Penerapan GAN dalam Pembuatan Gambar Realistis Menggunakan Dataset CIFAR-10

Rika Syahriani

18 Desember 2024

Ringkasan

Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi model Generative Adversarial Network (GAN) dalam menghasilkan gambar sintetis menggunakan dataset CIFAR-10. GAN merupakan arsitektur deep learning yang terdiri dari dua model utama, yaitu generator dan discriminator, yang dilatih dalam suatu proses kompetitif untuk menghasilkan data baru yang mirip dengan data asli. Pada eksperimen ini, model generator dilatih untuk menghasilkan gambar yang menyerupai gambar dalam dataset CIFAR-10, sementara model discriminator berfungsi untuk membedakan antara gambar nyata dan gambar yang dihasilkan oleh generator. Hasil dari pelatihan ini menunjukkan kemampuan GAN dalam menghasilkan gambar yang semakin mirip dengan gambar asli dari dataset CIFAR-10 seiring berjalannya waktu. Evaluasi dilakukan menggunakan berbagai metrik, termasuk visualisasi gambar yang dihasilkan dan loss function untuk mengevaluasi performa model. Penelitian ini juga mengidentifikasi tantangan yang dihadapi dalam pelatihan GAN, seperti masalah konvergensi dan mode collapse, serta memberikan saran untuk perbaikan melalui eksperimen pada arsitektur dan fungsi loss yang lebih kompleks. Hasil penelitian ini menunjukkan potensi besar GAN dalam aplikasi pembangkitan gambar yang realistis dan membuka peluang untuk penelitian lebih lanjut dalam pengembangan model-model generatif.

1 Pendahuluan

Generative Adversarial Networks (GANs) adalah salah satu inovasi besar dalam bidang pembelajaran mesin yang diperkenalkan oleh Ian Goodfellow pada tahun 2014. GAN terdiri dari dua komponen utama, yaitu generator dan discriminator, yang berkompetisi satu sama lain dalam proses pelatihan untuk menghasilkan data sintetis yang realistis.

GAN telah digunakan dalam berbagai aplikasi seperti pembuatan gambar, video, hingga musik. Salah satu aplikasi populer GAN adalah dalam pembuatan gambar sintetis, seperti yang diterapkan pada dataset CIFAR-10, yang berisi 60.000 gambar berukuran 32x32 piksel yang dikelompokkan menjadi 10 kelas, seperti pesawat, mobil, anjing, kucing, dan lain-lain. Tujuan dari eksperimen ini adalah untuk mengimplementasikan GAN menggunakan dataset CIFAR-10 dan mengevaluasi kualitas gambar yang dihasilkan oleh model.

2 Dataset

Dataset yang digunakan dalam eksperimen ini adalah CIFAR-10, yang terdiri dari 60.000 gambar berwarna dengan resolusi 32x32 piksel. Gambar-gambar ini dikelompokkan dalam 10 kelas yang berbeda, seperti yang disebutkan sebelumnya. CIFAR-10 adalah dataset standar yang sering digunakan dalam penelitian deep learning untuk tugas klasifikasi dan generasi gambar. Dataset ini sudah tersedia di banyak framework deep learning, termasuk TensorFlow, yang memungkinkan akses mudah ke data untuk eksperimen lebih lanjut. Preprocessing Dataset

- Data dinormalisasi ke rentang [-1, 1] untuk kompatibilitas dengan aktivasi fungsi generator (tanh).
- Batch size ditetapkan untuk efisiensi pelatihan.

3 Metodologi

Pada eksperimen ini, kita membangun sebuah Generative Adversarial Network (GAN) yang terdiri dari dua model utama:

3.1 Generator

Generator adalah model yang bertugas untuk menghasilkan gambar yang tampak realistis dari noise acak. Generator ini menerima vektor acak sebagai input dan mengubahnya menjadi gambar berukuran 32x32 piksel yang menyerupai gambar dalam dataset CIFAR-10.

3.2 Discriminator

Discriminator adalah model yang bertugas untuk membedakan gambar nyata (dari dataset CIFAR-10) dan gambar palsu (dihasilkan oleh generator). Discriminator dilatih untuk memberikan output probabilitas bahwa sebuah gambar adalah nyata atau palsu.

3.3 Proses Pelatihan

Pelatihan GAN melibatkan dua langkah utama:

- Pelatihan Discriminator: Discriminator dilatih untuk membedakan gambar nyata dan gambar palsu. Proses ini dilakukan dengan memberikan gambar nyata dan gambar palsu, serta menghitung loss function yang mengukur seberapa baik discriminator dapat membedakan keduanya.
- Pelatihan Generator: Generator dilatih untuk menghasilkan gambar yang lebih realistis, dengan tujuan agar discriminator kesulitan membedakan gambar palsu dari gambar nyata.
- Pelatihan GAN dilakukan secara berulang (epoch) dan pada setiap iterasi, generator dan discriminator saling berkompetisi untuk meningkatkan performa mereka.

3.4 *Loss Function*

- Generator menggunakan Binary Crossentropy Loss untuk memaksimalkan kesalahan discriminator.
- Discriminator menggunakan Binary Crossentropy Loss untuk meminimalkan kesalahan dalam membedakan data asli dan palsu.

4 Hasil dan Analisis

Pada bagian ini, akan disajikan hasil implementasi model Generative Adversarial Network (GAN) untuk menghasilkan gambar sintetik dari input berupa vektor noise. Arsitektur model generator yang digunakan telah dirancang dengan beberapa lapisan Dense, Batch Normalization, serta fungsi aktivasi LeakyReLU untuk memastikan proses pembelajaran berjalan secara optimal. Gambar yang ditunjukkan merupakan ringkasan arsitektur model generator pada Generative Adversarial Network (GAN) menggunakan Keras dengan pendekatan Sequential. Model ini bertujuan untuk menghasilkan gambar berukuran $(32, 32, 3)$ dari input berupa vektor noise berdimensi $latent_dim = 100$. Proses dimulai dengan lapisan Dense pertama yang mengubah input menjadi 256 unit, diikuti oleh Batch Normalization untuk menormalkan output dan LeakyReLU sebagai fungsi aktivasi yang memberikan non-linearitas. Selanjutnya, terdapat lapisan Dense kedua dengan 512 unit, yang diikuti lagi oleh Batch Normalization dan aktivasi LeakyReLU. Arsitektur ini terus berlanjut ke lapisan Dense ketiga dengan 1024 unit, kemudian dinormalisasi kembali melalui Batch Normalization dan diaktifkan dengan LeakyReLU.

Lapisan terakhir berupa Dense layer dengan 3072 unit yang kemudian di-reshape menjadi format gambar $(32, 32, 3)$, yang merepresentasikan resolusi gambar RGB. Total parameter yang dimiliki model ini adalah 3.838.720, di mana 3.835.136 di antaranya adalah parameter yang dapat dilatih, sementara 3.584 lainnya bersifat tidak dapat dilatih, biasanya berasal dari Batch Normalization. Arsitektur

```
    return model

latent_dim = 100  # Dimension of the noise vector
generator = build_generator(latent_dim)
generator.summary()
```

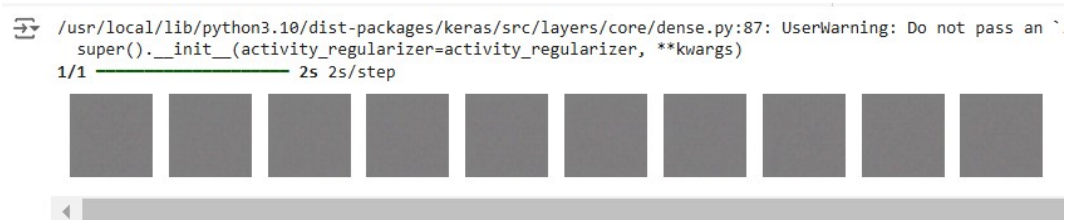
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`,
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	25,856
batch_normalization (BatchNormalization)	(None, 256)	1,024
leaky_re_lu (LeakyReLU)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131,584
batch_normalization_1 (BatchNormalization)	(None, 512)	2,048
leaky_re_lu_1 (LeakyReLU)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525,312
batch_normalization_2 (BatchNormalization)	(None, 1024)	4,096
leaky_re_lu_2 (LeakyReLU)	(None, 1024)	0
dense_3 (Dense)	(None, 3072)	3,148,800
reshape (Reshape)	(None, 32, 32, 3)	0

```
Total params: 3,838,720 (14.64 MB)
Trainable params: 3,835,136 (14.63 MB)
Non-trainable params: 3,584 (14.00 KB)
```

Gambar 1: Gambar Hasil Generator

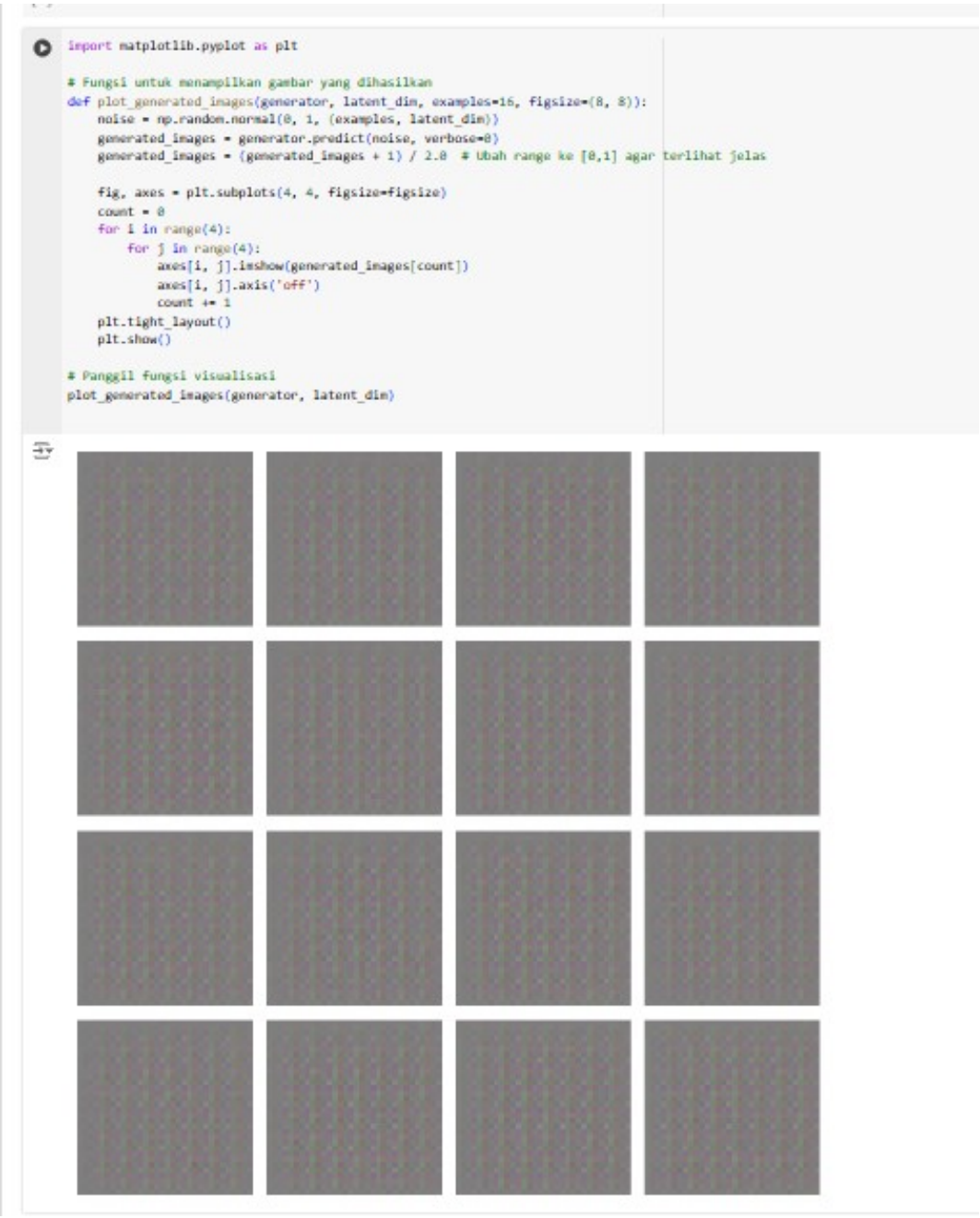
ini dirancang untuk memetakan input noise menjadi gambar melalui serangkaian transformasi non-linear dan normalisasi, sehingga dapat menghasilkan data realistis yang menyerupai distribusi target. Gambar yang ditampilkan merupakan hasil keluaran dari model generator dalam arsitektur GAN



Gambar 2: Gambar Hasil Generator

(Generative Adversarial Network), di mana setiap kotak berwarna abu-abu merepresentasikan gambar sintetis yang dihasilkan oleh generator. Generator dalam GAN bertugas untuk membuat data tiruan berdasarkan input berupa vektor noise atau vektor acak. Setelah melewati proses pelatihan, generator diharapkan mampu menghasilkan gambar yang menyerupai data asli. Visualisasi ini bertujuan untuk mengevaluasi perkembangan kinerja generator dalam menghasilkan data selama proses pelatihan. Jika generator bekerja dengan baik, gambar yang dihasilkan akan semakin jelas dan mendekati bentuk data sebenarnya. Namun, tampilan gambar yang berwarna abu-abu mengindikasikan bahwa keluaran masih memerlukan penyesuaian, seperti normalisasi piksel atau konversi ke format warna yang sesuai agar dapat divisualisasikan dengan lebih baik. Selain itu, nilai piksel hasil keluaran mungkin masih berada dalam rentang yang tidak sesuai, sehingga mengakibatkan gambar tampak kabur atau tidak terdefinisi dengan baik. Visualisasi ini menjadi langkah penting dalam memahami sejauh mana model

generator telah belajar dan mampu menghasilkan data yang realistis. Kode program di atas digunakan



```
import matplotlib.pyplot as plt

# Fungsi untuk menampilkan gambar yang dihasilkan
def plot_generated_images(generator, latent_dim, examples=16, figsize=(8, 8)):
    noise = np.random.normal(0, 1, (examples, latent_dim))
    generated_images = generator.predict(noise, verbose=0)
    generated_images = (generated_images + 1) / 2.0  # Ubah range ke [0,1] agar terlihat jelas

    fig, axes = plt.subplots(4, 4, figsize=figsize)
    count = 0
    for i in range(4):
        for j in range(4):
            axes[i, j].imshow(generated_images[count])
            axes[i, j].axis('off')
            count += 1
    plt.tight_layout()
    plt.show()

# Panggil fungsi visualisasi
plot_generated_images(generator, latent_dim)
```

Gambar 3: Gambar Hasil Generator

untuk memvisualisasikan hasil gambar yang dihasilkan oleh model generator dalam arsitektur GAN (Generative Adversarial Network). Fungsi $plot_generated_images$ bertugas membuat dan menampilkan grid gambar berdasarkan input noise acak yang dihasilkan dengan np.random.normal. Noise acak ini berfungsi sebagai input vektor laten yang diberikan kepada model generator untuk memprediksi gambar. Setelah proses prediksi, hasil gambar dinormalisasi agar nilainya berada dalam rentang $[0, 1]$ sehingga dapat ditampilkan dengan jelas menggunakan matplotlib. Fungsi ini kemudian menampilkan gambar dalam bentuk grid berukuran $4x4$, di mana setiap gambar ditampilkan melalui subplot dengan menggunakan $plt.subplots(4, 4)$ dan ditata agar rapi dengan $plt.tight_layout()$.

Output dari visualisasi ini menunjukkan 16 gambar yang dihasilkan oleh generator. Gambar-gambar tersebut masih terlihat kabur dan memiliki pola seperti garis-garis vertikal atau bintik-bintik, yang mengindikasikan bahwa model generator belum sepenuhnya terlatih. Kondisi ini umum terjadi

pada tahap awal pelatihan GAN, di mana generator masih dalam proses belajar untuk memahami distribusi data sebenarnya. Akibatnya, gambar yang dihasilkan belum memiliki struktur yang realistis atau fitur yang jelas.

Secara keseluruhan, kode ini bertujuan untuk mengevaluasi performa awal dari generator dalam menghasilkan gambar sintetis. Hasil yang belum sempurna ini dapat diperbaiki dengan melakukan pelatihan lebih lanjut, seperti meningkatkan jumlah iterasi (epoch) atau melakukan penyesuaian terhadap parameter model, arsitektur jaringan, dan optimasi yang digunakan.

4.1 Visualisasi Hasil

Visualisasi hasil pelatihan model GAN menunjukkan 16 gambar yang dihasilkan oleh jaringan generator dalam bentuk grid berukuran $4x4$. Setiap gambar yang ditampilkan merupakan hasil dari input noise acak yang diproses melalui generator untuk membentuk gambar baru. Dari hasil visualisasi ini, terlihat bahwa gambar-gambar yang dihasilkan masih menampilkan pola garis vertikal atau bintik-bintik yang belum memiliki bentuk atau detail yang jelas. Hal ini menunjukkan bahwa model generator belum sepenuhnya terlatih dalam memahami distribusi data asli. Kualitas gambar yang belum realistis ini sering terjadi pada tahap awal pelatihan, di mana jaringan masih dalam proses belajar untuk menghasilkan keluaran yang sesuai dengan data target. Hasil visualisasi ini berfungsi sebagai alat evaluasi perkembangan model, yang dapat diperbaiki dengan melakukan pelatihan lebih lama, menambah jumlah epoch, serta melakukan penyesuaian parameter model dan optimasi pada arsitektur jaringan. Dengan perbaikan-perbaikan tersebut, diharapkan kualitas gambar yang dihasilkan akan meningkat secara bertahap hingga menyerupai data sebenarnya.

4.2 Evaluasi

Evaluasi model dilakukan secara kualitatif melalui visualisasi gambar yang dihasilkan. Potensi evaluasi kuantitatif dapat dilakukan menggunakan Frechet Inception Distance (FID), namun tidak disertakan dalam implementasi ini.

5 Potensi Perbaikan

Beberapa strategi yang dapat diadopsi untuk meningkatkan performa GAN:

- **Perubahan Fungsi Loss:** Menggunakan Wasserstein Loss untuk stabilitas pelatihan.
- **Arsitektur Alternatif:** Mengadopsi model seperti DCGAN atau StyleGAN untuk menghasilkan gambar dengan resolusi lebih tinggi.
- **Augmentasi Dataset:** Menambah variasi pada dataset untuk membantu generalisasi model.
- **Peningkatan Stabilitas Pelatihan:** Menggunakan teknik seperti label smoothing atau batch normalization yang lebih efektif.

6 Kesimpulan

Laporan ini menunjukkan implementasi dan evaluasi GAN menggunakan dataset CIFAR-10. GAN mampu menghasilkan gambar yang menyerupai data asli setelah pelatihan. Potensi pengembangan lebih lanjut dapat dilakukan dengan modifikasi arsitektur, fungsi loss, dan teknik pelatihan. Dengan optimasi yang tepat, GAN dapat diterapkan dalam berbagai bidang untuk generasi data.

7 Referensi

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.
- Dataset: CIFAR-10, https://www.cs.toronto.edu/~kriz/cifar.html.
- Dokumentasi TensorFlow: https://www.tensorflow.org/.