untuk mengukur performa model deteksi objek. Metrik ini menghitung rata-rata presisi dari semua kelas yang terdeteksi berbagai *threshold confidence* dan lokalisasi *bounding box* berdasarkan IoU. Nilai mAP yang tinggi menunjukkan model tidak hanya benar dalam menentukan kelas, tetapi juga tepat dalam menempatkan *bounding box* pada objek. Nilai mAP dapat diperoleh dengan persamaan (14).

$$mAP = \frac{1}{N}\sum_{i=1}^{n} AP_i \tag{14}$$

1.3 Rumusan Masalah

Berdasarkan latar belakang masalah diatas maka rumusan masalah pada penelitian ini adalah sebagai berikut.

1. Bagaimana mendeteksi dan klasifikasi kualitas biji jagung yang mengalami efek *blur* ketika bergerak cepat di konveyor?
2. Bagaimana performa YOLO dalam mendeteksi dan klasifikasi kualitas biji jagung yang mengalami efek *blur* pada variasi kecepatan konveyor yang berbeda?

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Adapun tujuan yang ingin dicapai pada penelitian ini yaitu sebagai berikut.

1. Menghasilkan sistem deteksi dan klasifikasi kualitas biji jagung yang mengalami efek *blur* pada kecepatan konveyor.
2. Menganalisis performa sistem dalam mendeteksi dan klasifikasi kualitas biji jagung yang mengalami efek *blur* pada variasi kecepatan konveyor yang berbeda.

1.4.2 Manfaat

Manfaat yang dapat diperoleh dari penelitian yaitu sebagai berikut.

1. Menghasilkan sistem yang mampu mendeteksi kualitas biji jagung pada kondisi *blur* secara lebih akurat menggunakan algoritma YOLO dan TV Regularization *deblurring*.
2. Sebagai bahan rujukan dalam pengembangan ilmu pengetahuan dan teknologi bagi peneliti serta akademisi pada bidang terkait.

1.5 Batasan Masalah

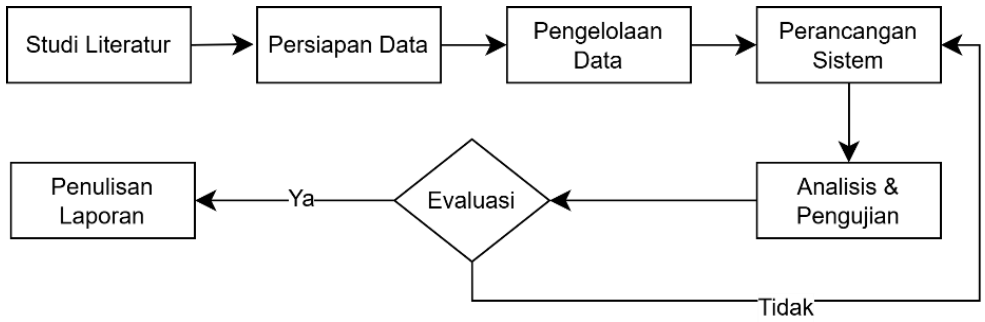
Adapun batasan masalah pada penelitian ini yaitu:

1. Citra yang di input adalah biji jagung yang dibagi tiga kelas yaitu biji normal, biji pecah, dan biji berjamur.
2. Kecepatan konveyor yang digunakan pada peneliti 0.30 m/s, 0.35 m/s, dan 0.40 m/s.
3. Jarak kamera yang digunakan pada penelitian ini yaitu 10 cm.
4. Sistem yang dikembangkan hanya berfokus pada proses deteksi dan klasifikasi biji jagung berdasarkan citra digital, tanpa mencakup tahap pengambilan keputusan lanjutan, seperti pemisahan fisik, penyortiran otomatis.

BAB II
METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini merupakan penelitian dengan jenis eksperimental. Pengumpulan data berupa studi literatur dari jurnal dan artikel ilmiah sehingga ditemukan ruang lingkup masalah pada penelitian yang dilakukan, berikut disajikan tahapan penelitian:



Gambar 4. Tahapan Penelitian

2.1.2 Studi Literatur

Tahapan awal yang dilakukan adalah studi literatur. Tahapan studi literatur merupakan tahapan pengumpulan informasi dan referensi – referensi yang terkait dengan penelitian yang akan dilakukan oleh penulis yaitu deteksi kualitas biji jagung. Tahap ini juga penulis menentukan metode yang akan digunakan dalam menyelesaikan masalah dan tantangan. Penelitian ini menggunakan metode total *variation deblurring* untuk mengurangi efek *blur* pada gambar dan kemudian akan diproses menggunakan algoritma YOLOv11 untuk deteksi dan klasifikasi kualitas biji jagung

2.1.3 Persiapan Data

Pada tahap ini, penulis melakukan pengumpulan dataset yang digunakan dalam penelitian. Dataset berupa citra biji jagung yang diambil pada sistem konveyor dengan tiga variasi kecepatan, yaitu 0.10 m/s, 0.15 m/s dan 0.20 m/s. Data dikumpulkan dalam tiga kategori kualitas, yaitu biji berjamur, biji normal, dan biji pecah. Dataset yang diperoleh selanjutnya digunakan untuk melatih model deteksi dan klasifikasi kualitas biji jagung.

2.1.4 Pengelolaan Data

Dilakukan pengolahan data citra biji jagung untuk memperoleh informasi visual, seperti warna dan tekstur. Proses ini mencakup beberapa tahapan *preprocessing*, yaitu pelabelan (*labelling),* perubahan ukuran citra (*resize*), pembagian data (*splitting),* serta perbaikan kualitas citra untuk mengatasi efek *blur*.

2.1.5 Perancangan Sistem

Tahapan perancangan sistem adalah tahap yang akan di kembangkan untuk menyelesaikan permasalahan dalam deteksi dan klasifikasi kualitas biji jagung pada citra yang mengalami *blur*. Pada tahap ini, sistem dirancang secara rinci agar alur kerja mudah dipahami, sehingga memudahkan proses implementasi sistem ke dalam program.

2.1.6 Pengujian dan Analisis

Tahapan pengujian dilakukan menggunakan dataset uji untuk mengevaluasi kinerja sistem yang di kembangkan. Setiap citra pada dataset uji dianalisis untuk proses deteksi dan klasifikasi kualitas biji jagung pada kecepatan konveyor yang berbeda. Hasil pengujian kemudian dievaluasi menggunakan metrik *precision*, *recall*, mAP50, dan mAP50–95 guna menilai ketepatan sistem dalam memprediksi kualitas biji jagung secara optimal.

2.1.7 Evaluasi

Evaluasi sistem ini dilakukan untuk mengukur sejauh mana sistem mampu melakukan deteksi dan klasifikasi kualitas biji jagung pada citra yang mengalami *blur* pada kecepatan konveyor yang berbeda.

2.1.8 Pembuatan Laporan

Bagian ini membahas hasil eksperimen dan analisis yang telah dilakukan pada sistem deteksi kualitas biji jagung, menginterpretasikan hasil pengujian yang diperoleh, serta membandingkan hasil yang diperoleh dengan penelitian sebelumnya.

2.2 Sumber Data

Data yang digunakan dalam penelitian ini diambil langsung dari hasil panen petani di Kabupaten Bone dan diproses di *Laboratorium Artificial Intelligence and Multimedia Processing* (AIMP), Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin yang dilengkapi dengan fasilitas untuk pemrosesan citra digital dan analisis data. Lingkungan laboratorium ini memungkinkan untuk pengaturan yang terkontrol dalam hal pencahayaan dan penempatan kamera untuk mendapatkan hasil gambar yang konsisten dari objek biji jagung yang diproses pada sabuk konveyor

Pengambilan data video menggunakan kamera Logitech Brio Ultra HD. Video direkam dengan posisi kamera terpasang pada *box* menghadap lurus ke bawah dengan jarak antara kamera dan objek 10 cm. Warna *belt* yang digunakan sekaligus sebagai *background* dari objek adalah putih untuk memaksimalkan kontras antara objek biji jagung dan *background.* Konveyor *belt* bergerak dengan kecepatan *gearbox* motor sebesar 0.10 m/s, 0.15 m/s dan 0.20 m/s untuk data training. Hasil rekaman video disimpan dalam format. MP4 dengan 30 *Frame Per Second* (FPS) dan resolusi 1920 x

1080 piksel. Ilustrasi pengambilan data biji jagung konveyor ditunjukkan pada Gambar 5.



Gambar 5. Konveyor

Akuisisi data *testing* dilakukan pada sistem konveyor dengan variasi kecepatan yang berbeda. Data *testing* dibagi menjadi tiga kategori berdasarkan kecepatan konveyor yang digunakan. Kategori pertama menggunakan kecepatan konveyor sebesar 0.30 m/s, kategori kedua dengan kecepatan 0.35 m/s, dan kategori ketiga dengan kecepatan 0.40 m/s. Variasi kecepatan ini digunakan untuk menganalisis pengaruh kecepatan konveyor terhadap kualitas citra serta kinerja sistem dalam melakukan deteksi dan klasifikasi kualitas biji jagung.

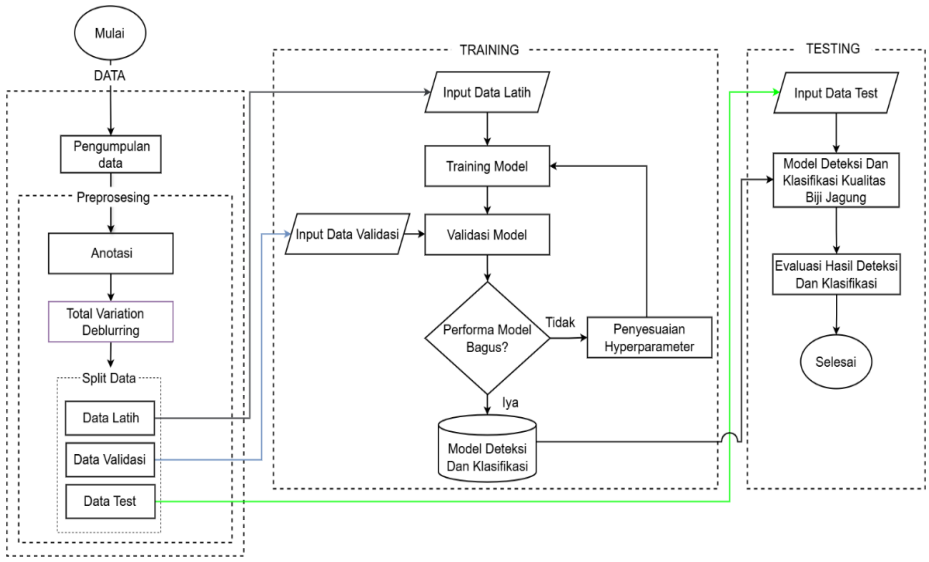
2.3 Perangkat Penelitian

Adapun perangkat-perangkat yang digunakan dalam menunjang penelitian ini yaitu sebagai berikut:

1. Perangkat Keras (Hardware)
   a. Kamera Logitech Brio Ultra HD
   b. Komputer Dell: Dengan prosesor intel Core i7-13700 (24 CPUs) dan RAM 16 GB
   c. DC motor rs 550
   d. Power supply
   e. Belt Konveyor
   f. LED strip
   g. Box kamera
2. Perangkat Lunak (Software)
   a. Operating Sistem Operasi Windows 11 Pro, 64 bit
   b. Jupyter Notebook
   c. Kaggle
   d. Roboflow
   e. Microsoft Excel
   f. Microsoft Wor

2.4 Rancangan Sistem

Alur perancangan sistem pada penelitian ini ditunjukkan pada Gambar 6 berikut:



Gambar 6. Flowchart Rancangan Sistem

2.4.1 Pengumpulan Data

Pengumpulan data pada penelitian ini dilakukan dengan merekam video biji jagung yang bergerak di atas sistem konveyor menggunakan kamera beresolusi 1080p. Proses perekaman video dilakukan pada dua skenario, yaitu data *training* dan data *testing*, dengan variasi kecepatan konveyor yang berbeda. Untuk data *training*, digunakan tiga kecepatan konveyor, yaitu 0.10 m/s, 0.15 m/s, dan 0.20 m/s, sedangkan untuk data *testing* digunakan kecepatan 0.30 m/s, 0.35 m/s, dan 0.40 m/s. Perbedaan kecepatan ini bertujuan untuk menguji kemampuan model dalam mengenali kualitas biji jagung pada kondisi kecepatan konveyor yang belum pernah dilatih sebelumnya. video diekstraksi menjadi citra (*frame*) dengan laju 8 *frame* per detik (8 fps), sehingga setiap detik video menghasilkan delapan citra. Hasil ekstraksi disimpan dalam format *.jpg* dan digunakan sebagai dataset untuk proses pelatihan dan pengujian sistem deteksi serta klasifikasi kualitas biji jagung. Gambar 7. Contoh citra hasil ekstraksi video menjadi frame (.jpg) sebagai dataset untuk pelatihan dan pengujian sistem deteksi dan klasifikasi kualitas biji jagung.



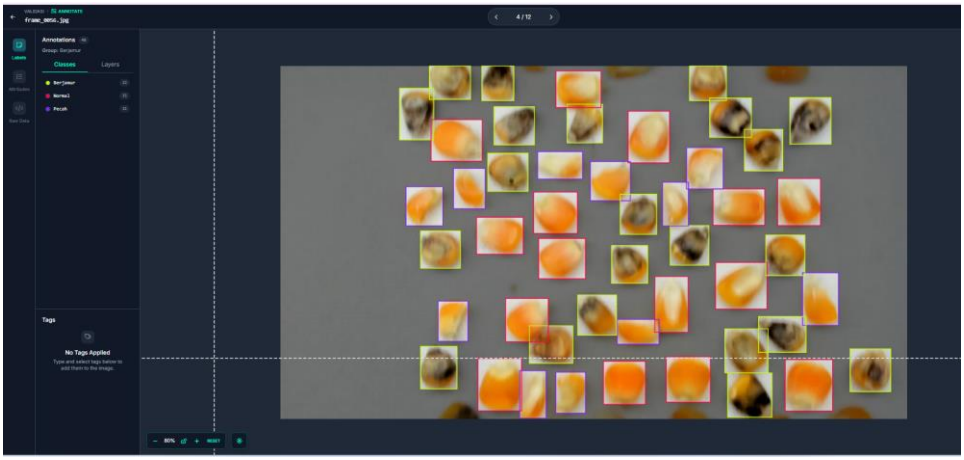
Gambar 7. Hasil Ekstrak Video Menjadi Gambar

2.4.2 Preprocessing

Setelah proses pengumpulan data dilakukan, dilanjutkan ke tahap pra-pemrosesan (*preprocessing*), untuk mempersiapkan gambar sebelum diproses lebih lanjut dengan tujuan untuk memperoleh hasil yang optimal. Tahapan dalam proses ini meliputi:

a. Anotasi dan labeling

Pada tahap ini, Pada tahap ini, proses anotasi gambar dilakukan dengan memberikan label pada citra biji jagung berbasis website Roboflow. Objek yang menjadi fokus deteksi adalah biji jagung dengan tiga klasifikasi kualitas, yaitu biji normal, biji berjamur, dan biji pecah. Hasil anotasi ini digunakan sebagai dasar bagi model dalam mempelajari pola visual serta posisi objek biji jagung pada citra. Hasil anotasi ini digunakan sebagai dasar bagi model dalam mempelajari pola visual serta posisi objek biji jagung pada citra.



Gambar 8. Anotasi dan Labelling Gambar

b. Resize

Proses *resize* atau perubahan ukuran citra dilakukan dengan mengubah ukuran *frame* asli yang memiliki resolusi 1920 × 1080 piksel menjadi 640 × 640 piksel. Perubahan ukuran ini bertujuan untuk mengurangi ukuran data sehingga dapat mempercepat proses pelatihan model serta mengoptimalkan penggunaan memori. Dengan melakukan *resize*, model YOLOv11 tetap dapat mengekstraksi dan mengenali fitur-fitur penting pada citra biji jagung tanpa kehilangan informasi visual yang dibutuhkan. Selain itu, penggunaan ukuran citra yang lebih kecil mendukung efisiensi pemrosesan, terutama dalam penerapan sistem deteksi kualitas biji jagung secara *real-time*.

c. Split Data

Setelah proses *resize* dilakukan, tahap selanjutnya adalah *split data* untuk membagi dataset citra biji jagung ke dalam tiga bagian, yaitu data *training*, data *validation*, dan data *testing*. Pembagian dataset ini bertujuan untuk melatih model, melakukan validasi selama proses pelatihan, serta menguji kinerja model dalam mendeteksi dan mengklasifikasikan kualitas biji jagung pada citra yang mengalami *blur* akibat kecepatan

konveyor. Proses deteksi dan klasifikasi dilakukan menggunakan algoritma YOLOv11. Adapun pembagian data dilakukan sebagai berikut:

1. Data Training digunakan untuk melatih model dalam mengenali dan mempelajari karakteristik visual objek biji jagung berdasarkan tiga kelas kualitas, yaitu biji normal, biji berjamur, dan biji pecah. Pada penelitian ini, jumlah data *training* yang digunakan sebanyak 555 citra biji jagung.



(a) (b) (c)

Gambar 9. Kecepatan 0.15 m/s (a) Kecepatan 0.20 m/s (b) Kecepatan 0.25 m/s

2. Data *Validation* digunakan sebagai data pendukung untuk mengevaluasi kinerja model selama proses pelatihan berlangsung. Data ini berperan dalam memantau perkembangan performa model serta membantu mengidentifikasi potensi terjadinya *overfitting* maupun *underfitting*. Pada penelitian ini, jumlah data *validation* yang digunakan sebanyak 159 citra.
3. Data *testing* digunakan untuk menguji kinerja akhir model setelah proses pelatihan selesai. Data ini bersifat terpisah karena tidak dilibatkan dalam proses *training* maupun *validation*. Data *testing* pada penelitian ini berupa citra biji jagung yang bergerak di atas sistem konveyor dengan variasi kecepatan yang berbeda. Data *testing* dibagi menjadi tiga kategori berdasarkan kecepatan konveyor, yaitu 0.30 m/s, 0.35 m/s, dan 0.40 m/s. Setiap kategori kecepatan terdiri dari 79 citra biji jagung yang mencakup tiga kelas kualitas, yaitu biji berjamur, biji pecah, dan biji normal. Peningkatan kecepatan konveyor menyebabkan bertambahnya efek *blur* pada citra akibat pergerakan biji jagung yang semakin cepat, sehingga detail visual seperti bentuk, tekstur, dan tepi objek menjadi kurang jelas. Variasi kecepatan ini digunakan untuk menganalisis pengaruh tingkat blur terhadap kinerja model dalam melakukan deteksi dan klasifikasi kualitas biji jagung. Secara keseluruhan, jumlah data *testing* yang digunakan dalam penelitian ini adalah 237 citra.



(a) (b) (c)

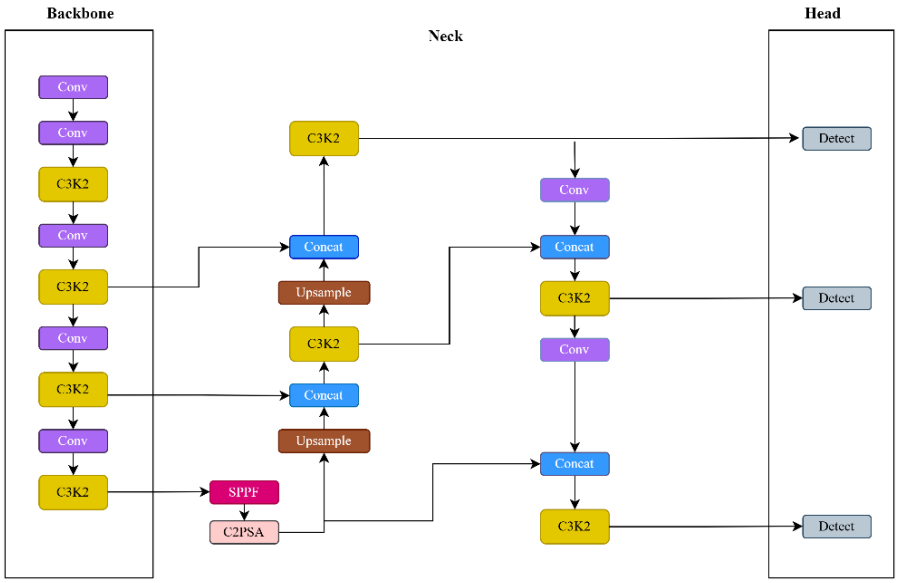
Gambar 10. Kecepatan 0.30 m/s (a) Kecepatan 0.35 m/s (b) Kecepatan 0.40 m/s

2.4.4 Training

Setelah tahap *preprocessing*, data *training* sebanyak 555 citra digunakan untuk melatih model deteksi kualitas biji jagung. Data ini digunakan sebagai bahan pembelajaran agar model dapat mengenali visual biji jagung berdasarkan tingkat kualitasnya. Melalui proses pelatihan, model belajar mengaitkan ciri visual dengan label kualitas yang diberikan, sehingga mampu mendeteksi dan klasifikasi kualitas biji jagung pada citra yang belum pernah digunakan sebelumnya.

1. Training YOLOv11

YOLOv11 merupakan arsitektur deteksi objek *single stage detector* yang mampu melakukan proses deteksi secara cepat dan efisien hanya dalam satu kali pemrosesan citra. arsitektur YOLOv11 tersusun atas tiga komponen utama, yaitu *backbone*, *head*. *Backbone* mengekstraksi fitur penting dari citra masukan, *neck* berfungsi untuk menggabungkan informasi fitur dari berbagai skala, sedangkan *head* digunakan untuk menghasilkan prediksi lokasi objek *bounding box* dan klasifikasi objek. Dengan struktur tersebut, YOLOv11 dapat mendeteksi objek dengan berbagai ukuran. Proses alur arsitektur YOLOv11 dapat ditunjukkan pada Gambar 11.



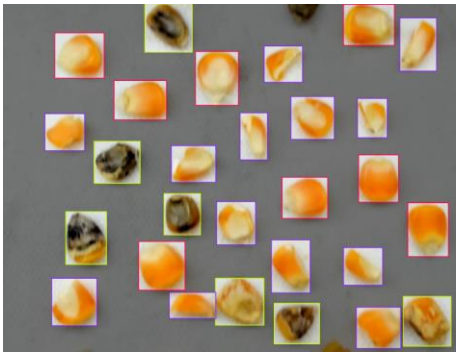
Gambar 11. Arsitektur YOLOv11

a. Inisiasi Bobot Awal

Sebelum proses pelatihan dimulai, bobot awal pada jaringan konvolusional diinisialisasi secara acak. Inisialisasi ini dilakukan untuk memberikan nilai awal pada parameter model agar proses optimisasi dapat berjalan. Pemilihan bobot secara random memungkinkan model memulai pembelajaran tanpa bias terhadap kelas atau fitur tertentu. Setelah inisialisasi, bobot akan diperbarui secara bertahap selama proses pelatihan berdasarkan loss yang dihasilkan dari prediksi model.

b. Input Data Training

Pada tahap awal dalam proses pelatihan model deteksi objek menggunakan YOLOv11, digunakan dataset yang terdiri dari citra objek sebagai data *input*. Setiap gambar dilengkapi dengan *ground truth* berupa anotasi yang menjelaskan posisi objek melalui *bounding box* serta label kelas yang sesuai. Anotasi ini digunakan sebagai data acuan bagi model dalam mempelajari letak dan jenis objek pada citra. *Label* kelas dipresentasikan dalam bentuk numerik sesuai dengan format anotasi YOLO, sehingga dapat diproses oleh model selama pelatihan. Gambar 12 menunjukkan contoh citra yang digunakan dalam proses *training* YOLO yang telah dianotasi.



Gambar 12. Contoh Input Citra Proses Training

Penelitian ini, kualitas biji jagung diklasifikasikan ke dalam tiga kelas, yaitu biji jagung normal, biji jagung pecah, dan biji jagung berjamur. Masing-masing kelas direpresentasikan dalam bentuk numerik sesuai dengan format anotasi YOLO, yaitu kelas normal diberi label 0, kelas biji pecah diberi *label* 1, dan kelas biji berjamur diberi *label* 3. Anotasi tersebut digunakan sebagai data masukan pada proses pelatihan model. Gambar 13 merupakan contoh anotasi inputan *training*.

	file_name	category	x	y	width	height
0	frame_0017_jpg.rf.47c3519bf590a45c65547a786e7f...	1	408	823	143.7	150.9
1	frame_0017_jpg.rf.47c3519bf590a45c65547a786e7f...	0	355	346	136.9	137.5
2	frame_0017_jpg.rf.47c3519bf590a45c65547a786e7f...	2	705	383	99.8	151.9
3	frame_0017_jpg.rf.47c3519bf590a45c65547a786e7f...	1	654	540	130.2	111.6
4	frame_0017_jpg.rf.47c3519bf590a45c65547a786e7f...	2	1540	889	112.0	135.7
...	...	...	...	...	...	...
2696	frame_0136_jpg.rf.bc3661451ddece5f0acdf8765012...	1	1437	413	142.7	158.8
2697	frame_0136_jpg.rf.bc3661451ddece5f0acdf8765012...	2	1642	547	122.6	170.5
2698	frame_0136_jpg.rf.bc3661451ddece5f0acdf8765012...	1	1224	89	130.7	151.8
2699	frame_0136_jpg.rf.bc3661451ddece5f0acdf8765012...	2	1414	894	139.9	145.4
2700	frame_0136_jpg.rf.bc3661451ddece5f0acdf8765012...	2	1798	628	122.5	157.1

Gambar 13. Contoh anotasi input training

c. Modul Convolution (Conv)

Modul Conv pada YOLOv11 disusun secara *sequential*, terdiri atas tiga tahapan utama, yaitu operasi konvolusi (Conv2D), *Batch Normalization* (BN), dan fungsi aktivasi *Sigmoid Linear Unit (SiLU)*.

Conv2D. merupakan operasi konvolusi dua dimensi yang menggunakan kernel untuk mengekstraksi fitur lokal dari citra melalui proses perkalian dan penjumlahan elemen matriks. Berikut Rumus perhitungan *convolution* yang akan menghasilkan *feature map*.

$$S(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N=1} I(i+m, j+n) \cdot K(m,n) \tag{15}$$

Keterangan:

$S(i,j)$: Nilai hasil konvolusi pada posisi (i, j)

$I(i+m, j+n)$: Nilai piksel pada posisi $(i+m, j+n)$ di citra asli (input)

$K(m, n)$: Nilai elemen kernel (filter) pada posisi (m, n)

M,N : Ukuran kernel (misalnya 2×2 berarti $M=N=2$)

Berikut contoh proses perhitungan *Convolution*:

Batch Normalization. Dilakukan dengan menormalkan keluaran layer konvolusi berdasarkan nilai rata-rata dan varians dari feature map. Proses normalisasi dilakukan sebagai berikut:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{16}$$

Output hasil normalisasi kemudian ditransformasikan menggunakan parameter skala γ dan pergeseran β:

$$y_i = \gamma \hat{x}_i + \beta \tag{17}$$

SiLU (Sigmoid Linear Unit). Pada setiap blok konvolusi dalam arsitektur YOLO Adalah penerapan fungsi aktivasi SiLU. Fungsi ini memperkenalkan sifat non – linear ke dalam jaringan saraf, sehingga model mampu mempelajari hubungan yang kompleks pada citra. Persamaan SiLU dituliskan sebagai:

$$\mathrm{SiLU}(x) = x \cdot \sigma(x) \; dengan \; \sigma(x) = \frac{x}{1 + e^{-x}} \tag{18}$$

Keterangan:

x : Nilai input hasil keluaran dari proses *Batch Normalization (BN)*.

$\sigma(x)$: Fungsi sigmoid yang memberikan pembobot *non-linear* terhadap nilai x

Setiap operasi konvolusi pada modul Conv tidak hanya berfungsi untuk mengekstraksi ciri-ciri penting dari citra masukan, tetapi juga menyebabkan perubahan pada dimensi data, baik dalam hal ukuran spasial (*height* dan *width*) maupun jumlah *channel* (*depth*). Ukuran keluaran dari proses konvolusi dua dimensi dapat dihitung menggunakan persamaan:

$$H_{out} = \lfloor \frac{H_{in} - K + 2P}{S} \rfloor + 1, \quad W_{out} = \lfloor \frac{W_{in} - K + 2P}{S} \rfloor + 1 \tag{19}$$

Dimana:

H_{in}, W_{in} : Tinggi dan lebar citra masukan,

H_{out}, W_{out}: Tinggi dan lebar keluaran,

K : Ukuran kernel (misalnya 3),

P : Jumlah *padding*

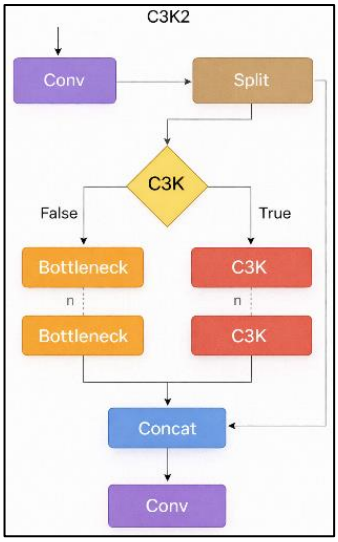
Selain mengubah ukuran, proses konvolusi juga menentukan berapa banyak *channel* yang dihasilkan. *Channel* dapat dianggap sebagai lapisan hasil olahan Gambar, dimaan setiap lapisan menyimpan jenis fitur yang berbeda. Pada YOLOv11, jumlah *channel* keluaran setiap layer sudah didtentukan sebelumnnya, untuk menghasilkan jumlah *channel*, konvolusi menggunakan jumlah kernel yang sama. Hal ini karena satu kernel menghasilkan satu *feature map*.

$$C_{out} = N_{kernel} \tag{20}$$

Sehingga dimensi akhir dari hasil konvolusi adalah $(C_{out}, H_{out}, W_{out})$. Pada YOLOv11, misalnya gambar masukan memiliki ukuran 640 x 640 piksel dengan 3 *channel* warna R, G, B. Kemudian gambar diproses menggunakan operasi konvolusi dengan kernel berukuran 3 x 3, sehingga setiap kernel membaca area kecil 3 x 3 piksel pada gambar. Penggunaan stride sebesar 2 membuat kernel bergeser dua piksel setiap kali, sehingga ukuran gambar keluaran menjadi lebih kecil. Akibatnya, ukuran spasial gambar berkurang dari 640×640 menjadi 320×320. Selain mengubah ukuran gambar, konvolusi juga menambah jumlah channel. Penggunaan 64 kernel menghasilkan 64 peta fitur, sehingga channel meningkat dari 3 menjadi 64. Peningkatan ini memperkaya representasi fitur dan membantu model mengenali pola visual yang lebih kompleks.

d. Modul C3K2

Modul C3K2 diawali dengan proses konvolusi pada *feature map* untuk menyesuaikan dimensi channel sekaligus mengekstraksi fitur awal. Menggabungkan konsep CSP (*Cross Stage Partial*) dengan pemrosesan fitur. fitur masuk Conv, lalu *split* menjadi dua bagian, satu bagian lewat jalur ringan (shortcut), sedangkan cabang lain diproses berulang (*n* kali) menggunakan *Bottleneck* jika c3k=False atau C3k jika c3k=True. Setelah itu kedua cabang *Concat* digabung pada channel dan diakhir Conv untuk fitur. Desain ini menjaga komputasi tetap ringan, tetapi menghasilkan fitur stabil untuk deteksi. Proses alur C3K2 ditunjukkan pada Gambar 14.



Gambar 14. Alur C3k2

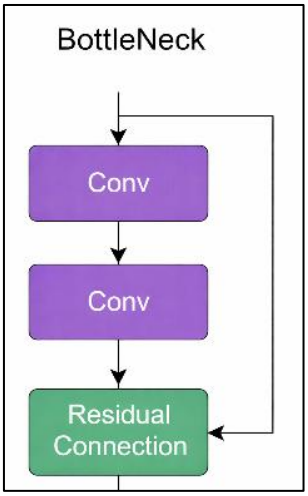
Tahap Split. Setelah proses konvolusi 1 x 1 sebelumnya, *feature map* yang dihasilakan dibagi menjadi dua bagian berdasarkan dimensi menggunakan operasi *chunk* atau *split*. Bagian A diproses lebih lanjut mengekstraksi fitur, dan jalur B yang dilewatkan secara langsung untuk mempertahankan informasi awal dan menjaga kestabilan gradien. Proses pembagian *feature map* tersebut dapat dinyatakan sebagai:

$$[A, B] = \text{split}(C) \tag{21}$$

C : Jumlah *channel* (kanal) pada *feature map* masukan.
A dan B : Dua bagian channel hasil pembagian yang digunakan pada proses selanjutnya,
Split (·) : Operasi pemisahan *feature map* berdasarkan dimensi *channel*.
Tahap ini penerapan prinsip *cross stage partial* (CSP), yaitu strategi yang membagi *feature map* menjadi dua bagian sehingga hanya sebagian fitur yang diproses lebih lanjut pada bagian utama, sementara sisanya dilewatkan secara langsung. Pendekatan ini mengurangi beban komputasi, serta stabilitas proses pelatihan model.
Bottleneck. Pada kondisi c3k=*False*, jalur utama di proses menggunakan *bottleneck* standar yang terdiri dari konvolusi 1x1 untuk melakukan penyesuaian *channel* misalnya dari 128 *channel* menjadi 64 agar komputasi lebih ringan. Tahap kedua konvolusi 3x3 yang mengekstraksi fitur spasial seperti tepi, tekstur dan bentuk karena kernel 3x3 menghubungkan antar piksel. Hasil konvolusi kemudian digabung dengan fitur awal melalui residual *connection* atau penjumlahan input dan output agar pelatihan lebih stabil. Proses *bottleneck* dapat dilihat pada Gambar 15. Persamaan proses *bottleneck* dapat dituliskan sebagai:

$$Y = X + f_{3\times3}\left(f_{1\times1}(X)\right) \tag{22}$$

Keterangan simbol:
X : Fitur masukan dari jalur transformasi
$f_{1\times1}$: Konvolusi 1×1
$f_{3\times3}$: Konvolusi 3×3
Y : Hasil keluaran dari blok *bottleneck*
$+$: Menunjukkan koneksi residu (*Residual connection*).



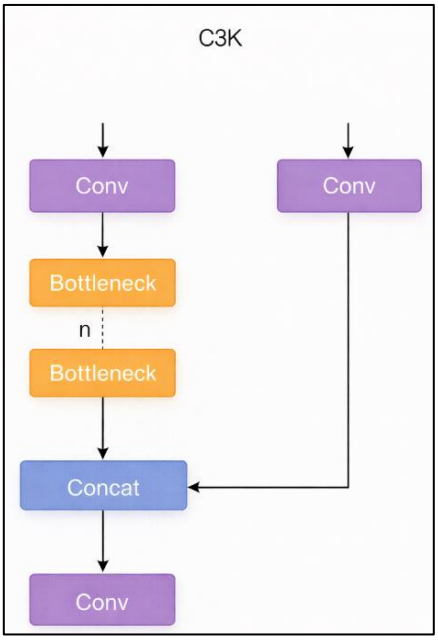
Gambar 15. Alur BottleNeck

C3k. Apabila parameter C3k=*True* jalur pemrosesan modul akan menggunakan block C3k sebagai pengembangan dari struktur C3. Pada blok ini fitur masukan terlebih dahulu dilewatkan melalui dua konvolusi 1x1 untuk membentuk dua bagian fitur dan menyesuaikan *channel*. Salah satu bagian diproses lebih lanjut menggunakan beberapa *bottleneck*, seperti yang ditunjukkan pada Gambar 16. Persamaan proses pada blok cek dapat dituliskan sebagai:

$$Y = f_{1\times1}\left(\left[f_{\text{Bottleneck}}\left(f_{1\times1}(X)\right), f_{1\times1}(X)\right]\right) \tag{23}$$

Dimana:

X : Fitur masukan
$f_{1\times1}$: Konvolusi 1×1
$f_{\text{Bottleneck}}$: Blok Bottleneck
$[\,,]$: Operasi *concatenation* antar fitur
Y : Hasil keluaran setelah penggabungan dua jalur fitur



Gambar 16. Alur C3k

Concat. Tahap *concatenation* (*Concat*) proses penggabungan dua *feature map* pada dimensi *channel*. Pada modul C3k2, satu *feature map* dari jalur utama yang telah diproses melalui blok C3k atau bottleneck, sedangkan *feature map* dari jalur *shortcut* yang melalui konvolusi. Kedua keluaran tersebut digabungkan sehingga jumlah *channel* pada output menjadi c_out= c1 + c2. Jika jumlah *channel* pada kedua jalur sama C1 = C2, maka jumlah *channel output* menjadi 2C dua kali lipat.

$$C_{out} = [C_{in\,1}, C_{in\,2}] \tag{24}$$

Misalkan $C_{in\,1}$ memiliki 32 kanal dan $C_{in\,2}$ juga memiliki 32 kanal maka C_{out} akan sama dengan 64.

e. Modul Upsample

Proses *upsample* pada YOLOv11 untuk meningkatkan ukuran spasial *height* dan *width* dari *feature map* yang sebelumnya telah di perkecil dari operasi konvolusi dengan *stride* atau *downsampling*. Tahap ini ini diperlukan agar *feature map* dari level resolusi yang berbeda dapat disesuaikan ukurannya sehingga dapat digabungkan di *concat* secara efektif pada bagian *neck*. Metode ini memperbesar *feature map* dengan cara menyalin nilai piksel terdekat menggandakan nilai yang sudah ada, tanpa menghitung nilai baru seperti bilinear. Karena itu prosesnya lebih cepat, ukuran *height* dan *width* menjadi lebih besar, sedangkan jumlah *channel* tetap. Jika *feature map* awal memiliki ukuran 3x3 dengan nilai:

0	2	0
0	2	0
0	2	0

Maka setelah dilakukan *upsampling* dengan metode *nearest* dan *scale_factor* = 2, ukuran hasilnya menjadi 6×6. Pada proses ini, setiap nilai pada fitur asal diperluas menjadi blok berukuran 2×2 dengan nilai yang sama, sehingga diperoleh:

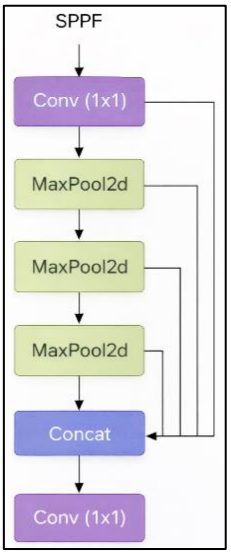
0	0	2	2	0	0
0	0	2	2	0	0
0	0	2	2	0	0
0	0	2	2	0	0
0	0	2	2	0	0
0	0	2	2	0	0

Dengan demikian, nilai 2 pada kolom tengah akan direplikasi menjadi dua kolom, dan setiap baris juga direplikasi menjadi dua baris, sesuai prinsip *nearest-neighbor.*

f. Modul SPFF (Spatial Pyramid Pooling - Fast)

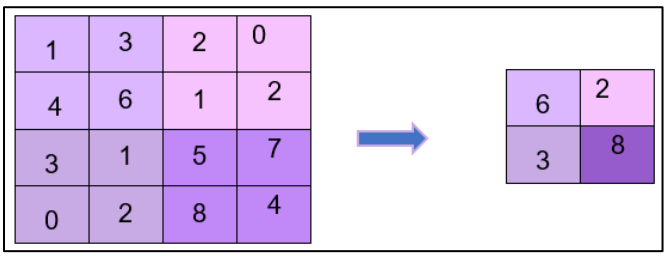
Modul SPFF (Spatial Pyramid Pooling - Fast) bagian yang membantu model mendapatkan informasi yang lebih luas dari feature map, tanpa mengubah ukuran *height* dan *width*. Prosesnya dimulai dengan konvolusi 1x1 untuk menyesuaikan jumlah *channel* agar lebih efisien. Setelah itu fitur dilewatkan ke MaxPool2d beberapa kali secara berurutan, sehingga model bisa menangkap informasi dari area yang semakin besar. Hasil dari beberapa pooling tersebut kemudian digabung (*concat*) pada dimensi *channel* agar informasi dari berbagai skala terkumpul. Terakhir, dilakukan konvolusi 1x1 lagi untuk

menyatukan dan merapikan hasil penggabungan, sehingga menghasilkan fitur dan siap digunakan pada tahap berikutnya. Proses alur SPPF ditunjukkan pada Gambar 17.



Gambar 17. Alur SPPF

Max Pooling adalah operasi downsampling untuk mengurangi ukuran *height* dan *width* pada *feature map* dengan cara mengambil nilai terbesar pada setiap area kecil kernel yang digeser. Proses ini merangkum informasi lokal sehingga fitur menjadi lebih ringkas dan komputasi pada layer berikutnya lebih ringan. Konsep *maxpooling* bekerja dengan menentukan ukuran kernel seperti 2x2 dan pergeseran kernel (*stride*), Kernel kemudian digeser dari kiri ke kanan dan dari atas ke bawah. Pada setiap posisi kernel, semua nilai di dalam area tersebut dibandingkan, lalu nilai paling besar dipilih sebagai satu nilai *output*. Proses ini diulang hingga seluruh *feature map* diproses. *Input feature map* 4x4 dan dibagi menjadi blok 2x2 lau di ambil nilai terbesar setiap blok.

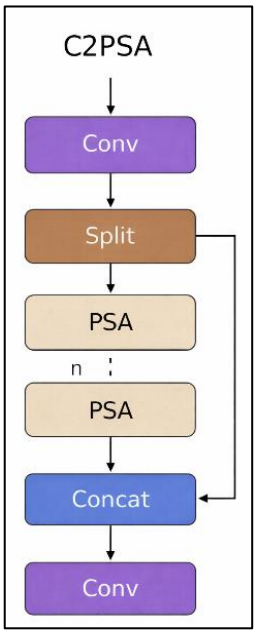


Gambar 18. Ilustrasi Max Pooling

g. Modul C2PSA (Cross-Stage Partial Spatial Attention)

Modul C2PSA (Cross-Stage Partial Spatial Attention) meningkatkan kualitas fitur dengan membuat model lebih fokus pada objek, sehingga deteksi lebih akurat terutama untuk objek kecil. Modul CSP lebih efisien karena hanya sebagian fitur yang diproses lebih berat. Mulai dari Conv untuk menyesuaikan *channel*, lalu *feature map* di split menjadi dua bagian, jalur *shortcut* yang diteruskan langsung dan jalur utama di proses oleh *PSA*

(spatial attention) sebanyak n kali dan kedua jalur di digabung di *concat* pada dimensi *channel* dan difusikan dengan Conv terakhir untuk menghasilkan fitur keluaran. Proses C2PSA ditunjukkan pada Gambar 19.



Gambar 19. Alur C2PSA

PSABlock merupakan blok pemrosesan yang menggabungkan dua komponen utama, yaitu *Attention dan Feed Forward Network* (FFN), untuk memperkuat representasi *feature map*. Komponen *Attention* berfungsi menekankan informasi pada lokasi fitur yang paling relevan misalnya area objek dengan cara mempelajari hubungan antar-posisi pada peta fitur. Jika X adalah fitur masukan, maka keluaran tahap *Attention* dikombinasikan kembali dengan masukan awal melalui residual *connection* dengan persamaan sebagai berikut**:**

$$F_1 = X + \text{Attention}(X) \tag{25}$$

Dimana:

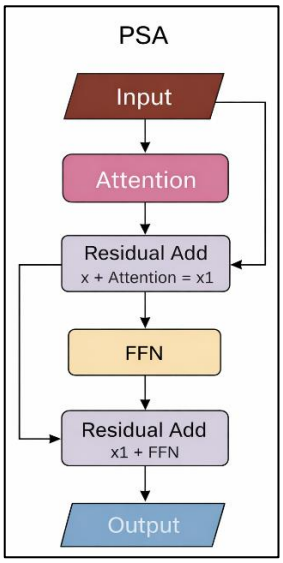
X : Merupakan fitur masukan
F_1 : Fitur hasil integrasi antara masukan awal dan *output attention*.

F_1adalah adalah fitur gabungan antara input dan *output attention*. Selanjutnya F_1diproses oleh FFN umumnya dua Conv 1×1 dan hasilnya dijumlahkan kembali dengan F_1 melalui residual *connection* untuk menghasilkan output akhir PSABlock

$$F_{out} = F_1 + \text{FFN}(F_1) \tag{26}$$

Di mana F_{out}merupakan keluaran akhir PSABlock. Dua residual *connection* membantu menjaga informasi awal dan menstabilkan aliran gradien saat pelatihan,

sehingga fitur yang dihasilkan lebih kuat untuk deteksi. Proses alur PSA dapat ditunjukkan pada Gambar 20.



Gambar 20. Alur PSA

Attention bertujuan membuat model lebih fokus pada bagian fitur yang penting. diawali dengan melewatkan fitur masukan X ke konvolusi 1×1 untuk membentuk representasi awal yang digunakan dalam pembentukan komponen *query* (Q), *key* (K), dan *value* (V). Keluaran konvolusi kemudian dilakukan *reshape* untuk menyesuaikan dimensi tensor misalnya dari bentuk peta fitur menjadi deret posisi, sehingga perhitungan attention antar posisi dapat dilakukan dengan lebih mudah. Setelah itu, fitur tersebut dipisahkan (*split*) menjadi tiga bagian, yaitu Q, K, dan V, sesuai pembagian *channel* yang telah ditentukan. Komponen Q dan K digunakan untuk menghitung skor *attention* dengan cara mengalikan matriks Q dengan transpose dari K. Skor ini menunjukkan tingkat keterkaitan antar-posisi pada *feature map*, dan dapat dituliskan sebagai:

$$S = (QK^T) \cdot \text{scale} \tag{27}$$

Nilai S kemudian dinormalisasi menggunakan fungsi *softmax* untuk menghasilkan matriks bobot *attention* A, sehingga bobot yang lebih besar diberikan pada posisi yang lebih relevan:

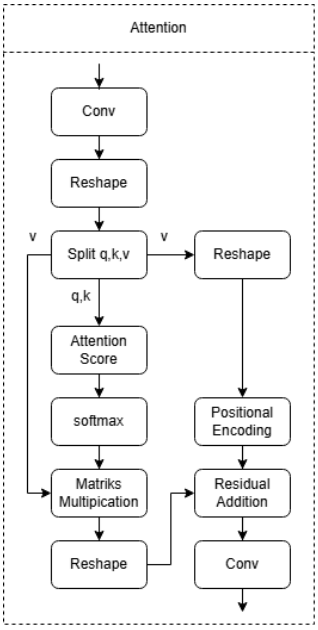
$$A = \text{Softmax}(S) \tag{28}$$

Matriks bobot A selanjutnya digunakan untuk membobotkan komponen V. Proses ini dilakukan melalui perkalian matriks sehingga menghasilkan keluaran fitur berbobot F_{att}:

$$F_{att} = AV \tag{29}$$

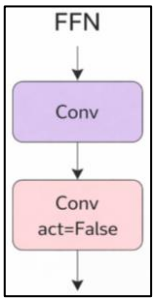
Output F_{att} kemudian di *reshape* kembali agar sesuai dengan bentuk *feature map* dua dimensi. Secara paralel, komponen V juga diproses untuk menambahkan informasi posisi positional *encoding*. Pada tahap ini, V diubah bentuknya *reshape* lalu dilewatkan ke konvolusi *depthwise* 3×3 untuk menangkap informasi lokal berdasarkan posisi. Hasil

positional *encoding* tersebut kemudian ditambahkan residual *addition* ke fitur hasil attention agar keluaran tetap mempertahankan informasi lebih stabil. Tahap terakhir menggunakan konvolusi 1×1 sebagai proyeksi untuk menyatukan dan merapikan fitur sehingga menghasilkan *output* akhir *blok attention*. Alur proses dari pembentukan Q, K, V hingga proyeksi akhir ditunjukkan pada Gambar 21.



Gambar 21. Alur Attention

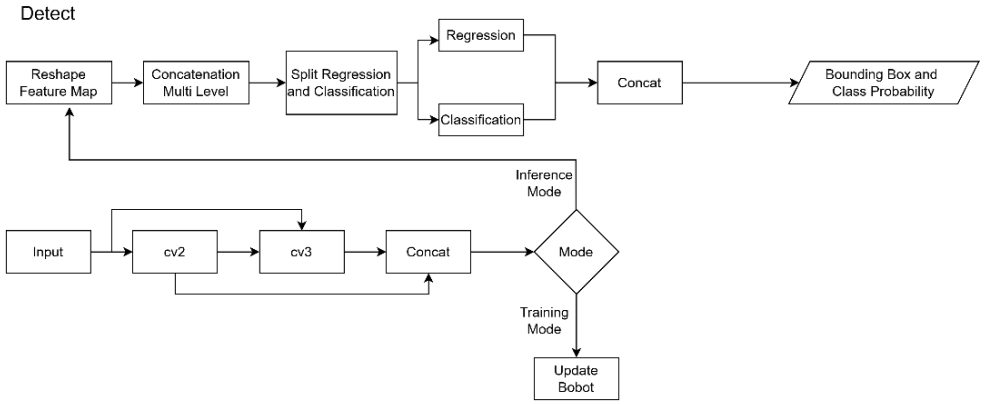
Feed Forward Network (FFN) pada PSABlock berfungsi memperkuat fitur setelah tahap *Attention*. Pada tahap awal, *feature map* hasil *attention* dilewatkan ke konvolusi 1×1 untuk melakukan transformasi kanal, sehingga informasi fitur menjadi lebih detail sebelum diteruskan. Selanjutnya, keluaran tahap pertama diproses oleh konvolusi 1×1 kedua untuk menyesuaikan kembali 2 jumlah *channel* ke dimensi yang diinginkan. Pada konvolusi kedua ini aktivasi dinonaktifkan (act=*False*), sehingga prosesnya bersifat linear dan menghasilkan keluaran akhir FFN. Alur pemrosesan FFN ini ditampilkan pada Gambar 22.



Gambar 22. Alur FFN

h. Modul Detect (Detection Head)

Modul *Detect* merupakan bagian akhir dari arsitektur YOLOv11 yang berfungsi menghasilkan prediksi objek berdasarkan *feature map* hasil pemrosesan *backbone* dan *neck*. Modul ini menerima tiga *feature map* multi skala (P3, P4, dan P5) dan memproses masing-masing level secara paralel untuk menghasilkan keluaran berupa regresi *bounding box* dan prediksi kelas objek. Secara umum, modul *detect* bekerja dalam dua mode, yaitu *training mode* dan *inference mode*. Pada *training mode*, keluaran modul *Detect* digunakan secara langsung untuk perhitungan *loss function*, sedangkan pada *inference mode*, keluaran tersebut diproses lebih lanjut untuk menghasilkan prediksi objek akhir yang memiliki makna spasial dan siap digunakan. Alur kerja modul *detect* secara keseluruhan ditunjukkan pada Gambar 23.



Gambar 23. Alur Detection Head

Training Mode setiap *feature map* hasil keluaran *neck* diproses secara independen pada masing-masing level skala melalui dua jalur prediksi, yaitu jalur regresi *bounding box* (cv2) dan jalur klasifikasi (cv3). Kedua jalur menerima masukan *feature map* yang sama, kemudian menghasilkan keluaran regresi dan klasifikasi yang selanjutnya digabungkan melalui operasi *concatenation* pada dimensi kanal. Keluaran hasil *concatenation* pada setiap level *feature map* dipertahankan dalam bentuk terpisah tanpa penggabungan antar level. Seluruh keluaran tersebut kemudian diteruskan ke modul *loss* untuk dilakukan pencocokan antara prediksi dan *ground truth*, serta perhitungan nilai loss yang digunakan dalam proses *backpropagation* pada tahap pelatihan model.

Inference Mode keluaran hasil *concatenation* dari setiap level *feature map* diproses lebih lanjut untuk menghasilkan prediksi objek akhir. Proses diawali dengan melakukan *reshape feature map* sehingga setiap lokasi spasial diperlakukan sebagai satu kandidat deteksi. Selanjutnya, seluruh kandidat deteksi dari berbagai level skala digabungkan melalui *concatenation multi-level* untuk membentuk satu representasi prediksi yang seragam. Representasi prediksi tersebut kemudian dipisahkan menjadi dua komponen, yaitu regresi dan klasifikasi. Pada komponen regresi dilakukan proses *decoding* untuk memperoleh koordinat *bounding box* yang merepresentasikan posisi objek pada citra masukan. Sementara itu, pada komponen klasifikasi dilakukan pemrosesan untuk menghasilkan probabilitas kelas objek. Hasil dari kedua komponen tersebut selanjutnya

digabungkan kembali melalui operasi *concatenation*. Keluaran akhir dari *inference mode* pada modul *detect* berupa koordinat *bounding box* dan probabilitas kelas yang merepresentasikan hasil prediksi objek.

2. Validasi Model

Validasi model tahap untuk mengevaluasi kemampuan model mendeteksi objek. Pelatihan dilakukan selama 200 *epoch*, dan setiap *epoch* model dievaluasi menggunakan 159 data validasi yang tidak digunakan saat pelatihan model. Pada tahap validasi model menjalankan prediksi seluruh data validasi tanpa memperbarui bobot. Hasil prediksi *bounding box* dan kelas objek kemudian dibandingkan dengan *ground truth* menggunakan ukuran kecocokan seperti IoU (*Intersection over Union*). Dari perbandingan ini dihitung metrik performa, seperti *precision*, *recall* dan mAP, serta dapat dilihat juga nilai *validation loss*. Hasil validasi digunakan untuk memantau perkembangan performa selama 200 *epoch*, mendeteksi kemungkinan *overfitting*, dan menentukan nilai metrik validasi terbaik berdasarkan kinerja pada data validasi.

3. Penyetelan Hyperparameter

Dalam hasil validasi menunjukkan bahwa model YOLO belum mencapai kinerja yang diharapkan misalnya *loss* masih tinggi, metrik *precision*, *recall*, dan mAP belum stabil, atau terdapat gejala *overfitting* seperti *training loss* membaik tetapi *validation* loss memburuk maupun *overfitting* maka dilakukan penyetelan hiperparameter. Penyetelan ini bertujuan mencari konfigurasi pelatihan yang lebih stabil sesuai, seperti penyesuaian *learning rate*, *batch size*, *weight decay*, sehingga proses pembelajaran menjadi lebih stabil dan hasil deteksi meningkat. Setelah hiperparameter diperbarui, model dilatih ulang dan kembali dievaluasi pada data validasi. Proses berulang *training* dan validasi, penyetelan ini dapat diulang sampai performa model pada validasi menunjukkan peningkatan dan mencapai kondisi yang lebih konsisten sesuai jumlah *epoch* yang ditetapkan, kemudian bobot terbaik dipilih berdasarkan hasil validasi. Tabel 1 merupakan beberapa contoh *hyperparameter* yang digunakan dalam pelatihan model YOLOv11.

Tabel 1. Hyperparameter yang Digunakan dalam Pelatihan Model

Hyperparameter	Deskripsi	Nilai default
Batch Size	Jumlah citra per iterasi.	16
Epochs	Jumlah putaran training.	200
Image Size	Ukuran input citra	640×640
IoU Threshold	Batas IoU untuk menentukan *true positive* evaluasi prediksi	0.5
Confidence Threshold	Batas skor minimum prediksi	0.25

Model. Proses pelatihan YOLOv11 selesai dan diperoleh nilai validasi yang memenuhi kriteria, diperoleh model yang membuat bobot hasil pelatihan. Model ini kemudian disimpan dalam format ".pt" sehingga dapat digunakan kembali untuk proses inferensi tanpa melatih ulang. Model YOLOv11 yang telah dilatih diintegrasikan sebagai tahap *preprocessing* untuk mengurangi *motion blur* pada sistem konveyor. Hasil citra yang lebih tajam dari proses *deblurring* kemudian diproses oleh YOLOv11 untuk melakukan deteksi dan klasifikasi kualitas biji jagung secara lebih akurat pada kondisi kecepatan konveyor yang menyebabkan *blur*.

2.4.5 Testing

1. Input Data Testing

Hasil model YOLOv11 yang telah diintegrasikan dengan metode total variation *deblurring*. Tahap berikutnya, memasukkan data *testing* yang telah disiapkan untuk diproses oleh sistem. Data pengujian ini digunakan untuk mengevaluasi kinerja deteksi dan klasifikasi kualitas biji jagung pada sistem konveyor, yang berpotensi menimbulkan blur pada kecepatan. Seluruh data *testing* kemudian dikelompokkan ke dalam beberapa kategori sesuai kondisi pengambilan, khususnya variasi kecepatan konveyor. Setiap kelompok data diproses secara terpisah agar hasilnya dapat dibandingkan dan dianalisis lebih lanjut.

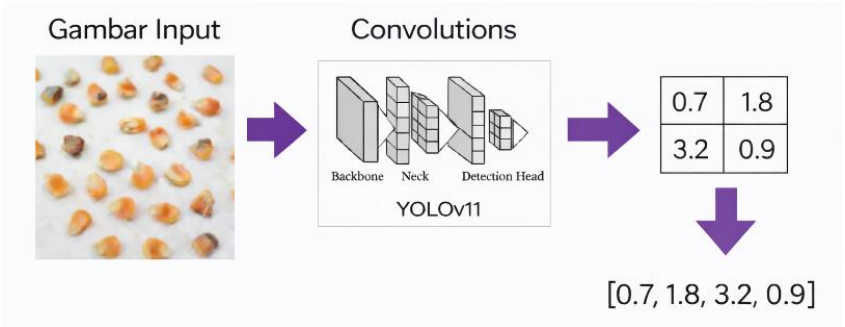
2. Resize

Tahap awal *testing*, sistem terlebih dahulu melakukan penyesuaian ukuran citra menjadi 640 x 640 piksel agar seluruh gambar uji memiliki ukuran dimensi yang sama dan sesuai ukuran *input* yang digunakan saat proses *training* model. Sehingga model dapat memproses data biji jagung yang lebih baik. Dengan ukuran yang konsisten, perhitungan pada feature map dan prediksi *bounding box* menjadi lebih tepat, serta objek pada gambar berada pada skala yang lebih sesuai dengan kondisi *training*.

3. Deteksi dan Klasifikasi Kualitas Biji Jagung

Setelah model YOLOv11 di latih dan diperoleh bobot terbaik, proses deteksi dilanjutkan pada tahap pengujian untuk mengenali biji jagung pada sistem konveyor. YOLOv11 melakukan ekstraksi fitur secara bertahap melalui lapisan konvolusi pada *backbone*, kemudian menggabungkan informasi multi skala pada *neck*, dan akhirnya menghasilkan prediksi pada *detection head*. Pada tahap ini, model menghasilkan *bounding box* untuk menunjukkan lokasi biji jagung serta klasifikasi untuk menentukan kategori kualitas biji jagung dalam satu alur inferensi. Sehingga proses deteksi berjalan cepat dan efisien.

Proses deteksi dimulai saat input gambar biji jagung yang diperoleh dari konveyor dimasukkan ke sistem. Setiap citra yang masuk disesuaikan ukuranya ke ukuran *input* yang digunakan saat pelatihan agar format data konsisten dan sesuai dengan konfigurasi model, kemudian diproses oleh model YOLOv11 melalui beberapa tahap utama. Pada tahap *backbone*, citra diproses dalam konvolusi untuk mengekstrak fitur seperti tepi, tekstur, dan bentuk. Hasil ekstraksi ini berupa *feature map*, yaitu representasikan spasial dari karakteristik visual citra dalam bentuk matriks dengan dimensi H×W×C, yaitu tinggi (H) dan lebar (W) sebagai posisi spasial, serta kanal (C) sebagai kumpulan pola fitur yang dipelajari model. Berikut merupakan contoh proses input pada Gambar 24.



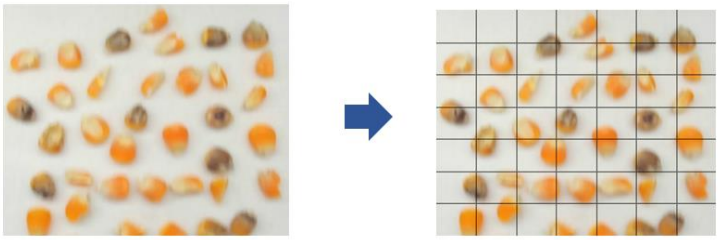
Gambar 24. Input Gambar Testing

Selanjutnya *feature map* dari beberapa level resolusi digabung dan diperkuat pada bagian *neck* untuk menangkap informasi objek pada berbagai skala. Tahap akhir dilakukan pada *detection head*, di mana setiap posisi *grid* pada *feature map* digunakan untuk memprediksi koordinat *bounding box* dan skor kelas. Dengan demikian, *output* YOLOv11 pada data *testing* berupa kumpulan prediksi yang terdiri dari lokasi objek (*bounding box*), label kualitas, dan nilai kepercayaan (*confidence score*), yang kemudian dapat dianalisis lebih lanjut untuk evaluasi performa sistem**.**

Hasil dari konvolusi berupa *feature map*, yaitu representasi spasial dari karakteristik visual citra dalam bentuk matriks. Sebagai contoh, feature map berukuran 2×2 menghasilkan nilai aktivasi [0.7, 1.8, 3.2, 0.9]. Nilai-nilai ini kemudian diratakan (*flattening*) menjadi vektor fitur berdimensi empat, yang akan digunakan sebagai *input* dalam prediksi kelas objek dan koordinat *bounding box* pada tahap berikutnya. Proses ini merupakan komponen penting dalam YOLO karena secara langsung mempengaruhi akurasi klasifikasi dan lokalisasi objek yang dideteksi.

Grid Point dan *Stride feature map* yang dihasilkan dari proses konvolusi berbagai skala P3 P4 P5, menetapkan *grid point* pada setiap sel *feature map* sebagai titik referensi untuk melakukan prediksi. *Grid point* ini merepresentasikan lokasi spasial tertentu pada *feature map* yang akan digunakan model untuk memperkirakan keberadaan objek. Selain itu, YOLOv11 juga menggunakan nilai stride pada tiap *level feature map*, yaitu perbandingan skala antara *feature map* dan citra *input*, sehingga prediksi yang dihasilkan pada ruang *feature map* dapat dipetakan kembali ke koordinat citra asli secara tepat. YOLOv11 dapat memprediksi lokasi *bounding box* sekaligus kelas objek secara lebih konsisten pada *berbagai level feature map.*

Prediksi kelas pada YOLOv11 adalah proses menentukan kategori objek untuk setiap deteksi yang dihasilkan. Pada setiap lokasi prediksi di *feature map*, *detection head* menghasilkan *skor* untuk tiap kelas. *Skor* ini kemudian diubah menjadi probabilitas 0–1. Kelas dengan probabilitas tertinggi dipilih sebagai label kelas pada *bounding box*, sedangkan nilai probabilitasnya menjadi tingkat keyakinan *confidence* misalkan 0.5 berarti model memiliki keyakinan 50% bahwa objek termasuk kelas tersebut. Dengan demikian, YOLOv11 tidak hanya memprediksi posisi objek, tetapi juga mengklasifikasikan objek yang terdeteksi Berikut merupakan gambar prediksi koordinat pada gambar 25.



Gambar 25. Prediksi koordinat bounding box

Setelah dilakukan deteksi dan klasifikasi sistem akan menghasilkan output berupa gambar yang telah memiliki *bounding box*, klasifikasi dan juga nilai *confidence*. Hasil tersebut akan dievaluasi berdasarkan kategori masing-masing.

2.4.6 Evaluasi Sistem

Mengevaluasi kinerja model YOLOv11 dalam mendeteksi dan mengklasifikasikan kualitas biji jagung, penelitian ini menggunakan confusion matriks deteksi objek yaitu *precision*, *recall*, mAP50, dan mAP50–95. *Precision* mengukur ketepatan deteksi, *recall* mengukur kemampuan menemukan objek, sedangkan mAP50 dan mAP50–95 menilai akurasi deteksi berdasarkan ambang IoU, yaitu tingkat tumpang tindih antara *bounding box* prediksi dan *ground truth*. Model dengan nilai mAP dan *precision–recall* terbaik dipilih sebagai model yang paling optimal untuk digunakan pada deteksi dan klasifikasi kualitas biji jagung.