BAB 2 LANDASAN TEORI

2.1 Batik

Batik adalah hasil karya seni yang berupa motif atau pola yang terdapat pada kain yang dihasilkan dengan berbagai Teknik. Pola batik dapat dibagi menjadi dua bagian, yaitu klowong dan isen. Klowong merupakan pola utama atau dasar pada suatu motif batik. Sedangkan isen merupakan pola hias kecil yang bertujuan untuk memberi ragam dan mengisi ruang-ruang kosong dari klowong[9].

2.2 Citra

Sebuah citra dapat didefinisikan sebagai sebuah fungsi dua dimensi, dimana f(x,y) dan nilai dari f pada pasangan koordinat (x, y) disebut sebagai intensitas atau *gray level*. Nilai x, y dan intensitas adalah *finite* dan diskrit, nilai ini didapat dari proses digitalisasi. Sebuah citra terdiri dari sejumlah elemen yang masing-masing mempunyai lokasi dan nilai intensitas tertentu. Elemen tersebut disebut *picture elements* atau piksel[10].

2.3 Komputer Visi

Komputer Visi adalah sebuah cabang ilmu yang mempelajari bagaimana computer dapat mengenali obyek yang diamati atau diobservasi. Komputer adalah kombinasi antara pengolahan citra dan pengenalan pola[11].

2.4 Pengolahan Citra

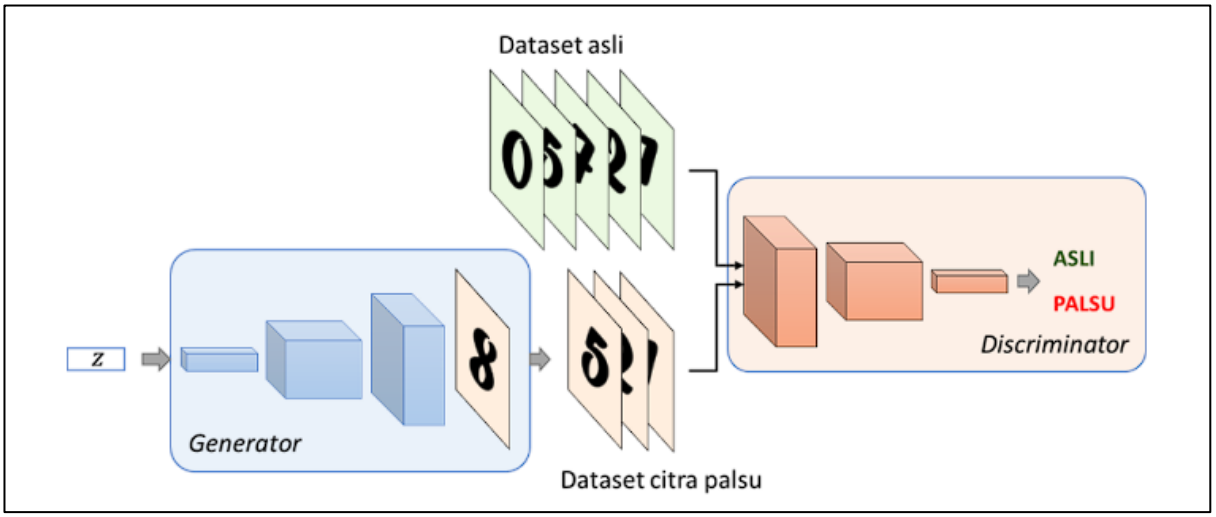
Pengolahan citra merupakan Teknik untuk memodifikasi atau menginterpretasi gambar yang ada seperti foto dan rangkaian gambar film yang bertujuan untuk meningkatkan kualitas gambar [12], atau memperbaiki gambar yang rusak[13].

2.5 Pengenalan Pola

Pengenalan pola salah satu cabang ilmu Kecerdasan Buatan, yang mengelompokan data numerik maupun simbolik (termasuk citra). Tujuan pengelompokkan adalah untuk mengenali obyek yang dalam suatu data berdasarkan ciri-ciri yang dimiliki obyek/pola tersebut. Sebulah pola adalah entitas yang terdefinisikan melalui ciri-ciri (features). Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola lainnya[14].

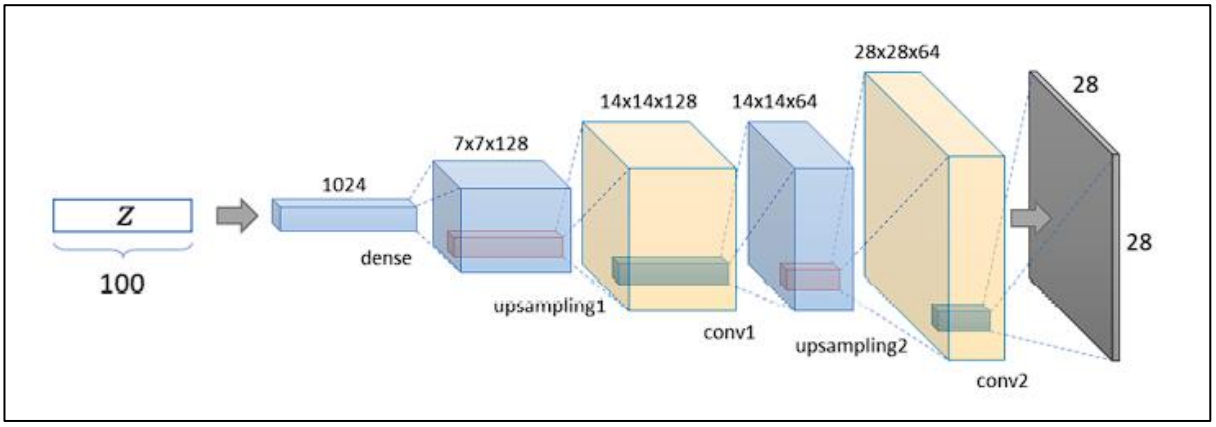
2.6 Generative Adversarial Network (GAN)

Generative Adversarial Network adalah keramgka kerja untuk model generatif yang memanfaatkan peraduan (adversary) antara dua jenis model. Model terdiri dari model generative (G) yang menangkap distribusi data, dan sebuah model diskriminatif (D) yang memperkirakan probabilitas sample yang datang berasal dari data training bukan dari model generatif (G)[2].



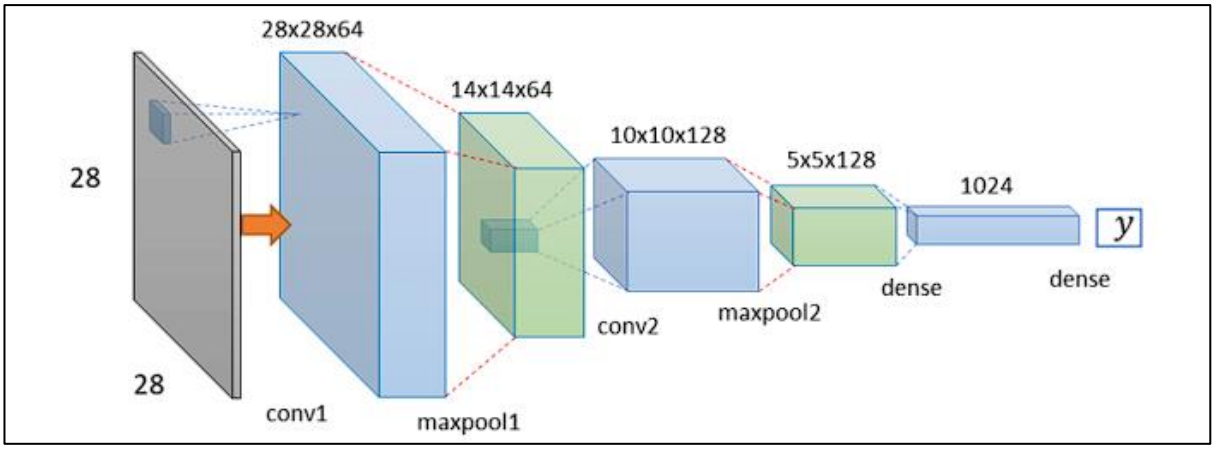
Gambar 2-1 Arsitektur GAN

Pada awal proses pelatihan, generator mendapatkan masukan berupa array yang dibangkitkan secara acak dengan distribusi tertentu yang disebut sebagai latent space (z), yang kemudian akan diubah menjadi sebuah citra melalui proses convolution.



Gambar 2-2 Arsitektur Generator

Pada proses berikutnya, discriminator akan menerima masukan yang dihasilkan oleh generator, kemudian akan menghasilkan keluaran berupa nilai biner yang menunjukan apakah citra masukan ada pada dataset asli (1) atau merupakan citra palsu (0).

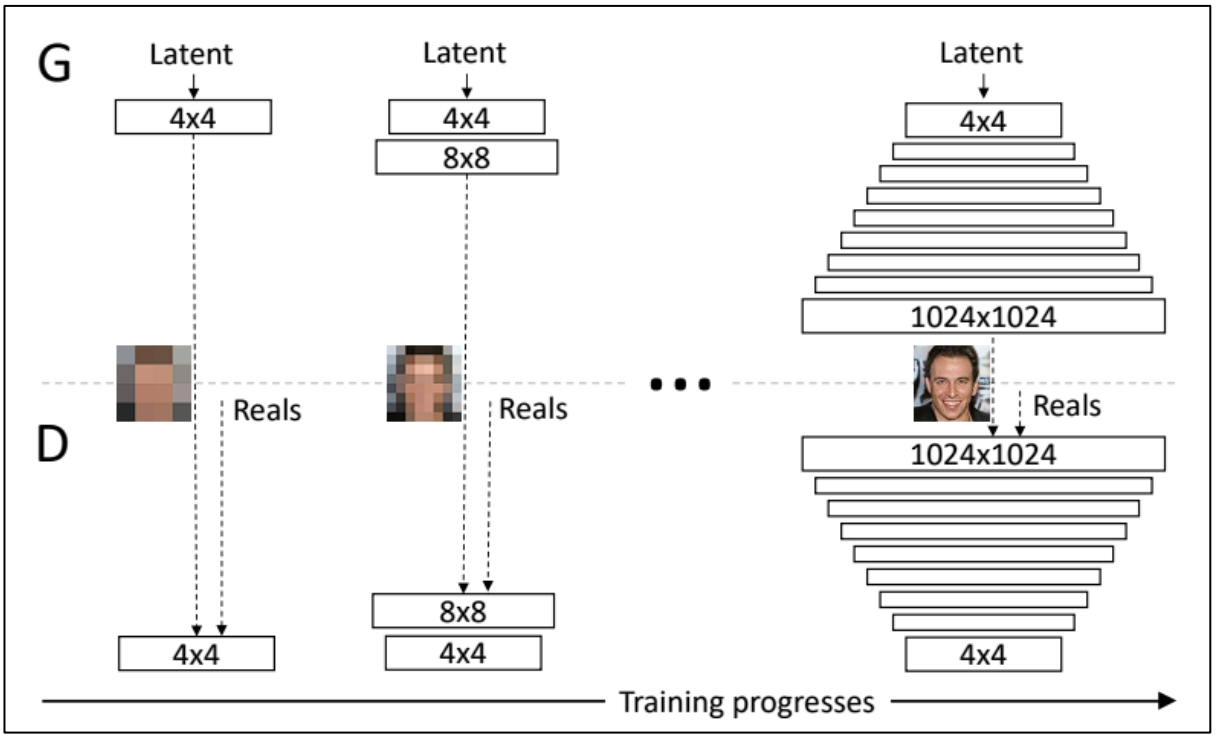


Gambar 2-3 Arsitektur Diskriminator

2.7 Progressive Growing GAN (ProGAN)

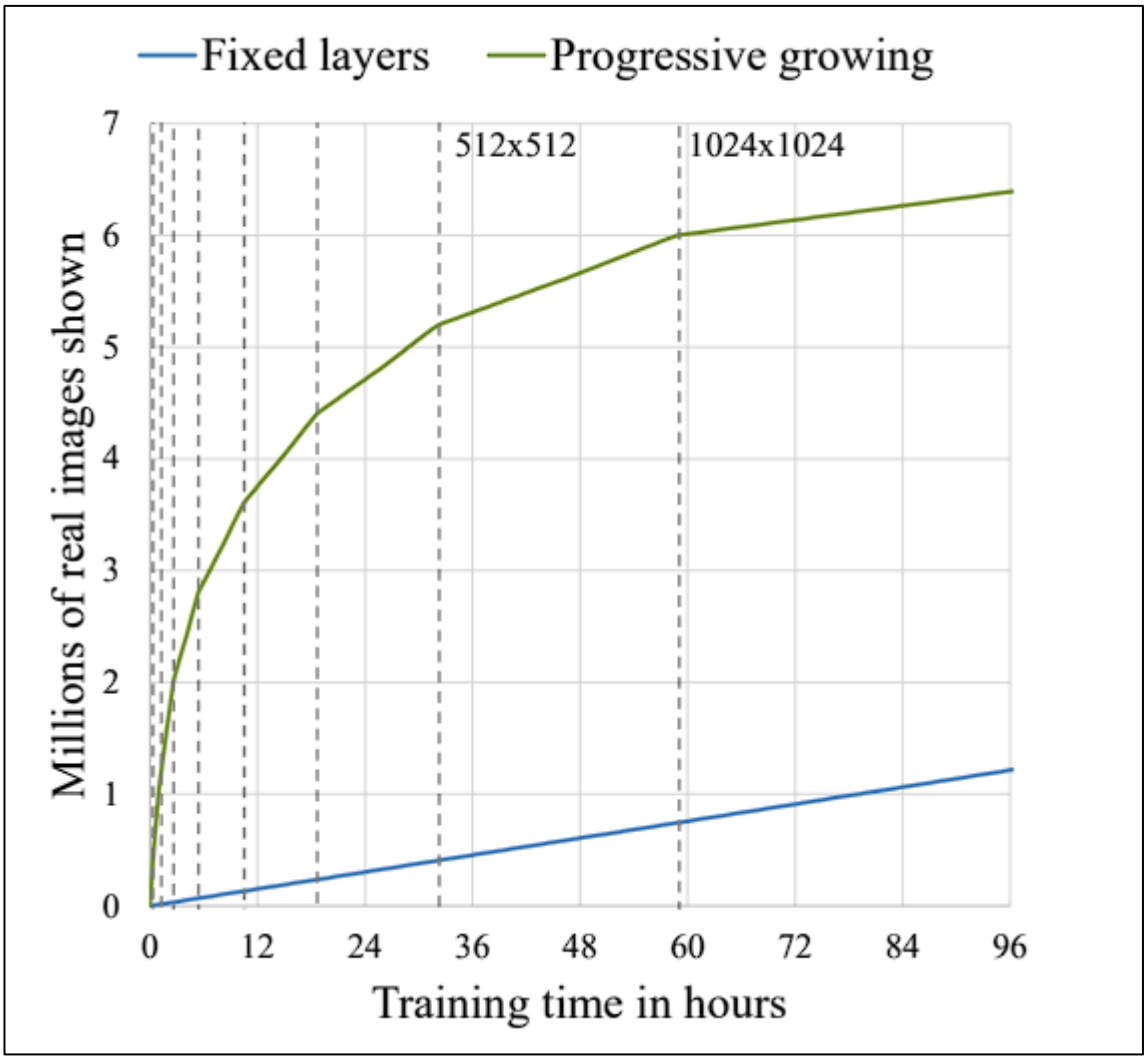
Pada perkembangannya GAN mengalami banyak perubahan, salah satunya adalah dari proses pelatihannya. Salah satu kerangka kerja pelatihan yang dapat digunakan adalah Progressive Growing atau sering disebut ProGAN. Metode ini memungkinkan model GAN untuk menghasilkan citra yang berresolusi tinggi dan bervariasi, serta memiliki kualitas yang baik[15].

Proses pelatihan ProGAN dimulai dengan generator yang menghasilkan gambar dengan resolusi rendah terlebih dahulu, dan kemudian seiring berjalannya proses training ditambahkan layer baru untuk menambah resolusi sedikit demi sedikit.



Gambar 2-4 Proses Training ProGAN

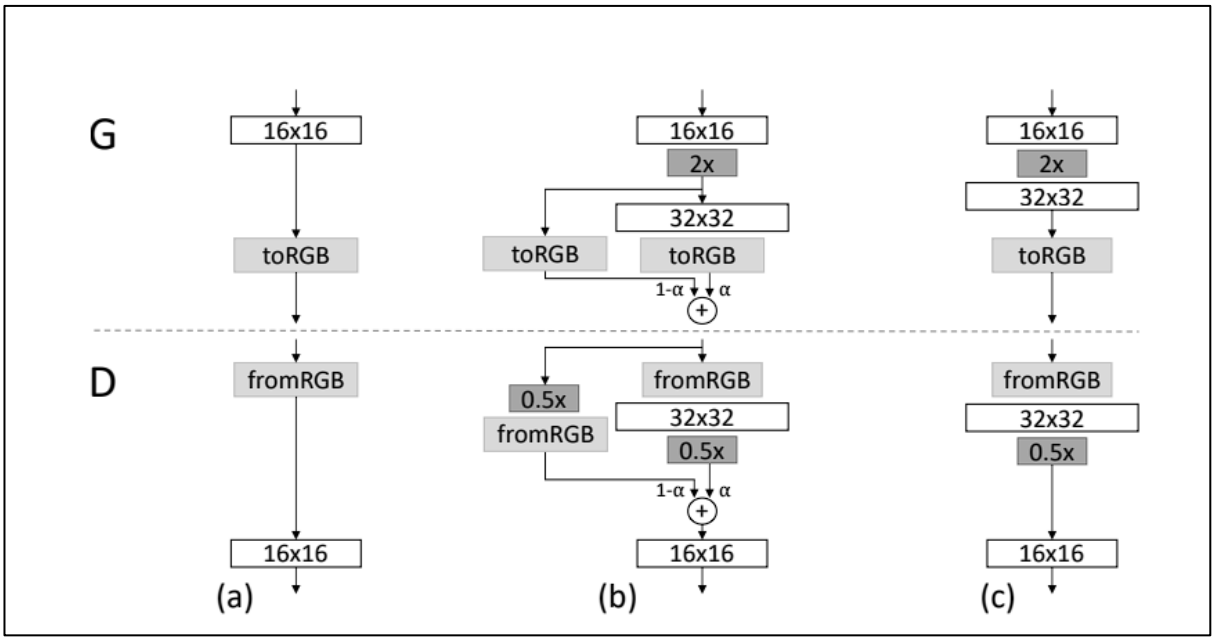
Pada Gambar 2-4, dapat dilihat bahwa pada awal pelatihan pada generator dan diskriminator dimulai dari resolusi 4x4 hingga mencapai epoch tertentu, kemudian arsitektur kedua model diperbaharui dengan menambahkan layer convolution dan upscaling/downscale baru. Dengan menggunakan metode pelatihan seperti ini, dapat meningkatkan kecepatan pelatihan yang cukup signifikan[15].



Gambar 2-5 Pengaruh Progressive Growing Terhadap Waktu Training

2.7.1 Fade In/Out Scaling

Pada saat menurunkan atau menaikan resolusi citra pada proses training ProGAN, proses tidak dilakukan secara langsung menggunakan algoritma yang ada tanpa adanya trainable parameter. Namun dilakukan interpolasi secara perlahan melalui parameter α, interpolasi antara upscale/downscale sederhana menggunakan algoritma nearest neighbor dan average pooling saja dengan bobot $(1-\alpha)$, lalu dijumlahkan dengan hasil upscale/downscale yang sama namun dijadikan masukan pada convolution layer dengan resolusi berikutnya dengan bobot α. Kedua hasil upscale/downscale dimasukan kedalam layer convolution dengan kernel 1x1 untuk melakukan proyeksi ke/dari warna RGB.



Gambar 2-6 Proses Upsampling dan Downsampling Seiring Berjalannya Pelatihan Model

Nilai bobot α dimulai dengan nilai yang sangat kecil dan bertambah seiring berjalannya proses training, dapat dilihat pada Gambar 2-6, di bagian (a), adalah awal dari training, di bagian (b) adalah tahap dimana transisi antar resolusi dimulai, dan bagian (c) adalah tahap akhir training dimana $\alpha = 1$ dan bobot dari citra upscaled/downscaled melalui algoritma nearest neighbor saja menjadi sama dengan 0 dan tidak berpengaruh lagi pada proses upscale/downscale.

2.7.2 Minibatch Standard Deviation

Model GAN pada umumnya memiliki kecenderungan untuk menghasilkan sedikit variasi daripada data yang ada pada data training, oleh karena itu Tero Karras dkk menemukan bahwa dengan menghitung simpangan baku pada setiap fitur yang ada di minibatch kemudian mengambil nilai rata-ratanya lalu menggabungkannya dengan feature map yang ada pada diskriminator. Dengan ditambahkannya layer ini, diskriminator akan mendapatkan informasi terkait variasi yang ada pada minibatch, sehingga model dapat menghasilkan citra yang lebih bervariasi[15].

2.8 Style-based Generative Adversarial Networks (StyleGAN)

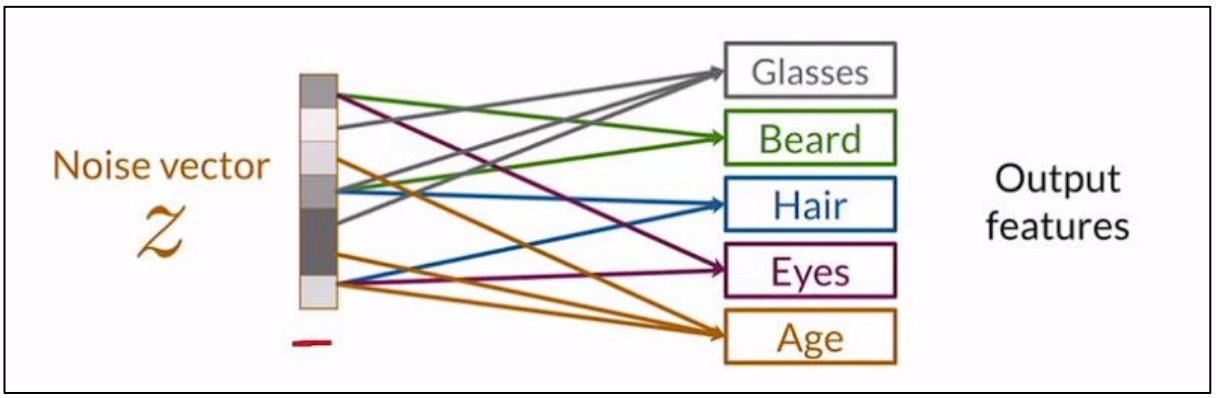
Pada penelitian terbaru terkait GAN yang dilakukan Tero Karras dkk, yang terinspirasi dari penelitian lain terkait style transfer, penelitian ini terfokus pada merancang arsitektur generator baru yang mampu menghasilkan citra wajah manusia yang sangat realistis dan juga berresolusi tinggi. Arsitektur ini dibangun berdasarkan arsitektur ProGAN dengan beberapa perubahan yang dilakukan pada generator [5].

2.8.1 Mapping Network

Mapping Network adalah salah satu komponen utama dari StyleGAN, ini digunakan untuk sebagai metode untuk menambahkan input noise pada generator, yang akan nantinya akan membantu untuk mengontrol style dari citra yang dihasilkan oleh generator.

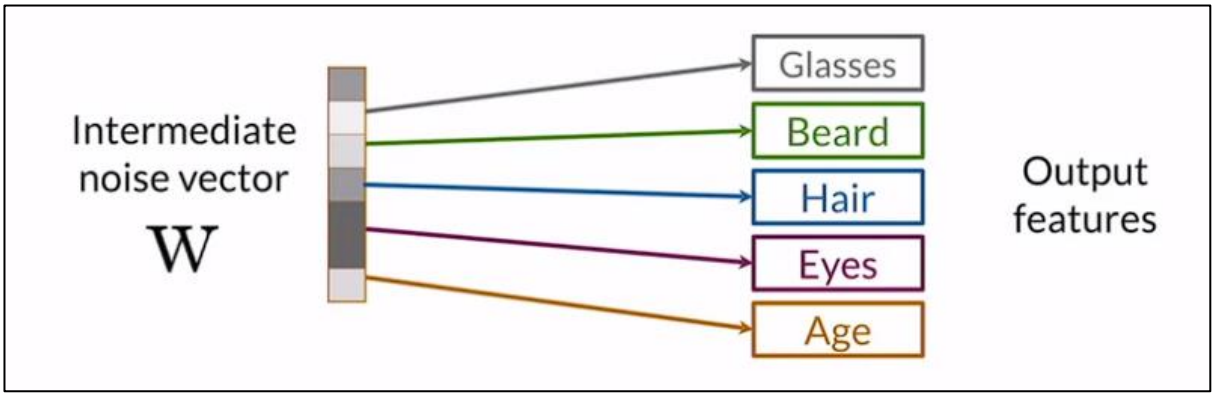
Mapping Network akan mengambil latent code Z sebagai input, kemudian akan menghasilkan intermediate latent code W sebagai output. Network ini terdiri dari multilayer perceptron dengan delapan layer. Network ini digunakan untuk mengurangi Entanglement yang terjadi pada model GAN.

Entanglement merupakan keadaan dimana satu elemen vektor pada latent code Z tidak terproyeksi pada satu fitur citra saja, namun pada banyak fitur. Karena keadaan inilah proses sintesis dalam model GAN umumnya tidak dapat dikontrol dan menghasilkan citra sesuai dengan keinginan[16]. Gambar 2-7 menjelaskan bagaimana entanglement mempengaruhi output fitur pada GAN.



Gambar 2-7 GAN Ketika Entanglement Terjadi

Entanglement sering terjadi karena data citra asli memiliki kepadatan distribusi tertentu, yang sulit untuk dipetakan ke Latent Z yang memiliki distribusi normal[5]. Namun dengan menambahkan Mapping Network, ini akan membantu Latent Code Z yang berdistribusi normal untuk belajar dan memiliki distribusi yang mirip dengan data citra yang telah dipelajari. Gambar 2-8 mengilustrasikan model GAN saat entanglement tidak terjadi.



Gambar 2-8 GAN Tanpa Entanglement

2.8.2 Adaptive Instance Normalization (AdaIN)

Layer Normalization banyak digunakan pada model Neural Network untuk mengstabilkan proses training[17]. Instance Normalization (IN) selain digunakan untuk menstabilkan training, ditemukan bahwa IN mengandung informasi terkait style yang didapat dari rata-rata fitur dan variansnya oleh karena itu IN banyak digunakan untuk style transfer [18].

$$IN(x) = \left(\frac{x - \mu(x)}{\sigma(x)}\right)$$

Persamaan 2-1 Instance Normalization

x = *input* fitur

$\mu(x)$ = Rata-rata dari input fitur

$\sigma(x)$ = Simpangan baku dari input fitur

AdaIN merupakan perkembangan dari IN, dimana selain menormalisasi input, AdaIN juga melakukan penyesuaian rata-rata dan simpangan baku dari *input* yang berupa konten dari suatu citra dengan *style* [19].

$$AdaIN(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

Persamaan 2-2 Adaptive Instance Normalization

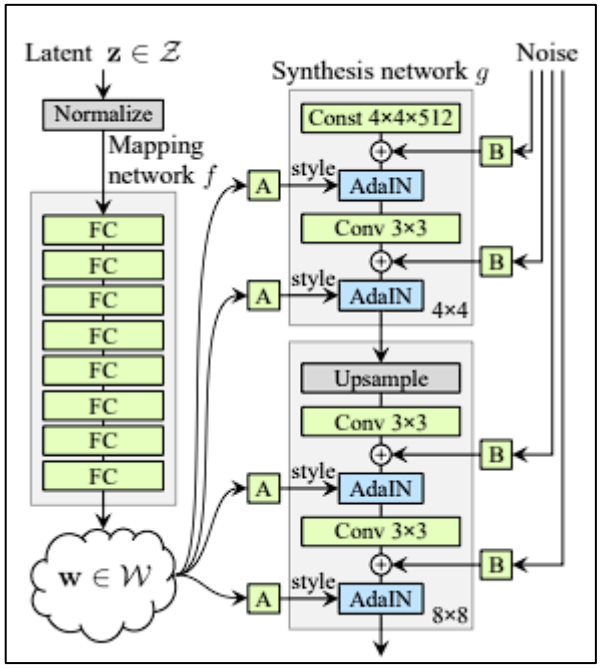
x = *input* fitur

$\mu(x)$ = Rata-rata dari input fitur

$\sigma(x)$ = Simpangan baku dari input fitur

$\mu(y)$ = Rata-rata dari style fitur

$\sigma(y)$ = Simpangan baku dari style fitur



Gambar 2-9 Arsitektur StyleGAN

Pada Arsitektur StyleGAN, Latent Code W hasil keluaran dari Mapping Network dijadikan sebagai style input y, dan fitur yang didapat dari hasil convolution dijadikan sebagai input pada AdaIN.

2.9 Activation Function

Fungsi aktivasi membentuk output dari sebuah jaringan saraf buatan, oleh karena itu fungsi ini adalah bagian yang tak dapat dipisahkan dari sistem jaringan saraf buatan, khususnya pada deep learning. Fungsi aktivasi mempengaruhi kemampuan sebuah jaringan saraf buatan dalam memperkirakan fungsi tujuan, oleh karena itu fungsi ini harus merupakan fungsi yang bersifat non-linear dan juga fungsi yang dapat diturunkan. Fungsi yang bersifat linear hanya dapat menghasilkan output yang juga linear, dan penggunaan fungsi yang dapat diturunkan akan memudahkan perhitungan gradien yang menjadi bagian penting dalam proses optimasi sistem [20].

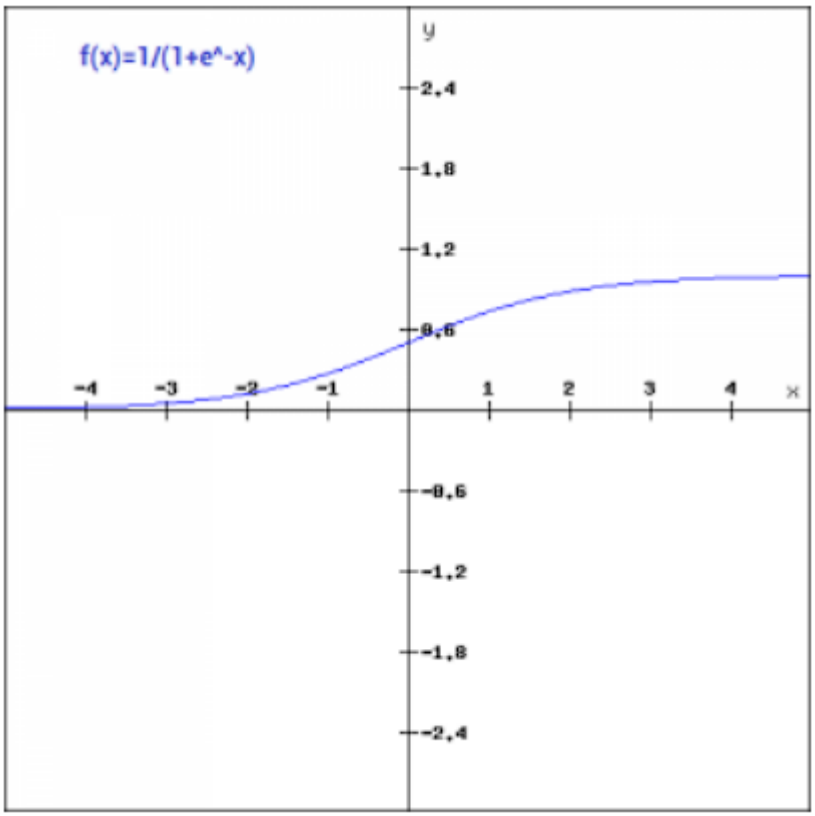
2.9.1 Sigmoid

Sigmoid merupakan fungsi aktivasi yang mengubah nilai dengan rentang 0 hingga 1. Selain Sigmoid, fungsi aktivasi Softmax juga menghasilkan output dengan nilai rentang 0 hingga 1 akan tetapi perbedaannya ada pada jumlah kelas yang dapat diterapkan, softmax dapat menerima n jumlah kelas (multi-class) sedangkan Sigmoid hanya dapat menerima 2 buah kelas[20]. Oleh karena itu fungsi ini cocok digunakan pada diskriminator yang hanya akan melakukan klasifikasi biner, fungsi Sigmoid didefinisikan dengan Persamaan 2-3.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Persamaan 2-3 Sigmoid

Gambar 2-10 menjuelaskan hasil plot grafik pada fungsi Sigmoid



Gambar 2-10 Grafik Sigmoid

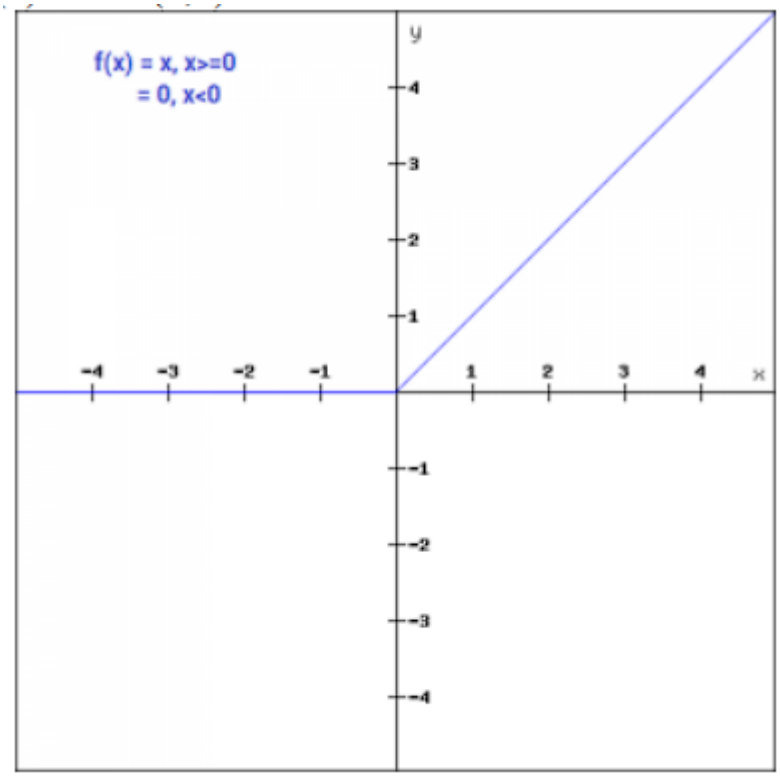
2.9.2 ReLU

ReLU merupakan singkatan dari rectified linear unit yang merupakan fungsi yang banyak digunakan pada neural network. Kelebihan dari fungsi ReLU adalah tidak semua neuron dalam neural network diaktifkan secara bersamaan, oleh karena itu fungsi ReLU lebih efisien daripada fungsi lain[20]. Secara matematis, fungsi ReLU didefinisikan dengan Persamaan 2-4.

$$f(x) = \max(0, x)$$

Persamaan 2-4 ReLU

Gambar 2-11 menjelaskan hasil plot grafik dari fungsi ReLU.



Gambar 2-11 Grafik ReLU

2.9.3 LeakyReLU

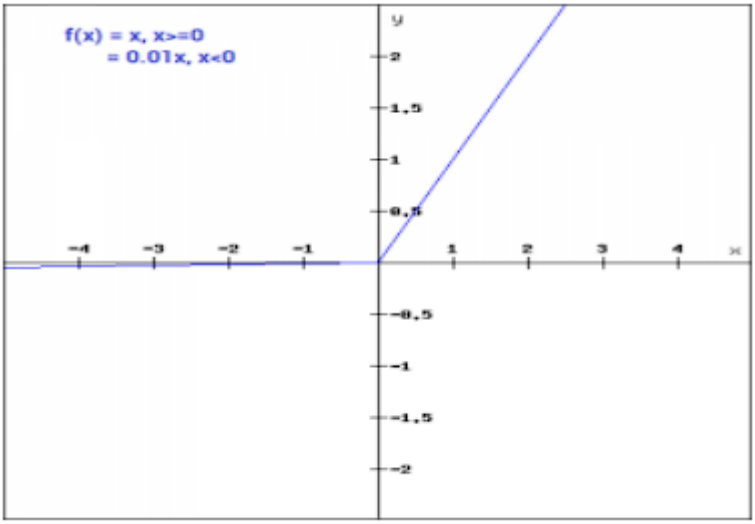
LeakyReLU adalah pembaharuan dari fungsi aktivasi ReLU dimana untuk nilai negatif x, hasil fungsinya didefinisikan sebagai komponen linear x yang sangat kecil[20]. Secara matematis, fungsi LeakyReLU didefinisikan dengan Persamaan 2-5.

$$f(x) = x, \quad x \geq 0$$

$$f(x) = 0.01x, \quad x < 0$$

Persamaan 2-5 Leaky ReLU

Gambar *2-12* menjelaskan hasil plot grafik dari fungsi LeakyReLU.



Gambar 2-12 Grafik LeakyReLU

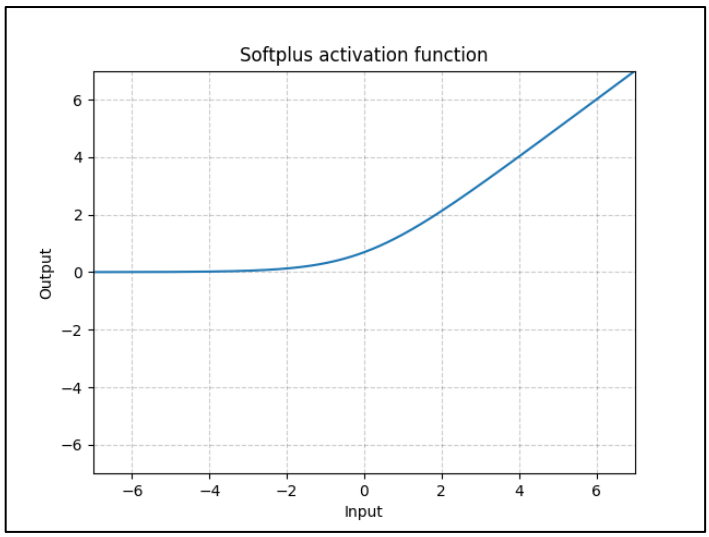
2.9.4 Softplus

Softplus adalah pendekatan halus dari ReLU, fungsi ini dapat digunakan untuk membatasi keluaran agar selalu bernilai positif. Secara matematik Softplus didefinisikan pada Persamaan 2-6.

$$f(x) = \log(1 + e^x)$$

Persamaan 2-6 Persamaan Softplus

Gambar *2-13* menjelaskan hasil plot grafik dari fungsi Softplus.



Gambar 2-13 Grafik Softplus

2.10 Loss Function

Loss Function berfungsi untuk menghitung margin error dari output yang dihasilkan dengan tujuan yang sudah ditentukan.

2.10.1 BCE Loss

BCE atau Binary Cross Entropy merupakan loss function yang umum digunakan untuk model klasifikasi yang memiliki dua kategori. BCE merupakan turunan dari Cross Entropy dimana CE dapat menerima banyak kelas c. Cross Entropy sendiri didefinisikan sebagai ukuran perbedaan suatu distribusi probabilitas untuk variable acak yang diberikan atau pada serangkaian peristiwa[21]. Secara matematis, Cross Entropy didefinisikan pada Persamaan 2-7.

$$CE = -\sum_{i=1}^{C} y_i \log p_i$$

Persamaan 2-7 Cross Entropy

y = label kelas

p = nilai hasil prediksi

c = banyak kelas

Namun jika banyak kelas hanya terdapat dua, maka *Binary Cross Entropy* dapat didefinisikan menjadi sebagai berikut:

$$BCE = -(\mathrm{ylog}(p) + (1-y)\log(p))$$

Persamaan 2-8 Binary Cross Entropy

y = label kelas

p = nilai hasil prediksi

2.10.2 Wasserstein Distance

Pada awal dikemukakannya arsitektur GAN, loss function yang digunakan adalah BCE Loss, namun seiring berkembangnya arsitektur tersebut ditemukan bahwa penggunaan BCE Loss dapat menyebabkan masalah seperti vanishing

gradient yang dapat menyebabkan mode collapse dalam proses training-nya. Wasserstein Distance menghitung jarak dari distribusi data asli dengan distribusi data yang dibangkitkan oleh generator. Dengan menggunakan Wasserstein Distance, discriminator tidak lagi menggunakan Sigmoid untuk activation function pada outputnya, namun menggunakan linear. Output ini dapat diinterpretasikan sebagai nilai keaslian dari citra [22].

Namun untuk menggunakan loss function ini, ada syarat yang harus dipenuhi oleh diskriminator, yaitu diskriminator harus 1-Lipschitz Continuous. 1-Lipschitz Continuous berarti gradien yang dimiliki oleh diskriminator harus < 1. Kondisi ini sangat penting untuk memastikan bahwa Wasserstein Loss tidak hanya fungsi kontinu, dan memiliki turunan, namun juga memastikan bahwa gradien tidak tumbuh terlalu tinggi yang dapat mengakibatkan ketidakstabilan pada proses training [22].

$$\mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$$

Persamaan 2-9 Wasserstein Loss

2.11 Adam Optimizer

Optimizer adalah sebuah fungsi atau algoritma yang digunakan untuk memperbaharui atribut dari neural network, seperti weight dan bias secara iteratif yang didasarkan pada data training. Adam merupakan salah satu optimizer yang banyak digunakan. Diperkenalkan oleh Diedrik dkk pada tahun 2015 [23]. Adam merupakan kombinasi dari dua optimizer lain yaitu RMSProp dan Stocastic Gradient Descent dengan Momentum.

Secara matematis Adam didefinisikan pada persamaan

$$\theta_{t+1} = \theta_t - \frac{\alpha * \widehat{m_t}}{\sqrt{\hat{v}} + \varepsilon}$$

Persamaan 2-10 Adam Optimizer

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

$$m_t = (1-\beta_1)g_t + \beta_1 * m_{t-1}$$

$$v_t = (1-\beta_2)g_t^2 + \beta_2 * v_{t-1}$$

2.12 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.

Secara matematis FID dideskripsikan pada **Persamaan *2-11***.

$$d^2((m,C),(m_w,C_w)) = \left|\left|m - m_2\right|\right|_2^2 + Tr(C + C_w - 2(CC_w)^{\frac{1}{2}})$$

Persamaan 2-11 Persamaan Frechet Inception Distance

$m =$ rata-rata pada citra asli

$C =$ matriks kovarian pada citra asli

$m_w =$ rata-rata pada citra yang telah dibangkitkan

$C_w =$ matriks kovarian pada citra yang telah dibangkitkan

2.13 Python

Python adalah Bahasa pemrograman interpretatif multiguna, Tidak seperti Bahasa lain yang sulit untuk dibaca dan dipahami, Python lebih menekankan pada keterbacaan kode agar lebih mudah dipahami, khususnya oleh orang yang tidak familiar pada sintaks-sintaks bahasa pemrograman yang lebih kompleks.

Bahasa ini pertama kali dikembangkan pada tahun 1991, oleh seorang Bernama Guido van Rossum. Sampai saat ini bahasa Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hamper semua sistem operasi, dan bahkan untuk sistem operasi linux hamper semua distronya sudah menyertakan Python terinstall di dalam sistemnya.

2.14 Pytorch

Pytorch adalah framework komputasi ilmuah berbasis Python yang ditargetkan untuk menggantikan NumPy agar dapat menggunakan kekuatan GPU untuk mendapatkan kemampuan komputasi parallel yang besar, oleh karena itu framework ini cocok digunakan untuk penelitian yang membutuhkan fleksibilitas dan kecepatan yang maksimum.

2.14.1 Tensor

Tensor adalah sebuat struktur data array pada Pytorch, secara konseptual tensor identic dengan array NumPy; Tensor adalah array dengan n-dimensi, dan Pytorch menyediakan banyak fungsi utilitas yang dapat digunakan untuk memanipulasi data yang ada pada Tensor ini.

2.14.2 AutoGrad

Paket autograd di Pytorch menyediakan fungsi mengotomatiskan perhitungan mencari gradien untuk proses backpropagation dalam sistem jaringan saraf buatan. Saat menggunakan autograd, forward pass pada jaringan model akan membuat computational graph; node dalam graph tersebut akan menjadi Tensor, dan edge akan menjadi fungsi yang menghasilkan Tensor input dan output [24]. Dengan melakukan backpropagation melalui graph maka akan memudahkan proses perhitungan gradien [25].