Kecerdasan Buatan

KELOMPOK 2

IMPLEMENTASI GAN: MENGHASILKAN GAMBAR ANGKA TULISAN TANGAN

Anggota Kelompok:

- 2303010115, Adi Rosyadi, Universitas Perjuangan Tasikmalaya
- 2303010126, Juan Pernando A, Universitas Perjuangan Tasikmalaya
- 23.11.5668, Raditya M. W. H, Universitas Amikom Yogyakarta
- 23.11.5646, Rifqi Aryo Mulyadi, Universitas Amikom Yogyakarta

Struktur Pembahasan

Mengenal mekanisme kerja GAN pada dataset mnist.	01
Penjelasan kode program dan konfigurasi parameter model.	02
Mengamati perkembangan model dari noise hingga membentuk angka.	03
Penilaian keberhasilan model	04

Konsep Dasar: Mekanisme Kerja Generative Adversarial Network (GAN)

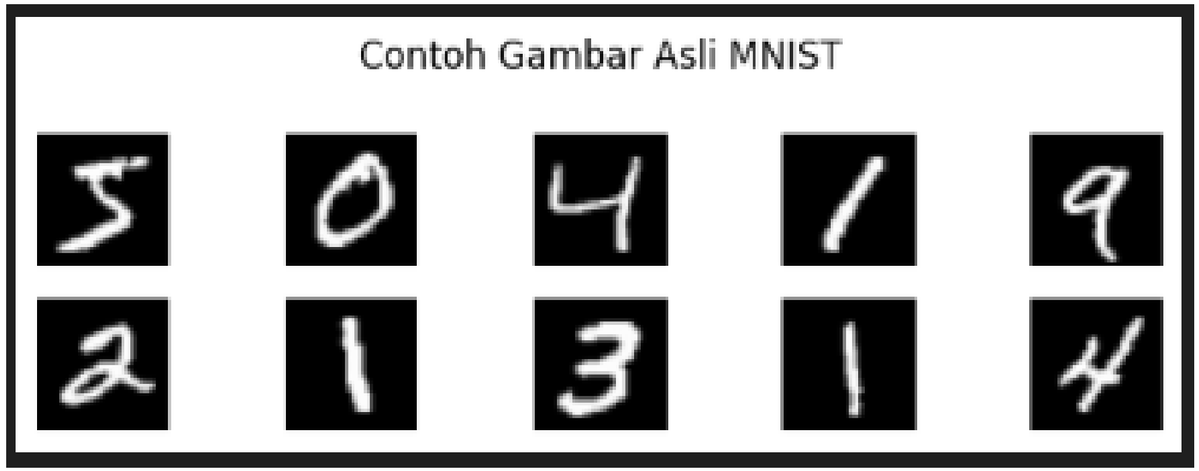
GAN bekerja melalui interaksi dua aktor utama yang memiliki tujuan berlawanan:

- Generator: Bertugas mengubah random noise menjadi gambar angka yang meyakinkan (seperti pemalsu lukisan).
- Discriminator: Bertugas membedakan antara data asli dan data buatan Generator (seperti detektif seni).

Dataset MNIST digunakan untuk melatih (Discriminator) agar tahu seperti apa angka yang asli.

- Cara Memuat Data: Di dalam kode, kita menggunakan fungsi:
  (train_images, _), (_, _) = tf.keras.datasets.mnist.load_data()
- Sebelum masuk ke jaringan saraf, data asli harus "diselaraskan" dengan cara:
  Normalisasi, mengubah nilai piksel (0-255) menjadi rentang -1 hingga 1.
- Tujuannya: Agar seimbang dengan output dari Generator yang menggunakan fungsi aktivasi tanh. Jika rentang datanya berbeda, Discriminator akan terlalu mudah menebak mana yang palsu, dan proses belajar akan gagal.

Konsep Dasar: Mekanisme Kerja Generative Adversarial Network (GAN)



Visualisasi Dataset MNIST

- Karakteristik Data: Terdiri dari 60.000 data latih berupa citra angka tulis tangan digit 0 sampai 9.
- Dimensi Citra: Setiap gambar memiliki ukuran 28 x 28 piksel dengan format grayscale (hitam putih).
- Tantangan Model: Keberagaman gaya penulisan tangan dari berbagai orang menjadi tantangan utama bagi model untuk mempelajari pola visual yang akurat.

Arsitektur Model: Jaringan dan Konfigurasi Pelatihan

1.Sisi Generator:

```
def make_generator():
    model = tf.keras.Sequential(name="generator")
    model.add(layers.Input(shape=(NOISE_DIM,)))

    model.add(layers.Dense(7*7*256, use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(0.2))

    model.add(layers.Reshape((7, 7, 256)))

    model.add(layers.Conv2DTranspose(128, 5, strides=1, padding="same", use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(0.2))

    model.add(layers.Conv2DTranspose(64, 5, strides=2, padding="same", use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(0.2))

    model.add(layers.Conv2DTranspose(1, 5, strides=2, padding="same",
                                     use_bias=False, activation="tanh"))
    return model
```

Arsitektur Model: Jaringan dan Konfigurasi Pelatihan

- Generator bekerja seperti "Lensa Pembesar". Ia menerima input berupa vektor angka acak (100 dimensi) yang tidak bermakna, lalu memperluasnya secara bertahap.
- Proses Layer:
  - Menerima 100 titik data.
  - Diproses melalui lapisan saraf (Dense layers) untuk mengekstraksi pola awal.
  - Diubah bentuknya (Reshape) menjadi struktur 2D.
  - Output Akhir: Sebuah matriks 28 x28 piksel yang menyerupai angka.
- Fungsi Aktivasi Tanh: Digunakan di akhir untuk memastikan warna piksel berada di rentang yang stabil (hitam ke putih).

Arsitektur Model: Jaringan dan Konfigurasi Pelatihan

1.Sisi Discriminator:

```
def make_discriminator():
    model = tf.keras.Sequential(name="discriminator")
    model.add(layers.Input(shape=(28, 28, 1)))

    model.add(layers.Conv2D(64, 5, strides=2, padding="same"))
    model.add(layers.LeakyReLU(0.2))
    model.add(layers.Dropout(0.2))

    model.add(layers.Conv2D(128, 5, strides=2, padding="same"))
    model.add(layers.LeakyReLU(0.2))
    model.add(layers.Dropout(0.2))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))
    return model
```

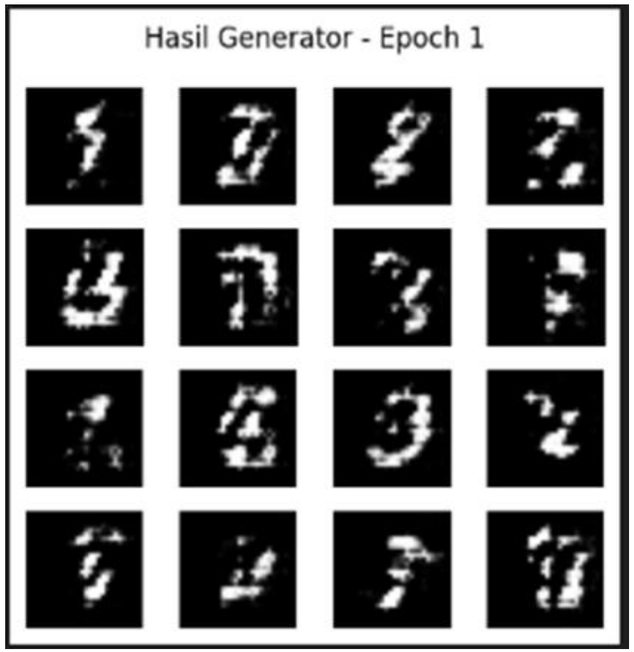
Arsitektur Model: Jaringan dan Konfigurasi Pelatihan

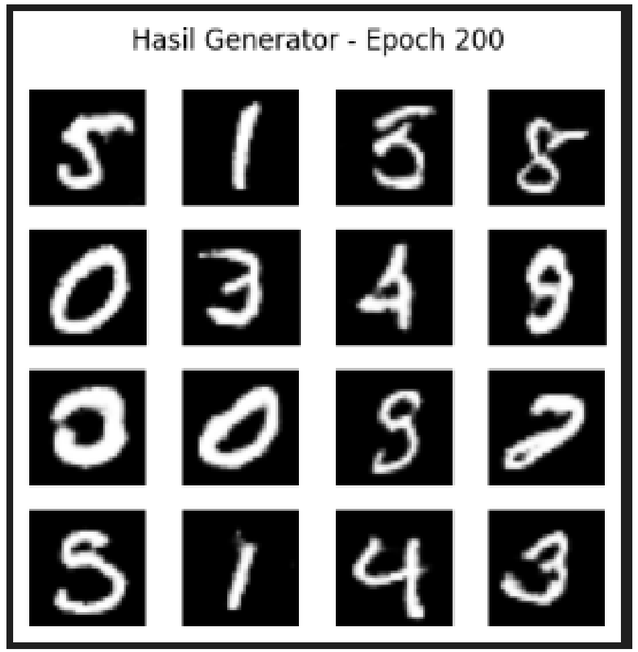
- Berbeda dengan Generator, Discriminator bekerja seperti "Filter Penyaring". Ia menerima gambar lengkap, lalu mengecilkan informasinya untuk mengambil keputusan.
- Proses Layer:
- Menerima gambar 28 x 28 (baik dari dataset asli maupun dari Generator).
- Merapratkan gambar menjadi satu baris data (Flattening).
- Melewati lapisan saraf untuk mencari kejanggalan atau ciri khas angka yang asli.
- Output Akhir: Satu angka probabilitas antara 0 (Palsu) hingga 1 (Asli).

Dalam kedua model ini, kita menggunakan LeakyReLU sebagai "katup pengaman". Fungsinya adalah membiarkan sedikit informasi tetap mengalir meskipun nilainya negatif, sehingga saraf model tidak "mati" (dead neuron) selama proses kompetisi yang keras.

Perkembangan Visual: Bagaimana Model Belajar (Epoch 1 - 1000)

1.Tahap Awal (Epoch 1 - 200): Mencari Bentuk

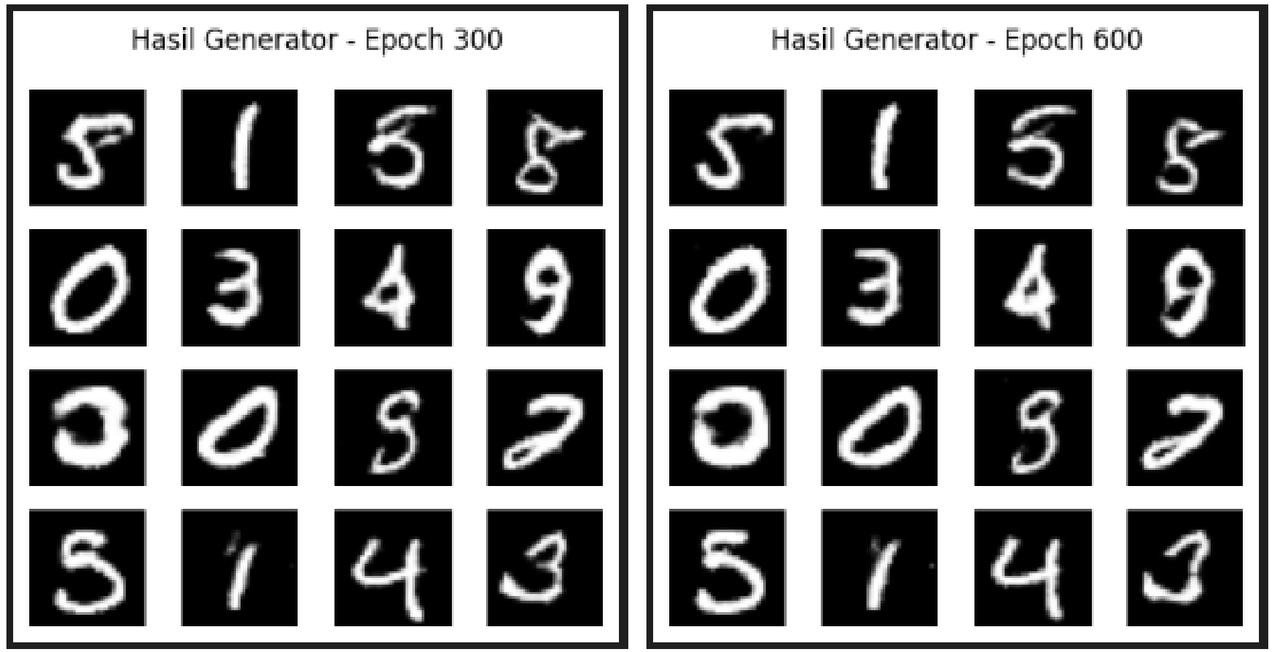




- Kondisi: Gambar bertransisi dari gangguan acak (noise) total menuju pola dasar yang mulai menyerupai struktur angka.
- Penjelasan: Di fase ini, Generator mulai membangun "logika" visualnya. Dari bobot yang awalnya acak pada Epoch 1, model secara bertahap belajar mengenali posisi piksel dan struktur geometri dasar (seperti garis dan lengkungan). Meskipun masih kasar, pada Epoch 200 model sudah mampu membedakan objek angka dari latar belakangnya.

Perkembangan Visual: Bagaimana Model Belajar (Epoch 1 - 1000)

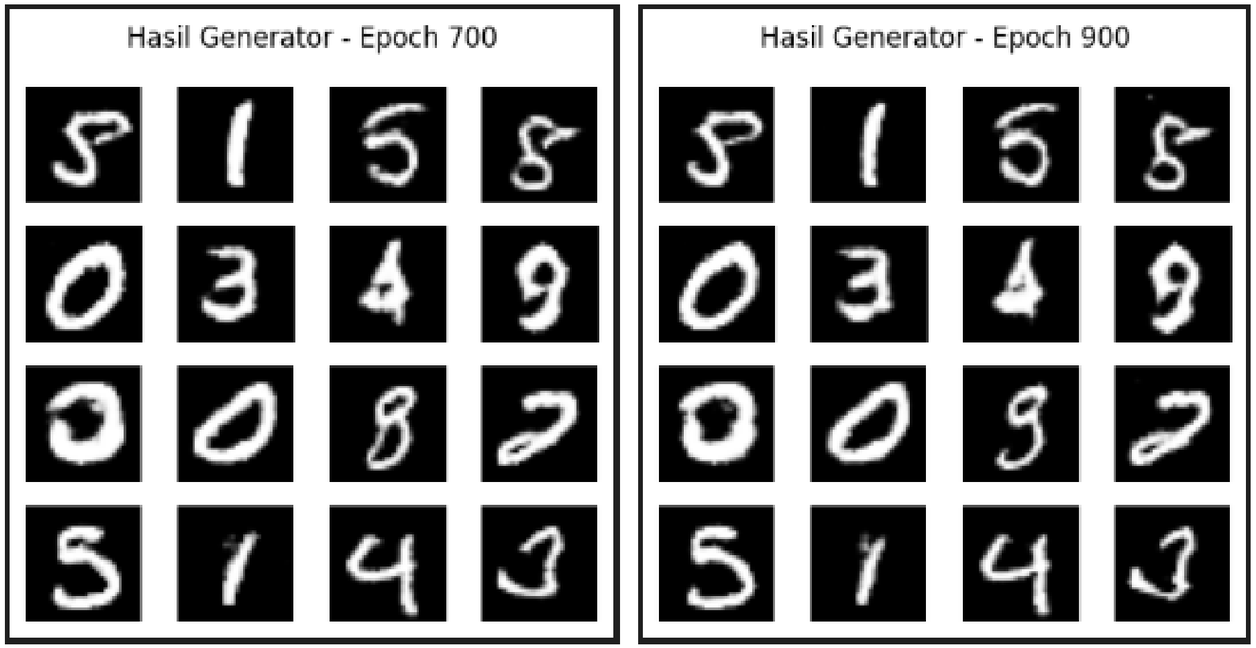
1.Tahap Pembentukan (Epoch 300 - 600): Mulai Mengenal Pola



- Kondisi: Tampilan gambar semakin bersih dari gangguan (noise) dan pola angka terlihat jauh lebih solid serta tegas dibandingkan tahap sebelumnya.
- Penjelasan: Pada fase ini, model sudah melewati tahap pencarian bentuk dasar dan mulai fokus pada konsistensi struktur. Generator kini lebih mahir dalam meniru detail lengkungan dan sudut angka secara spesifik, sehingga variasi antar angka (seperti perbedaan antara '3', '5', dan '8') menjadi lebih jelas dan mudah dibedakan.

Perkembangan Visual: Bagaimana Model Belajar (Epoch 1 - 1000)

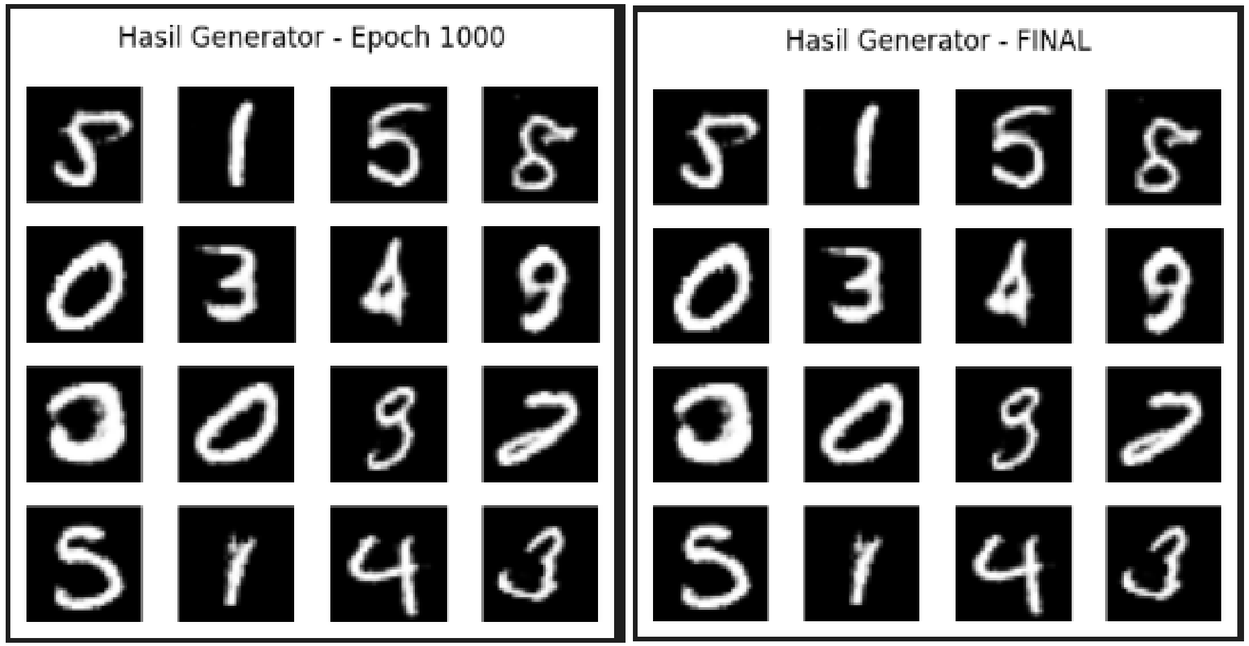
1.Tahap Penajaman (Epoch 700 - 900): Struktur Angka Terlihat



- Kondisi: Angka-angka seperti 1, 8, atau 0 sudah mulai bisa dikenali dengan jelas oleh mata manusia.
- Penjelasan: Gangguan latar belakang semakin hilang karena model sudah memahami "anatomi" angka tulisan tangan. Pada fase ini, pola dasar sudah sangat kuat dan Generator fokus pada pemantapan detail agar bentuk angka terlihat lebih konsisten dan menyatu.

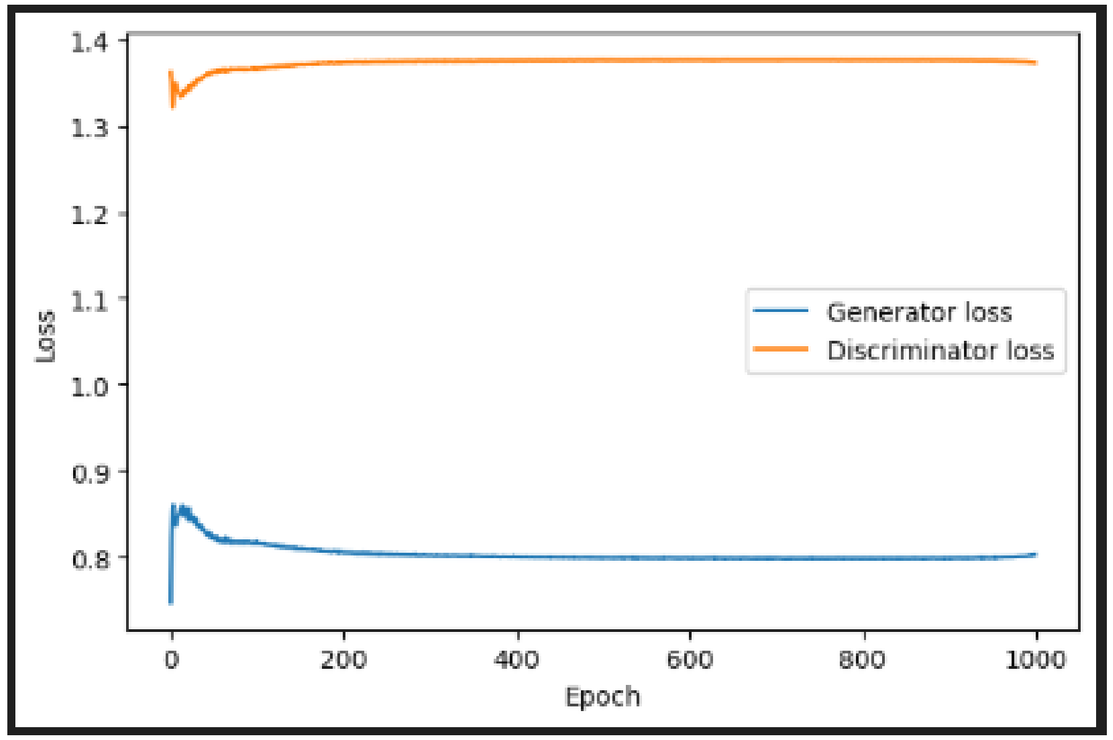
Perkembangan Visual: Bagaimana Model Belajar (Epoch 1 - 1000)

1.Tahap Matang (Epoch 1000): Hasil Optimal



- Kondisi: Angka terlihat halus, solid, dan minim gangguan visual sehingga mudah dibedakan secara jelas.
- Penjelasan: Generator telah sukses menguasai variasi gaya penulisan angka secara natural. Hasil pada Epoch 1000 ini merupakan representasi performa optimal dalam siklus pelatihan saat ini; meskipun detail dapat terus ditingkatkan hingga 6000 epoch, capaian ini sudah sangat efektif membuktikan keberhasilan model dalam menghasilkan data yang realistis.

Kesimpulan: Penilaian Keberhasilan



- Stabilitas Grafik: Grafik Loss yang mendatar menunjukkan tercapainya Nash Equilibrium, menandakan proses pelatihan antara Generator dan Diskriminator berjalan stabil.
- Efektivitas Model: Pada Epoch 1000, model berhasil menghasilkan angka yang realistis dan terbaca jelas, membuktikan efisiensi arsitektur yang digunakan.
- Potensi Pengembangan: Hasil ini merupakan fondasi yang kuat; meskipun detail dapat terus dipertajam hingga 6000 epoch, pencapaian saat ini sudah sangat representatif dalam menunjukkan keberhasilan model.

Kecerdasan Buatan

SEKIAN DAN

TERIMA KASIH !