Laporan Final Project
Kecerdasan Buatan (Lanjut)

Implementasi Generative Adversarial Network (GAN): Menghasilkan Gambar Angka Tulisan Tangan

Kelompok 2

Anggota Kelompok:

1.	2303010115	Adi Rosyadi	Universitas Perjuangan Tasikmalaya
2.	2303010126	Juan Pernando A.	Universitas Perjuangan Tasikmalaya
3.	23.11.5668	Raditya M. W. H	Universitas Amikom Yogyakarta
4.	23.11.5646	Rifqi Aryo Mulyadi	Universitas Amikom Yogyakarta

1. Latar Belakang

Perkembangan Kecerdasan Buatan saat ini sangat pesan termasuk pada pengolahan citra.*Deep Learning a*dalah salah satu yang menjadi fondasi digital dalam memecahkan masalah terkait kecerdasan buatan. Penerapan ini sudah sangat luas mulai dari data sintesis wajah hingga ke tulisan yang menyerupai tangan manusia[1].

Konsep Generative Adversarial Networks (GAN) merupakan Solusi dari deep learning untuk memecahkan masalah tersebut. GAN bekerja dengan dua cara yaitu Generator dan Diskriminator[2].

Pada pembelajaran GAN ini,Generator bekerja sebagai pembuat replka data baru dan Diskriminator yang menilai hasilnya. Melalui pengulangan lebih lanjut Generator terus diuji sampai Diskriminator sulit membedakan yang asli dengan yang palsu.

Generator dan Diskriminator saling berlawan satu sama lain sampai mengahsilkan epoch atau waktu yang terus belurang sampai hasilnya bagus.

Secara sederhananya GAN ini merupakan Generative Model untuk menciptakan data baru, hal ini bermanfaat untuk mengembangkan Kecerdasan Buatan yang semakin pesat.

2. Dataset

Projek ini menerapkan dataset,didalam lingkungan deep learning data mnist diakui sebagai hal yang mendasar, terutama untuk peneliti yang ingin menguji sesuatu algoritma.

Data mnist dibagi jadi dua bagian utama, antara lain:

- **Data Pelatihan:** Berjumlah 60.000 sampel gambar.
- **Data Pengujian:** Berjumlah 10.000 sampel gambar.

Kejelasan Teknis Citra

Setiap entitas gambar dalam dataset ini diatur dalam jenis tertentu.

Dimensi Resolusi: Memiliki ukuran 28 x 28 piksel.

1. **Skala Warna:** Menggunakan format *grayscale* (skala abu-abu).
2. **Kedalaman Piksel:** Nilai intensitas setiap piksel berada pada rentang 0 (hitam) hingga 255 (putih).

Metode Perolehan Data

Data mnist tidak diunduj dari luar, melainkan sudah terintegrasi API dengan suatu data. Berikut adalah potongan kode yang digunakan untuk memuat data tersebut:

```
# LOAD & PREPROCESS MNIST
(train_images, _), (_, _) = tf.keras.datasets.mnist.load_data()
```

```
train_images = train_images.astype("float32")
train_images = (train_images - 127.5) / 127.5   # -> [-1, 1]
train_images = np.expand_dims(train_images, axis=-1)  # (N, 28, 28, 1)
```

Penyesuaian skala piksel ke dalam interval [-1, 1] merupakan langkah penting. yang disamakan dengan arsitektur model GAN. Hal ini dikarenakan lapisan *output* pada komponen **Generator** mengimplementasikan fungsi aktivasi **Tanh**, yang secara matematis memetakan nilai ke dalam rentang [-1, 1].

3. Metode

3.1 Alur Projek

Pelakasanna projek ini dilakukan dengan tahapan yang terintegrasi.Dimulai dari tahap perolehan data hingga ke validasi model,setiap langkahnya di tingkatkan untuk memperkirakan sebuah model. Untuk lebih dalam lagi data ini bekerja meliputi:

Perolehan Data: Tahap awal berupa pengumpulan dan pemuatan dataset MNIST ke dalam lingkungan pemrograman.

1. **Pra-pemrosesan Citra:** Melakukan normalisasi dan penyesuaian dimensi data agar siap diproses oleh algoritma.
2. **Konstruksi Arsitektur Generator:** Merancang sususnan jaringan saraf yang bertanggung jawab menciptakan data sintetis.
3. **Konstruksi Arsitektur Discriminator:** Membangun unit jaringan saraf yang berfungsi sebagai penguji keaslian gambar.
4. **Integrasi Model GAN:** Menggabungkan komponen Generator dan Discriminator ke dalam satu kesatuan sistem pembelajaran saling berlawanan.
5. **Fase Pelatihan:** Menjalankan proses pengulangan pembelajaran untuk melatih kemampuan kedua jaringan.
6. **Interpretasi dan Analisis Output:** Melakukan visualisasi terhadap hasil generasi angka serta mengevaluasi kualitas gambar yang diproduksi.

3.2 Perolehan dan Pra-pemrosesan Data

Setelah menjelaskan karakter dari dataset MNIST pada bagian sebelumnya, langkah selanjutnya adalah menerapkan serangkaian prosedur pra-pemrosesan. Tahapan ini sangat penting untuk mempersiapkan data agar sesuai dengan pembelajaran pada arsitektur GAN.

Proses dimulai dengan memuat data ke dalam memori, yang kemudian dilanjutkan dengan pengiriman skala piksel menjadi rentang [-1, 1].Penerapan kerja dari nornalisasi ini diterapkan dengan kode sebagai berikut:

```
print("min/max:", train_images.min(), train_images.max())
```

Penerapan kode tersebut bermanfaat sebagai prosedur validasi untuk memastikan bahwa intensitas piksel gambar telah terpetakan secara tepat, dalam interval [-1, 1]. Penyamaan ini bersifat wajib karena arsitektur Generator mengadopsi fungsi aktivasi **Tanh**, yang memerlukan input dan menghasilkan output dalam rentang numerik yang identik guna menjaga stabilitas.

Untuk memperbaiki ketetapan perhingan sistem selama dalam pelatihan, data citra yang telah dinormalisasi kemudian dikirim ke dalam objek **tf.data.Dataset** melalui pustaka TensorFlow. Prosedur penyusunan dataset ini mencakup langkah-langkah sebagai berikut:

```
BUFFER_SIZE = train_images.shape[0]
BATCH_SIZE = 128

train_dataset = (
    tf.data.Dataset.from_tensor_slices(train_images)
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE, drop_remainder=True)
    .prefetch(tf.data.AUTOTUNE)
)
```

3.3 Pengaturan Arsitektur Model

3.3.1 Bagian Generator

Generator bertugas untuk mengembangkan data sintesis citra tiruan. Proses ini meniru data mnist dengan acak, yang berperan sebagai keacakan atau ketidakpastian untuk menjamin adanya keragaman pada output gambar angka yang dihasilkan.

Susunan tingkatan yang menyusun bagian Generator ini diterapkan melalui kode berikut:

```
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
```

3.3.2 Bagian Discriminator

Discriminator bekerja sebagai penguji dari hasil Generator, Tugas utamanya adalah melakukan proses untuk menentukan keaslian data.

Pengaturan cara kerja dan susunan terkait Diskriminator, dicantumkan dalam kode sebagai berikut

```
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
```

3.3.3 Penyatuan Sistem dan Algoritma GAN

Setelah tahap perancangan masing-masing bagian selesai. GAN dijelaskan sebagai model generatif yang mengambil prinsip pembelajaran saling bersaing.

.

```python
# LOSS & OPTIMIZER
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def discriminator_loss(real_logits, fake_logits):
    # label smoothing untuk real agar training lebih stabil
    real_labels = tf.ones_like(real_logits) * 0.9
    fake_labels = tf.zeros_like(fake_logits)

    real_loss = cross_entropy(real_labels, real_logits)
    fake_loss = cross_entropy(fake_labels, fake_logits)
    return real_loss + fake_loss

def generator_loss(fake_logits):
    return cross_entropy(tf.ones_like(fake_logits), fake_logits)

gen_optimizer  = tf.keras.optimizers.Adam(learning_rate=2e-4, beta_1=0.5)
disc_optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4, beta_1=0.5)
```

Pada pengerjaan ini, integrasi antara unit Generator dan Discriminator tidak menggunakan pendekatan model sekuensial konvensional. Sebagai gantinya, sistem mengimplementasikan **siklus pelatihan kustom** dengan memanfaatkan fungsionalitas **tf.data.GradientTape**.

3.4 Prosedur Pelatihan Model

```python
# TRAIN LOOP
import time

EPOCHS = 150

gen_history, disc_history = [], []

NOISE_START = 0.10
NOISE_END   = 0.00

for epoch in range(1, EPOCHS + 1):
    start = time.time()

    # linear decay stddev
    t = (epoch - 1) / (EPOCHS - 1)
    noise_std = NOISE_START + t * (NOISE_END - NOISE_START)

    gen_losses, disc_losses = [], []

    for batch in train_dataset:
        gl, dl = train_step(batch, noise_std)
        gen_losses.append(gl)
        disc_losses.append(dl)

    g = float(tf.reduce_mean(gen_losses))
    d = float(tf.reduce_mean(disc_losses))
    gen_history.append(g)
    disc_history.append(d)

    print(f"Epoch {epoch:03d}/{EPOCHS} | noise_std={noise_std:.3f} | gen_loss={g:.4f} | disc_loss={d:.4f} | {time.time()-start:.1f}s")

    if epoch == 1 or epoch % 10 == 0:
        generate_images(f"Hasil Generator - Epoch {epoch}")

generate_images("Hasil Generator - FINAL")
```

Tahapan pelatihan model dilakukan melalui epoch atau waktu sekitar 1000 epoch, pojek ini menerapkan cara kerja **Linear Decay** pada penyimpanan standar *noise* untuk menjaga stabilitas interaksi saling berlawan.
Untuk proses lebih mendalamnya meliputi hal seperti:

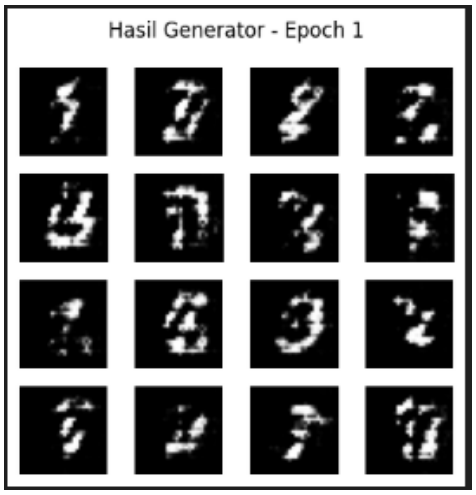
Pengurangan Noise Dinamis: Parameter noise_std disesuaikan secara bergerak maju dimulai dari nilai 0,10 pada awal pengulangan dan menyusut secara sejajar hingga mencapai 0,00 pada *epoch* terakhir.

- **Tahapan Saling Berlawanan:** Pada setiap kelompok data, fungsi train_step diproses untuk melakukan pembaruan bobot pada komponen Generator dan Discriminator secara serentak
- **Memantau dan catatan kinerja:** Data kerugian dari Generator (g_1) dan Discriminator (d_1) diperoleh pada setiap kelompok, kemudian dirata-ratakan di akhir setiap *epoch*.
- **Hasil Akhir:** Setelah menyelesaikan 1000 *epoch*, fungsi generate_images dipanggil untuk mensahkan output akhir. Pada tahap ini, model diharapkan telah mencapai titik menuju pusat.

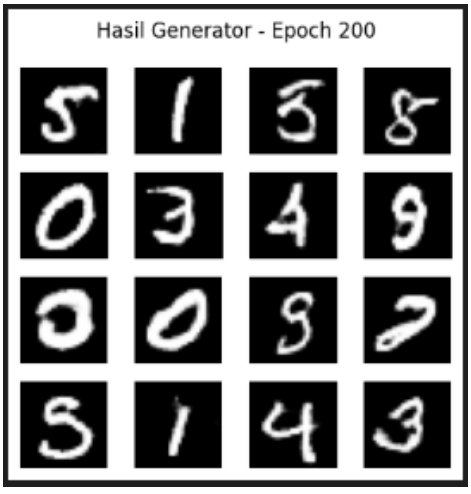
4. Hasil dan Pembahasan

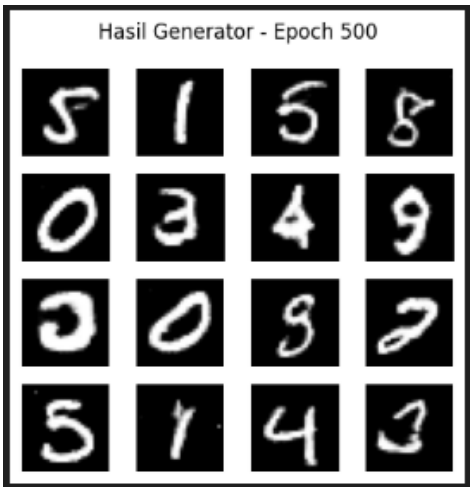
Pemantauan dilakukan secara berskala untuk melihat suatu model dari setiap tahapan, Perkembangan tersebut dapat dijelaskan ke dalam beberapa tahapan penting, dimulai dari tahap awal sebagai berikut:

- **Tahap Awal(Epoch 1):** Kondisi pada Epoch ini masih dalam keadaan acak dan buram

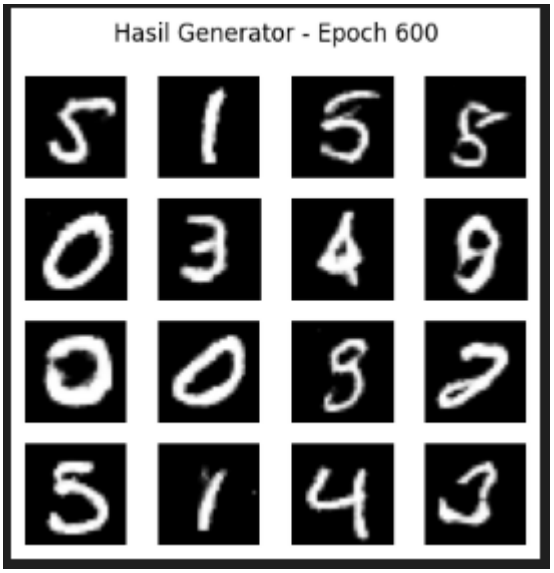


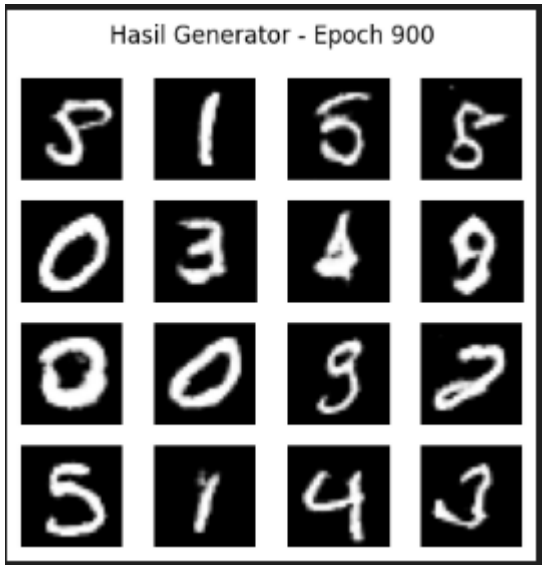
- **Tahap Keseimbangan dan Pengurangan Noise (Epoch 200 – 500):** Seiring dengan penurunan nilai noise_std secara sedikit demi sedikit, kualitas nya sudah menunjukan kejelasan angka tulisan tangan.



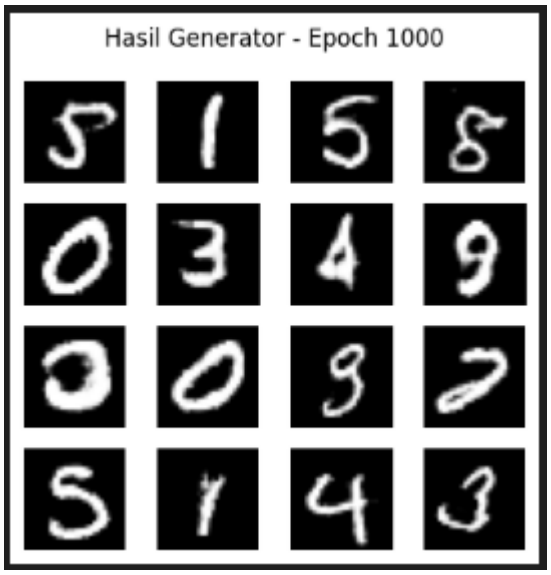


- **Tahap Perpindahan dan Susunan Pola (Epoch 600 – 900):** Selama proses ini pengurangan reduksi mulai sangat terlihat Generator bisa mengahsilkan angka yang lebih bagus.





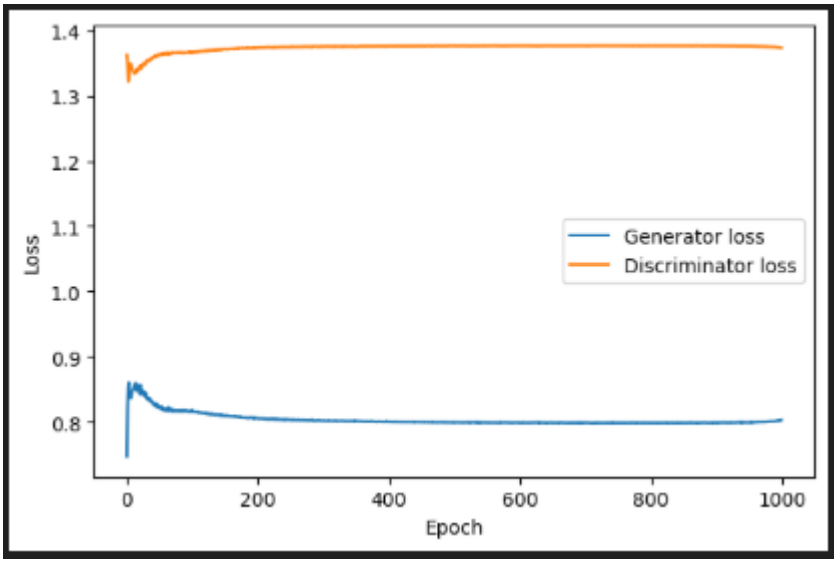
- **Tahap Optimal (Epoch 1000):** Pada tahap final ini, model telah mencapai titik keseimbangan yang diharapkan. Citranya menunjukan kualitas yang jelas, meskipun secara pembelajaran selanjutnya dapat terus ditingkatkan hingga 6000 epoch, capaian ini sudah sangat tepat untuk membuktikan keberhasilan model dalam menghasilkan data yang nyata.



5. Pemecahan Kinerja dan Batasan Model

Hasil menunjukan bahwa pembelajaran Generator dilakukan secara bertahap dan memakan waktu yang cukup lama.

Meskipun model sudah menghasilkan data sisntesis yang tepat , masih ada Batasan terkait ketajaman dan kejelasan lebih lanjut., seperti adanya efek blur atau pada garis angka.



Keseimbangan sistem ini sangat dipengaruhi oleh penerapan strategi **Linear Noise Decay**. Cara kerja tersebut secara tepat untuk mengurangi dominasi sebelum waktunya oleh Discriminator, sehingga memberikan ruang bagi Generator untuk memisahkan informasi dari data kea rah perbaikan hingga mencapai titik pusat akhir yang seimbang.

6. Kesimpulan

Berdasarkan rangkaian penerapan projek ini dapat disimpulkan dengan titik dasar sebagai berikut:

- **Kemampuan Arsitektur GAN:** Projek ini menyatakan bahwa kerangka kerja GAN memiliki ketetapan yang tinggi dalam memproses data visual yang rumit.
- **Signifikan Dataset MNIST:** MNIST terbukti menjadi acuan yang sangat cocok untuk tahap awal pembelajaran sistem generatif.
- **Menetapkan Pengaturan Pelatihan:** Kuliatas citra dipengaruhi oleh epoch semakin tinggi epoch makan hasil semakin jelas.
- **Kreativitas Sintesis Data:** Jaringan Generator terbukti tidak melakukan proses langsung terhadap dataset referensi. Model ini berhasil mengambil inti dari penulisan angka, yang memungkinkannya untuk memproduksi entitas visual baru yang bersifat unik namun tetap mempertahankan angka yang tepat dan tetap dengan data asli.

7. Referensi

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] I. Goodfellow et al., “Generative Adversarial Nets,” in Proc. Advances in Neural Information Processing Systems (NIPS), 2014, pp. 2672–2680.

[3] Y. LeCun et al., “The MNIST database of handwritten digits,” 1998.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," in *Proc. 30th International Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 2234–2242.

[6] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Amortised MAP Inference for Image Super-resolution," in *Proc. 5th International Conference on Learning Representations (ICLR)*, 2017. (Referensi utama untuk teknik penambahan *noise* pada input discriminator).

8. Kontribusi & distribusi anggota kelompok

- Rifqi Aryo Mulyadi: Implementasi kode dan training model
- Juan Pernando: Analisis hasil dan pengujian
- Adi Rosyadi: Penyusunan laporan dan dokumentasi
- Raditya M. W. H: Studi literatur dan referensi

Link Colab:
https://colab.research.google.com/drive/1oB20FhQKXRZ4-jzSzsPA7FRuvLyW_Gb9?usp=sharing