BAB II

TINJAUAN PUSTAKA

2.1 Batik dan Motif Batik

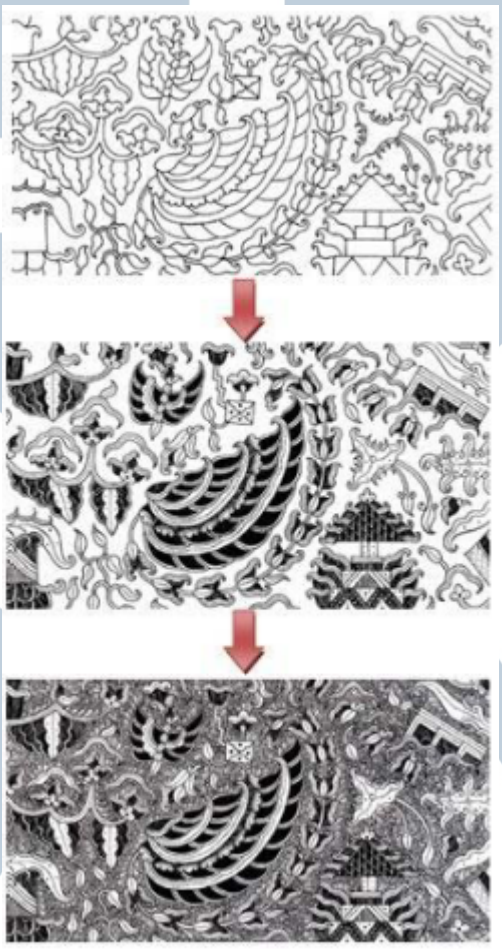
Batik adalah sebuah *wastra*, yaitu sebuah potongan kain yang dibuat secara tradisional dengan motif batik tertentu. Secara etimologi batik mempunyai makna dari akhiran "*tik*", dalam bahasa Jawa yang berarti "*mbatik*". Kata tersebut merupakan turunan dari kata menetes, disebut juga sebagai "menulis" atau menulis dengan lilin. "Mbatik" diartikan sebagai menulis atau menggambar dengan memperhatikan detail yang kecil [7].

Motif batik selain rumit dan indah juga memiliki makna simbolis. Motif tradisional mengambil tema dari alam dan lingkungan sekelilingnya sehingga motif yang dihasilkan dapat menggambarkan lingukungan dimana batik itu dibuat. Salah satu motif batik yaitu Kawung. Motif ini terinspirasi oleh pohon aren yang buahnya berwarna putih lonjong atau disebut kolang-kaling. Corak Ceplok adalah salah satu variasi dalam motif Kawung dengan perubahan-perubahan pada bulatannya menjadi segi-empat atau berbentuk bintang yang juga diatur secara geometris [8].

Motif adalah desain secara keseluruhan dari kain batik, sementara ornamen adalah bentuk objek yang berfungsi sebagai penghias atau pengisi. Ada beberapa motif batik yang terkenal di Yogyakarta salah satunya Parang. Ciri-ciri dari motif ini yaitu mempunyai pola hias bergaris miring dan berjajar [9]. Sebuah tekstil tradisional tidak bisa disebut batik hanya karena terdapat ornamen-ornamen yang tergambar pada kainnya karena batik tidak dapat terlepas dari proses *mbatik*. Dengan demikian, ornamen batik tidak dapat dipisahkan dari proses membuat batik. Mengamati batik tidak dapat hanya dari sekedar dekorasi pola batik melainkan keseluruhan proses mulai dari pembuatan sampai apresisasi estetika

batik itu sendiri. Proses pembuatannya terdiri dari beberapa langkah seperti pada gambar 2.1 dengan penjabaran seperti dibawah ini [10]:

1. Klowongan, proses menggambar elemen-elemen dasar dari keseluruhan desain.
2. Isen-isen, proses mengisi bagian-bagian dari gambar-gambar yang sudah selesai dari proses sebelumnya dengan beberapa motif.
3. Penempatkan latar belakang dari desain sebagai keseluruhan desain.



Gambar 2.1: Proses pembuatan batik : Klowongan (atas), Isen-isen (tengah), penempatan latar belakang (bawah).

Menilai batik secara keseluruhan, banyak produk memiliki motif-motif yang terinspirasi dari pola batik. Tetapi hal tersebut tidak bisa disebut batik apabila tidak termasuk tekstil dan tidak ada pekerjaan metode membatik didalamnya. Dengan

demikian dapat dibuat formulasi sederhana dari apa yang bisa disebut batik [10]:

1. Proses pembuatannya, dengan menggunakan canting, bisa cap atau tulis. Hal ini termasuk bahan-bahan yang digunakan dalam prosesnya;
2. Ornamen dekoratif seperti yang muncul di tekstil batik; dan
3. Apresesiasi dari penggunan batik.

2.2 Generative Adversarial Network

Generative Adversarial Network merupakan kerangka kerja untuk memperkirakan model generatif melalui proses adversarial yang secara bersama melatih dua model yaitu sebuah model generatif $\mathbb{G}$ yang menangkap distribusi data training dan sebuah model diskriminatif yang memperkirakan kemungkinan data yang datang bukan dari model $\mathbb{G}$. Prosedur pelatihan menghasilkan $\mathbb{G}$ untuk memaksimalkan kemungkinan $\mathbb{D}$ untuk membuat kesalahan. Model *Generative* dapat dianalogikan sebagai sebuah tim yang tugasnya untuk memalsukan uang, sementara model *discriminative* adalah tim yang harus mencari tahu uang palsu. Kompetisi diantara keduanya akan mengembangkan metode mereka sampai uang palsu tidak dapat dibedakan [5].

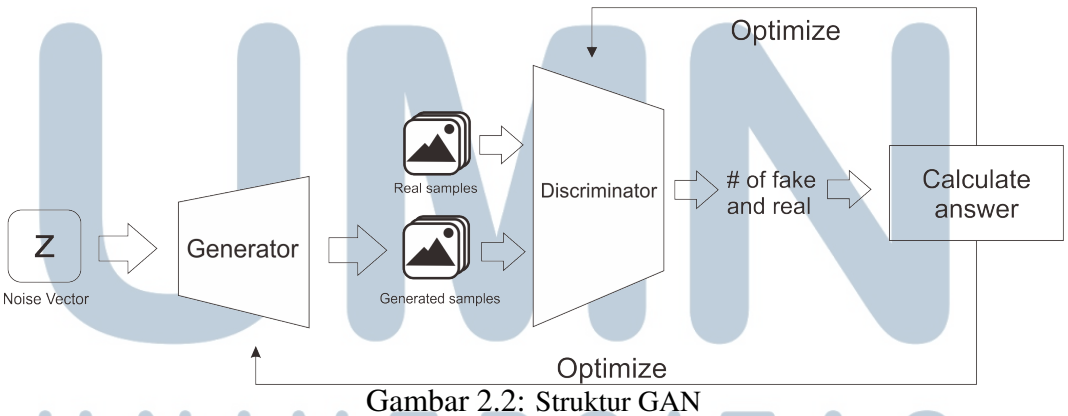
Dibandingkan dengan model generatif lain seperti *variational autoencoders*, gambar yang dihasilkan oleh *GAN* biasanya lebih realistis dan tidak terlalu blur. Hal ini juga terbukti secara teori bahwa *GAN* yang optimal dapat dicapai, *generator* dapat dengan sempurna menghasilkan gambar yang sesuai dengan persebaran dari gambar aslinya, dan *discriminator* tidak dapat membedakan gambar asli dengan hasil *generator* [11].

Dalam pengimplementasiannya, *training GAN* cukup sulit karena beberapa alasan yaitu, cakupan *network* terlalu sulit untuk dicapai. Kedua, *GAN* selalu masuk kedalam *mode collapse*, yang *generator* menghasilkan gambar yang sama

atau mirip untuk berbagai *noise vectors* $\vec{z}$. Berbagai macam ekstensi dari *GAN* telah dikembangkan untuk meningkatkan kestabilan *training* [11].

Penelitian yang sudah dilakukan oleh Alec Radford yang berjudul "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks" membahas tentang penggunaan *convolutional network* pada layer *generator* dan *discriminator* [6]. Hasil dari penelitian tersebut adalah sebuah arsitektur bernama *DCGAN* (*Deep Convolutional GANs*) dengan pedoman :

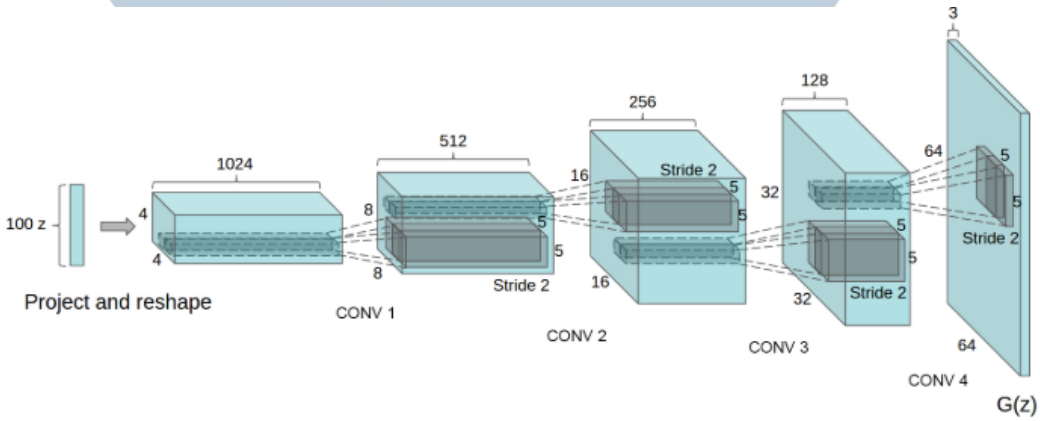
- Mengganti *pooling layer* dengan *strided-convolutions* untuk *discriminator* dan *fractional-strided convolutions* untuk generator;
- Menggunakan *bacthnorm* untuk kedua *generator* dan *discriminator*;
- Menghapus semua *hidden-layer*;
- Menggunakan fungsi aktivasi ReLU pada *generator* untuk semua *layer* kecuali untuk *output*; dan
- Menggunakan fungsi aktivasi LeakyReLU pada *discriminator* untuk semua *layer*.



Gambar 2.2: Struktur GAN

Input dari *generator* berupa *noise vector* yang kemudian akan didekonvolusi sehingga menjadi sebuah deret angka dengan *channel* dan ukuran tertentu. Angka-angka didalamnya merepresentasikan nilai dari satu *pixel* dan setiap *channel* merepresentasikan warna apa saja yang akan digunakan seperti

misalnya *RGB*. *Output* dari *generator* berupa gambar hasil dekonvolusi yang kemudian akan menjadi *input discriminator*. Ilustrasi *generator* bisa dilihat di gambar 2.3. *Discriminator* yang berupa *convolutional network* menerima *input* berupa hasil *output generator* dan data asli. Kemudian, *discriminator* akan mencoba untuk membedakan mana data asli dengan data *output generator*. Hasil dari proses membedakan ini akan menjadi parameter untuk mengoptimasi *generator* dan *discriminator*. Proses ini akan dilakukan berkali-kali dalam sebuah fungsi *train* didalam fungsi tersebut terdapat fungsi optimasi sehingga keduanya akan saling "belajar". Hasil akhirnya *generator* diharapkan mampu untuk menghasilkan keluaran yang menyerupai persebaran data asli dan *generator* dapat membedakan data asli dan hasil *generator* [6]. Ilustrasi DCGAN ada pada gambar 2.2.



Gambar 2.3: Ilustrasi struktur *generator* [6]

2.3 Tensorflow

Tensorflow merupakan sebuah antar muka untuk menuliskan algoritma *machine learning* dan implementasinya. Perhitungan menggunakan Tensoflow dapat dilakukan oleh banyak sistem mulai dari *mobile device* sampai *large-scale distributed system* yang terdiri dari banyak mesin dan *computational devices* seperti kartu grafis. Tensoflow bersifat fleksibel dan bisa digunakan untuk menuliskan sebagian besar algoritma termasuk *training* dan algoritma pengambil

keputusan untuk model *deep neural network* dan sudah digunakan untuk produksi penelitian *machine learning* dalam banyak bidang *computer science* dan bidang lain seperti *natural image processing*, *computer vision*, dan robotik [4].

2.4 Gradient Descent Optimization

Dalam hal *machine learning* diperlukan sebuah fungsi matematika untuk membuat sebuah *AI* dapat seolah-olah "belajar", fungsi tersebut disebut sebagai *objective function*. *Objective function* yang akan digunakan dalam penelitian ini yaitu algoritma *gradient descent optimization*.

Gradient descent optimization merupakan salah satu algoritma yang populer untuk melakukan optimasi dan algoritma yang umum digunakan untuk mengoptimasi *neural network*. *Gradient descent* merupakan cara untuk meminimalkan nilai dari *objective function* dengan *input* berupa parameter *model* [12].

Ada tiga macam *gradient descent* berdasarkan banyaknya data yang digunakan menghitung gradien dari fungsi objektif yaitu batch gradient descent, stochastic gradient descent, dan mini-batch gradient descent. *Batch gradient descent* menghitung *objective function* menggunakan keseluruhan data untuk sekali *update*, tetapi memungkinkan untuk konvergen ke global minimum atau ke local minimum. Stochastic gradient descent (SGD) menyelesaikan masalah komputasi *batch gradient descent* yang *redundant* dengan melakukan satu update setiap kali. *SGD* bersifat fluktuatif sehingga ada kemungkinan dapat lompat menuju *local minimum* yang lebih baik tetapi ada kemungkinan dapat melampaui itu. Hal ini bisa dicegah dengan perlahan mengurangi *learning rate* sehingga sifatnya menyerupai *batch gradient descent*. *Mini-batch gradient descent* mengambil keuntungan dari keduanya dan melakukan *update* untuk setiap *mini-batch* ukuran n dari setiap sampel *training*[12]. *RMSProp (Root Mean*

Square) optimizer merupakan fungsi optimasi *mini-batch SGD*. *RMSProp* dapat mencapai konvergen dengan cepat tanpa *overshoot* karena sifat *adaptive learning* yang secara perlahan mengurangi *learnnig rate* secara otomatis. *AdamOptimizer* merupakan fungsi pengembangan dari *RMSOptimizer* dengan tambahan momentum untuk mencapai konvergen lebih cepat. Penelitian *DCGAN* oleh Alec Radford [6] menggunakan dua *optimizer* yaitu *mini-batch SGD* dan *AdamOptimizer*. Penelitian ini menggunakan *RMSProp* untuk mengetahui bagaimana hasil dari metode *mini-batch batch gradient descent* dengan hasil keluaran dan mencoba menggunakan *AdamOptimizer* untuk mengetahui apakah dengan *Adam* dapat menghasilkan *output* yang lebih baik.

2.5 Penelitian Terkait

Berikut adalah beberapa hasil penelitian yang membahas terkait dengan peneitian ini:

1. Penelitian oleh Yohanes Gultom, dkk. Batik Classification Using Deep Convolutional Network Transfer Learning. Dalam penelitian tersebut dibahas tentang implementasi Deep Convolutional Neural Network untuk pengklasifikasian motif batik [2]. Penelitian tersebut menggunakan *dataset* motif batik Parang, Nitik, Kawung, Ceplok, dan Lereng yang akan digunakan kembali dalam penelitian ini sehingga dapat menghemat waktu untuk pencarian *dataset* motif batik;
2. Penelitian oleh Ian J. Goodfellow, dkk. Generative Adversarial Nets. Dalam penelitian tersebut dibahas mengenai sebuah kerangka kerja baru untuk memperkirakan model generatif dengan proses adversarial (*GAN*) [5] yang dijadikan sebagai landasan metode dalam penelitian ini;
3. Penelitian oleh Yang Zhou, dkk. Non-Stationary Texture Synthesis by Adversarial Expansion. Dalam penelitian tersebut dibahas tentang

penggunakan *GAN* untuk memperbesar tekstur dari sebuah gambar [13], penelitian ini membuktikan bahwa sintesis motif dengan *GAN* dapat dilakukan;

4. Penelitian oleh Xian Wu, dkk. A Survey of Image Synthesis and Editing with Generative Adversarial Networks. Dalam penelitian tersebut membahas tentang perbandingan *GAN* model generatif lain seperti *variational autoencoder*, *cGAN* (*Conditional GAN*), *DCGAN* (*Deep Convolutional GANs*), dan *LAPGAN* (*Laplacian GAN*) [11], dalam penelitian ini variasi DCGAN yang akan digunkan karena lebih mudah dipahami dan diimplementasikan oleh penulis; dan
5. Penelitian oleh Alec Radford dan Luke Metz. ”Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks”. Dalam *paper* tersebut dibahas mengenai implementasi *GAN* menggunakan *convolutional network* [6]. Hasil penelitian tersebut akan dijadikan sebagai landasan metode untuk perancangan *DCGAN* dalam penelitian menggunakan Python dan Tensorflow-gpu.