BAB 2
TINJAUAN PUSTAKA

2.1 Tanaman Padi

Padi (*Oryza sativa L.*) merupakan biji-bijian penting yang menjadi sumber makanan utama bagi lebih dari separuh populasi global, terutama di Asia Timur dan Tenggara. Ini adalah makanan pokok yang dapat disiapkan dengan berbagai cara, termasuk direbus dan digiling menjadi tepung. Secara budaya dan kuliner, padi memiliki peranan yang signifikan dan dijadikan bahan dalam berbagai hidangan di Asia, Timur Tengah, dan lainnya. Tanaman padi sendiri adalah rumput tahunan, tumbuh hingga sekitar 1,2 meter tingginya, dengan sistem akar yang lebar dan serabut serta malai yang menghasilkan biji. Budidaya padi memiliki sejarah yang kaya, dengan bukti tertua yang berasal dari sekitar 7000–5000 SM di Cina. Saat ini, padi banyak ditanam di Asia, dengan produksi yang signifikan di Cina, India, Indonesia, dan Bangladesh [1], [2].

2.2 Penyakit Tanaman Padi

Di Indonesia, di mana beras sangat penting sebagai makanan utama, tanaman ini sering menghadapi masalah dengan penyakit daun yang beragam. Salah satu yang paling dikhawatirkan adalah Hawar Daun Bakteri (BLB) atau dalam bahasa inggris disebut *bacterial leaf blight*. BLB disebabkan oleh bakteri *Xanthomonas oryzae pv. oryzae* dan menunjukkan gejala awal berupa bercak berair yang kemudian berubah menjadi kuning dan coklat, dan akhirnya membuat daun menjadi layu dan mati. Dampaknya sangat besar karena bisa mengurangi hasil panen beras. Untuk mengatasinya, petani sering menggunakan varietas beras yang tahan terhadap BLB, memberikan perlakuan khusus pada benih, dan berusaha mencegah genangan air di sawah. Selain BLB, ada juga penyakit seperti Bercak Coklat atau *brown spot* yang disebabkan oleh jamur *Cochliobolus miyabeanus*, dan Hawar Daun atau *leaf smut* yang disebabkan oleh jamur *Ustilaginoidea virens*. Bercak Coklat membuat daun berbintik-bintik coklat dengan lingkaran kuning di sekitarnya, yang bisa mengurangi kualitas dan hasil panen [15], [16]. Untuk

mengendalikannya, petani biasanya menggunakan fungisida dan mengubah tanaman yang ditanam di sawah. Hawar Daun muncul sebagai bercak hitam seperti bubuk di atas daun, dan bisa merusak biji beras. Pengendalian Hawar Daun melibatkan perlakuan khusus pada benih dan menjaga kebersihan sawah.

2.3 Citra Digital

Secara garis besar, pengolahan citra digital merujuk pada penggunaan komputer untuk memproses gambar dua dimensi. Dalam lingkup yang lebih luas, pengolahan citra digital mencakup pemrosesan data dua dimensi secara umum. Citra digital adalah kumpulan nilai-nilai numerik yang direpresentasikan dalam bentuk larik (*array*) dan dapat berupa bilangan real atau kompleks, yang diwakili oleh serangkaian bit tertentu. Sebuah citra dapat dianggap sebagai fungsi f(x, y) dengan M baris dan N kolom, di mana x dan y adalah koordinat spasial, dan nilai amplitudo f pada titik koordinat (x, y) disebut sebagai intensitas atau tingkat keabuan citra pada titik tersebut. Jika nilai x, y, dan amplitudo f semuanya bersifat terbatas (*finite*) dan diskrit, maka citra tersebut dapat disebut sebagai citra digital [17].

2.4 Pengolahan Citra

Pengolahan citra atau *image processing* cabang dari ilmu komputer yang mempelajari teknik-teknik untuk menganalisis, memanipulasi, dan memproses gambar digital. Tujuan utamanya adalah untuk mendapatkan informasi yang bermanfaat dari gambar, seperti meningkatkan kualitas gambar, mengurangi *noise*, mendeteksi objek, atau melakukan segmentasi untuk memisahkan bagian-bagian yang berbeda dari gambar. Proses ini melibatkan penerapan berbagai teknik dan algoritma komputer yang dirancang khusus untuk menangani data gambar [17].

2.5 Citra RGB

Citra RGB, yang juga dikenal sebagai citra *true color*, adalah jenis citra digital yang memiliki matriks data dengan ukuran M × N × 3 yang mewakili warna merah (*red*), hijau (*green*), dan biru (*blue*) untuk setiap pikselnya. Setiap warna dasar memiliki rentang nilai tersendiri, dimulai dari 0 hingga 255. Warna dari setiap

piksel ditentukan oleh campuran intensitas merah, hijau, dan biru yang terkandung dalam matriks tersebut [17].

2.6 Augmentasi Data

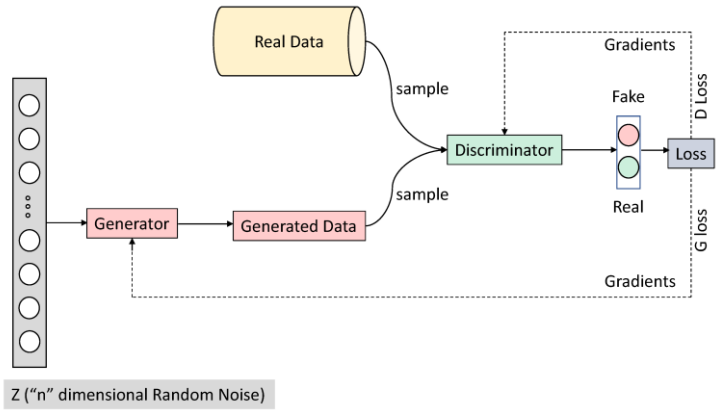
Augmentasi data merupakan salah satu teknik pra-pemrosesan standar dalam visi komputer, yang digunakan terutama untuk mengatasi masalah penurunan kinerja akibat pelatihan yang kurang memadai atau *overfitting* karena kurangnya data. Metode augmentasi data tradisional mencakup translasi gambar, rotasi gambar, pemutaran gambar, serta pemotongan dan reposisi gambar [18].

2.7 GAN

Generative Adversarial Networks (GAN) diperkenalkan oleh Ian J. Goodfellow et al. pada tahun 2014 sebagai kerangka kerja baru untuk mengestimasi model generatif melalui proses adversarial. Dalam GAN, dua model dilatih secara bersamaan: model generatif *G* yang menangkap distribusi data, dan model diskriminatif *D* yang mengestimasi probabilitas bahwa sebuah sampel berasal dari data pelatihan atau dari *G*. Pelatihan *G* dilakukan untuk memaksimalkan probabilitas *D* melakukan kesalahan, menciptakan permainan minimax antara *G* dan *D*.

GAN terdiri dari dua jaringan saraf: generator dan diskriminator. Generator mengambil variabel *noise* sebagai input dan menghasilkan data sintetis yang menyerupai data asli. Diskriminator, di sisi lain, berusaha membedakan antara data asli dan data yang dihasilkan oleh generator. Kedua jaringan ini dilatih secara bergantian dengan tujuan akhir agar generator dapat menghasilkan data yang tak dapat dibedakan dari data asli oleh discriminator [19].

Berikut Gambar 2.1 Arsitektur GAN (Generative Adversarial Network) menunjukkan *arsitektur Generative Adversarial Network* (GAN), yang terdiri dari dua komponen utama: Generator dan Discriminator [13].



Gambar 2.1 Arsitektur GAN (*Generative Adversarial Network*) [13]

Fungsi tujuan (*objective function)* dari GAN ditunjukkan dalam Persamaan 2.1 berikut.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim Pdata(x)}[logD(x)] + \mathbb{E}_{z\sim Pz(z)}[\log(1 - D(G(z)))] \quad \textbf{(2.1)}$$

Distribusi z adalah distribusi *noise input* dari *G*. *G* dan *D* bertarung satu sama lain, dengan *G* terus-menerus meningkatkan kemampuannya untuk menangkap distribusi sampel yang sebenarnya dan menghasilkan gambar-gambar berkualitas tinggi, dan *D* meningkatkan kemampuannya untuk membedakan gambar-gambar yang dihasilkan. GAN asli telah terbukti memberikan *output* yang lebih realistis dibandingkan dengan algoritma gambar generatif lainnya. Sepanjang proses ini, kedua bagian diperbarui secara simultan, dan ini berlanjut hingga tercapai keseimbangan *Nash*, yaitu suatu kondisi dalam teori permainan yang menyatakan bahwa tidak ada pemain yang bisa mendapatkan keuntungan dengan mengubah strategi awal mereka. Hal ini mengimplikasikan bahwa generator dan diskriminator telah mencapai kondisi optimal mereka [19], [20] .

2.8 DCGAN

Deep Convolutional Generative Adversarial Networks (DCGAN) adalah salah satu pengembangan dari *Generative Adversarial Networks* (GAN) yang bertujuan untuk memanfaatkan arsitektur *Convolutional Neural Networks* (CNN) dalam konteks pembelajaran representasi tanpa supervisi. DCGAN pertama kali diperkenalkan oleh Alec Radford, Luke Metz, dan Soumith Chintala pada tahun

2016. Arsitektur ini dirancang untuk mengatasi masalah stabilitas dalam pelatihan GAN dan untuk meningkatkan kualitas representasi fitur yang dipelajari dari data gambar yang tidak berlabel. DCGAN memperkenalkan beberapa perubahan arsitektural pada GAN konvensional untuk mencapai stabilitas pelatihan dan performa yang lebih baik. Perubahan utama meliputi:

1. Penggantian Lapisan Pooling

   Mengganti lapisan pooling dengan konvolusi bertingkat (*strided convolution*) dalam diskriminator dan konvolusi bertingkat fraksional (*fractional-strided convolution*) dalam generator.

2. *Batch Normalization*

   Menerapkan normalisasi *batch* pada hampir semua lapisan di generator dan diskriminator untuk menstabilkan proses pelatihan.

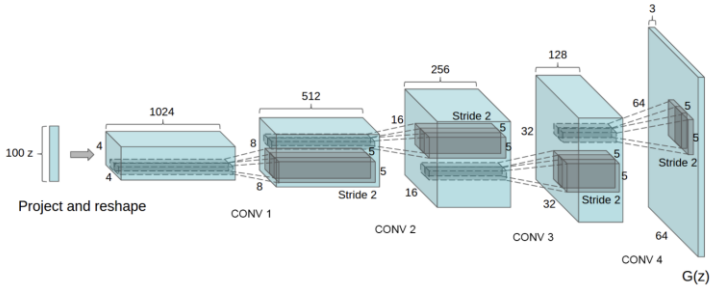
3. Penghapusan Lapisan *Fully Connected*

   Menghilangkan lapisan *fully connected* pada arsitektur yang lebih dalam untuk meningkatkan stabilitas.

4. Aktivasi ReLU dan Tanh

   Menggunakan fungsi aktivasi ReLU pada semua lapisan di generator kecuali lapisan output yang menggunakan Tanh, serta menggunakan aktivasi LeakyReLU pada semua lapisan di diskriminator.

Pelatihan DCGAN menggunakan pendekatan standar GAN di mana generator dan diskriminator dilatih secara bergantian dalam permainan *minimax*. Generator menghasilkan gambar dari distribusi noise input, sementara diskriminator berusaha membedakan antara gambar asli dan gambar yang dihasilkan. Tujuan dari diskriminator adalah memaksimalkan kemampuan membedakan, sementara tujuan generator adalah meminimalkan kemampuan diskriminator untuk mendeteksi gambar palsu. Berikut Gambar 2.2 Arsitektur DCGAN – Generator adalah arsitektur dari DCGAN yang diambil dari jurnal DCGAN [10]:



Gambar 2.2 Arsitektur DCGAN – Generator [10]

Dalam proses pelatihan DCGAN, penting untuk memastikan bahwa data input sudah dinormalisasi ke rentang [−1,1]. Karena pada lapisan output generator menggunakan fungsi aktivasi Tanh. Selain itu, gambar-gambar input diubah ukurannya menjadi 128x128 piksel untuk menjaga keseimbangan antara detail gambar dan kompleksitas komputasi. Langkah-langkah praproses ini sangat penting untuk memastikan stabilitas pelatihan dan performa optimal dari model DCGAN [10].

2.9 Generator

Generator pada DCGAN dimulai dengan vektor laten berdimensi rendah, yang direpresentasikan sebagai sebuah persegi panjang kecil pada Gambar 2.2 Arsitektur DCGAN – Generator *Vektor noise* acak adalah vektor yang berisi nilai-nilai acak yang diambil dari suatu distribusi probabilitas, biasanya distribusi normal (Gaussian). Nilai-nilai dalam vektor diambil dari distribusi normal adalah *mean* 0 dan standar deviasi 1 [10]. Vektor ini kemudian diproyeksikan dan diubah bentuknya menjadi bentuk 3D dengan ukuran yang lebih besar. Selanjutnya, generator menggunakan serangkaian konvolusi *transpose*.

Cara Kerja Konvolusi *Transpose*:

1) Upsampling

Konvolusi *transpose* meningkatkan ukuran spasial data. Berbeda dengan konvolusi biasa yang memperkecil ukuran gambar, konvolusi *transpose* memperbesarnya. Ini dicapai dengan menambahkan nol di antara elemen-elemen data asli sebelum menerapkan konvolusi.

2) Konvolusi

Setelah *upsampling*, filter konvolusi diterapkan pada data yang diperbesar. Filter ini mempelajari pola dan fitur tertentu dalam data.

Dengan menerapkan konvolusi *transpose* secara berulang, representasi pada setiap lapisan generator semakin besar. Dimulai dari fitur-fitur kasar (warna, bentuk dasar), generator secara bertahap membangun detail yang lebih halus dan menghasilkan gambar output berdimensi tinggi yang menyerupai gambar asli [10]. Lapisan terakhir menggunakan fungsi aktivasi Tanh untuk menghasilkan nilai piksel dalam rentang [-1, 1].

2.10 Diskriminator

Discriminator pada DCGAN menggunakan arsitektur CNN standar, dengan lapisan konvolusi yang memperkecil ukuran spasial data pada setiap lapisan. Diskriminator ini mengambil gambar, baik yang dihasilkan oleh generator (palsu) maupun gambar asli, dan mengklasifikasikannya sebagai asli atau palsu.

Discriminator pada DCGAN terdiri dari beberapa lapisan konvolusi yang secara berturut-turut mengurangi resolusi spasial data. Setiap lapisan konvolusi biasanya diikuti oleh fungsi aktivasi LeakyReLU. Lapisan terakhir menggunakan fungsi aktivasi Sigmoid untuk menghasilkan probabilitas bahwa gambar input adalah asli (1) atau palsu (0) [10].

2.11 Loss Function

Seperti yang ditunjukkan oleh persamaan (2.1) Generator mencoba mengurangi nilai fungsi ini sementara discriminator mencoba meningkatkan nilainya. Dalam konteks *min-max*, pendekatan ini terlihat efektif. Namun, dalam praktiknya, hal ini bisa menyebabkan generator "jenuh". Ini berarti bahwa jika generator tidak dapat mengejar kemampuan discriminator, sering kali generator akan berhenti belajar atau meningkatkan kemampuannya lebih lanjut. Saat discriminator dilatih, ia bertugas mengklasifikasikan data asli dan data palsu yang dibuat oleh generator.

Loss function GAN dapat dikategorikan lebih lanjut menjadi dua bagian dan keduanya akan digunakan dalam penelitian ini untuk mengoptimalkan hasil yang diperoleh [19]:

1. Discriminator Loss

   Discriminator memberikan penalti pada dirinya sendiri jika salah mengklasifikasikan data asli sebagai palsu atau data palsu (yang dibuat oleh generator) sebagai asli. Ini dilakukan dengan memaksimalkan fungsi berikut:

$$L_D = -(\mathbb{E}_{x\sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]) \quad \textbf{(2.2)}$$

   Dimana :

   a) $\mathbb{E}_{x\sim P_{data}(x)}[\log D(x)]$ adalah ekspektasi dari log probabilitas bahwa sampel nyata x dianggap nyata oleh discriminator D.
   b) $\mathbb{E}_{z\sim P_z(z)}[\log (1 - D(G(z)))]$ adalah ekspektasi dari log probabilitas bahwa sampel palsu G(z) dianggap palsu oleh discriminator D.

2. Generator Loss

$$\mathrm{L_G} = -\mathbb{E}_{\mathrm{z}\sim \mathrm{P_z(z)}}[\log \mathrm{D(G(z))}] \quad \textbf{(2.3)}$$

   Dimana :

   $\mathbb{E}_{z\sim P_z(z)}[\log D(G(z))]$ adalah ekspektasi dari log probabilitas bahwa sampel palsu G(z) dianggap nyata oleh discriminator D.

Saat generator dilatih, ia menghasilkan gambar dari *noise* acak. Gambar ini kemudian melewati discriminator, yang akan mengklasifikasikannya sebagai "Asli" atau "Palsu" berdasarkan kemampuannya untuk membedakan antara gambar asli dan gambar palsu. Kerugian pada generator dihitung berdasarkan klasifikasi oleh discriminator. Generator mendapat "*reward*" jika berhasil mengelabui discriminator sehingga gambar palsu dianggap asli, dan mendapat "penalti" jika gagal. Untuk melatih generator, persamaan diatas diminimalkan.

2.12 Backpropagation

Backpropagation, atau *backward propagation of errors*, adalah algoritma yang digunakan untuk melatih jaringan neural dengan cara meminimalkan fungsi loss. Algoritma ini bekerja dengan menghitung gradien dari fungsi loss terhadap parameter-parameter dalam jaringan neural (misalnya bobot dan bias), dan menggunakan informasi ini untuk memperbarui parameter tersebut sehingga fungsi loss dapat diminimalkan.

Dalam konteks *Generative Adversarial Networks* (GANs), terdapat dua jaringan neural yang dilatih secara bersamaan: generator dan discriminator. *Backpropagation* digunakan untuk menghitung gradien dan memperbarui parameter pada kedua jaringan ini [19].

2.12.1 Gradien untuk Generator

Rumus ini digunakan untuk memperbarui parameter dari generator dalam GAN. Rumus ini dinotasikan sebagai berikut:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \tag{2.4}$$

Di mana θg adalah parameter dari generator dan $z^{(i)}$ adalah vektor noise input ke generator. Tujuan dari rumus ini adalah meminimalkan *log-likelihood* dari discriminator untuk mengidentifikasi data yang dihasilkan sebagai palsu, sehingga generator dapat belajar menghasilkan data yang lebih realistis.

2.12.2 Gradien untuk Diskriminator

Rumus ini merupakan bagian dari algoritma *Generative Adversarial Networks* (GANs) yang digunakan untuk memperbarui parameter dari discriminator. Rumus ini dinotasikan sebagai berikut:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log \mathrm{D}\left(x^{(i)}\right) + \left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right] \tag{2.5}$$

Di mana θd adalah parameter dari discriminator, m adalah jumlah contoh data, $x^{(i)}$ adalah contoh data nyata, D adalah fungsi discriminator, G adalah generator, dan $z^{(i)}$ adalah vektor *noise* input ke generator. Tujuan dari rumus ini adalah memaksimalkan *log-likelihood* dari discriminator untuk membedakan antara data nyata dan data yang dihasilkan oleh generator.

2.12.3 Adam Optimizer

Adam adalah algoritma optimisasi yang dapat digunakan sebagai ganti dari prosedur *classical stochastic gradient descent* untuk memperbarui bobot secara iteratif yang didasarkan pada data training. Adam dapat dikatakan merupakan kombinasi antara RMSprop dan *Stochastic Gradient Descent* dengan momentum. Adam menghitung *learning rate* individu untuk parameter yang berbeda. Nama "Adam" berasal dari "*adaptive moment estimation*" karena Adam menggunakan estimasi gradien momen pertama dan kedua untuk mengadaptasi *learning rate* untuk setiap bobot jaringan saraf [21].

1. Pembaruan momen pertama dan kedua

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad \textbf{(2.6)}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_{t^2} \quad \textbf{(2.7)}$$

2. Bias-koreksi momen

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \textbf{(2.8)}$$

3. Pembaruan bobot

$$\theta = \theta - \alpha \frac{\widehat{m}t}{\sqrt{\widehat{v}_t + \varepsilon}} \quad \textbf{(2.9)}$$

Dimana

$\alpha =$ laju pembelajaran atau $learning\ rate$ (0.001)

$\beta_1, \beta_2 =$ parameter momen $(\beta_1 = 0.9, \beta_2 = 0.999)$

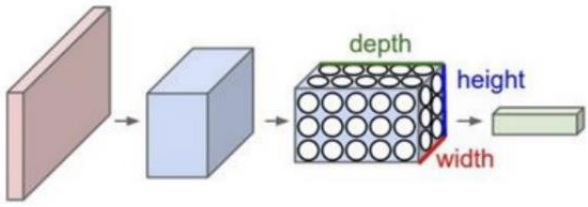
$\theta =$ bobot yang diperbarui

2.13 Convolution Neural Network

Convolutional Neural Network (CNN) adalah jenis jaringan saraf tiruan *deep feedforward* yang sangat banyak digunakan dalam analisis citra. CNN terinspirasi oleh mekanisme biologis, khususnya pola konektivitas antara neuron yang mirip dengan organisasi visual korteks pada hewan. Pada korteks visual, neuron kortikal merespons stimulus hanya di area terbatas dari bidang visual, yang dikenal sebagai bidang reseptif (*receptive field*) [22].

CNN memiliki satu lapisan masukan (*input layer*), satu lapisan keluaran (*output layer*), dan beberapa lapisan tersembunyi yang biasanya terdiri dari lapisan konvolusi (*convolutional layers*), lapisan pooling, lapisan normalisasi, lapisan ReLU dan lapisan *fully connected*. Semua lapisan ini diatur secara bertingkat, mirip dengan cara membuat sandwich dengan berbagai bahan yang disusun berlapis-lapis.

Tidak seperti jaringan perceptron multilapis (MLP) yang arsitekturnya berbentuk dua dimensi, CNN menggunakan arsitektur tiga dimensi yang mencakup lebar (*width*), tinggi (*height*), dan kedalaman (*depth*). Setiap lapisan dalam CNN mengubah volume input tiga dimensi menjadi volume output tiga dimensi yang terdiri dari aktivasi neuron. Gambar berikut ini mengilustrasikan dimensi dalam CNN:



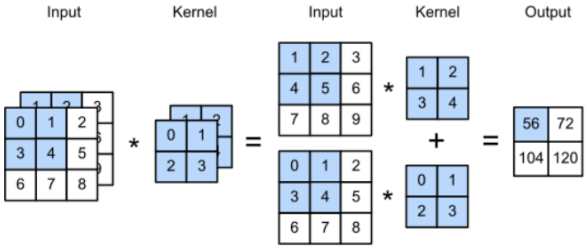
Gambar 2.3 Dimensi CNN [22]

2.13.1 Convolution Layer

Convolution layer inti dari CNN yang melakukan operasi konvolusi, yang berarti menerapkan beberapa filter ke gambar untuk mengekstrak fitur. Filter ini disebut kernel. Filter ini diterapkan pada setiap sub-bagian gambar, yang

selanjutnya ditentukan oleh lapisan parameter konektivitas lokal. Setiap aplikasi filter menghasilkan nilai skalar untuk lokasi piksel tertentu, yang bila digabungkan di semua lokasi piksel, sering disebut sebagai peta fitur. Peta fitur akan dihitung sesuai dengan filter konvolusi tertentu.

Convolutional layer dalam arsitektur CNN umumnya menggunakan lebih dari satu filter. Jika menggunakan empat filters, maka *convolutional layer* akan berisi sejumlah neurons yang tersusun dalam kisi berukuran 28x28x4. Dengan demikian, akan ada empat neurons yang melihat area yang sama pada (citra) masukkan tersebut [23].



Gambar 2.4 Proses Convolution Layer [23]

CNN umumnya menggunakan lebar langkah atau stride = 1 dengan zero padding seperti pada persamaan 2.10 berikut dengan F adalah ukuran filter.

$$P = \frac{(F-1)}{2} \tag{2.10}$$

Sementara untuk menghitung ukuran matriks *full padding* keluaran dari proses konvolusi ini, dapat digunakan rumus sebagai berikut.

$$H = \frac{I - \mathrm{K} + 2\mathrm{P}}{\mathrm{S}} + 1 \tag{2.11}$$

Dengan H = ukuran feature maps keluaran,

I = ukuran citra masukan

K = ukuran kernel atau filter

P = jumlah padding

S = jumlah stride

2.13.2 Fully Connected Layer (Dense Layer)

Lapisan *fully connected* menghubungkan setiap neuron di lapisan input ke setiap neuron di lapisan output. Ini dilakukan dengan operasi matriks linear, yaitu komputasi menggunakan suatu perkalian matriks yang diikuti dengan bias *offset*. Lapisan ini mengubah vektor input menjadi tensor dengan dimensi lebih tinggi. Perhitungan pada layer ini dilakukan dengan persamaan sebagai berikut [22].

$$fc = \sum_{j=1} w_{i,j}^{T} x_i + b_j \quad \textbf{(2.12)}$$

Di mana:

$b =$ nilai vector bias

$w =$ nilai matriks bobot

$x =$ vector input

2.13.3 Convolutional Layer Transpose

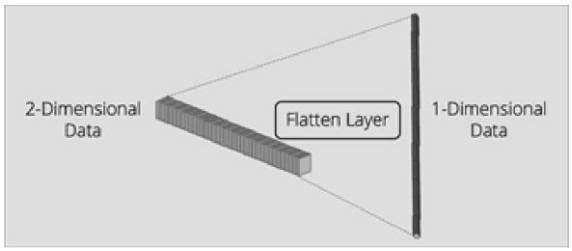
Konvolusi *transpose* (juga dikenal sebagai *deconvolution*) adalah operasi yang meningkatkan ukuran spasial dari *input feature map*. Operasi ini sering digunakan dalam model generator untuk menghasilkan gambar yang lebih besar dari representasi yang lebih kecil [24].

Secara sederhana, konvolusi *transpose* dapat dibayangkan sebagai kebalikan dari konvolusi biasa. Jika konvolusi biasa mengurangi ukuran *feature map*, maka konvolusi *transpose* akan memperbesarnya. Proses ini melibatkan penempatan kernel konvolusi pada *input feature map*, melakukan perkalian *elemen-wise*, dan menjumlahkan hasilnya.

Dalam model generator ini, *layer-layer Conv2DTranspose* digunakan untuk secara bertahap meningkatkan ukuran spasial dari representasi awal yang dihasilkan oleh *layer Dense*. Setiap *layer Conv2DTranspose* juga mengurangi jumlah *channel,* sehingga representasi menjadi lebih terfokus pada fitur-fitur spasial yang relevan untuk menghasilkan gambar.

2.13.4 Flatten Layer

Flatten Layer berguna seperti jembatan yang menghubungkan antara *Convolutional Network* dan *Fully Connected Layer*. *Layer* ini biasa dipakai dikarenakan gambar dan *output* dari *Convolutional Network* berupa 2 dimensi, sedangkan di sisi lain *Fully Connected Layer* hanya menerima 1 dimensi input. Oleh karena itu *Flatten Layer* dipakai untuk mengubah data 2 dimensi menjadi 1 dimensi berupa vektor [25]. Konsep ini sangat mirip dengan operasi *reshape* pada *array* di berbagai bahasa pemrograman dan pustaka pemrosesan data seperti NumPy. Ilustrasinya seperti pada Gambar 2.5 berikut.



Gambar 2.5 Ilustrasi Flatten Layer [25]

2.14 Reshape Array

Operasi *reshape* mengubah bentuk tensor tanpa mengubah datanya. Ini berguna untuk mempersiapkan data dari lapisan dense menjadi format yang sesuai untuk operasi konvolusi. Radford et al. [10] menjelaskan bahwa vektor input Z awalnya diproyeksikan menggunakan layer dense untuk mendapatkan ukuran yang sesuai, misalnya 16 elemen, sebelum di-*reshape* menjadi tensor 4D dengan bentuk 4×4×1 [20]. Langkah ini melibatkan penggunaan operasi *reshape* yang dinyatakan dengan rumus:

$$Z"[i,j,k] = Z'[(i \times w \times c) + (j \times c) + k \quad \textbf{(2.13)}$$

Dimana i, j, k adalah indeks untuk tinggi, lebar, dan channel dalam tensor 4D. serta w adalah lebar (*width*) dari tensor setelah *reshape*. c adalah jumlah channel (*channels*) dari tensor setelah *reshape*.

2.15 Batch Normalization

Layer *Batch Normalization* menormalisasi output dari layer Dense untuk menstabilkan dan mempercepat pelatihan model. Normalisasi ini dilakukan dengan menghitung rata-rata (*mean*) dan standar deviasi (*standard deviation*) dari output layer Dense di sepanjang *batch*, lalu menskalakan dan menggeser nilai-nilai tersebut.

Perhitungan *batch normalization* dapat menggunakan rumus berikut [22]:

$$BN = \gamma \cdot \frac{(I - \mu)}{\sqrt{\sigma^2 + \varepsilon}} + \beta \quad \textbf{(2.14)}$$

Dimana:

I = output dari layer dense atau nilai matriks

μ = nilai rata − rata

σ^2 = nilai varians

ε = nilai kecil atau konstan stabilitas numerik

γ dan β = gamma dan beta, parameter yang dipelajari selama pelatihan

2.16 Activation Function

Activation function atau yang biasa disebut dengan fungsi aktivasi, dimaksudkan untuk mengetahui apakah suatu neuron dalam jaringan saraf teraktivasi. Untuk memperbarui parameter bobot jaringan saraf multi-layer, gunakan jaringan fungsi aktivasi non-linier. Jenis aktivasi ini memungkinkan penggunaan beberapa lapisan tersembunyi dalam arsitektur neural network. Berikut adalah beberapa fungsi aktivasi yang akan digunakan dalam penelitian ini [23].

1. ***Rectified Linear Unit*** **(ReLU)**

   Fungsi aktivasi ReLU yang akan digunakan dalam Generator dapat dilihat pada persamaan berikut.

$$f(x) = max(0, x) \quad \textbf{(2.15)}$$

Dengan, $f(x) = ReLu(x) = \begin{cases} x, (x \geq 0) \\ 0, (x < 0) \end{cases}$

Fungsi ini mengem balikan nilai maksimum antara nol dan *input* x. Secara sederhana, jika input x adalah positif atau nol, outputnya adalah x itu sendiri. Sebaliknya, jika input x adalah negatif, outputnya adalah nol. Ini meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada *convolutional layer*.

2. *Leaky Rectified Linear Unit* (Leaky ReLU)

Fungsi aktivasi ReLU yang akan digunakan dalam Diskriminator dapat dilihat pada persamaan berikut.

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \frac{x_i}{a_i} & if\ x_i < 0 \end{cases} \quad \textbf{(2.16)}$$

Dimana a_i merupakan sebuah nilai dengan rentang antara satu (1) sampai tak hingga (∞).

3. Sigmoid

Fungsi aktivasi sigmoid yang akan digunakan dalam Diskriminator, berfungsi mengatur seberapa banyak informasi bisa lewat. Fungsi sigmoid menghasilkan output yang merupakan kisaran nilai antara 0 dan 1 [21]. Persamaan dari fungsi sigmoid dapat dilihat pada persamaan berikut:

$$Sigmoid(x_i) = (\frac{1}{1 + e^{-x_i}}) \quad \textbf{(2.17)}$$

4. Tanh

Fungsi aktivasi *hyperbolic tangent* yang akan digunakan dalam Diskriminator, yaitu tipe aktivasi fungsi dalam *deep learning* dan memiliki beberapa varian, selain itu fungsi aktivasi *hyperbolic* tangent dikenal juga sebagai fungsi tanh yang menghasilkan nilai -1 sampai 1.

$$tanh(x_i) = \frac{2}{1 + e^{-2x}} - 1 \quad \textbf{(2.18)}$$

2.17 Interpolasi Bilinear

Interpolasi bilinear adalah salah satu metode yang umum digunakan dalam penskalaan gambar. Metode ini bekerja dalam dua arah, horizontal dan vertikal, yang dapat diuraikan sebagai berikut [26]:

$$I' = S\left(I, r_x, r_y\right) = S_y(S_x(I, r_x), r_y) = S_x(S_y(I, r_y), r_x) \quad \textbf{(2.19)}$$

Di mana S adalah fungsi penskalaan, I adalah gambar asli, I' adalah gambar yang telah diskalakan, dan rx dan ry adalah rasio penskalaan horizontal dan vertikal masing-masing. Fungsi penskalaan dapat diuraikan menjadi Sx dan Sy, yang merupakan fungsi penskalaan horizontal dan vertikal. Tidak peduli fungsi penskalaan mana yang dijalankan terlebih dahulu, hasil dari penskalaan gambar tetap sama.

Misalkan ada gambar digital berukuran $H \times W$, di mana H dan W menunjukkan tinggi dan lebar gambar. Dimensi gambar yang diinterpolasi adalah $H' \times W'$ Koordinat di posisi (Y', X') dari gambar yang diinterpolasi dipulihkan dari posisi (Y, X),$(Y+1, X)$,$(Y, X+1)$,$(Y+1, X+1)$ dari gambar asli. Hubungan yang sesuai ditunjukkan pada persamaan berikut:

$$ry = \frac{H}{H'}$$
$$rx = \frac{W}{W'} \quad \textbf{(2.20)}$$

$$Y = Y' \cdot \frac{H}{H'}$$
$$X = X' \cdot \frac{W}{W'} \quad \textbf{(2.21)}$$

Interpolasi bilinear adalah metode yang sering digunakan untuk memperbesar atau memperkecil citra dalam pemrosesan gambar digital. Teknik ini bekerja dengan menggunakan empat piksel tetangga terdekat untuk memperkirakan nilai piksel baru. Secara matematis, nilai piksel di posisi (Y', X') dari citra hasil interpolasi dapat dihitung dengan persamaan berikut:

$$f(Y', X') = (1 - dx)(1 - dy)f(Y, X) + dx(1 - dy)f(Y, X + 1) + (1 - dx)dyf(Y + 1, X) + dxdyf(Y + 1, X + 1) \quad \textbf{(2.22)}$$

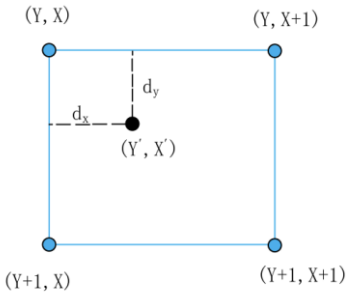
Dimana dx dan dy adalah jarak normalisasi dari koordinat asli ke piksel tetangga, yang dihitung sebagai:

$$dx = X' \cdot \frac{W}{W'} - X \quad \textbf{(2.23)}$$

$$dy = Y' \cdot \frac{H}{H'} - Y \quad \textbf{(2.24)}$$

Pada dasarnya, interpolasi bilinear memadukan nilai-nilai piksel dari empat tetangga terdekat berdasarkan jarak mereka dari titik yang diinginkan, memberikan hasil yang lebih halus dibandingkan metode tetangga terdekat.

Gambar ini menunjukkan hubungan koordinat antara piksel yang diketahui dan piksel tujuan dalam metode interpolasi bilinear. Dalam gambar ini, empat piksel tetangga yang diketahui adalah (Y, X), (Y, X+1), (Y+1, X), dan (Y+1, X+1). Piksel tujuan yang diinterpolasi adalah (Y', X').



Gambar 2.5 Hubungan Peta Koordinat [26]

Dalam interpolasi bilinear, pembulatan ke bawah sering digunakan untuk menentukan piksel integer yang paling dekat dengan posisi piksel target yang akan diinterpolasi. Dengan pembulatan ke bawah, akan dipastikan bahwa koordinat piksel tetangga yang digunakan dalam perhitungan adalah integer dan berada di dalam batas-batas gambar asli [27].

2.18 Normalisasi Piksel

Teknik normalisasi piksel adalah bagian dari proses untuk memastikan bahwa nilai piksel dari fitur lapisan jaringan neural dapat dianalisis secara visual dalam rentang yang standar. Teknik ini penting dalam meningkatkan stabilitas dan kecepatan konvergensi model pembelajaran mesin. Rumus dibawah bertujuan untuk memetakan nilai fitur lapisan yang dinormalisasi kembali ke rentang piksel tampilan [0, 255] dan nantinya akan disesuaikan dengan kebutuhan penelitian penulis untuk menormalisasi ke rentang sebaliknya yaitu [-1, 1]. Rumus tersebut adalah [28]:

$$X(i,j) = (X'(i,j) + 1.0) \times 127.5 \quad \textbf{(2.25)}$$

Dimana $X'(i,j)$ adalah nilai piksel lapisan fitur yang dinormalisasi dan $X(i,j)$ adalah nilai piksel tampilan. Ini bertujuan untuk menampilkan detail struktural yang lebih jelas dalam visualisasi hasil, sehingga menunjukkan bagaimana modul SEBlock meningkatkan kemampuan generalisasi dan ekstraksi fitur dari jaringan. Proses ini bekerja dengan pertama-tama menambah 1 ke nilai $X'(i,j)$, yang mengubah rentang dari [-1, 1] menjadi [0, 2]. Kemudian, hasil ini dikalikan dengan 127.5 untuk mendapatkan nilai dalam rentang [0, 255].

2.19 GoogleNet

GoogLeNet, yang juga dikenal sebagai Inception V1, adalah salah satu arsitektur dalam *transfer learning* yang digunakan untuk klasifikasi gambar. Arsitektur ini merupakan pemenang kompetisi ILSVRC 2014 dengan kesalahan top-5 sebesar 6,67%. Prestasi ini menunjukkan betapa sulitnya bagi manusia untuk mencapai akurasi serupa. GoogLeNet mengambil pendekatan yang inovatif dengan memperkenalkan modul inception, berbeda dengan VGG yang mengikuti beberapa konsep dari LeNet dan AlexNet. Modul inception ini memanfaatkan normalisasi batch, RMSprop, dan distorsi gambar. Meskipun memiliki arsitektur yang lebih dalam dengan 22 lapisan, model ini sangat efisien karena menggunakan jauh lebih sedikit parameter, hanya sekitar 1/12 dari AlexNet.

GoogLeNet mengandalkan modul *inception* yang menggabungkan beberapa ukuran kernel konvolusi (1x1, 3x3, 5x5) dan lapisan *max-pooling* untuk menangkap berbagai fitur, yang kemudian digabungkan untuk membentuk keluaran modul *inception.* Karena banyaknya parameter dalam lapisan fully connected yang dapat menyebabkan *overfitting* dan komputasi berat, GoogLeNet tidak menggunakan lapisan ini seperti AlexNet. Sebagai gantinya, digunakan metode *average pooling* dan *dropout* setelah modul inception untuk mengurangi dimensi dan mencegah overfitting. Arsitektur GoogLeNet terdiri dari total 9 modul *inception*, yang menjadikannya sangat efisien dan efektif dalam mengklasifikasikan gambar [29].

2.20 Frechet Inception Distance

Frechet Inception Distance (FID) adalah sebuat metrik pengujian untuk model GAN yang menangkap kemiripan dari citra yang dibangkitkan dengan citra asli, metode pengujian ini menggunakan model Inception V3. Metode ini menghasilkan bilangan skalar yang dapat di interpretasikan sebagai skor kemiripan. Semakin kecil skor yang didapat maka semakin mirip citra yang dihasilkan oleh model GAN [13].

Secara matematis FID dideskripsikan pada persamaan berikut.

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + \mathrm{Tr}(C + C_w) - 2(CC_w)^{\frac{1}{2}}) \qquad \textbf{(2.26)}$$

Dimana, m = rata-rata pada citra asli

C = matriks kovarian pada citra asli

mw = rata-rata pada citra yang telah dibangkitkan

Cw = matriks kovarian pada citra yang telah dibangkitkan

Tr = jumlah dari elemen diagonal utama

2.21 Tensorflow

TensorFlow adalah sebuah *framework open-source* yang dikembangkan oleh Google untuk keperluan machine learning dan deep learning [30]. TensorFlow menyediakan berbagai alat dan library yang memudahkan proses pengembangan model *neural network*, termasuk *Generative Adversarial Networks* (GANs). Dalam

konteks DCGAN (*Deep Convolutional Generative Adversarial Networks*), TensorFlow menawarkan fleksibilitas dan efisiensi untuk membangun, melatih, dan menguji model. DCGAN adalah jenis GAN yang menggunakan jaringan saraf konvolusional dalam arsitekturnya, yang sering digunakan untuk menghasilkan gambar realistis dari data acak. Penggunaan TensorFlow dalam pengembangan DCGAN memungkinkan implementasi yang lebih mudah dan cepat berkat API yang intuitif dan dukungan ekstensif untuk berbagai fungsi neural network.

Dalam penelitian "*On Aliased Resizing and Surprising Subtleties in GAN Evaluation*," [31] ditemukan bahwa proses mengubah ukuran (*resize*) gambar menggunakan TensorFlow secara default menggunakan interpolasi bilinear, yang tidak menggunakan antialiasing. Hal ini penting untuk diperhatikan dalam evaluasi GAN karena metode interpolasi yang dipilih dapat mempengaruhi kualitas gambar yang dihasilkan serta evaluasi kinerja model. Menurut dokumentasi resmi TensorFlow, metode tf.image.resize_bilinear memang dirancang untuk mengubah ukuran gambar menggunakan interpolasi bilinear, dengan beberapa parameter opsional seperti *align_corners* untuk menyelaraskan sudut gambar *input* dan *output* Dalam berbagai aplikasi pemrosesan citra, interpolasi bilinear sering dipilih karena kemampuannya untuk menghasilkan gambar yang lebih halus tanpa memerlukan komputasi yang intensif.

2.22 Literatur Review

Penelitian tentang *Deep Convolutional Generative Adversarial Networks* (DCGAN) untuk augmentasi data telah menunjukkan berbagai keunggulan dalam meningkatkan akurasi model melalui variasi dan kualitas dataset yang lebih baik. Di bidang pertanian, Wu et al. [3] berhasil meningkatkan akurasi model dari 85% menjadi 92% dalam identifikasi penyakit daun tomat menggunakan DCGAN. Talukdar [5] menemukan bahwa DCGAN dapat menangani ketidakseimbangan kelas dalam klasifikasi penyakit tanaman, memperbaiki kinerja model dengan menghasilkan lebih banyak contoh untuk kelas yang kurang terwakili. Barbedo [6] menegaskan bahwa ukuran dan variasi dataset penting untuk efektivitas *deep learning* dan *transfer learning*, dan menunjukkan bahwa DCGAN dapat

meningkatkan akurasi model secara signifikan. Penelitian lainnya, Zhu et al.[11], menunjukkan bahwa penggunaan cDCGAN yang ditingkatkan dapat mengurangi overfitting dalam penilaian vigor tanaman. Patmawati et al. [32] mengeksplorasi aplikasi DCGAN untuk augmentasi data gambar tanah, yang membantu dalam klasifikasi jenis tanah. Di bidang seni, Purba [33] menggunakan GAN untuk menciptakan karya seni digital abstrak, menunjukkan fleksibilitas teknik ini. Di bidang kesehatan, Venu et al. [13] menemukan bahwa augmentasi data dengan DCGAN dapat meningkatkan akurasi deteksi kondisi kesehatan kompleks dari 88% menjadi 94%. Mourad [34] menggunakan GAN untuk menghasilkan gambar otak sintetis, mendukung penelitian neuroimaging. Zhang et al. [18] menunjukkan bahwa augmentasi data dengan DCGAN meningkatkan akurasi deteksi cacat pada buah pir. Selain itu, kombinasi DCGAN dengan transfer learning terbukti efektif dalam meningkatkan akurasi model. Wu et al. [3] dan Venu et al. [13] menunjukkan bahwa integrasi DCGAN dan transfer learning dapat memanfaatkan pengetahuan dari dataset lain untuk lebih meningkatkan kinerja model di berbagai aplikasi.

Berikut tabel 2.1 menyajikan beberapa literatur review yang berkaitan dengan penelitian.

Tabel 2.1 Literatur Review

No	Judul dan Penulis	Tahun dan Tempat	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
1	DCGAN-Based Data Augmentation for Tomato Leaf Disease Identification Penulis: Qiufeng Wu, Yiping Chen, Jun Meng.	2020, China	Augmentasi data daun tomat menggunakan DCGAN	Penelitian ini dijadikan sebagai acuan utama dalam merancang arsitektur yang menggunakan teknik augmentasi data menggunakan DCGAN, karena pada kasus digunakan dataset daun yang terinfeksi penyakit juga. Hasil dari DCGAN yang menghasilkan gambar yang lebih baik kemudian digunakan untuk klasifikasi dengan pendekatan transfer learning.

2	Handling of Class Imbalance for Plant Disease Classification with Variants of GANs Penulis: Barshneya Talukdar	2020, India	Ketidakseimbangan kelas dalam klasifikasi penyakit tanaman menggunakan varian GAN (DCGAN, CGANs)	Penelitian ini memberikan wawasan tentang penanganan ketidakseimbangan kelas, yang dapat diterapkan dalam penelitian untuk menghasilkan data yang lebih seimbang pada dataset penyakit tanaman yang besar. Berdasarkan penelitian ini, penulis akan menerapkan DCGAN pada datasetnya karena telah terbukti efektif dalam mengatasi masalah tersebut.
3	Impact of Dataset Size and Variety on the Effectiveness of Deep Learning and Transfer Learning for Plant Disease Classification Penulis: Jayme Garcia Arnal Barbedo	2018, Brazil	Pengaruh ukuran dan variasi dataset pada klasifikasi penyakit tanaman	Menekankan pentingnya dataset yang besar dan bervariasi untuk meningkatkan kinerja model. Relevan untuk augmentasi data daun padi dengan DCGAN.
4	Evaluation of Deep Convolutional Generative Adversarial Networks for Data Augmentation of Chest X-ray Images Penulis: Sagar Kora Venu, Sridhar Ravula	2021, Amerika Serikat (USA)	Augmentasi data gambar X-ray medis menggunakan DCGAN	Penelitian ini akan mengadopsi ukuran gambar yang digunakan sebagai parameter penting, dengan memperhatikan detail arsitektur untuk setiap output layernya. Selain itu, pengujian akan dilakukan menggunakan FID score sebagai metrik evaluasi utama untuk mengukur kualitas hasil generasi dari DCGAN.

5	Pear Defect Detection Method Based on ResNet and DCGAN Penulis: Yan Zhang, Shiyun Wa, Pengshuo Sun , Yaojun Wang	2021, China	Augmentasi data buah pir menggunakan DCGAN, khususnya yang berasal dari daerah Korla, Xinjiang. Penelitian ini fokus pada deteksi cacat pada buah pir, seperti blackspot.	Penelitian ini dijadikan sebagai acuan utama dalam pemilihan parameter mean dan standar deviasi serta arsitektur model. Dalam penelitian ini, mean dan standar deviasi dipilih berdasarkan analisis statistik dari data gambar cacat yang tersedia, memastikan bahwa variasi yang dihasilkan realistis dan representatif terhadap cacat sebenarnya.
6	Synthetic Brain Images: Bridging The Gap In Brain Mapping With Generative Adversarial Model Penulis: Drici Mourad, Dr. Kazeem Oluwakemi Oseni	2024, United Kingdom	Objek penelitian ini adalah penerapan arsitektur DCGAN untuk sintesis gambar MRI otak. Studi ini menggunakan dataset gambar MRI sagittal otak yang sehat.	Penelitian ini akan mengadopsi visualisasi data sebelum melakukan DCGAN, serta penggunaan Binary Cross-Entropy dan Adam sebagai fungsi loss dan optimizer.
7	Implementation Of Generative Adversarial Networks For Creating Digital Artwork In The Form Of Abstract Images Penulis: Eric Secada Purba, Hendry	2022, Indonesia	Objek penelitian ini adalah gambar lukisan abstrak.	Penelitian ini dijadikan acuan dalam penerapan arsitektur DCGAN untuk gambar seni. DCGAN menggunakan convolutional layers pada discriminator dan transpose convolutional layers pada generator, dengan batch normalization untuk meningkatkan kecepatan dan stabilitas pelatihan.
8	Augmentasi Data Menggunakan DCGAN Pada Gambar Tanah (Data Augmentation	2023, Indonesia	Objek penelitian ini adalah gambar tanah dari empat kategori: black soil, red soil, clay soil, dan alluvial soil.	Penelitian ini dijadikan acuan utama dalam pemilihan parameter latent space dimension untuk menghasilkan citra sintetis yang berkualitas. Variasi parameter latent space dimension (64, 100, 128,

	Using DCGAN For Soil Image) Penulis: Patmawati, Andi Sunyoto, Emha Taufiq Luthfi			256, dan 512) diuji untuk melihat pengaruhnya terhadap kualitas gambar yang dihasilkan. Evaluasi dengan FID menunjukkan bahwa latent space dimension yang tepat sangat penting untuk menghasilkan gambar sintetis yang realistis, dengan nilai FID terbaik tercapai pada latent space dimension 100.
9	Data augmentation using improved cDCGAN for plant vigor rating Penulis: Fengle Zhu, Mengzhu He, Zengwei Zheng	2020, China	Objek penelitian ini adalah gambar daun tanaman anggrek, yang digunakan untuk menilai kekuatan tanaman.	Walaupun pada penelitian ini digunakan cDCGAN, tetapi arsitektur yang digunakan akan serupa hanya saja tidak menggunakan label. Menggunakan cDCGAN untuk menghasilkan gambar tanaman berkualitas tinggi, relevan untuk augmentasi data menggunakan DCGAN pada penyakit daun padi.