BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penulis menemukan beberapa penelitian terdahulu yang terkait dengan *segmantic segmentation* model dengan *backbone* yang berbeda-beda, seperti:

2.1.1 EffiSegNet: Gastrointestinal Polyp Segmentation through a Pre-Trained EfficientNet-based Network with a Simplified Decoder

Penelitian dengan judul "EffiSegNet: Gastrointestinal Polyp Segmentation through a Pre-Trained EfficientNet-based Network with a Simplified Decoder" yang dilakukan oleh Ioannis A. Vezakis, et al pada tahun 2024 [11] bertujuan untuk mengembangkan model segmentasi baru bernama EffiSegNet, yang didasarkan pada pendekatan transfer learning menggunakan keluarga EfficientNet. Penelitian tersebut menggunakan *backbone EfficientNet* sebagai dasar model, dan menemukan variasi *EfficientNetB4* memiliki performa terbaik dengan nilai F1-score = 0.9552, mIoU = 0.9056, Precision = 0.9679, dan Recall = 0.9429.

Dari penelitian ini, *backbone EfficientNetB3* akan digunakan daripada *EfficientNetB4* dengan alasan keterbatasan kekuatan komputasi, alasan lain *EfficientNetB3* dipilih dikarenakan akurasi yang digapai sangat tinggi dan ingin melihat performa *EfficientNetB3* saat dibandingkan dengan *backbone* lainnya.

2.1.2 Comparison of Backbones for Semantic Segmentation Network

Penelitian dengan judul "*Comparison of Backbones for Semantic Segmentation Network* " yang dilakukan oleh Rongyu Zhang, et al pada tahun 2020 [9] bertujuan untuk membandingkan kelebihan dan kekurangan berbagai macam backbone. Penelitian ini menggunakan model AD-LinkNet untuk *backbone ResNet101* dan *Xception*, sedangkan model UNET digunakan untuk *backbone VGG-16.* Pembandingan *backbone* dilakukan dengan menggunakan metrik IoU, *backbone VGG-16* mendapatkan nilai terendah dengan 62.94%,

sedangkan *ResNet101* mendapatkan nilai 63.37%, backbone Xception mendapatkan nilai tertinggi dengan 64.81%

Dari penelitian ini, bisa dilihat bahwa pengunaan backbone yang berbeda dapat menghasilkan model dengan performa yang bervariasi, dengan itu, *Xception* dan *ResNet101* akan digunakan sebagai backbone model untuk dibandingkan performanya saat melakukan *semantic segmentation* kepada dataset peneltian ini.

2.1.3 Deep Learning-Based Semantic Segmentation of Turmeric Crop Images Using the DeepLabV3+ Architecture

Penelitian dengan judul "*Deep Learning-Based Semantic Segmentation of Turmeric Crop Images Using the DeepLabV3+ Architecture*" yang dilakukan oleh Preethi R, et al pada tahun 2024 [7] dilatarbelakangi oleh kebutuhan untuk mengklasifikasikan tanaman kunyit yang sakit dan yang sehat. Penelitian ini memiliki tujuan untuk melakukan pertanian presisi dengan memberikan informasi yang berguna kepada petani untuk intervensi yang cepat dan pengelolaan tanaman yang lebih baik. Tujuan utama dari usulan ini adalah untuk mengotomatisasi proses pertanian dan meningkatkan hasil dari proses pertanian.

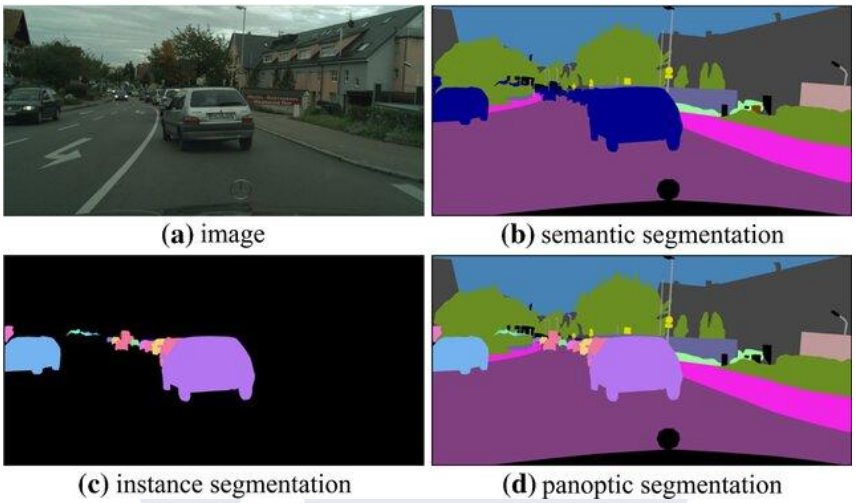
Hasil penelitian menunjukkan bahwa arsitektur DeepLabV3+ mampu melakukan *semantic segmentation* dengan sangat baik pada gambar tanaman kunyit, memberikan hasil yang akurat dalam mengidentifikasi area tanaman, penyakit, dan latar belakang.

2.2 Tinjauan Teori

2.2.1 Semantic Segmentation

Image Segmentation adalah teknik dalam *computer vision* yang mengelompokkan semua *pixel* pada sebuah foto digital ke dalam kelompok-kelompok yang memiliki sifat atau karakteristik yang sama. Gambar 2.1 menunjukkan perbedaan antara jenis-jenis segmentasi yang umum

digunakan. Terdapat tiga jenis algoritma yang umum digunakan dalam *image segmentation*, yaitu *semantic segmentation*, *instance segmentation*, dan *panoptic segmentation*. Masing-masing memiliki tujuan dan aplikasi yang berbeda, seperti terlihat dalam ilustrasi yang menampilkan cara kerja setiap jenis segmentasi.



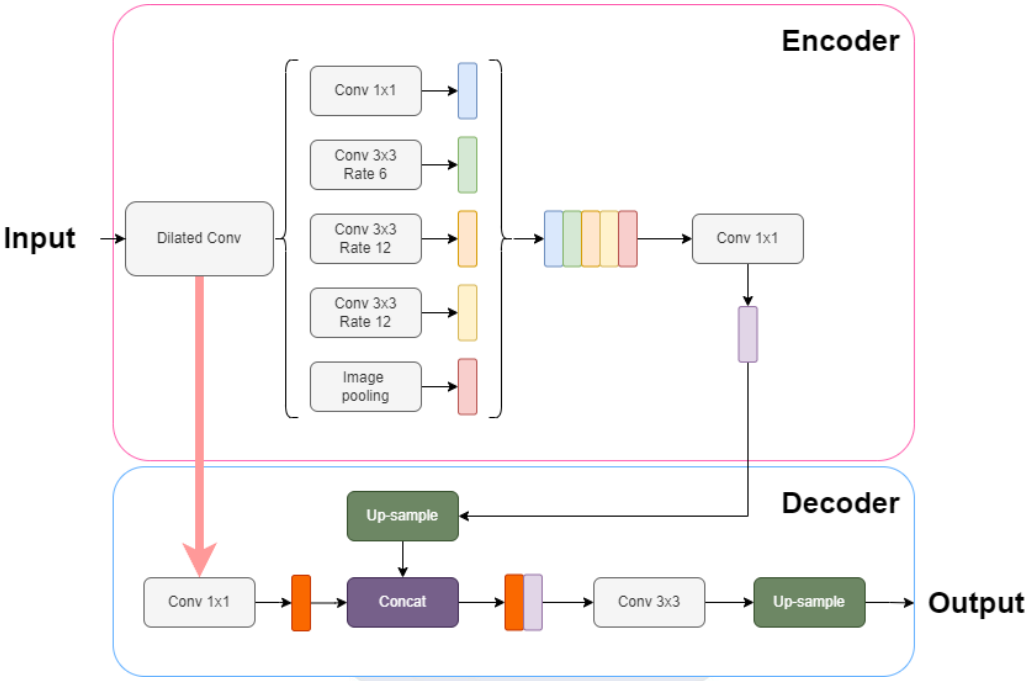
Gambar 2.1 Contoh tiga jenis segmentation

Semantic segmentation adalah algoritma yang memberikan label pada setiap *pixel* dalam gambar, memungkinkan identifikasi dan pengenalan kelompok *pixel* yang membentuk pola tertentu. Algoritma *instance segmentation* mengklasifikasikan setiap objek yang ada sebagai objek unik. Algoritma *panoptic segmentation* menggabungkan *instance* dan *semantic*, menghasilkan algoritma yang mengklasifikasikan objek pada gambar ke dalam kategori, serta membedakan objek yang masuk ke dalam kategori yang sama dengan objek lainnya. Algoritma *semantic segmentation* digunakan untuk mengklasifikasikan setiap *pixel* dari gambar ke dalam kategori tertentu, seperti membedakan antara objek dan latar belakan

Untuk membangun model *semantic segmentation*, diperlukan arsitektur yang menjadi dasar model tersebut. Beberapa arsitektur yang umum digunakan antara lain Fully Convolutional Networks (FCN), U-Net, dan DeepLab.

2.2.2 DeepLabV3+

DeepLabV3+ adalah model *deep learning* yang dirancang untuk *semantic segmentation* dalam bidang *computer vision*. DeepLabV3+ merupakan pengembangan dari model DeepLabV3, dengan tambahan fitur-fitur yang memperbaiki hasil segmentasi [12].



Gambar 2.2 Arsitektur DeepLabV3+

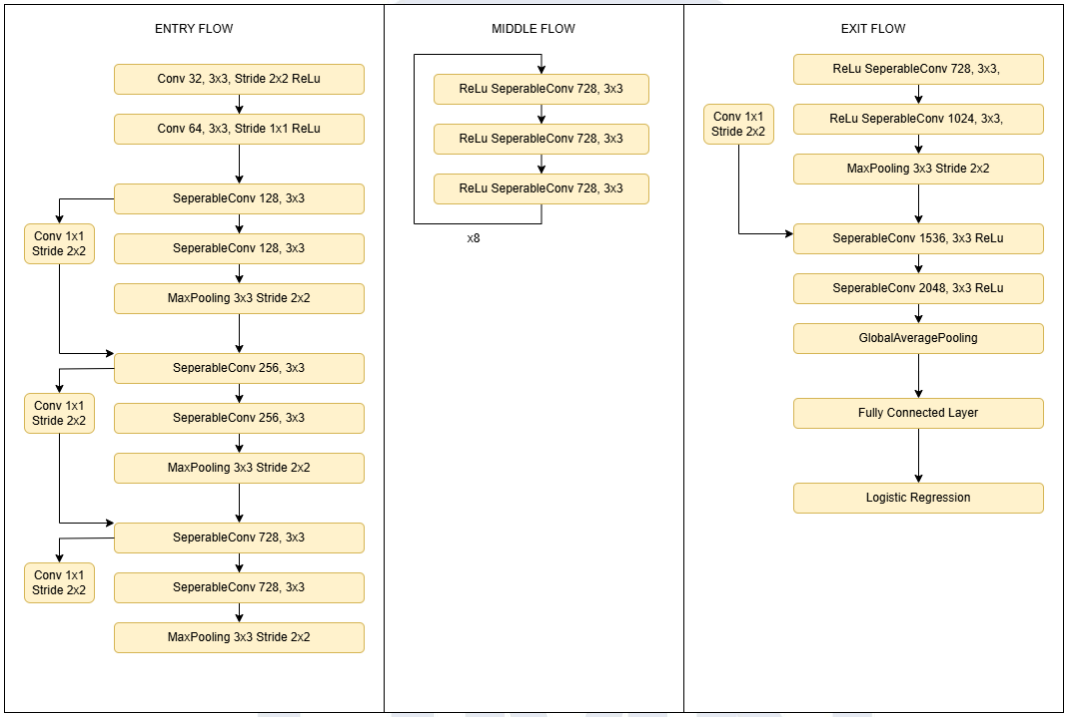
Seperti yang bisa dilihat pada gambar 2.2, model ini mengadopsi arsitektur *encoder-decoder*, di mana bagian *encoder* menggunakan *dilated convolutions* untuk memperluas *field of view* tanpa mengorbankan resolusi spasial. Bagian *decoder* kemudian memulihkan detail spasial yang mungkin hilang selama proses *encoding*, menghasilkan peta segmentasi yang lebih detail. Selain itu, DeepLabV3+ memanfaatkan *Atrous Spatial Pyramid Pooling* (ASPP) untuk menangkap informasi kontekstual pada berbagai skala, yang sangat berguna dalam menangani objek dengan ukuran dan bentuk yang bervariasi.

Keunggulan DeepLabV3+ terletak pada kemampuannya menghasilkan segmentasi yang akurat dan efisien. Kemampuan DeepLabV3+ untuk menangani kompleksitas objek dan variasi skala membuatnya menjadi

pilihan yang kuat untuk berbagai aplikasi yang memerlukan segmentasi gambar yang presisi.

2.2.3 Xception

Xception adalah arsitektur *convolutional neural network* yang didasarkan sepenuhnya pada *depthwise seperable convolutional layers*. *Xception* berasal dari hypotesis arsitektur *Inception* yang dimodifikasi menjadi lebih baik kinerjanya [13].



Gambar 2.3 Arsitektur *Xception*

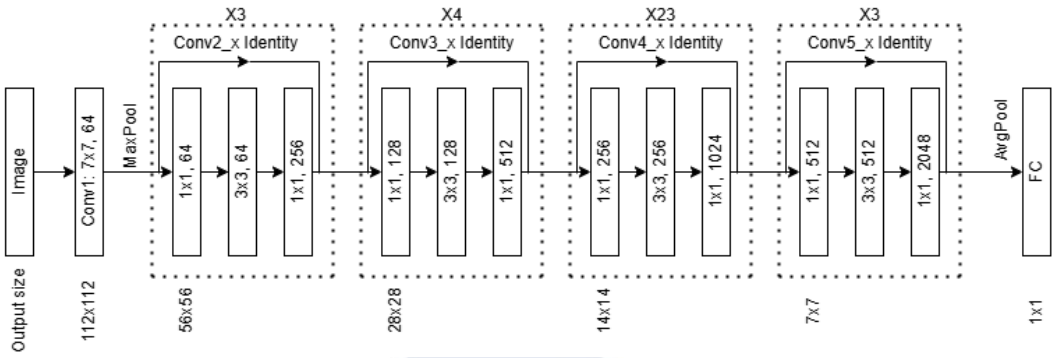
Seperti yang bisa dilihat pada gambar 2.3, Xception membagi arsiterkturnya menjadi 3 bagian, *Entry*, *Middle*, dan *Exit Flow*. *Entry Flow* digunakan untuk mengekstrak fitur dasar dari gambar *input*, *middle flow* menggunakan *Depthwise Separable Convolutions* yang direpetisi sebanyak 8 kali untuk menyempurnakan fitur yang ada, dan *exit flow* digunakan untuk memproses fitur yang sudah ada kemudian melakukan prediksi pada gambar tersebut.

Keunggulan *Xception* terletak pada arsitekturnya yang sederhana, mudah dipahami, dan dimodifikasi. Berdasarkan jurnal *Xception* karya François

Chollet, ketika dibandingkan dengan model populer lainnya seperti VGG-16, *ResNet*-152, dan *Inception* V3, *Xception* menunjukkan peningkatan performa yang signifikan. Oleh karena itu, banyak tugas segmentasi semantik menggunakan *Backbone Xception* yang dilatih dengan *ImageNet*.

2.2.4 ResNet-101

ResNet adalah arsitektur *convolutional neural network* yang menggunakan pendekatan *residual learning* untuk mempermudah pelatihan *deep learning*. ResNet101 merupakan bagian dari keluarga *Residual Networks* (ResNet) yang diperkenalkan oleh Kaiming He et al. pada tahun 2015 [14].



Gambar 2.4 Arsitektur *ResNet-101*

Arsitektur ResNet101 terdiri dari 101 lapisan jaringan yang dapat dilihat pada gambar 2.4, dengan struktur utama berbasis blok *residual*. Setiap blok *residual* terdiri dari *shortcut connection* seperti "conv2_x" yang memungkinkan data melewati beberapa lapisan tanpa diubah selama dimensi data masih sama. Mekanisme ini membantu mencegah masalah degradasi yang sering terjadi pada *deep learning*, di mana performa model justru menurun meskipun jumlah lapisan meningkat.

Keunggulan ResNet101 terletak pada kemampuannya untuk memberikan performa yang tinggi dengan tingkat komputasi yang tidak terlalu besar. Dibandingkan dengan model lain seperti VGG dan AlexNet, ResNet101 menunjukkan hasil yang lebih baik dalam berbagai tugas seperti *classification*, *semantic segmentation*, dan *objeck detection*. Model ini

sering digunakan sebagai *backbone* untuk berbagai aplikasi *deep learning* karena efisiensinya dalam mempelajari fitur kompleks dari data.

2.2.5 EfficientNetB3

EfficientNetB3 adalah salah satu arsitektur *convolutional neural network* dari keluarga *EfficientNet* yang dirancang untuk mencapai efisiensi tinggi dalam performa dan penggunaan komputasi sistem. *EfficientNet* dibuat oleh Mingxing Tan dan Quoc V. Le pada tahun 2019 [15] berdasarkan arsitektur *MobileNet* dengan tambahan metode yang mengoptimalkan *scaling* model menggunakan metode *compound scaling*.

Tabel 2.1 Tabel arsitektur *baseline*

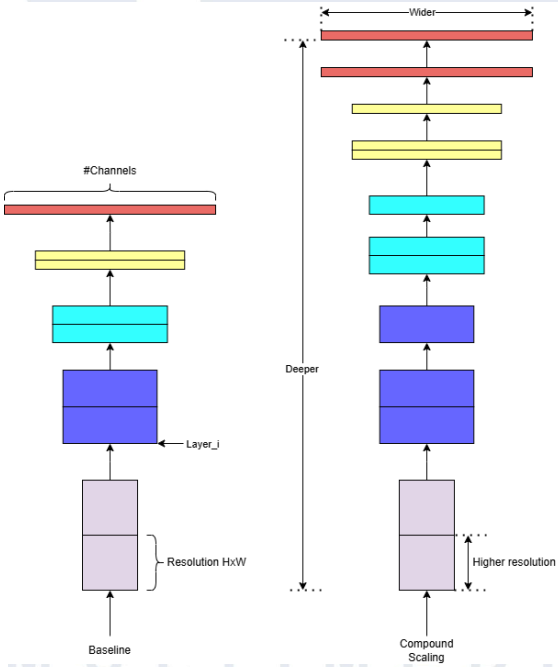
Stage	Operator	Resoultion	#channels	#layers
1	Conv3x3	224 X 224	32	1
2	MbConV1, K3x3	112 X 112	16	1
3	MbConV6, K3x3	112 X 112	24	2
4	MbConV6, K5x5	56 X 56	40	2
5	MbConV6, K3x3	28 X 28	80	3
6	MbConV6, K5x5	14 X 14	112	3
7	MbConV6, K5x5	14 X 14	192	4
8	MbConV6, K3x3	7 X 7	320	1
9	Conv1x1 & Pooling & FC	7 X 7	1280	1

Semua variasi dari keluarga *EfficientNet* didasarkan dengan arsitektur *baseline* pada tabel 2.1, untuk mendapatkan arsitektur yang lebih besar dapat dilakukannya *scaling* dengan rumus *compound model scaling.* rumus sebagai berikut:

$$Depth: d = \alpha^{\varphi}$$
$$Withd: w = \beta^{\varphi}$$
$$Resolution: r = \gamma^{\varphi}$$

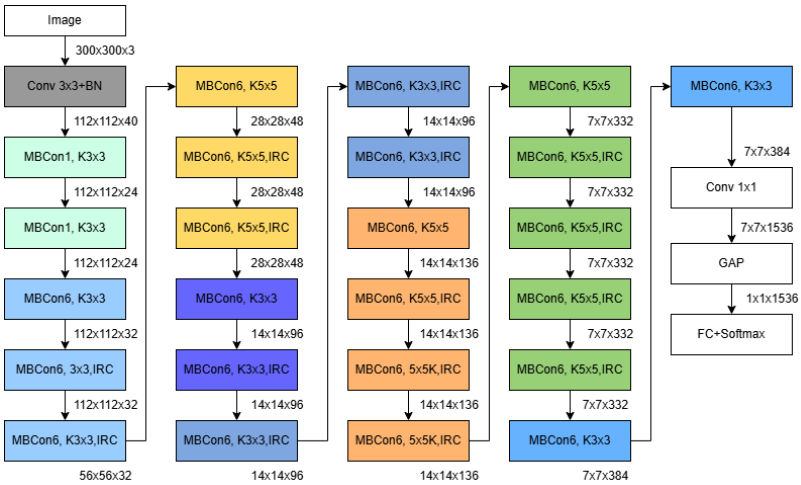
φ merupakan nilai yang diatur oleh penguna, nilai tersebut mengatur seberapa banyak daya komputasi yang dapat digunakan. Perubahan nilai φ akan selalu meningkatkan nilia FLOPS sebesar 2^{φ}.

Saat ingin melakukan *scaling* dengan *compound model scaling*, ada 2 langkah yang perlu dilakukan. Pertama dimulai dari *baseline* EfficientNetB0, nilai φ akan diset menjadi 1 dengan asumsi daya komputasi cukup, kemudian melakukan pencarian nilai α, β, γ dengan mengikuti rumus $\alpha * \beta^2 * \gamma^2 \approx 2$. Setelah perhitungan, nilai α = 1.2, β = 1.1, dan γ = 1.15. kedua, nilai-nilai tersebut kemudian dimasukkan ke dalam rumus *compound model scaling* berserta nilai φ untuk mendapatkan nilai terbaik untuk *scaling baseline*. Nilai φ didapatkan dari ukuran *scaling* yang di inginkan, contoh jika ingin mencari nilai φ untuk *EfficientNetB7* maka nilai $\varphi = 7$.



Gambar 2.5 Perbandingan arsiterktur sebelum dan sesudah *Compound model scaling*

Gambar 2.5 menujukan perbandingan baseline dengan baseline yang sudah berhasil *scaling*. Penelitian ini menggunakan EfficientNetB3 yang arsitekturnya dapat dilihat pada gambar 2.6.



Gambar 2.6 Arsitektur *EfficientNetB3*

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA