

**TUGAS UJIAN AKHIR SEMESTER  
EKSPLORASI *GENERATIVE MODEL***

**LAPORAN  
IF5281 *Deep Learning***

**JUDUL TOPIK**

***Zoo Animals Image Generation using Conditional Deep Convolutional  
Generative Adversarial Network (cDCGAN)***

**Oleh:**

**MUHAMMAD RAZIF RIZQULLAH  
(23522050)**



**INSTITUT TEKNOLOGI BANDUNG  
Mei 2023**

## DAFTAR ISI

DAFTAR ISI.....	1
I. Abstrak.....	2
II. Pengantar.....	3
III. Studi Literatur.....	4
III.1. Generative Adversarial Network (GAN).....	4
III.2. Conditional Generative Adversarial Network (cGAN).....	5
III.3. Deep Convolutional Generative Adversarial Network (DCGAN).....	6
III.4. Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) untuk arabic handwritting.....	6
III.5. Improved Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) untuk plant vigor rating.....	8
IV. Dataset.....	10
V. Metode.....	14
VI. Eksperimen.....	18
VI.1. Eksperimen untuk nilai S_SIZE, NGF, dan NDF.....	18
VI.2. Eksperimen untuk nilai LR.....	23
VI.3. Eksperimen untuk nilai NZ.....	25
VII. Penutup.....	28
VII.1. Kesimpulan.....	28
VII.2. Saran.....	29
DAFTAR PUSTAKA.....	30

## I. Abstrak

Kebutuhan akan data merupakan salah satu hal utama yang perlu dipenuhi dalam membangun model *machine learning*. Beberapa pendekatan dilakukan untuk mengatasi hal ini diantara adalah *data augmentation* dan *data generation*. Salah satu pendekatan *data generation* dilakukan pada penelitian ini dengan mengembangkan arsitektur dari GAN, cGAN, dan DCGAN sehingga menghasilkan cDCGAN, sebuah arsitektur *deep convolution* pada GAN dengan kemampuan menghasilkan data yang *conditional* berdasarkan label atau kelas. Penelitian ini menggunakan data *Zoo Animals*, dipilih 8 kategori binatang yang menjadi target penelitian. Eksperimen pada penelitian ini membahas tentang pengaruh dari jumlah data tiap kelas (S\_SIZE), *number of generator filter* (NGF), *number of discriminator filter* (NDF), *learning rate* (LR), dan *number of latent space* (NZ). Berdasarkan penelitian yang dilakukan diperoleh hasil terbaik dengan konfigurasi S\_SIZE 1500, NGF 128, NDF 128, LR 2e-4, dan NZ 20.

**Keyword:** *GAN, cDCGAN, image generation*

## **II. Pengantar**

Perkembangan teknologi terutama *Artificial Intelligence* (AI) telah mencakup ke banyak aspek dalam kehidupan. Salah satu penggunaannya adalah pada *task* klasifikasi binatang di kebun binatang. Untuk mengatasi hal ini telah banyak dikembangkan model *Machine Learning* (ML) untuk bisa melakukan klasifikasi. Namun, terdapat permasalahan utama pada ML, yaitu butuh jumlah data yang cukup banyak untuk bisa memberikan hasil klasifikasi yang baik (Jordan dan Mitchell, 2015).

Masalah kurangnya data latih bisa diatasi dengan *image augmentation*, prosesnya adalah membuat data baru hasil modifikasi yang tidak menghilangkan ciri dari data aslinya (Kostrikov dkk., 2004). *Image augmentation* sudah diteliti dan digunakan sejak lama, meski demikian, hal ini masih belum bisa mengatasi kekurangan data yang ada karena proses augmentasi juga memiliki keterbatasan tentang seberapa banyak kombinasi augmentasi yang bisa dihasilkan.

Untuk mengatasi jumlah kombinasi data yang terbatas, selanjutnya diteliti metode baru yang bisa membuat kombinasi gambar lebih banyak dari pada proses augmentasi atau kombinasi hingga tak terhingga, yaitu *image generation* (Gregor dkk., 2015). Beberapa pendekatan berbasis ML sudah dilakukan, yaitu mulai dari *Recurrent Neural Network* (RNN) (Gregor dkk., 2015), *Generative Adversarial Network* (GAN) (Goodfellow dkk., 2014), *Conditional Generative Adversarial Network* (cGAN) (Mirza dan Osindero, 2014), *Deep Convolutional Generative Adversarial Network* (DCGAN) (Radford dkk., 2015), dan *Conditional Deep Convolutional Generative Adversarial Network* (cDCGAN) (Zhu, 2020).

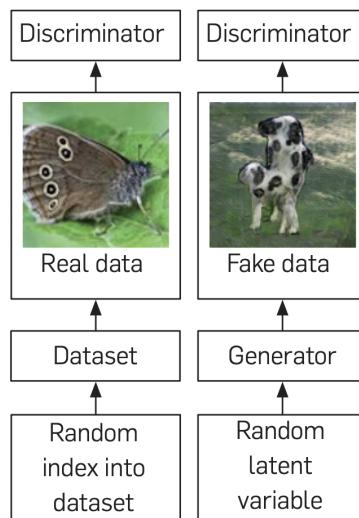
Data yang digunakan pada penelitian ini adalah dataset baru yang dibuat oleh Dabberger (2023), belum ada penelitian yang menggunakan dataset ini untuk diterapkan pada *image generation*. Penelitian ini fokus pada eksperimen yang menghasilkan suatu model untuk *image generation* berdasarkan kategori dari binatang yang ada. Berdasarkan pertimbangan jenis data dan penelitian-penelitian sebelumnya, pendekatan berbasis cDCGAN dipilih pada penelitian ini.

### III. Studi Literatur

Penelitian ini membahas tentang penggunaan *image generation* untuk dataset *Zoo Animals*. Untuk memahami lebih lanjut tentang konsep dan perkembangan teknologi *image generation*, berikut merupakan beberapa literatur yang terkait.

#### III.1. *Generative Adversarial Network (GAN)*

Goodfellow dkk. (2014) melakukan penelitian untuk membangun model *image generation* berbasis pada teknik *machine learning* menggunakan *neural network* untuk mengatasi keterbatasan data. Penelitian ini adalah awal dari tercetusnya metode baru yang belum ada sebelumnya yaitu *Generative Adversarial Network* (GAN). Perbedaan utama dari GAN dengan *generative model* lain adalah pada cara kerjanya. Generative model lain dibentuk berdasarkan teori optimisasi namun GAN dibentuk berbasis *game theory*.



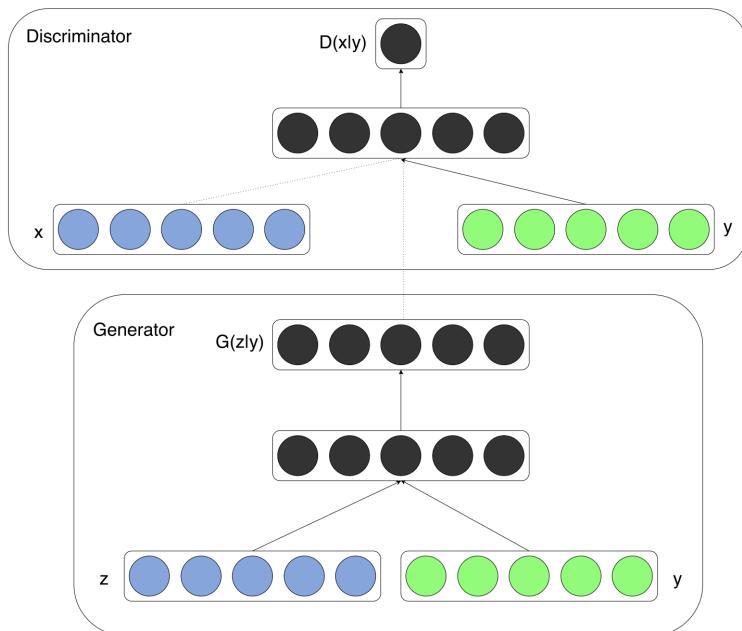
Gambar III.1. Cara kerja GAN (Goodfellow dkk., 2014)

Seperti yang ditunjukkan pada Gambar III.1 GAN bekerja menggunakan dua bagian utama, yaitu *generator* dan *discriminator*. Bagian *generator* mendapatkan masukan berupa *random latent variable*, yang mana menghasilkan suatu *fake data*. Kemudian bagian *discriminator* dilatih untuk bisa membedakan antara *fake data* hasil *generator model* dan *real data* yang berasal dari dataset. Penelitian awal

GAN sudah mampu melakukan generasi gambar, namun hal ini kurang baik karena proses *training* yang tidak stabil. Permasalahan lain ada di bagaimana data masukan dibangun, penggunaan *latent space* secara *random* dinilai kurang berguna karena kelas dari gambar yang dihasilkan tidak terkontrol dengan baik.

### III.2. Conditional Generative Adversarial Network (cGAN)

Salah satu permasalahan utama dari *vanilla* GAN adalah ketidakmampuan melakukan generasi gambar untuk suatu kelas tertentu. Mirza dan Osindero (2014), mengembangkan GAN untuk bisa mengatasi hal ini dengan nama *Conditional Generative Adversarial Nets* (cGAN), dengan arsitektur yang ditunjukkan pada Gambar III.2 berikut.

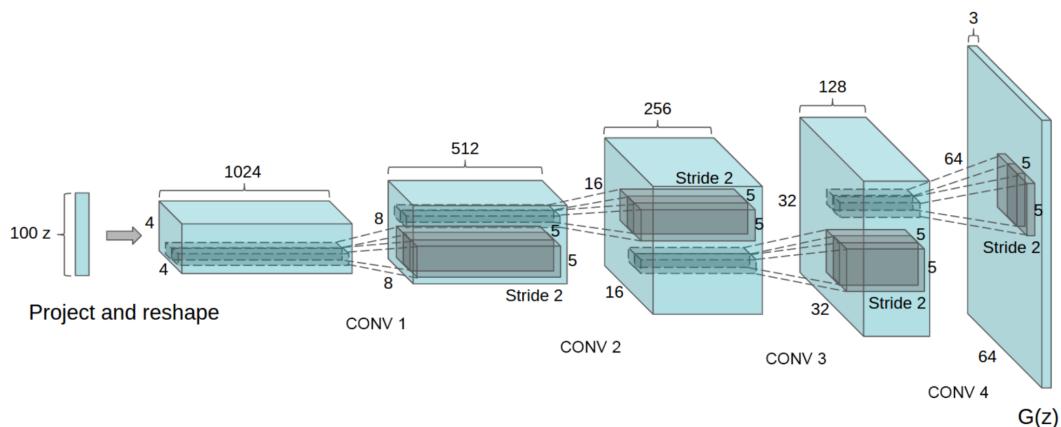


Gambar III.2. Arsitektur cGAN (Goodfellow dkk., 2014)

Perbedaan utama antara cGAN dan GAN adalah bagian input dari model *generator* dan *discriminator*. Pada cGAN masing-masing *input* dari model ditambahkan *value* dari kelas atau label ( $y$ ). Tujuannya adalah membuat gambar ( $x$ ) dan *latent space* ( $z$ ) memuat informasi label ( $y$ ). Penelitian ini diterapkan pada gambar *Handwriting MNIST dataset*, dan mampu menghasilkan gambar yang sesuai dengan labelnya. Arsitektur cGAN hanya mengganti bagian *input* saja, sehingga permasalahan GAN untuk proses *training* yang tidak stabil masih ada.

### III.3. Deep Convolutional Generative Adversarial Network (DCGAN)

Radford dkk. (2016) mengusulkan suatu pendekatan baru yang mengembangkan GAN untuk bisa mengatasi permasalahan GAN yang tidak stabil pada proses training, dan *generator* yang kurang baik dalam menangkap karakteristik dari gambar. Pengembangan ini memanfaatkan *Convolutional Neural Network* (CNN) yang dilatih dengan pendekatan *unsupervised*.



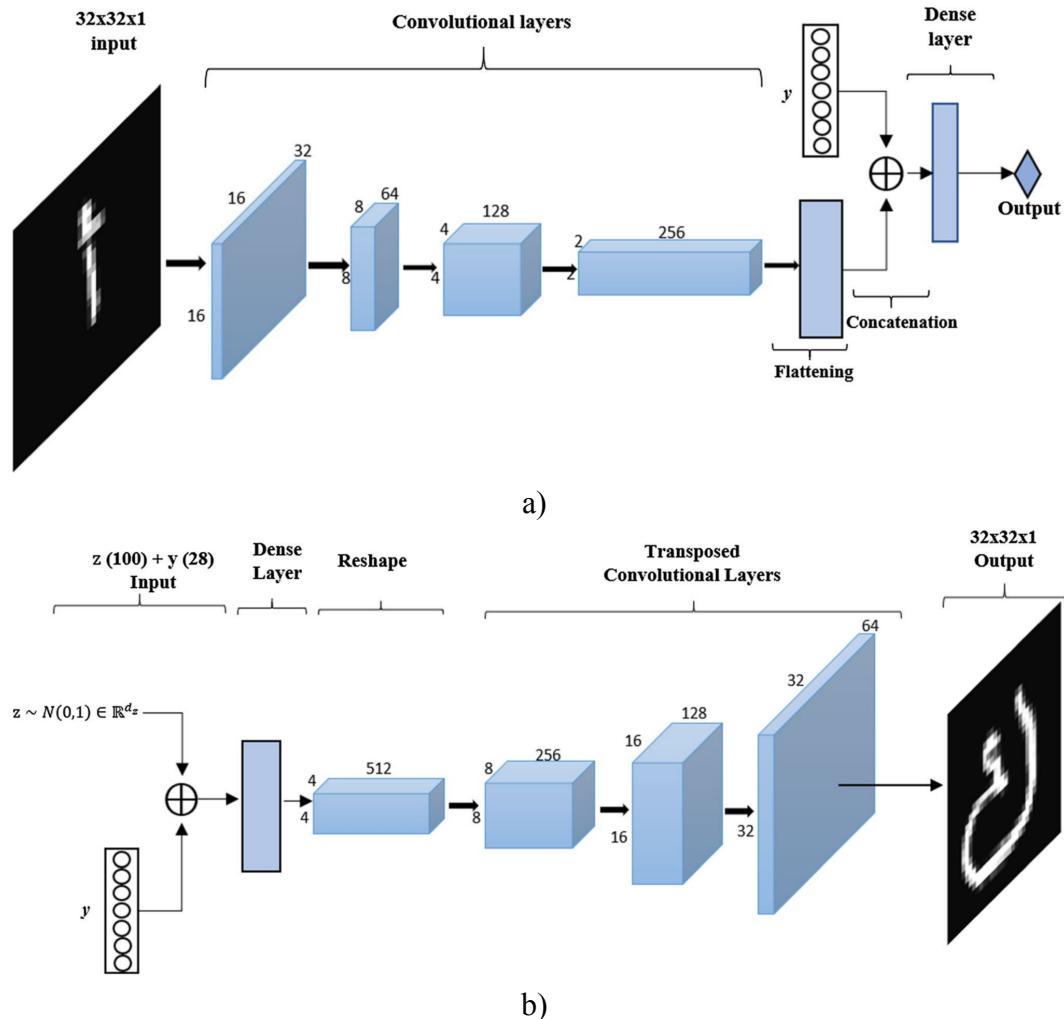
Gambar III.3. Arsitektur *Generator* pada DCGAN (Radford dkk., 2016)

Arstektur baru yang diusulkan ada pada Gambar III.3, yang mana pada arsitektur ini bagian *generator* didesain menggunakan beberapa *layer CNN* dengan masukan berupa *latent space* ( $z$ ) dan keluaran berupa gambar berukuran  $3 \times 64 \times 64$ . Hasil dari DCGAN ini dinilai lebih baik dari pada *vanilla GAN*, namun, penelitian ini masih melakukan generasi berdasarkan sebuah *random latent space*, untuk bisa membuat generasi yang bisa diatur targetnya (label) maka gabungan antara pendekatan cGAN dan DCGAN dilakukan oleh penelitian selanjutnya.

### III.4. Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) untuk arabic handwriting

Mustapha dkk. (2022) melakukan penelitian untuk melakukan generasi tulisan arab, dalam penelitiannya Mustapha dkk. menggabungkan pendekatan DCGAN dan cGAN. Arsitektur yang digunakan dalam penelitian tersebut ditunjukkan pada Gambar III.4. Terdapat gambar III.3.a yaitu bagian *discriminator*, yang mana *input* model adalah sebuah gambar dengan dimensi  $32 \times 32 \times 1$ , gambar ini masuk ke

dalam layer CNN. Pada bagian ujung setelah terjadi proses *flattening*, ada informasi tambahan berupa label yang masuk ke dalam proses dengan cara *concatenation*. Informasi ini kemudian diolah menggunakan *fully connected layer* sebelum akhirnya menjadi *output*.



Gambar III.4. Arsitektur cDCGAN untuk *arabic handwritting* a) *discriminator*; dan b) *generator* (Mustapha dkk., 2016)

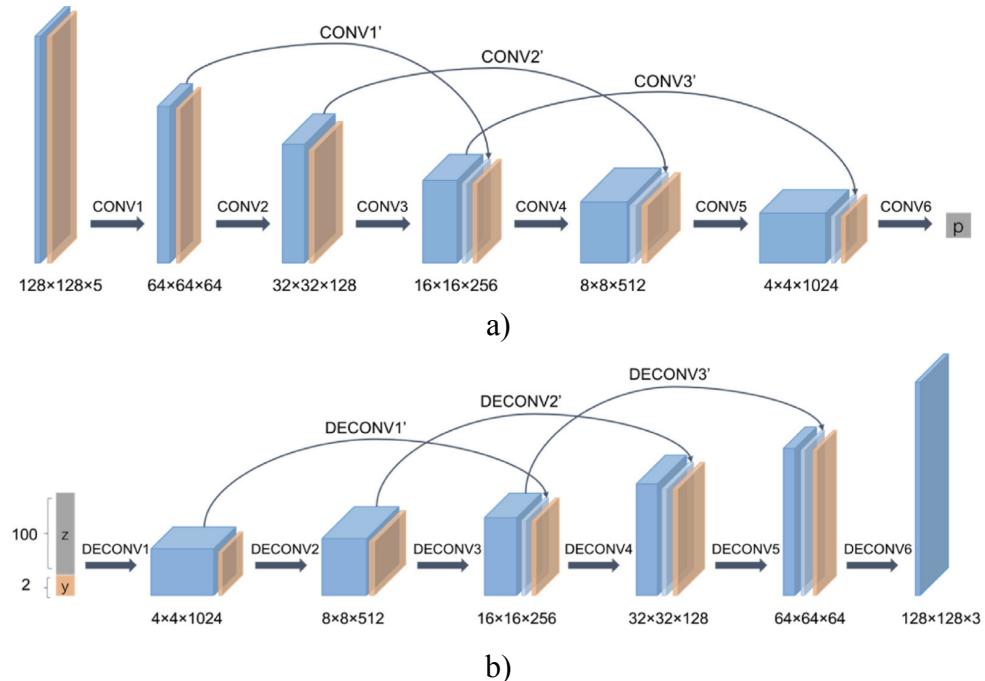
Mirip dengan apa yang terjadi pada bagian *discriminator*, bagian *generator* juga melibatkan label. Proses yang melibatkan label terjadi pada bagian awal jaringan, dimana masukan berupa *latent space* ( $z$ ) diproses secara *concatenation* dengan label. Setelah informasi hasil *concat* terbentuk ada satu lapisan *fully connected layer* lagi untuk selanjutnya masuk ke dalam arsitektur CNN. Terdapat empat lapis

*layer CNN* yang digunakan dengan output berupa gambar dengan dimensi 32x32x1.

Penelitian ini berhasil membentuk gambar *arabic handwritting* baru berdasarkan data yang sudah ada, namun terdapat aspek lain yang belum diteliti yaitu penggunaan arsitektur ini pada gambar yang memiliki tiga *channel* atau RGB *channel*.

### III.5. Improved Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) untuk *plant vigor rating*

Zhu dkk., (2020) melakukan penelitian untuk membuat data augmentasi pada tanaman untuk selanjutnya diterapkan pada tugas *plant vigor rating*. Tanaman memiliki tingkat kesegaran yang berbeda-beda, masalahnya adalah jumlah data yang masih sedikit sehingga perlu ada data augmentasi. Data augmentasi yang standar dinilai kurang baik sehingga pendekatan berbasis GAN dilakukan. Arsitektur yang digunakan pada penelitian ini ditunjukkan pada Gambar III.5 berikut.



Gambar III.5. Arsitektur cDCGAN untuk a) *discriminator*; dan b) *generator* (Zhu dkk., 2016)

Penelitian ini menggunakan data dengan ukuran gambar 128x128 dan tiga channel RGB, jumlah label yang digunakan ada dua yaitu *weak* dan *healthy*. Berdasarkan arsitektur yang digunakan, terlihat ada perbedaan arsitektur cDCGAN dibandingkan dengan penelitian yang dilakukan oleh Mustapha dkk. (2022). Pada penelitian Zhu dkk. (2020) bagian *generator* tetap menggunakan label sebagai salah satu masukan, namun label tersebut sudah di-*concat* dengan gambar *input* mulai dari awal arsitektur. Pada penelitian ini di bagian *generator* juga berbeda, ada tambahan bagian DECONV1', DECONV2', dan DECONV3' yang merupakan hubungan antar *convolution layer*. Hasil dari bagian generator merupakan gambar dengan tiga channel RGB, dimensinya adalah 3x128x128. Hasil penelitian ini menunjukkan bahwa penggunaan gambar hasil generasi pada proses latih model klasifikasi memberikan kinerja yang lebih baik dibandingkan mdoel klasifikasi tanpa data hasil generasi.

#### **IV. Dataset**

Penelitian ini menggunakan dataset yang berasal dari situs Kaggle dengan judul *Zoo Animals* yang diunggah oleh Jirka Daberger pada Maret 2023. Dataset ini berisi gambar yang diperoleh dari berbagai sumber lain. Konten yang ada dalam dataset adalah gambar dan label dengan format *Pascal Visual Object Classes* (Pascal VOC) berbentuk file XML. Namun, pada penelitian ini label tersebut tidak digunakan karena fokusnya adalah untuk melakukan generasi gambar berdasarkan kategori dan kategori ini yang selanjutnya akan digunakan sebagai label. Terdapat total 19 kategori dari dataset ini seperti pada tabel berikut.

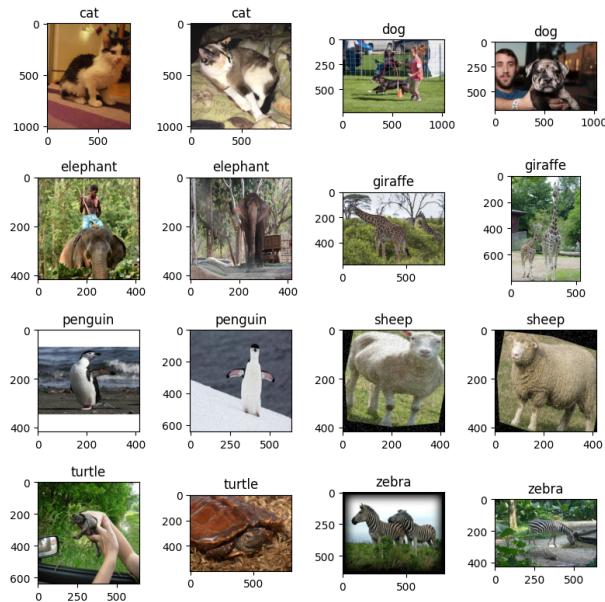
Tabel IV.1. Kategori yang tersedia dan jumlah gambar

Kategori	Jumlah Gambar
cow	7068
rhino	2412
flamingo	100
parrot	3618
elephant	4112
lion	2492
penguin	5026
capybara	1344
tiger	3870
dog	5720
buffalo	502
turtle	1670
giraffe	1902
zebra	4068
kangaroo	2640
cat	5588
sheep	5220
deer	6066
jaguar	1984

Dari keseluruhan kategori seperti yang tertampil pada tabel di atas, terdapat beberapa kesenjangan jumlah gambar tiap kategori, ada yang hanya punya 100 dan ada yang jumlahnya hingga 7068. Berdasarkan pertimbangan jumlah data dan kemampuan mesin komputasi yang tersedia, maka diambil 8 kategori saja untuk digunakan pada penelitian ini. Detail kategori yang digunakan ditunjukkan pada Tabel IV.2.

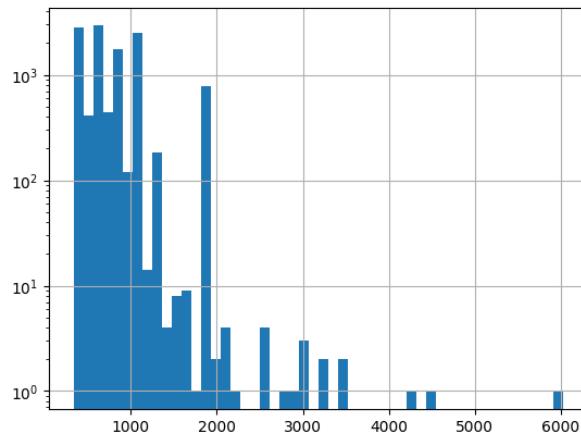
Tabel IV.2. Kategori yang dipilih dan jumlah gambar

Kategori	Jumlah Gambar
cat	5588
dog	5720
penguin	5026
zebra	4068
giraffe	1902
elephant	4112
sheep	5220
turtle	1670

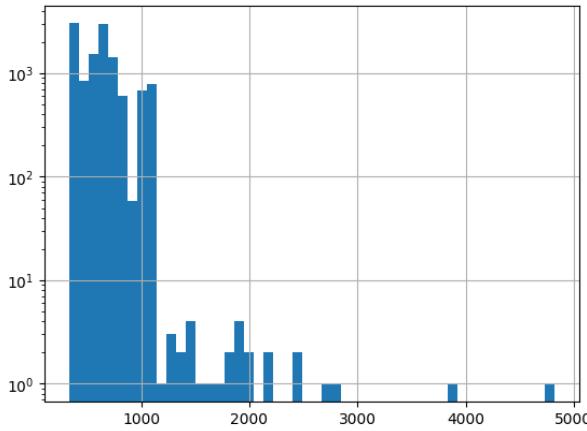


Gambar IV.1. Sampel data dengan jumlah dua gambar untuk masing-masing kategori

Pada Gambar IV.1 ditampilkan beberapa contoh gambar yang akan digunakan pada penelitian, masing-masing kategori ditampilkan dua sampel data. Terdapat perbedaan dimensi untuk masing-masing gambar. Untuk mempelajari lebih jauh tentang persebaran dimensi ini, maka dibuat histogram plot untuk melihat persebarannya. Histogram plot tersebut ditampilkan pada Gambar IV.2 berikut.



a)



b)

Gambar IV.2. *Histogram* untuk a) panjang gambar (*width*) dan b) tinggi gambar (*height*) untuk data yang akan digunakan pada proses latih

Berdasarkan Gambar IV.2, terlihat bahwa persebaran ini tidak teratur dengan nilai antara ratusan hingga lima ribu piksel. Oleh karena ini proses resize diperlukan untuk membuat gambar dengan ukuran dimensi yang sama. Gambar pada data diubah ukurannya menjadi  $64 \times 64$  dengan warna yang tetap yaitu 3 channel RGB. Gambar di bawah menunjukkan hasil proses resize.

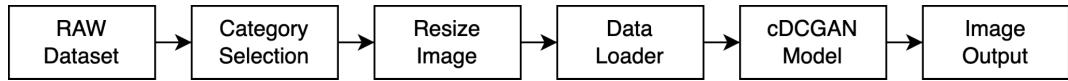


Gambar IV.3. Sampel data yang sudah diubah ukurannya menjadi 64 x 64

Terlihat pada gambar di atas, setelah *di-resize* informasi dari gambar masih bisa terbaca sehingga gambar ini bisa digunakan sebagai dataset. Tidak semua gambar yang ada pada dataset dijadikan gambar latih. Pada penelitian ini digunakan dua jenis jumlah data tiap kategori untuk dibandingkan hasilnya, yaitu dengan jumlah 500 dan 1500 gambar setiap kategorinya, sehingga total data yang digunakan adalah 4000 dan 12000 gambar, diambil secara acak dari *dataset* sesuai kategori.

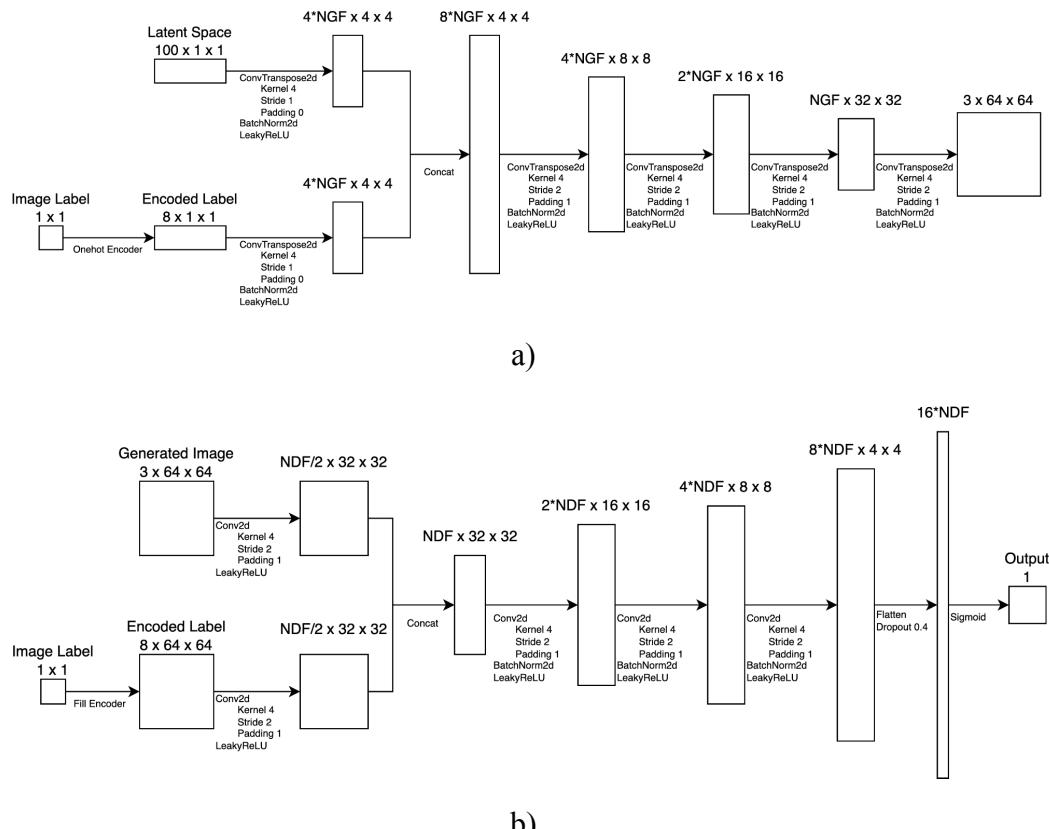
## V. Metode

Metode yang dipilih pada penelitian ini adalah *Conditional Deep Convolutional Generative Adversarial Network* (cDCGAN). Proses latih yang ada pada metode ini ditunjukkan pada Gambar V.1 berikut.



Gambar V.1. Alur dari dataset untuk menjadi gambar keluaran cDCGAN

Untuk arsitektur cDCGAN yang digunakan pada penelitian ini ditunjukkan oleh Gambar V.2 berikut.



Gambar V.2. Arsitektur cDCGAN yang digunakan pada penelitian ini a) bagian *generator* dan b) bagian *discriminator*

Pada Gambar V.2 terdapat *number of generator filter* (NGF), *number of discriminator filter* (NDF), dan 4 lapis *convolution layer*. Berdasarkan gamabr tersebut, terlihat bahwa arsitektur cDCGAN yang digunakan pada penelitian ini mirip dengan DCGAN (Radford dkk., 2016). Perbedaan utama dari DCGAN dan cDCGAN adalah bagian *input* pada model. Pada DCGAN, *input* pada *generator* hanya *latent space* (NZ) saja dan pada *discriminator* hanya *fake image* hasil *generator* saja, namun berbeda dengan cDCGAN. Pada cDCGAN ada label yang menjadi bagian dari input model, hal ini bisa dilakukan dengan proses *encoder*.

Proses *label encoder* yang digunakan pada bagian *generator* sedikit berbeda dengan *discriminator*. Pada *generator* label di encode menggunakan *onehot encoder*, yang mana cara kerjanya adalah membuat membuat vektor dengan indeks yang sama dengan label diberi nilai 1 dan sisanya akan menjadi nilai nol.

Diasumsikan, label yang akan masuk proses *encoding* pada *generator* merupakan kelas dengan nilai 1, untuk rentang kelas 0 hingga 7 (total terdapat 8 kelas). Maka hasil dari proses encoding menggunakan *onehot encoder* adalah vektor dengan nilai [0, 1, 0, 0, 0, 0, 0, 0]. Terlihat pada vektor, indeks 1 diberi nilai 1 karena label yang masuk memiliki nilai 1. Jika label yang masuk memiliki nilai 5, maka hasil *label encoding* pada *generator* adalah [0, 0, 0, 0, 0, 1, 0, 0].

Proses *label encoding* pada bagian *discriminator* sedikit berbeda dalam hal ukuran, namun konsepnya sama, yaitu mengisi nilai pada vektor dengan 1 sesuai dengan nilai dari label yang masuk. Perbedaannya, *onehot encoder* pada *generator* menghasilkan ukuran (8, 1, 1), sedangkan pada *fill encoder* di *discriminator* memiliki dimensi vektor (8, 64, 64). Nilai 64 tersebut menyesuaikan dimensi dari gambar yang digunakan.

Gambar V.3 berikut merupakan implementasi dari *label encoder* untuk selanjutnya digunakan pada model *generator* dan *discriminator*. Pada gambar tersebut, terlihat *shape* dari *encoded label* memiliki dimensi (8, 8, 1, 1) untuk *onehot* dan (8, 8, 64, 64) untuk *fill*, ini dikarenakan angka 8 yang pertama menunjukkan jumlah kelas pada data yang digunakan. Ini untuk memudahkan pembuatan *encoder*, pada penggunaannya hanya perlu memanggil `onehot[label]` atau `fill[label]`.

Dimensi *encoded label* yang masuk ke dalam model sama seperti penjelasan sebelumnya yaitu (8, 1, 1) untuk *generator* dan (8, 64, 64) untuk *discriminator*.

```
# label preprocess using onehot encoder
onehot = torch.zeros(CLASS_NUM, CLASS_NUM)
onehot = onehot.scatter_(1, torch.LongTensor(list(range(CLASS_NUM))).view(CLASS_NUM,1), 1).view(CLASS_NUM, CLASS_NUM, 1, 1)
fill = torch.zeros([CLASS_NUM, CLASS_NUM, IMG_SIZE, IMG_SIZE])
for i in range(CLASS_NUM):
    fill[i, i, :, :] = 1

onehot, fill = onehot.to(DEVICE), fill.to(DEVICE)
onehot.shape, fill.shape

(torch.Size([8, 8, 1, 1]), torch.Size([8, 8, 64, 64]))
```

Gambar V.3. Label encoder dengan metode onehot

Selanjutnya, *encoded label* masuk ke proses ConvTranspose2d dan mengubah ukuran yang sebelumnya 8x1x1 menjadi 4\*NGF x 4 x 4. Bagian *latent space* juga masuk ke proses yang serupa yaitu ConvTranspose2d, mengubah ukuran sebelumnya 100x1x1 menjadi 4\*NGF x 4 x 4. *Input* dari bagian *encoded label* dan *latent space* pada *generator* digabungkan dengan *concat*, sehingga ukurannya menjadi 8\*NGF x 4 x 4, selanjutnya data ini masuk ke dalam arsitektur DCGAN. Arsitektur pada DCGAN ini memiliki empat tahap ConvTranspose2d untuk mengubah ukuran dimensi 8\*NGF x 4 x 4 menjadi 3x64x64, sesuai dengan dimensi target gambar.

Arsitektur model *discriminator* memiliki dua bagian *input*, yaitu sebuah gambar dengan ukuran 3x64x64 dan label. Label dengan dimensi 1x1 masuk proses *fill encoder*, hasilnya memiliki dimensi 8x64x64. Gambar dan juga *encoded label* masing-masing masuk ke *layer Conv2d* untuk mereduksi dimensi menjadi NDF/2 x 32 x 32. Kemudian hasilnya masuk ke proses concat dan menghasilkan dimensi NDF x 32 x 32. Data dengan ukuran NDF x 32 x 32 ini masuk ke dalam arsitektur DCGAN, terdapat empat *convolutional layer*, setelah itu terdapat satu *flatten layer*, bagian output dari arsitektur ini adalah fungsi *sigmoid* dengan *output* 0 atau 1. *Output* dengan nilai 0 menandakan gambar diklasifikasi sebagai *fake* dan value 1 merupakan gambar *real*.

Proses latih metode cDCGAN ini melibatkan model bagian *generator* dan *discriminator*, yang mana perbedaan dengan DCGAN ada pada bagian *label encoder*, sehingga label ikut masuk ke dalam arsitektur. Hasil dari penggunaan

model dibahas pada bagian eksperimen. Implementasi dari program ini tersedia pada GitHub *repository* dengan alamat.

<https://github.com/rizquuula/cDCGAN-ZooAnimals>

## VI. Eksperimen

Desain eksperimen penelitian ini ditujukan untuk menilai kinerja dari model cDCGAN dengan studi ablati untuk jumlah *filter* yang digunakan, penentuan *hyperparameter*, dan pengukuran waktu *training*. Karena ukuran model yang dalam dan butuh komputasi paralel yang tinggi, maka pada eksperimen ini hanya menggunakan mesin GPU. Mesin yang digunakan berasal dari Google Colab versi gratis, yaitu GPU model Tesla T4, jumlah GPU 1, dan *memory* 15360 MB. Karena keterbatasan GPU, maka Kaggle juga digunakan dengan spesifikasi GPU model Tesla P100, jumlah GPU 1, dan *memory* 16280 MB.

### VI.1. Eksperimen untuk nilai S\_SIZE, NGF, dan NDF

Desain eksperimen yang dilakukan melibatkan beberapa hal, yaitu jumlah sampel gambar yang digunakan untuk masing-masing kategori (S\_SIZE), jumlah filter pada generator (NGF), dan jumlah filter pada discriminator (NDF). Perlu disampaikan bahwa pada bagian ini terdapat beberapa hyperparameter yang tidak berubah, seperti jumlah epoch 100, learning rate 2e-4, Beta1 0.5, batch size 32, dan ukuran latent space 100. Eksperimen bagian VI.1 ini fokus untuk mengetahui pengaruh jumlah filter terhadap gambar hasil *generator* dan waktu *training* dengan desain skenario yang ditunjukkan pada Tabel VI.1.

Tabel VI.1. Desain eksperimen pada penelitian S\_SIZE, NGF, dan NDF

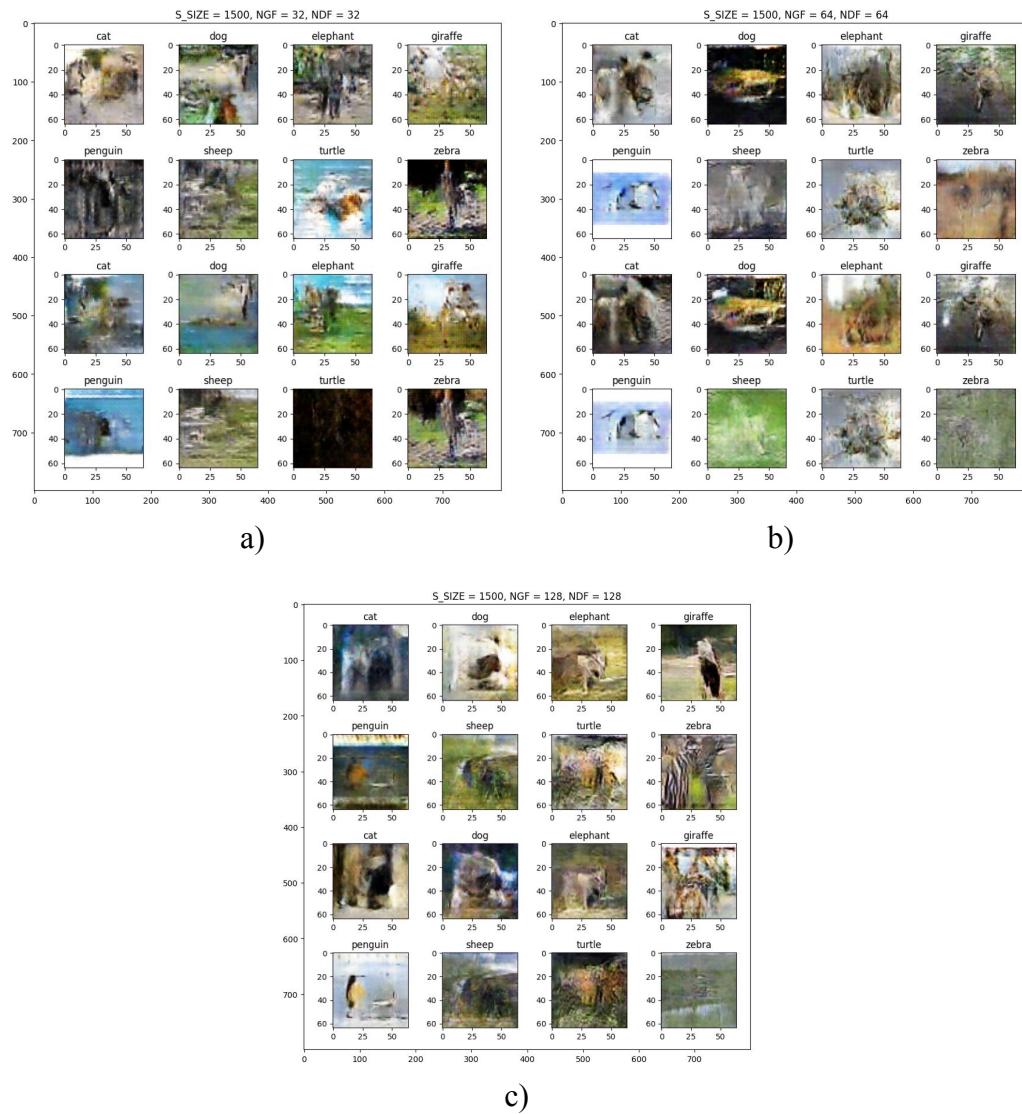
ID	NGF	NDF	S_SIZE
1	32	32	500
2	64	64	500
3	64	128	500
4	128	64	500
5	128	128	500
6	32	32	1500
7	64	64	1500
8	64	128	1500
9	128	64	1500
10	128	128	1500

Ukuran S\_SIZE tiap kategori dipilih sebagai salah satu parameter penelitian untuk melihat hubungan antara jumlah data yang digunakan dengan hasilnya. Jumlah NGF dan NDF juga diteliti untuk melihat pengaruh dari nilai tersebut terhadap hasil gambarnya dan juga waktu proses training yang diperlukan. Hasil eksperimen akan dibahas dalam bentuk kelompok-kelompok kecil dari eksperimen, sehingga perbandingan hasil bisa dibahas lebih detail dan rinci. Berikut merupakan perbandingan hasil untuk S\_SIZE 500 pada NGF dan NDF yang jumlahnya sama.



Gambar VI.1. Hasil untuk S\_SIZE 500 dengan a) NGF dan NDF = 32; b) NGF dan NDF = 64; c) NGF dan NDF = 128

Berdasarkan Gambar VI.1., dapat diperhatikan bahwa hasil dari nilai NGF dan NDF yang lebih tinggi menghasilkan gambar yang lebih mudah dikenali. Untuk jumlah NGF dan NDF 32 gambar terlihat masih bercampur warnanya dan sulit untuk dibedakan, begitupun pada NGF dan NDF 64 meski untuk beberapa bagian bentuk dari warnanya sudah menyerupai hewannya, yang lebih baik ada pada NGF dan NDF 128 karena bentuk hewan sudah semakin terlihat dan warnanya sudah sesuai, meski masih belum terlalu jelas.



Gambar VI.2. Hasil untuk S\_SIZE 1500 dengan a) NGF dan NDF = 32; b) NGF dan NDF = 64; c) NGF dan NDF = 128

Pada Gambar VI.2., terlihat bahwa kesimpulan hasilnya mirip dengan Gambar VI.1. yang menggunakan S\_SIZE 500, tidak ada perubahan yang drastis namun

dapat terlihat bahwa penggunaan S\_SIZE yang lebih besar membuat model bisa belajar lebih baik. Perbedaan yang cukup jelas terlihat pada NGF dan NDF = 128, untuk S\_SIZE 500 warna masih terlalu bercampur namun pada S\_SIZE 1500 warna sudah dipelajari lebih baik. Hasil terbaik diperoleh pada S\_SIZE 1500, NGF dan NDF 128, gambar *elephant*, *penguin*, dan *zebra* terlihat lebih jelas bentuk dan warnanya. Selanjutnya adalah eksperimen untuk nilai NGF dan NDF yang tidak seimbang, untuk melihat bagaimana kinerja dari model jika sebagian filter hilang. Berikut merupakan hasil dari eksperimen tersebut.



Gambar VI.3. Hasil dengan NGF dan NDF tidak seimbang a) S\_SIZE 500, NGF 64, NDF 128; b) S\_SIZE 500, NGF 128, NDF 64; c) S\_SIZE 1500, NGF 64, NDF 128; dan d) S\_SIZE 1500, NGF 128, NDF 64

Berdasarkan Gambar VI.3, dapat terlihat cukup jelas hasilnya untuk model dengan jumlah NGF > NDF seperti pada bagian b) dan d), memiliki hasil yang lebih buruk dibandingkan hasil untuk jumlah NGF < NDF seperti pada bagian a) dan c). Jika dibandingkan antara a) dan c), maka dapat terlihat bahwa c) dengan S\_SIZE yang lebih besar menghasilkan gambar hasil *generator* dengan kualitas yang lebih baik dibandingkan bagian a). Oleh karena itu, nilai NDF berpengaruh signifikan pada kinerja dari arsitektur cDCGAN secara keseluruhan, namun hasil yang terbaik tetap diperoleh ketika nilai NGF dan NDF seimbang. Berdasarkan Gambar VI.1, VI.2, dan VI.3. maka dapat terlihat bahwa gambar dengan S\_SIZE 1500, NGF 128, dan NDF 128 memiliki hasil yang terbaik.

Selanjutnya adalah analisis waktu training yang dibutuhkan dalam melatih model dengan jumlah epoch yang sama yaitu 100. Tabel VI.2. menunjukkan jumlah waktu training yang digunakan untuk setiap model yang menghasilkan gambar pada Gambar VI.1, VI.2, dan VI.3.

Tabel VI.2. *Training time* untuk masing-masing skenario sebelumnya

<b>ID</b>	<b>NGF</b>	<b>NDF</b>	<b>S_SIZE</b>	<b>Training Time (s)</b>	<b>Training Time (H:MM:SS)</b>
1	32	32	500	586.0322123	0:09:46
2	64	64	500	1021.844561	0:17:02
3	64	128	500	2790.996404	0:46:31
4	128	64	500	1729.198487	0:28:49
5	128	128	500	3557.511505	0:59:18
6	32	32	1500	1838.337111	0:30:38
7	64	64	1500	2909.312079	0:48:29
8	64	128	1500	9269.438068	2:34:29
9	128	64	1500	4830.129345	1:20:30
10	128	128	1500	11109.74546	3:05:09

Berdasarkan Tabel VI.2, dapat terlihat bahwa S\_SIZE yang lebih kecil memiliki waktu *training* yang lebih cepat. Hal yang menarik adalah bagian dengan NGF

dan NDF yang tidak seimbang, yang mana pada ID 3, 4, 8, dan 9 terlihat bahwa model dengan penggunaan NDF yang lebih besar memerlukan waktu latih yang lebih tinggi. Penggunaan waktu yang lebih tinggi ini juga berhubungan dengan gambar hasil *generator* yang dihasilkan pada Gambar VI.3, yang mana model dengan NDF yang lebih besar menghasilkan gambar dengan kualitas lebih baik. Berdasarkan beberapa hasil tersebut, maka dapat disimpulkan bahwa peran dari *discriminator* sangat diperlukan untuk bisa membuat *generator* menghasilkan gambar yang baik.

Hasil lengkap dari eksperimen VI.1 tersedia di google drive dengan masing-masing dalam file format zip.

<https://drive.google.com/drive/folders/1qJ4igSl1s2oKNhAF-xq557IUwqv65cIw?usp=sharing>

## VI.2. Eksperimen untuk nilai LR

Learning Rate (LR) merupakan sebuah angka yang mempengaruhi seberapa cepat perubahan bobot di dalam model dilakukan pada proses *backpropagation*. Semakin kecil nilai dari LR maka perubahan bobot akan semakin kecil, namun jika terlalu kecil maka akan terjebak pada *local optimal*. Jika nilai LR terlalu besar maka perubahan bobot akan besar dan rawan terjadi gradient swing, yang mana model tidak akan pernah mencapai titik optimalnya. Berdasarkan hal tersebut, memilih LR yang tepat sangat berpengaruh pada kinerja model.

Eksperimen mencari model terbaik berdasarkan LR diujikan dengan menggunakan konfigurasi NGF dan NDF model terbaik pada eksperimen sebelumnya. Pada eksperimen sebelumnya LR bernilai tetap, sedangkan pada eksperimen ini nilai LR bervariasi antara 2e-2, 2e-3, 2e-4, dan 2e-5 dan nilai NGF dan NDF tetap yaitu 128. Hasil dari eksperimen ini dapat dilihat pada Gambar VI.4 berikut.



Gambar VI.4. Hasil eksperimen untuk LR yang berbeda yaitu a) LR 2e-2; b) LR 2e-3; c) LR 2e-4; dan d) LR 1e-4

Berdasarkan hasil eksperimen dengan menggunakan LR berbeda seperti yang ditampilkan pada Gambar VI.4, terlihat bahwa LR yang besar yaitu 2e-2 di bagian a) menghasilkan gambar yang tidak konvergen, bahkan tidak mampu untuk menangkap informasi dari data latih. Untuk LR 2e-3 di bagian b) terlihat model sudah menangkap warna namun sisi-sisi yang membentuk objek masih belum terlihat. Untuk LR 1e-4 di bagian d) hasilnya cukup baik dengan beberapa gambar seperti zebra yang mulai terlihat, namun masih belum jelas untuk bentuk dari binatangnya. Hasil terbaik diperoleh dengan menggunakan 2e-4 pada bagian c) yang mana ini memperlihatkan warna dan bentuk objek, terutama pada gambar

*penguin*, *elephant*, *giraffe*, dan *zebra*. Untuk waktu latih yang dibutuhkan ditunjukkan pada Gambar VI.5 berikut.

#### Version History

	2e-2	2h ago
	Save & Run All • Diff: +1 -1	...
	Ran in 1 hour and 36 minutes	
	1e-4	2h ago
	Save & Run All • Diff: +1 -1	...
	Ran in 1 hour and 37 minutes	
	2e-3	11h ago
	Save & Run All • Diff: +0 -0	...
	Ran in 1 hour and 37 minutes	
	2e-4	11h ago
	Save & Run All • Diff: +630 -0	...
	Ran in 1 hour and 38 minutes	

Gambar VI.5. Waktu *training* dengan LR yang berbeda

Berdasarkan Gambar VI.5, terlihat bahwa waktu *training* dengan LR yang berbeda tidak menimbulkan perbedaan waktu yang signifikan, oleh karena itu LR tidak terlalu mempengaruhi waktu. Hasil untuk eksperimen dengan menggunakan LR 2e-2, 2e-3, 2e-4, dan 2e-5 tersedia di Kaggle, saya menggunakan Kaggle karena Google Colab-nya sudah mencapai batas penggunaan.

<https://www.kaggle.com/code/linkgish/conditional-image-generation-cdcgan-lr-ex.ipynb>

#### VI.3. Eksperimen untuk nilai NZ

Selain nilai dari S\_SIZE, NGF, NDF, dan LR, terdapat satu hal lagi yang membuat cDCGAN ini menarik, bagian jumlah informasi yang digunakan sebagai input pada *generator*, yaitu *latent space* (NZ). Jumlah NZ pada eksperimen kali ini bervariasi antara 20, 100, dan 200, pada beberapa eksperimen sebelumnya jumlah NZ yang digunakan adalah tetap pada angka 100. Eksperimen ini ditujukan untuk mempelajari pengaruh dari NZ terhadap gambar yang dihasilkan oleh generator. Untuk konstanta yang digunakan, yaitu S\_SIZE 1500, NGF 128, NDF 128, dan LR 2e-4. Berikut merupakan hasil dari eksperimen ini.



Gambar VI.6. Hasil eksperimen untuk NZ yang berbeda yaitu a) 10; b) 20; c) 100; dan d) 200

Berdasarkan hasil eksperimen yang ditunjukkan pada Gambar VI.6, terlihat bahwa hasil generasi dengan nilai NZ 10 menghasilkan gambar yang masih belum terlalu jelas, informasi dari gambar tidak terlihat dengan baik, ini bisa disebabkan karena NZ yang terlalu kecil. Pada hasil dengan NZ 100 dan 200, terlihat bahwa gambar juga kurang jelas, meski beberapa detail sudah lebih baik, ini bisa disebabkan karena informasi yang disimpan pada NZ terlalu banyak sedangkan epoch yang dilakukan masih sedikit. Hasil terbaik diperoleh pada NZ 20, dimana warna cukup jelas dan bentuk juga sudah cukup terlihat.

## Version History

	NZ = 100	2h ago
	Save & Run All • Diff: +1 -1	...
	Ran in 1 hour and 36 minutes	
	NZ = 10	2h ago
	Save & Run All • Diff: +1 -1	...
	Ran in 1 hour and 35 minutes	
	NZ = 200	4h ago
	Save & Run All • Diff: +1 -1	...
	Ran in 1 hour and 37 minutes	
	NZ = 20	4h ago
	Save & Run All • Diff: +3 -1	...
	Ran in 1 hour and 36 minutes	

Gambar VI.7. Waktu *training* dengan NZ yang berbeda

Untuk perbandingan dari segi waktu *training* seperti ditunjukkan pada Gambar VI.7, terlihat bahwa waktu yang dibutuhkan tidak terlalu berbeda untuk NZ 10, 20, 100, maupun 200. Hal ini bisa disimpulkan bahwa perbedaan dari jumlah NZ tidak mempengaruhi waktu dari proses training namun sangat berpengaruh dengan output hasil generator. Hasil untuk eksperimen dengan menggunakan NZ 10, 20, 100, dan 200 tersedia di Kaggle.

<https://www.kaggle.com/linkgish/conditional-image-generation-cdcgan-nz-exp>

## **VII. Penutup**

Bagian penutup pada penelitian ini memiliki dua poin utama yaitu kesimpulan hasil penelitian dan saran untuk pengembangan yang selanjutnya.

### **VII.1. Kesimpulan**

Berdasarkan beberapa eksperimen dan analisis yang dilakukan, penelitian ini menghasilkan beberapa kesimpulan sebagai berikut.

1. Penambahan jumlah data, yaitu dengan penggunaan S\_SIZE 1500 menghasilkan hasil yang lebih baik daripada S\_SIZE 1500.
2. Penggunaan NGF dan NDF yang tidak seimbang memberikan hasil yang kurang baik, dan pada arsitektur yang digunakan pada penelitian ini peran dari jumlah NDF lebih dominan daripada NGF. Terbukti dengan eksperimen NGF=64, NDF=128 yang memberikan hasil lebih baik daripada NGF=128, NDF=64.
3. Hasil dari eksperimen untuk nilai S\_SIZE, NGF, dan NDF diperoleh hasil terbaik untuk nilai S\_SIZE 1500, NGF 128, dan NDF 128.
4. Hasil eksperimen untuk menentukan LR terbaik, pada epoch ke-100 diperoleh hasil bahwa LR 2e-4 menghasilkan gambar yang lebih baik dari pada 2e-2, 2e-3, dan 1e-4. Jika LR terlalu tinggi maka model tidak bisa konvergen sedangkan jika LR terlalu rendah maka model akan butuh waktu lama untuk konvergen.
5. Hasil eksperimen untuk jumlah *latent space* (NZ) menunjukkan bahwa pada penggunaan NZ 20 memperoleh hasil yang lebih baik daripada 10, 100, dan 200. Ini menandakan bahwa jumlah NZ mempengaruhi hasil *generator* dan perlu untuk diketahui nilai NZ yang paling optimal.
6. Model terbaik yang dihasilkan pada eksperimen ini dengan epoch 100 dan ukuran gambar 64x64x3 adalah ketika menggunakan S\_SIZE 1500, NGF 128, NDF 128, LR 2e-4, dan NZ 20.

## **VII.2. Saran**

Keterbatasan dari mesin komputasi menjadi salah satu hambatan pada penelitian ini, sehingga percobaan tentang pengaruh data yang lebih besar masih belum bisa dilakukan. Terdapat beberapa saran yang diberikan untuk penelitian lebih lanjut. Diantaranya perlu dilakukan eksperimen untuk jumlah kelas yang berbeda serta pengaruhnya dengan penggunaan jumlah *latent space*. Penelitian ini juga masih sebatas penerapan *image generation*, perlu adanya penelitian lebih lanjut tentang penggunaan data hasil generasi pada penelitian ini untuk mendukung *task classification* pada data.

## DAFTAR PUSTAKA

- Dabberger J., 2023. Zoo animals. Object detection dataset of animals using Pascal VOC labeling format - XML files. <https://www.kaggle.com/datasets/jirka-dabberger/zoo-animals>
- Jordan, M.I. and Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), pp.255-260.
- Kostrikov, I., Yarats, D. and Fergus, R., 2020. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*.
- Gregor, K., Danihelka, I., Graves, A., Rezende, D. and Wierstra, D., 2015, June. Draw: A recurrent neural network for image generation. In *International conference on machine learning* (pp. 1462-1471). PMLR.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. *Communications of the ACM*, 63(11), pp.139-144.
- Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.
- Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Mustapha, I.B., Hasan, S., Nabus, H. and Shamsuddin, S.M., 2022. Conditional deep convolutional generative adversarial networks for isolated handwritten arabic character generation. *Arabian Journal for Science and Engineering*, 47(2), pp.1309-1320.

Zhu, F., He, M. and Zheng, Z., 2020. Data augmentation using improved cDCGAN for plant vigor rating. Computers and Electronics in Agriculture, 175, p.105603.

## LAMPIRAN

### 1. Dataset

Zoo Animals: <https://www.kaggle.com/datasets/jirkadaberger/zoo-animals>

### 2. GitHub Repository

Repository: <https://github.com/rizquuula/cDCGAN-ZooAnimals>

### 3. Google Colab

S\_SIZE, NGF, dan NDF experiment:

<https://drive.google.com/drive/folders/1qJ4igSl1s2oKNhAF-xq557IUwqv65cIw?usp=sharing>

### 4. Kaggle Notebook

LR experiment:

<https://www.kaggle.com/code/linkgish/conditional-image-generation-cdcgan-lr-exp>

NZ experiment:

<https://www.kaggle.com/linkgish/conditional-image-generation-cdcgan-nz-exp>

### 5. Hasil terbaik dengan S\_SIZE 1500, NGF 128, NDF 128, LR 2e-4, dan NZ 20

