Menghasilkan Background Game Music dengan Menggunakan Deep Convolutional Generative Adversarial Network

Daniel Widjojo, Henry Novianus Palit, Alvin Nathaniel Tjondrowiguno
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: widjojo.daniel98@gmail.com, hnpalit@petra.ac.id, alvin.nathaniel@petra.ac.id

ABSTRAK

Salah satu faktor penting dalam menghasilkan game yang baik adalah dengan adanya *Background Game Music* (BGM) yang baik. Namun sulitnya mendapatkan *asset* musik *game* membuat *game develop2er* harus mengeluarkan biaya, waktu, dan tenaga ekstra untuk mendapatkan musik *game* dengan kualitas yang baik. Hal ini dapat menghambat dalam proses pembuatan *game*. Untuk itu apabila terdapat program yang mampu menghasilkan BGM dengan baik maka akan sangat membantu pekerjaan dari *game developer*.

Metode yang digunakan pada penelitian ini adalah *Deep Convolutional Generative Adversarial Network* (DCGAN). Data yang digunakan adalah *file Musical Instrument Digital Interface* (MIDI) yang kemudian akan diubah ke dalam bentuk *pianoroll*. *Pianoroll* tersebut kemudian akan diubah menjadi matriks dan masuk pada model DCGAN. Sebelum melakukan *training*, diperlukan pembuatan proses *preprocessing, postprocessing* serta model dari DCGAN itu sendiri. Pengujian dilakukan dengan mencari parameter dan arsitektur terbaik pada DCGAN untuk menghasilkan *Background Game Music* dengan kualitas yang baik.

Hasil pengujian menunjukkan bahwa DCGAN merupakan model yang sangat sensitif terhadap arsitektur dan *hyperparameter*, oleh karena itu perlu perhatian ekstra dalam melakukan *tuning* arsitektur dan *hyperparameter* pada DCGAN. Selain itu musik yang diubah kedalam bentuk *pianoroll* kurang dapat menonjolkan fiturnya sehingga sulit untuk dipelajari oleh DCGAN. Dari hasil yang ada, DCGAN mampu menghasilkan *Background Game Music* namun dengan kualitas yang buruk.

Kata Kunci: *Deep Convolutional Generative Adversarial Network*, *Music Generation*, Tensorflow, *Keras*, *Pianoroll*, *Background Game Music*.

ABSTRACT

One of many important factor in producing a good quality game is the existence of a good quality Background Game Music (BGM). But the difficulty of getting game music assets makes game developers have to pay extra money, waste more time and effort to get a good quality game music. This can hinder the process of making a game. For that reason, if there is a program that is able to produce BGM with good quality, it will greatly help the work of the game developer.

The method used in this study is the Deep Convolutional Generative Adversarial Network (DCGAN). The data that being used is Musical Instrument Digital Interface (MIDI) file format which will then be converted into a pianoroll format. The pianoroll will then be converted into a matrix and entered into the DCGAN model. Before conducting training process, it is necessary to make a preprocessing, postprocessing and a model DCGAN. Testing in this study is done by finding the best parameters and architecture of DCGAN to produce Background Game Music with good quality.

The test results show that DCGAN is a very sensitive model in terms of architecture and hyperparameter, therefore it needs extra attention in tuning architecture and hyperparameter for DCGAN. Besides that, music that is converted into pianoroll format lacks the ability to highlight its features and making it difficult to learn by DCGAN. From the end results, DCGAN is able to produce Background Game Music but with poor quality.

Keywords: *Deep Convolutional Generative Adversarial Network*, *Music Generation*, Tensorflow, *Keras*, *Pianoroll*, *Background Game Music*.

1. PENDAHULUAN

Musik merupakan penunjang penting game, *game developer* pasti ingin agar *game*-nya baik dengan musik yang baik. Untuk mendapatkan musik ada 2 cara, yang pertama adalah mencari gratis di internet. Hal ini mudah dilakukan namun seringkali musik yang ditemukan monoton. Cara yang kedua adalah membeli musik secara *online* atau menyewa *music composer*. Hal ini akan membebani *game developer* dalam hal *cost* yang harus dikeluarkan. Masalah serupa disampaikan oleh game developer di Surabaya yaitu Regulus Studio [12] dan *Maulidan Games* [8]. Keduanya sering mengalami kesulitan dalam mencari BGM yang gratis dan berkualitas. Karena memakan banyak waktu, maka seringkali developer membeli musik dengan harga kurang lebih 10 *US Dollar* di internet. Untuk harga *music composer* di Indonesia cukup mahal karena harganya menggunakan standar Internasional. Masalah diatas dapat diselesaikan jika *Artificial Intelligence* (AI) mampu dimanfaatkan untuk menghasilkan BGM secarta otomatis. Dasar masalah yang diangkat dari penelitian ini adalah mengurangi beban biaya dari *developer game* dalam pembuatan BGM. Penelitian tentang menghasilkan musik menggunakan AI sudah banyak dilakukan sebelumnya [2]. Penelitian ini melakukan

automatic music composition dengan *text based Long-Short Term Memory* (LSTM). Hasilnya cukup baik namun metode tersebut tidak dapat mendeteksi *drum track*. Penelitian dalam menghasilkan musik secara otomatis juga pernah dilakukan sebelumnya [6]. Peneliti menggabungkan 2 *layer* dari LSTM dan *Deep Q Learning* (DQN). Metode ini membutuhkan waktu *training* yang lama dalam menghasilkan *output* yang baik. Penelitian serupa juga pernah dilakukan sebelumnya [4]. Peneliti melakukan generasi musik dengan menggunakan *Generative Adverserial Network* (GAN) Hasilnya sangat baik, metode tersebut mampu mendeteksi ke 5 instrumen dengan baik. Tetapi kualitas musik yang dihasilkan tidak terlalu baik jika dibandingkan dengan musik yang diproduksi oleh manusia. Demi menyelesaikan masalah tersebut maka penelitian ini berfokus untuk menghasilkan musik menggunakan (DCGAN), DCGAN sendiri dipilih karena merupakan salah satu desain GAN yang paling populer dan mudah untuk digunakan[5].

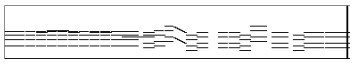
2. DASAR TEORI

2.1 *Background Game Music*

BGM adalah musik yang mengisi atau melengkapi sebuah *video game*. Musik tersebut dimainkan sebagai latar dalam permainan. Biasanya musik dimainkan untuk beberapa saat lalu selanjutnya akan diulang dari awal, hal ini dilakukan agar dapat menghemat ukuran dan menghemat waktu dalam pembuatan musik. Jenis musik dalam game memiliki jenis yang bermacam – macam tergantung dari suasana maupun skenario yang sedang terjadi dalam game. Berikut merupakan beberapa contoh tema suasana yang sering ditemui dalam *game* : *Mystery*, *Warning*, *Combat*, *Chase / fast movement*, *Victory*, *Walking*, *Title Screen* [11]. Yang terpenting adalah selalu menggunakan musik yang lebih menarik dari pada suasana atau momen yang sedang berlangsung pada layar pemain[11].

2.2 *Pypianoroll*

Pianoroll adalah representasi musik secara simbolis yang merekam adanya nada dalam setiap *time step* sebagai matriks biner. Nada direpresentasikan secara vertikal dan waktu/*time* step direpresentasikan secara horisontal.



Gambar 1. Contoh pianoroll

Mutlitrack pianoroll adalah sebuah list dari *pianoroll*, dimana setiap *track* merepresentasikan 1 instrumen musik. Gambar 1 merupakan contoh representasi dari pianoroll. Pypianoroll merupakan *library* bahasa *python* untuk merepresentasikan *file* musik MIDI menjadi *multitrack pianoroll* [3].

2.3 DCGAN

DCGAN merupakan pengembangan dari *algoritma* GAN. Perbedaan yang utama pada DCGAN dengan *original* GAN adalah pada DCGAN ditambahkan *layer Convolutional Neural Network* (CNN) pada *discriminator* sebagai *classifier* dari data yang dihasilkan oleh *generator*[10]. Berikut merupakan arsitektur dari DCGAN :Mengganti *pooling layers* pada *discriminator* dengan *strided convolutions* dan pada *generator* dengan *fractional-strided convolutions*; Menggunakan *batch normalization* baik pada *generator* maupun *discriminator*; Menghapus *fully connected hidden layer* untuk arsitektur yang lebih dalam; Menggunakan ReLU *activation* untuk semua *layer* pada *generator* kecuali *layer* output yang menggunakan *Tanh*; Menggunakan *Leaky* ReLU *activation* pada untuk semua *layer* pada *discriminator*.

3. DESAIN SISTEM

3.1 *Dataset*

Data yang digunakan dalam penelitian ini berupa musik *background* yang digunakan pada *game* mulai pada tahun 1996 hingga 2019. Semua data diperoleh secara *online* dengan mengunduh dari halaman *vgmusic.com*. *Website* ini mempunyai koleksi BGM yang beragam dari berbagai *platform* seperti *Nintendo*, *Sega*, *Sony*, *Microsoft*, *NEC*, *SNK*, *Atari*, *Computer*, hingga *platform game arcade*. Ekstensi dari *file* musik yang didapat berupa .mid sebanyak kurang lebih 30.000 *file* dengan durasi, instrumen, dan *genre* yang berbeda – beda. Karena pengambilan data dilakukan secara acak dan masal maka banyak musik yang sama karena ada beberapa *game* yang tersedia dalam beberapa *platform*.[9]

3.2 *Preprocessing*

Pypianoroll digunakan dalam memproses pada MIDI *files* sebelum masuk kedalam DCGAN. Pypianoroll nantinya juga digunakan setelah proses *postprocessing* pada untuk mengubah *pianoroll* menjadi MIDI *files*. MIDI *files* akan dimasukkan ke library Pypianoroll untuk mengubah MIDI Files menjadi bentuk objek *multitrack*. Selanjutnya instrumen yang ada pada MIDI akan diurutkan berdasarkan *program* yang terdapat disetiap *track*. Prioritas pengurutan yang digunakan adalah yang pertama *track* dengan tipe drum dan yang kedua sesuai dengan urutan program/jenis instrument. Selanjutnya panjang dan jumlah instrumen pada setiap *pianoroll* akan disesuaikan dengan ketentuan yang telah diatur sebelumnya. Tahap yang selanjutnya adalah *Binarize*, pada tahapan ini *pianoroll* akan diubah dalam bentuk biner sehingga *pianoroll* hanya memiliki 2 simbol yaitu angka 0 dan 1. Tahap ini memanfaatkan fungsi *binarize* pada *library* dari Pypianoroll. Tahap yang terakhir adalah *Matrix*, tahap ini akan mengubah bentuk dari *pianoroll* menjadi matriks 3 dimensi. Bentuk dari matriks 3 dimensi yang dimaksud adalah dimensi X merepresentasikan waktu, dimensi Y merepresentasikan tinggi nada/*pitch* dan dimensi Z merepresentasikan bunyi setiap instrumen/*track* yang digunakan. Selanjutnya matriks yang sudah melalui tahap ini akan masuk kedalam DCGAN.

3.3 *Postprocessing*

Proses ini berlangsung setelah mendapatkan *input* berupa matriks 3 dimenasi yang merupakan hasil *training* dari DCGAN. Matriks yang dihasilkan oleh DCGAN akan diubah kembali kedalam bentuk *pianoroll*. Tahap ini akan memanfaatkan *library* dari Pypianoroll untuk mengubah matriks 3 dimensi menjadi bentuk *pianoroll*. Setiap matriks akan dipotong sesuai dengan jumlah instrumen yang ada, kemudian potongan matriks tersebut akan dibuatkan masing masing sebuah *object* yang disebut dengan *track*. Semua *track* instrumen tadi akan disatukan menjadi 1 *pianoroll*. Selanjutnya *pianoroll* tersebut akan diubah

kembali menjadi bentuk MIDI *file* dengan menggunakan fungsi *write* pada *library* Pypianoroll.

3.4 DCGAN

Ada beberapa 3 tahapan utama yang dilakukan pada proses DCGAN diantaranya adalah *Generator*, *Discriminator*, dan *Calculate Loss and Update with Adam Optimizer*. *Generator* menggunakan *deconvolution layer* atau *convolution transpose layer*. Hal ini berarti mengubah dimensi dari *input* data menjadi dimensi yang lebih besar. *Generator* menerima *input* berupa *noise* yang kemudian akan melalui *dense*, *batch normalization*, RELU dan *reshape layer*. Untuk selanjutnya *noise* tersebut akan melalui bagian selanjutnya yaitu *convolution transpose layer* 2 dimensi, *batch normalization* dan RELU secara berulang hingga 4 bagian. Pada bagian *layer* yang paling akhir, *noise* tersebut akan melalui sebuah *convolution transpose layer* 2 dimensi dengan *Tanh* sebagai *activation function* dari *layer* tersebut. Output dari generator ini adalah sebuah matriks 3 dimensi yang selanjutnya akan disebut dengan data palsu. Selanjutnya data palsu ini akan dimasukkan kedalam *discriminator* untuk mendapatkan penilaian *decision* dari *discriminator*.Data yang menjadi input dari *discriminator* akan masuk pada bagian pertama yang terdiri dari *layer convolutional 2 dimensi*, *Leaky* RELU, dan *dropout*. Hal yang sama akan dilalui oleh data hingga 5 kali, hanya saja perbedaan ada pada bagian 4 dan 5. Pada bagian 4 dan 5 terdapat tambahan *layer* yaitu *batch normalizatio*n. Selanjutnya pada bagian akhir, data akan melalui *layer flatten* dan *dense* sehingga menghasilkan 1 nilai berupa *decision*. Jika data yang dimasukkan adalah data asli, seharusnya *discriminator* menghasilkan nilai mendekati 1. Sebaliknya jika data yang dimasukkan adalah data palsu maka yang ideal adalah *discriminator* menghasilkan nilai mendekati 0. Setelah *generator* dan *discriminator* menerima *input* data, maka *loss* dari *discriminator* akan dihitung. *Generator* berusaha agar data palsu yang dihasilkannya dinilai seperti data asli maka nilai *loss* akan dihitung dengan melihat seberapa dekat *decision* untuk data palsu dengan nilai 1. *Discriminator* berusaha membedakan data palsu dan data asli dengan benar maka *loss discriminator* merupakan nilai dari seberapa dekat *decision* untuk data asli dengan nilai 1 dijumlah dengan nilai dari seberapa dekat *decision* untuk data palsu dengan nilai 0. Selanjutnya nilai *loss* dari *generator* dan *disciminator* akan menjadi *gradient* untuk *adam optimizer* melakukan *update* bobot pada masing – masing model. *Learning r*ate yang digunakan adalah sebesar $1x10^{-4}$ pada kedua *adam optimizer* baik *generator* maupun *discriminator*.

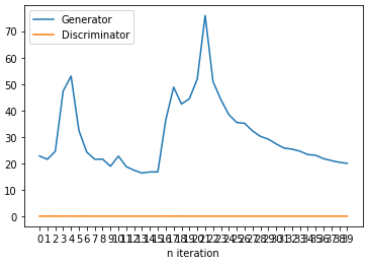
4. PENGUJIAN SISTEM

Pengujian ini dimaksudkan untuk mendapatkan arsitektur model yang paling stabil serta dapat menghasilkan BGM dengan kualitas baik. Pengujian dilakukan dengan memanfaatkan *Google Colab Notebook* yang memiliki spesifikasi *GPU NVIDIA Tesla K80* dengan *memory* sebesar 12 *GB*. Selain itu *Google Colab* juga dilengkapi dengan *RAM* sebesar 25 *GB* dan *temporary disk* sebesar 68 *GB*. Limitasi yang terdapat pada *Google Colab* adalah waktu proses dengan kuota sebesar 6 jam, setelah lebih dari kuota maka proses akan diputus

4.1 Model DCGAN *Conv2D*

4.1.1 Tanpa Penentuan Instrumen

Ukuran data yang dipakai adalah panjang waktu/*ticks* sebesar 5600 dan jumlah *track*/instrumen sebanyak 12. Nilai dari jumlah instrumen didapatkan melalui *density* dari jumlah instrumen keseluruhan data. Arsitektur *generator* yang digunakan pada pengujian ini memiliki 6 *layer* berulang yang terdiri dari Conv2D *transpose*, *batch normalization*, dan RELU. Sedangkan pada *input* menggunakan *layer dense*, *batch normalization*, *reshape*, dan RELU sebagai fungsi aktivasi. Arsitektur *discriminator* yang digunakan pada pengujian ini adalah 6 *layer* dengan 1 layer merupakan *layer output*. Tiga *layer* pertama terdiri dari *Conv2D*, *Leaky* RELU, dan *dropout* sebesar 0,3, sedangkan 2 *layer* berikutnya ditambahkan *batch normalization* setelah layer *Conv2D*. Pada *layer output* terdapat *layer flatten* dan *dense* dengan dimensi sebesar 1 yang digunakan untuk menghasilkan *decision*. Proses *training* berlangsung selama 11,9 jam dengan *epoch* sebesar 40. Data yang diproses sebesar 5000 data yang dibagi kedalam 16 *batch*.



Gambar 2. Grafik *loss* tanpa penentuan Instrumen

Grafik *loss* pada Gambar 2 terdapat beberapa lonjakan tajam pada *loss generator* selama proses *training*, hal ini menunjukkan bahwa model yang digunakan masih belum stabil. Garis pada *loss discriminator* tetap lurus yang berarti *discriminator* tidak belajar dengan baik. Pada pengujian ini DCGAN memberikan hasil yang kurang baik, *pianoroll* yang dihasilkan terdapat banyak *noise* sehingga tidak dapat di dengarkan. DCGAN tidak mampu menangkap pola dari data MIDI selama proses *training*.

4.1.2 Pengujian dengan Penentuan Instrumen

Pengujian dengan menentukan instrumen dilakukan dengan ukuran data yang lebih kecil dari pengujian yang sebelumnya. Untuk panjang waktu/*tick* mengambil nilai sebesar 1000 yang dibagi kedalam 16 *batch*, sedangkan instrumen diambil sejumlah 10. Penentuan jenis instrumen menggunakan proporsi 7:3 dimana 7 merupakan instrumen non perkusi dan 3 merupakan instrumen perkusi. Penentuan 10 jenis instrumen berdasarkan instrumen yang paling sering digunakan dalam keseluruhan data. Jenis instrumen yang terpilih adalah 0, 30, 33, 48, 56, 80, 81 untuk non perkusi dan 0, 16, 24 untuk instrumen perkusi.

4.1.2.1 Pengujian Parameter Default

Pada *Generator* terdapat 6 *layer* berulang ditambah dengan 1 *layer input* dan 1 *layer output*. *Discriminator* pada pengujian ini memiliki 7 *layer* berulang ditambah dengan 1 *layer* sebagai *output*. Pada pengujian ini diperlukan waktu *training* selama 5,6 jam untuk 100 *epoch*.

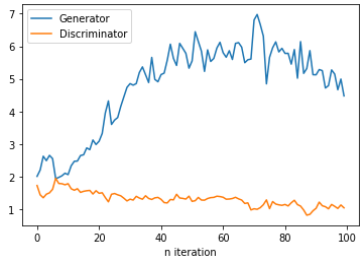
Gambar 3. Grafik *loss paremeter default*

Terlihat melalui grafik Gambar 3 pada iterasi awal terjadi proses belajar yang ditandai dengan nilai *loss discriminator* yang cukup tinggi mendekati *loiss generator*. Untuk *loss generator*, tidak terlalu banyak terdapat *spike* tajam. Grafik ini menandakan bahwa proses *training* berjalan dengan baik, nilai *loss* dari *generator* pun terlihat stabil dengan *trend* yang naik. Untuk akurasi dari *discriminator* pada model ini masih tergolong tinggi yaitu berada dikisaran 90%. Hal ini menunjukkan bahwa *generator* masih belum mampu mengelabuhi *discriminator* dengan baik. Hasil *pianoroll* pada *epoch* yang ke 100 untuk pengujian parameter *default* dapat dikatakan buruk. Instrumen – instrumen yang ada bermain sendiri – sendiri tanpa ada hubungan dengan instrumen yang lainnya. Nada yang dimainkan pun juga asal berbunyi dan memiliki durasi sangat singkat/putus - putus, hampir sama seperti orang yang baru pertama kali belajar memainkan alat musik. Pada pengujian ini tempo diberikan secara manual karena model DCGAN tidak mampu menghasilkan variasi tempo. Namun model DCGAN dapat mengenali pola – pola umum pada data walaupun nadanya tidak tepat.

4.1.2.2 Pengujian dengan Improved Technique

Dalam pengujian ini ditambahkan beberapa *trick* yang dapat membuat GAN menjadi lebih stabil [1]. Beberapa poin – poin yang diterapkan pada pengujian ini adalah sebagai berikut :

Penggunaan *Gaussian distribution* dalam membuat *noise*, Pemanfaatan *layer batch normalization* pada *generator* dan *discriminator*, Pengunaan *activation function Leaky* RELU pada *generator* dan *discriminator*, Penambahan *smooth labeling* pada *loss function discriminator* untuk data *real*, dan Penggunaan *layer dropout* pada *generator* sebanyak 50% baik untuk fase *training* maupun *test*.

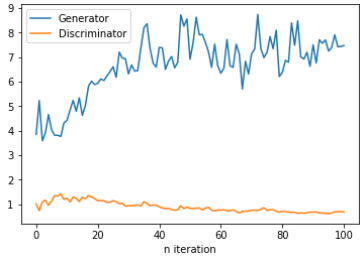


Gambar 4. Grafik *loss improved technique*

Dapat dilihat pada Gambar 4 bahwa proses *training* dari pengujian ini berjalan dengan lancar. Garis dari *generator* terlihat stabil dengan *trend* yang cenderung naik lalu kemudian sedikit turun sekitar *epoch* 80 sampai 100. Garis pada grafik tidak terlalu banyak memiliki *spike* bila dibandingkan dengan pengujian sebelumnya yang menggunakan *default parameter*. Secara keseluruhan baik *generator* dan *discriminator* memiliki nilai *loss* yang lebih rendah dari pengujian *default parameter*. Untuk akurasi dari *discriminator*, berkisar diangka 87%. Walaupun angka 87% ini terlihat tinggi namun hal ini menandakan bahwa *generator* mulai membuat *discriminator* semakin sulit membedakan data yang asli dan yang palsu. Sedangkan *discriminator* juga tidak boleh mendapatkan akurasi yang terlalu rendah, sehingga *discriminator* dan *generator* dapat saling belajar dengan cara saling bersaing. Pada hasil *pianoroll* terlihat bahwa model mampu menangkap beberapa pola – pola *pianoroll* yang terdapat pada data asli. Hasil pola terlihat jelas namun pola yang dihasilkan kasar seolah – olah masih tercampur dengan *noise*. Saat didengarkan, lagu masih terdengar kurang baik karena belum terdengar adanya kolerasi antar instrumen. Setiap instrumen bermain sendiri – sendiri secara acak tanpa ada harmonisasi. Walaupun durasi setiap nada yang dimainkan tidak terputus – putus.

4.1.2.3 Pengujian dengan Decreasing Dropout Rate

Arsitektur *generator* pada pengujian ini Terdapat 7 *layer* berulang yang digabungkan dengan 1 *layer input* dan 1 *layer output*. Untuk arsitektur *discriminator* memiliki 6 *layer* berulang dengan tambahan 1 *layer output*. Dalam pengujian ini setiap berjalan 20 *epoch*, *dropout value* pada *generator* yang dimulai dari 0,9 akan diturunkan sebanyak 0,2 hingga *epoch* terakhir memiliki *dropout value* sebesar 0,1. Pengujian ini dilakukan sebanyak 100 *epoch* dengan lama waktu 5,4 jam.

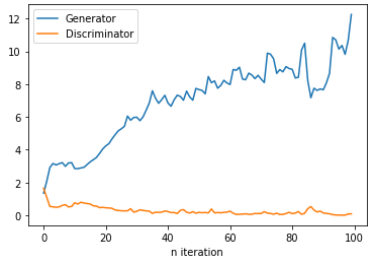


Gambar 5. Grafik *loss decrasing dropout rate*

Pada grafik Gambar 5, Nilai *loss* dari *generator* cukup tinggi. *Generator* memiliki *trend* yang naik dengan *spike* yang tidak terlalu tajam namun cukup sering terjadi. Hal ini menandakan bahwa proses *training* pada generator kurang stabil. Sedangkan untuk *discriminator* berjalan dengan stabil dan baik. Akurasi discriminator dalam pengujian ini terlihat cukup tinggi yaitu berikisar diantara 90%. Hal ini berari bahwa *generator* kurang mampu untuk mengelabuhi *discriminator* dalam proses *training*. *Pianoroll* yang dihasilkan memiliki kualitas yang kurang baik karena masih terdapat banyak *noise* pada *pianoroll*. Karena memiliki terlalu banyak *noise* maka *pianoroll* ini tidak dapat didengarkan. Walaupun tidak dapat didengarkan, namun dapat dilihat beberapa *track* telah mampu menangkap pola – pola berupa garis dan titik seperti data *pianoroll* yang asli.

4.1.2.4 Pengujian dengan All Layer Dropout

Pada pengujian *all layer dropout* kali ini model *generator* yang digunakan memiliki 5 *layer* berulang dengan 1 *layer input* dan 1 *layer output*. Untuk model *discriminator* terdapat 6 *layer* berulang dan 1 *layer output*.



Gambar 6. Grafik *loss all layer dropout*

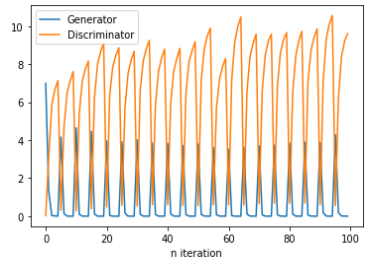
Pada grafik *loss* Gambar 6 pengujian ini terlihat bahwa proses *training* berjalan dengan baik dan stabil. Namun masih terdapat beberapa *spike* pada loss *generator epoch* 80 hingga 100. Pada awal *epoch*, terlihat *loss generator* dan *discriminator* yang berdekatan, hal ini berarti *generator* dan *discriminator* saling bersaing dalam memahami data. Sekitar *epoch* 15, *discriminator* mulai menungguli *generator* secara mutlak. Hal ini lah yang menyebabkan *trend loss* dari *generator* cenderung naik. Akurasi *discriminator* dalam pengujian ini termasuk sangat tinggi yaitu dikisaran 97%. Hal ini menandakan bahwa selama proses *training*, *generator* gagal untuk mengelabuhi *discriminator* dengan data yang dihasilkan. Hasil *pianoroll* tidak terlalu ada banyak pola, seolah – olah *generator* kurang berani dalam melukiskan polanya pada *pianoroll*. Seperti pada percobaan sebelumnya, musik yang dihasilkan kurang baik. Semua instrumen bermain sendiri – sendiri secara acak tanpa adanya harmonisasi. Durasi nada yang dihasilkan terdengar putus – putus

4.1.3 Dengan Menggunakan Skip Update

Pengujian selanjutnya dilakukan dengan melewati proses *update* bobot pada *discriminator* selama proses *training*. Untuk parameter yang digunakan dalam pengujian ini adalah dengan menggunakan *improved technique* dan *all dropout*. Pengujian *skip update* ini dilakukan karena melihat melalui pengujian sebelumnya bahwa grafik *loss* dari *discriminator* terlalu rendah jika dibandingkan dengan *generator*, hal ini menandakan bahwa *discriminator* terlalu kuat sehingga *generator* sulit untuk menipu *discriminator*. Pada pengujian ini data, model, dan parameter masih sama dengan pengujian sebelumnya yang membedakan adalah pada proses training *discriminator* hanya akan mendapatkan *update* bobot saat *epoch* kelipatan yang ditentukan yaitu 2 dan 5.

4.1.3.1 Skip Update Kelipatan 5

Untuk pengujian *skip update* dengan kelipatan 5 ini, parameter yang digunakan adalah *all dropout.*

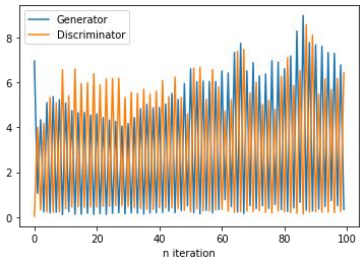


Gambar 7. Grafik *loss skip update* 5

Pengujian ini memiliki epoch sebesar 100 yang memakan waktu selama 5,9 jam. Dapat dilihat melalui grafik Gambar 7, terjadi osilasi yang cukup tinggi. Hal ini terjadi dikarenakan pada saat *discriminator* tidak dilakukan *update* bobot maka *loss* dari *generator* rendah dan *loss* dari *discriminator*, sebaliknya pada saat *epoch* pada kelipatan 5 *discriminator* akan mendapatkan *update* bobot sehingga *loss* dari *generator* tinggi dan *loss* dari *discriminator* menjadi rendah. Untuk akurasi *discriminator* secara perlahan menurun hinggak ke kisaran 55%. Hal ini dikarenakan *discriminator* tidak terlalu banyak belajar sehingga *generator* mampu menipu *discriminator* dalam membedakan data yang asli dan yang palsu. Untuk hasil *pianoroll* yang dihasilkan dapat dikatakan cukup baik, model mampu menghasilkan pola – pola yang cukup jelas. Jika *pianoroll* ini didengarkan masih terdengar aneh dan terputus – putus. *Pianoroll* masih kekurangan harmonisasi antar alat musiknya.

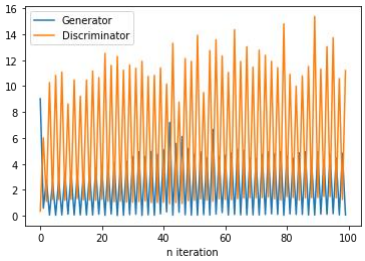
4.1.3.2 Skip Update Kelipatan 2

Untuk pengujian *skip update* kelipatan 2, parameter yang digunakan ada 2 yaitu *all dropout* dan *improved technique.*



Gambar 8 Grafik *loss skip update* kelipatan 2 *all dropout*

Proses *training* untuk parameter *all dropout* berlangsung selama 6,1 jam dengan *epoch* sebesar 100. Dapat dilihat pada Gambar 8 bahwa terjadi osilasi pada grafik *loss* untuk pengujian *skip update* kelipatan 2 dengan *all dropout*. Osilasi yang dihasilkan lebih rapat bila dibandingkan dengan pengujian dengan *skip update* kelipatan 5, hal ini wajar karena *discriminator* hanya dilakukan *update* bobot pada *epoch* dengan kelipatan 2. Untuk akurasi *discriminator* pada pengujian ini berada pada kisaran 75%, hal ini cukup baik bila dibandingkan dengan pengujian tanpa *skip update* yang memiliki *akurasi discriminator* jauh lebih tinggi dari pengujian ini. Hasil pianoroll pada pengujian ini terlihat cukup baik. Terlihat banyak pola yang dihasilkan model pada hasil pianoroll. Ketika pianoroll didengarkan, setiap instrumen masih tidak mampu berharmonisasi secara bersama, hasil musik yang dihasilkan masih terdengar putus – putus namun tidak separah pada pengujian dengan *skip* 5.

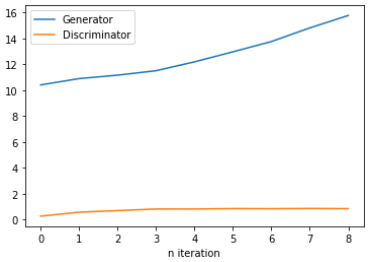


Gambar 9 Grafik *loss skip update kelipatan* 2 *improved technique*

Pada bagian ini menggunakan parameter improved technique dengan epoch sebesar 100 yang memakan waktu selama 5,3 jam. Seperti pada pengujian *skip update* yang lain, pada pengujian ini melalui Gambar 9 dapat dilihat grafik *loss* yang dihasilkan juga terjadi osilasi. Namun menariknya *loss discriminator* pada pengujian ini tidak memiliki osilasi yang tinggi seperti pada pengujian yang lain. Untuk akurasi pada pengujian ini berada pada kisaran 65%, hal ini menandakan bahwa *generator* berhasil menipu *discriminator* beberapa kali namun *discriminator* sendiri masih mampu membedakan data yang asli dan yang palsu. Grafik ini cukup baik karena baik *generator* dan *discriminator* dapat bersaing secara seimbang tanpa ada salah satu yang terlalu mendominasi. Hasil *pianoroll* dari *skip update* kelipatan 2 dengan *improved technique* terlihat sangat baik. *Pianoroll* memiliki banyak pola garis dan titik, seolah – olah model ini sudah lebih berani dalam menggambar pola – pola dari *pianoroll*. Jika didengarkan, hasil musik sudah tidak terdengar putus – putus, namun masih kurang adanya harmonisasi antar instrumen.

4.2 Model DCGAN *Conv3D*

Pada bagian ini pengujian model DCGAN akan melibatkan *layer Convolutional* 3 dimensi. Ukuran dari data yang dipakai sama dengan pengujian *Conv2D* dengan penentuan instrumen. Namun karena jenis *layer* yang digunakan adalah layer 3 dimensi maka diperlukan 1 dimensi tambahan yaitu *depth*. Pada pengujian ini besar *depth* yang ditetapkan adalah 5. Untuk instrumen yang digunakan sama seperti pengujian *Conv2D*. Data yang digunakan sebesar 1000 data yang dibagi kedalam 4 *batch size*. Model *generator* pada pengujian ini terdapat 5 *layer* berulang ditambah dengan 1 *layer input* dan 1 *layer output*. *Discriminator* pada pengujian ini memiliki 6 *layer* berulang dan 1 *layer output*.



Gambar 10 Grafik *loss Conv3D*

Pada grafik *loss* Gambar 10 dari pengujian ini, *generator* memiliki nilai yang tinggi dibandingkan dengan dengan pengujian yang lain. Pada *epoch* awal tidak terjadi perlawanan yang ketat antara *generator* dan *discriminator*, melainkan *generator* langsung memiliki nilai *loss* yang tinggi dan memiliki *trend* meningkat. Hal ini dapat menandakan bahwa *generator* dan *discriminator* tidak melakukan *learning* dengan baik selama proses *training* berlangsung. Terlihat melalui hasil *pianoroll* bahwa model *Conv3D* kurang mampu menangkap *feature* pada data. Hal ini dikarenakan model untuk *Conv3D* sesungguhnya tidak cocok dengan data pada pengujian ini. *Conv3D* biasanya digunakan untuk data *medical imaging* dan data *video* [7].

4.3 Pengujian Survei

Untuk mengetahui kualitas lagu lebih lanjut maka dilakukan sebuah survei dengan responden sebanyak 15 orang yang terdiri dari 6 *game developer*, 7 penggemar *game*, dan 2 penggemar musik. Lagu pengujian yang digunakan telah dipilih yang sekiranya paling baik dan dapat didengarkan oleh manusia. Hasil pengujian yang digunakan antara lain: *Conv2D* dengan penentuan instrumen parameter *default*; *Conv2D* dengan penentuan instrumen parameter *improved technique*; *Conv2D* dengan penentuan instrumen parameter *all dropout*; *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *improved technique*; *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *all dropout*; *Conv2D* dengan menggunakan *skip update* kelipatan 5 parameter *all dropout*

Berikut merupakan perolehan skor masing – masing dari keenam lagu pengujian pada survei:

Tabel 1. Penilaian skor terhadap 6 lagu pengujian melalui survei

Default	Improved	All dropout	Skip 2 Improved	Skip 2 All Dropout	Skip 5 All Dropout
5.2	5.47	5.2	7.27	5.47	5.6

Pada bagian pertama ini, responden diminta untuk memberikan skor penilaian dari angka 1 – 10 untuk setiap lagu pengujian. Dapat dilihat melalui Tabel 1 bahwa rata – rata tertinggi dari penilaian 15 responden model pengujian *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *improved technique* mendapatkan nilai tertinggi yaitu 7,27

Tabel 2. Penilaian dalam mengurutkan lagu terbaik

Default	Improved	All dropout	Skip 2 Improved	Skip 2 All Dropout	Skip 5 All Dropout
2.73	2.67	3	5.2	3.8	3.6

Pada bagian kedua ini, responden diminta untuk mengurutkan lagu mana yang terbaik dengan angka 6 merupakan lagu terbaik dan angka 1 merupakan lagu terburuk. Berdasarkan dengan perhitungan Tabel 2 maka hasil yang terbaik menurut responden adalah model pengujian *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *improved technique* dengan nilai tertinggi yaitu 5,2

4.4 Analisa Pengujian

Berdasarkan pada hasil pengujian dapat disimpulkan bahwa model yang paling stabil adalah model pengujian *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *improved technique*. Model tersebut memiliki *range loss* yang tinggi jika dibandingkan dengan model lain, namun akurasi *discriminator* cukup stabil. *Discriminator* tidak terlalu kuat mendominasi proses *training* sehingga *generator* memiliki banyak kesempatan untuk belajar. Untuk hasil *pianoroll* terlihat punya pola yang jelas dan bervariasi. Berdasarkan pengujian disimpulkan bahwa parameter yang paling berdampak adalah *standard deviation* untuk menghasilkan *gaussian noise* dan *dropout layer*. Nilai *standard deviation* berdampak pada seberapa bervariasi angka acak yang dihasilkan untuk *input*. Jika variasi angka yang berikan berkisar antara 0 dan 1 maka hal ini akan meringankan pekerjaan *generator*. *Dropout* berfungsi untuk memberikan stabilisasi pada model, *dropout* dapat menyaring *noise* berlebih pada *input*. Namun filter yang terlalu kuat akan membuat pola semakin memudar. Dalam DCGAN, proses *training* yang stabil dan seimbang perlu diperhatikan. Melalui

pengujian dapat disimpulkan bahwa *discriminator* yang terlalu mendominasi dapat membuat proses *training* menjadi tidak stabil.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan pada sistem, maka dapat disimpulkan bahwa

- DCGAN mampu menghasilkan BGM namun dengan kualitas yang kurang baik.
- Arsitektur dan parameter yang terbaik adalah pada pengujian *Conv2D* dengan menggunakan *skip update* kelipatan 2 parameter *improved technique*
- Berdasarkan dari hasil pengujian, model DCGAN merupakan model yang sensitif terhadap hyperparameter, arsitektur dan stabilisasi *training*.
- *Layer dropout* dapat membantu *generator* untuk lebih stabil. Namun terlalu banyak menggunakan *dropout* membuat hasil pola dari *pianoroll* semakin menghilang.
- Penggunaan *noise* dengan *gaussian distribution* memiliki performa lebih baik dari pada menggunakan *noise* biasa pada model DCGAN.
- Proses *training* yang stabil dan seimbang sangat berpengaruh terhadap performa DCGAN. Baik *discriminator* maupun *generator* harus seimbang
- Bentuk *pianoroll* kurang menonjolkan *feature* dari musik sehingga tidak dapat ditangkap model DCGAN dengan baik.
- DCGAN kurang mampu mendapatkan *feature* berupa pola pada dimensi yang terlalu besar.
- Performa DCGAN lebih baik pada saat memiliki arsitektur yang dalam dibandingkan dengan arsitektur yang tidak terlalu dalam namun memuliki dimensi yang besar.
- Untuk melakukan pemrosesan dalam bidang musik dibutuhkan *resource power* yang kuat, karena sulit untuk melakukan *resize* pada musik.

5.2 Saran

Saran yang dapat diberikan untuk menyempurnakan dan mengembangkan program ini lebih lanjut antara lain:

- Menggunakan *optimizer* selain ADAM pada *discriminator* misalnya seperti SGD. Berdasarkan pengujian, kecepatan model saat belajar dapat mempengaruhi proses *training*.
- Menggunakan model *Hybrid* GAN selain DCGAN seperti *Style*-GAN, *Wasserstein* GAN, dan BigGAN yang merupakan metode GAN terbaru. Hal ini disarankan karena metode – metode tersebut memiliki arsitektur yang lebih kompleks dari pada DCGAN.
- Menggunakan representasi musik selain *pianoroll* seperti *wave* dan *spectogram*. karena *pianoroll* kurang dapat menonjolkan *feature* musik.

6. REFERENCES

[1] Chintala, S., Denton, E., Arjovsky, M. & Mathieu, M., 2020. *How to Train a GAN? Tips and tricks to Make GANs Work.* (5 March 2020). URI= https://github.com/soumith/ganhacks.

[2] Choi, K., Fazekas, G. & Sandler, M., 2016. Text-based LSTM Networks for Automatic Music Composition. pp. 1-8, 2016. DOI= https://arxiv.org/abs/1604.05358.

[3] Dong, H.-W., Hsiao, W.-Y. & Yang, Y.-H., 2018. Pypianoroll: Open Source Python Package for Handling Multitrack Pianorolls. *in Late-Breaking Demos of the 18th International Society for Music Information Retrieval Conference (ISMIR),* pp. 1-2. pp. 1-2, 2018. DOI= https://salu133445.github.io/pypianoroll/

[4] Dong, H.-W., Hsiao, W.-Y., Yang, L.-C. & Yang, Y.-H., 2018. MuseGAN: Multi-Track Sequential Generative. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 34-41, 2018. DOI= https://arxiv.org/abs/1709.06298

[5] J. Hui, Hui, J., 2018. *GAN — DCGAN (Deep convolutional generative adversarial networks),* (18 Juny 2018). URI= https://medium.com/@jonathan_hui/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f.

[6] Kotecha, N., 2018. Bach2Bach: Generating Music Using A Deep Reinforcement Learning Approach. 2018. DOI= ttps://arxiv.org/abs/1812.01060

[7] Lane, T., 2018. *1D & 3D Convolutions explained with… MS Excel!.* (17 October 2018). URI= https://medium.com/apache-mxnet/1d-3d-convolutions-explained-with-ms-excel-5f88c0f35941.

[8] Nararya, H. R., 2019. *Wawancara Bersama Maulidan Games.* [Interview]. (25 November 2019).

[9] Newman, M. et al., 2019. *VGMusic - Game Music MIDI files.* (16 September 2019). URI= http://vgmusic.com.

[10] Radford , A., Metz, L. & Chintala, S., 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings,* pp. 1-16, 2016. DOI= https://arxiv.org/abs/1511.06434

[11] Rogers, S., 2010. *Level Up! The Guide to Great Video Game Design.* Chichester: Wiley.

[12] Wibisono, A., 2019. *Wawancara Bersama Regulus Studio* [Interview] (23 November 2019).