Analisis Kinerja LSTM dan GRU sebagai Model Generatif untuk Tari Remo

Lukman Zaman[1,3], Surya Sumpeno[1,2], Mochamad Hariadi[1,2]

***Abstract*— Creating dance animations can be done manually or using a motion capture system. An intelligent system that able to generate a variety of dance movements should be helpful for this task. The recurrent neural network such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) could be trained as a generative model. This model is able to memorize the training data set and reiterate its memory as the output with arbitrary length. This ability makes the model feasible for generating dance animation. Remo is a dance that comprises several repeating basic moves. A generative model with Remo moves as training data set should make the animation creating process for this dance simpler. Because the generative model for this kind of problem involves a probabilistic function in form of Mixture Density Models (MDN), the random effects of that function also affect the model performance. This paper uses LSTM and GRU as generative models for Remo dance moves and tests their performance. SGD, Adagrad, and Adam are also used as optimization algorithms and drop-out is used as the regulator to find out how these algorithms affect the training process. The experiment results show that LSTM outperforms GRU in term of the number of successful training. The trained models are able to create unlimited dance moves animation. The quality of the animations is assessed by using visual and dynamic time warping (DTW) method. The DTW method shows that on average, GRU results have 116% greater variance than LSTM's.**

***Intisari*— Pembuatan animasi tari bisa disusun secara manual atau menggunakan *motion capture*. Sebuah sistem cerdas yang dapat menghasilkan gerakan tari yang bervariasi bisa membantu proses ini. *Recurrent neural network* yang berupa *Long Short-Term Memory* (LSTM) atau *Gated Recurrent Unit* (GRU) dapat dilatih menjadi model generatif. Model ini akan menghafal data pelatihan dan mengeluarkan apa yang diingatnya dengan variasi yang tak terbatas panjangnya. Kemampuan ini sesuai untuk menghasilkan tarian yang terdiri atas perulangan gerakan-gerakan dasar. Tari Remo adalah tari yang memiliki banyak variasi dan tersusun dari gerakan-gerakan dasar yang pendek dan berulang. Bila tari ini dilatihkan menjadi model generatif, maka animasi tarian yang merupakan variasi dari gerakan-gerakan tari Remo menjadi mudah dihasilkan. Karena model generatif untuk tari Remo melibatkan penggunaan *Mixture Density Network* (MDN) yang berupa fungsi probabilitas, maka kinerja model ini juga terpengaruh efek acak (*random*). Makalah ini melakukan sejumlah percobaan menggunakan LSTM dan GRU yang dilatih untuk menjadi model generatif bagi gerakan tari Remo. Hasil percobaan memberi informasi seberapa besar kesuksesan proses pelatihan. Efek penggunaan teknik optimisasi SGD, *AdaGrad*, dan *Adam*, serta teknik regularisasi *drop-out* juga dicoba. Pada pelatihan yang sukses, variasi animasi tari yang dihasilkan dianalisis menggunakan *Dynamic Time Warping* (DTW). Hasil percobaan menunjukkan keberhasilan LSTM dalam pelatihan lebih besar daripada GRU. Perhitungan DTW menunjukkan bahwa gerakan yang dihasilkan GRU memiliki variasi 116% lebih besar daripada variasi hasil LSTM.**

***Kata Kunci*—LSTM, GRU, DTW, model generatif, tari Remo.**

I. PENDAHULUAN

Saat ini ranah seni sudah mulai banyak dijadikan sebagai objek bagi riset-riset yang menggunakan *deep learning*. Hal ini tidak terlepas dari terjangkaunya teknologi GPU yang memungkinkan penanganan proses yang memerlukan beban komputasi tinggi. Capaian yang hendak diraih riset-riset ini adalah dibangunnya sistem cerdas yang bertindak sebagai artis digital yang mampu menciptakan karya seni yang tidak terbedakan dengan karya buatan manusia. Di seni gambar dan lukisan, satu gaya lukisan telah dapat disalin ke lukisan yang lain [1], [2]. Terdapat pula sistem cerdas yang mampu membuat sketsa *doodle* [3] dan menciptakan foto dari coretan [4]. Karya musik dapat dihasilkan dengan bantuan *generative adversarial network* [5]. Di ranah seni tari, pembuatan gerakan tari dapat diatur lewat musik [6] dan video seseorang yang menari juga dapat dibuat dari video penari yang lain [7].

Beberapa karya seni direpresentasikan dalam data yang berurutan. Misalnya dalam musik, karya dapat dinyatakan sebagai rangkaian nada sepanjang dimensi waktu. Tarian juga dapat dinyatakan dalam rangkaian pose tubuh yang merentang sepanjang dimensi waktu. Dalam bentuk data sekuensial semacam ini, suatu informasi baru dapat diperoleh bila serangkaian data yang berurutan dapat diproses. Sebagai contoh, untuk menghasilkan gerakan tari, maka sistem harus memproses gerakan yang telah terjadi sebelumnya untuk dapat memprediksi gerakan selanjutnya. Untuk dapat menangani data berurutan semacam ini, diperlukan arsitektur khusus semacam *Hidden Markov Model* (HMM) atau *Recurrent Neural Netwok* (RNN).

RNN adalah struktur jaringan saraf tiruan (JST) yang menggunakan umpan balik, yaitu menggunakan keluarannya menjadi salah satu masukan bagi proses selanjutnya [8]. Sebenarnya, RNN secara konsep merupakan arsitektur yang sederhana. Namun, dalam praktiknya, cara paling optimal untuk melatih jaringan semacam ini adalah dengan melakukan *unrolling*. *Unrolling* akan mengurai perulangan dalam RNN

[1] *Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS, Sukolilo, Surabaya 60111, Jawa Timur, Indonesia (tlp: 031-5947302, fax: 031-5931237, e-mail: lukman12@mhs.ee.its.ac.id, surya@ee.its.ac.id, mochar@ee.its.acid)*

[2] *Departemen Teknik Komputer, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS, Sukolilo, Surabaya 60111, Jawa Timur, Indonesia (tlp: 031-5922936, email: surya@ee.its.ac.id, mochar@ee.its.ac.id)*

[3] *Departemen Teknik Informatika, Sekolah Tinggi Teknik Surabaya, Jl. Ngagel Jaya Tengah 73-77, Surabaya, Jawa Timur, Indonesia 60284 (tlp:031-5027920; e-mail: lz@stts.edu)*

menjadi untaian *clone* sepanjang jumlah perulangan. Dengan penguraian ini, bentuk RNN menjadi linear dan dapat dilatih menggunakan mekanisme *back propagation* biasa.

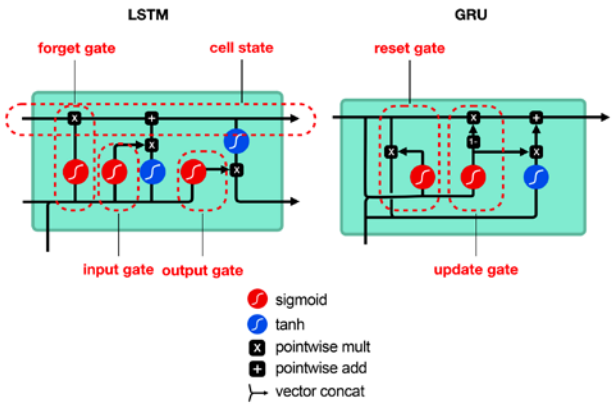
Jumlah perulangan dalam RNN berubah menjadi jumlah *layer* pada bentuk yang telah terurai. Bila jumlah *layer* lebih dari dua, maka bentuk ini dapat disebut sebagai *deep network*. Bila jumlah *layer* mencapai lima atau lebih, maka pelatihan akan menghadapi masalah *vanishing gradient*. Untuk mengatasi masalah ini, disusunlah arsitektur yang memanfaatkan sistem gerbang. Sistem gerbang yang telah menunjukkan keunggulan di berbagai riset adalah *Long Short-Term Memory* (LSTM) [9]. Sistem gerbang yang lebih baru dan mulai populer digunakan adalah *Gated Recurrent Unit* (GRU) [10].

Penggunaan RNN memiliki beberapa jenis skenario. Pada analisis sentimen misalnya [11], rangkaian data masukan digunakan untuk melakukan klasifikasi. Rangkaian masukan juga dapat menghasilkan rangkaian keluaran pada kasus penerjemah bahasa [12] dan *image captioning* [13]. RNN juga dapat digunakan sebagai model generatif [14]. Pada model ini, RNN dilatih untuk menghafalkan suatu *dataset*, lalu RNN disuruh mengeluarkan apa yang telah diingatnya sebagai keluaran. Pada skenario ini, keluaran dapat lebih panjang daripada masukan.

Berdasarkan keberhasilan RNN sebagai model generatif, maka dalam makalah ini dibangun model generatif untuk menghasilkan karya seni yang belum banyak ditangani, yaitu animasi tari. Tari Remo dipilih karena tari ini dapat dibagi menjadi gerakan-gerakan dasar yang relatif pendek dan berulang, sehingga dapat ditangani oleh batasan memori GPU yang ada saat ini. Walaupun teknologi GPU saat ini banyak membantu proses pelatihan agar lebih cepat, tetapi kompleksitas perhitungan yang diperlukan masih menjadi beban. Untuk waktu pelatihan yang optimal, pemilihan arsitektur RNN dan penggunaan teknik optimasi yang tepat perlu dilakukan. Makalah ini menggunakan dua macam arsitektur RNN, yaitu LSTM dan GRU. Kinerja LSTM dan GRU dibandingkan. Ukuran kinerja yang digunakan adalah tingkat keberhasilan LSTM dan GRU dalam proses pelatihan serta penilaian atas gerakan yang dihasilkan. Hal-hal yang memengaruhi kinerja, yaitu penggunaan beberapa algoritme optimasi dan regularisasi juga dicoba. Evaluasi hasil gerakan dilakukan secara visual dan dengan menggunakan perhitungan jarak *Dynamic Time Warping* (DTW). Hasil riset ini akan membantu pembentukan model generatif untuk tarian-tarian lainnya dan menuntun pemilihan arsitektur dan teknik optimisasi yang terbaik untuk proses pelatihannya.

II. MODEL GENERATIF UNTUK TARIAN

Tari Remo yang dijadikan *dataset* pelatihan direkam menggunakan perangkat *motion capture*. Digitalisasi tarian tradisional Indonesia dengan *motion capture* telah dilakukan untuk beberapa tarian, misalnya tari Incling, tari Merak, dan tari Yapong [15]. Format penyimpanan tari dalam bentuk *Biovision Hierarchy* (BVH) dipilih karena format ini telah menunjukkan kemudahan rekonstruksi dan juga kemudahan untuk kepentingan klasifikasi, analisis, dan pencarian [16]. Format ini berbentuk data sekuensial, jadi penanganannya memerlukan



Gbr. 1 Arsitektur LSTM dan GRU.

arsitektur khusus. HMM dan RNN memiliki kemampuan tersebut, tetapi RNN lebih fleksibel dalam memberikan variasi hasil. Sebelum digunakan untuk menangani tarian, RNN telah berhasil digunakan untuk menangani data sekuensial yang berupa teks [17], film [18], tulisan tangan [14],urutan tindakan dalam suatu permainan [19], dan musik [6].

Tarian merupakan *subset* dari gerakan manusia, jadi riset di ranah gerakan manusia dapat dijadikan dasar. Sebelum LSTM populer, RNN telah digunakan untuk menghasilkan gerakan. Arsitektur RNN yang digunakan adalah *fully connected* dan metode pelatihannya menggunakan algoritme genetik [20]. RNN pada sistem tersebut berhasil membuat animasi orang berjalan tanpa menggunakan contoh ataupun data pelatihan. *Deep learning* juga telah digunakan untuk membuat gerakan manusia dengan menggunakan *auto-encoder* [21]. Sistem ini membangun abstraksi dari gerakan yang berdimensi tinggi ke dimensi yang lebih sederhana. Dengan menelusuri parameter dimensi rendah dan memetakan kembali ke bentuk gerakan, berbagai gerakan yang berkisar pada data pelatihan dan kombinasinya dapat dihasilkan. Penelitian lain berhasil mengatur gerakan tari dengan panduan musik [22]. Pada metode ini, hanya gerakan dan musik dari *dataset* pelatihan yang dapat dihasilkan. Animasi tarian yang berupa video dapat dihasilkan dari video tarian lain yang dilakukan penari profesional [7]. Walaupun sistem ini menggunakan video sebagai masukan dan keluarannya, model kerangka semacam BVH tetap digunakan sebagai perkiraan pose. Model generatif yang juga menggunakan RNN untuk gerakan tari adalah *chor-RNN* [23]. Perbedaannya dengan riset ini adalah *chor-RNN* menggunakan data posisi *marker* langsung dari *motion capture* dan data tarian yang digunakan adalah tarian kontemporer.

III. LSTM DAN GRU

LSTM pertama kali dibuat oleh Sepp Hochreitern dan Jürgen Schmidhuber pada tahun 1997 [9]. LSTM juga disempurnakan oleh banyak orang, seperti Felix Gers, Fred Cummins, Santiago Fernandez, Justin Bayer, Daan Wierstra, Julian Togelius, Faustian Gomez, Matteo Gagliolo, dan Alex Graves.

LSTM menangani masalah *vanishing gradient* dalam pelatihan yang pasti terjadi di RNN dasar. Dalam LSTM, terdapat sebuah *cell state*. *Cell state* ini berfungsi sebagai memori atau ingatan untuk sebuah *layer*. Nilai *cell state*

dimanipulasi menggunakan sebuah *gating system* atau sistem gerbang. Sistem gerbang ini terdiri atas beberapa arsitektur JST sederhana untuk mengatur waktu sebuah data harus disimpan, digunakan, atau dilupakan.

Terdapat tiga jenis unit gerbang berbeda yang digunakan dalam LSTM, yaitu *input gate*, *forget gate*, dan *output gate*, seperti yang ditunjukkan pada Gbr. 1. *Input gate* berfungsi untuk menentukan sebuah masukan akan ditambahkan ke dalam memori *cell state* saat itu atau tidak. *Forget gate* berguna untuk menentukan sebuah memori pada waktu sebelumnya harus dilupakan atau tidak. Sedangkan *Output gate* berguna untuk menentukan seberapa besar pengaruh memori *cell state* terhadap hasil prediksi yang akan dihasilkan.

GRU adalah arsitektur yang diciptakan oleh Kyunghun Cho pada tahun 2014 [10]. Serupa dengan LSTM, GRU juga menggunakan sistem gerbang, seperti ditunjukkan pada Gbr. 1. Arsitektur GRU lebih sederhana daripada LSTM. GRU tidak menggunakan *cell state*, tetapi memanfaatkan *hidden state* untuk menyimpan informasi. *Reset gate* dalam GRU menentukan informasi baru harus dilupakan atau tidak, sedangkan *update gate* untuk mengingat.

Dari Gbr. 1 dapat dilihat bahwa LSTM memiliki tiga gerbang *sigmoid* dan dua gerbang *tanh*, sedangkan GRU hanya memerlukan dua *sigmoid* dan sebuah *tanh*. Perhitungan yang terlibat untuk setiap *update* LSTM ditunjukkan pada (1) sedangkan untuk GRU adalah (2).

$$\begin{aligned}
\widetilde{c_t} &= \tanh(W_c[a_{t-1}, x_t] + b_c) \\
G_u &= \sigma(W_u[a_{t-1}, x_t] + b_u) \\
G_f &= \sigma\big(W_f[a_{t-1}, x_t] + b_f\big) \\
G_o &= \sigma(W_o[a_{t-1}, x_t] + b_o) \\
c_t &= G_u * \widetilde{c_t} + G_f * c_{t-1} \\
a_t &= G_o * c_t
\end{aligned} \tag{1}$$

$$\begin{aligned}
\widetilde{c_t} &= \tanh(W_c[G_r * c_{t-1}, x_t] + b_c) \\
G_u &= \sigma(W_u[c_{t-1}, x_t] + b_u) \\
G_r &= \sigma(W_r[c_{t-1}, x_t] + b_r) \\
c_t &= G_u * \widetilde{c_t} + (1 - G_u) * c_{t-1} \\
a_t &= c_t
\end{aligned} \tag{2}$$

Karena GRU melibatkan perhitungan yang lebih sedikit daripada LSTM, secara teori GRU dapat dilatih lebih cepat daripada LSTM. Namun, pada praktiknya, untuk masing-masing kasus harus dicoba, agar diperoleh yang lebih cocok untuk menyelesaikan suatu masalah.

IV. Algoritme Optimisasi

Pelatihan RNN dapat dilakukan dengan menggunakan algoritme genetika [20]. Namun, saat ini pelatihan dilakukan dengan *backpropagation* seperti ANN pada umumnya. *Backpropagation* dilakukan untuk melakukan *update* berdasarkan fungsi *loss* yang merupakan fungsi dari bobot (*W*) dan *bias* (*b*). Bobot dan *bias* adalah parameter internal yang terlatihkan dalam ANN. Algoritme optimisasi dalam proses pelatihan adalah usaha untuk melakukan minimalisasi fungsi tersebut sehingga ANN dapat memetakan parameter yang mewakili fitur X ke target label Y.

Untuk melakukan minimalisasi fungsi *loss*, algoritme optimisasi menggunakan turunan pertama atau turunan keduanya. Turunan pertama yang biasa disebut dengan gradien adalah vektor yang *tangential*, yaitu menyentuh dan searah dengan permukaan fungsi *loss*. Karena ini adalah fungsi multivariabel, maka gradien dinyatakan dalam matriks Jacobian. Alternatif lain adalah menggunakan turunan kedua yang dinyatakan dalam matriks Hessian. Teknik optimisasi paling dasar yang menggunakan gradien adalah *gradient descent* (GD). Persamaan yang digunakan adalah $\theta = \theta - \eta.\nabla J(\theta)$, dengan η adalah *learning rate* dan $\nabla J(\theta)$ adalah gradien dari fungsi *loss* dari parameter θ. Kelemahan dari GD adalah lambat dan memerlukan banyak memori karena *update* dilakukan untuk keseluruhan *dataset* pelatihan.

Variasi dari GD adalah *stochastic gradient descent* (SGD). SGD melakukan satu *update* untuk setiap sampel pelatihan. Walaupun SGD dapat mempercepat dan mengurangi memori, tetapi metode ini mudah terombang-ambing oleh fluktuasi yang terjadi pada parameter internal. Untuk meredam fluktuasi, pelatihan dilakukan dalam suatu *mini-batch*, yaitu sekumpulan data pelatihan dilatih bersama. Besarnya *batch* yang optimal tergantung pada kasusnya (kasus per kasus). Secara umum, saat ini SGD sudah menggantikan GD (GD adalah SGD dengan ukuran *batch* sebesar *dataset* pelatihan).

Algoritme optimasi yang lain adalah *AdaGrad* [24]. Algoritme ini menggunakan *learning rate* yang beradaptasi berdasarkan parameter θ. *Learning rate* bernilai besar untuk nilai parameter yang relatif unik, dan bernilai rendah pada parameter yang sering muncul. Namun, karena *learning rate* makin lama makin rendah, terkadang algoritme ini bisa berhenti tanpa menemukan solusi optimal.

Alternatif yang lain adalah *Adaptive Moment Estimation* (*Adam*) [25]. Pada *Adam*, *learning rate* juga adaptif. Perubahan *learning rate* dihitung dari estimasi momentum. Momentum adalah nilai yang dihitung berdasarkan arah dari pelatihan sebelumnya. Dalam *Adam*, kuadrat gradien disimpan dan disimpan juga rata-rata gradien sebelumnya untuk perhitungan momentum.

V. Tari Remo dan Gerakannya

Tari Remo adalah tari tradisional dari Jawa Timur. Tari ini umumnya dipentaskan di awal kesenian Ludruk, atau di pembukaan acara-acara penting. Tari ini dapat ditarikan penari tunggal ataupun dalam grup. Walaupun tari ini menggunakan kostum pria, tetapi penari bisa dari gender apa pun. Gbr. 2 menunjukkan contoh pementasan tari Remo yang dilakukan oleh seorang penari perempuan.

Interpretasi paling umum untuk tari Remo adalah penggambaran tentang prajurit atau kesatria [26]. Namun, ada penafsiran bahwa sebenarnya tari Remo menggambarkan pengembaraan seseorang menuju kedewasaan spiritual [27]. Tafsir ini merujuk ke tari yang lebih kuno, yaitu tari Nglana [28]. Tafsir-tafsir ini yang membentuk gerakan tari Remo

Gbr. 2 Pementasan tari Remo.

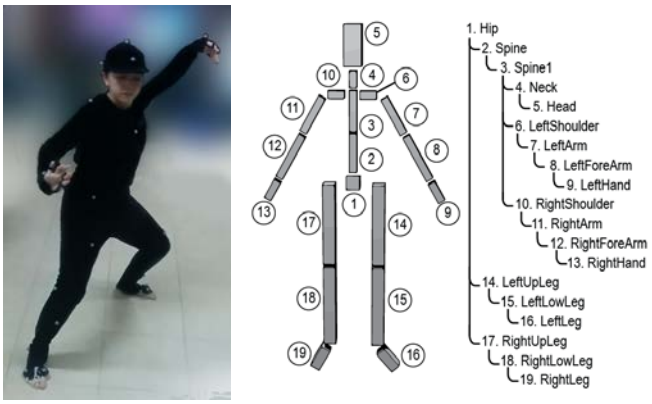
beserta variasi pengembangannya. Tari Remo sendiri saat ini memiliki beberapa variasi. Biasanya variasi-variasi ini dinamai dengan nama koreografer atau daerah asalnya. Beberapa variasi tari Remo adalah Remo Tubi, Remo Trisnawati, dan Remo Jombangan. Variasi yang lain adalah panjang tariannya. Remo bisa ditarikan selama 30 atau 12 menit. Untuk pemula, ada pula variasi pendek yang hanya berdurasi 4 menit [26].

Seperti tari Jawa pada umumnya, tari Remo disusun dari sederetan gerakan-gerakan dasar. Gerakan dasar ini secara tradisi memiliki nama dan definisi, tetapi saat ini belum ada dokumen yang secara formal mencatat gerakan tari Remo. Walaupun demikian, dari nama dan jenis gerakannya, tari Remo terlihat masih berhubungan dengan gerakan tari Jawa yang lain [29].

Kompleksitas gerakan-gerakan dasar tari Remo memiliki rentang variasi yang lebar. Ada gerakan-gerakan yang sederhana, semacam *tindak* (gerakan berjalan), *kawung* (gerakan menaikkan dan menurunkan tangan secara bergantian), dan *pacak gulu* (menggerakkan leher hingga kepala bergeser ke kiri dan kanan). Ada pula gerakan yang kompleks semacam *iket* (gerakan tangan yang menirukan gerakan mengikat sesuatu) dan *ayam alas* (gerakan menirukan ayam hutan). Beberapa gerakan bahkan tidak melibatkan gerakan tubuh, tetapi sekadar sikap atau pose. Sebagai contoh, gerakan *ngendewa* yang menirukan pose memanah.

VI. PROSES AKUISISI DATA

Data tarian Remo yang akan dilatihkan direkam dengan perangkat *motion capture* (*mocap*) Optitrack dengan delapan kamera. Varian tari yang dipilih adalah versi pendek 4 menit. Data asli dari *motion capture* dalam 60 *frame per second* (fps) dan berupa posisi *marker* yang dipasang pada seluruh tubuh penari, seperti terlihat pada Gbr. 3(a). Untuk penyederhanaan, data diubah menjadi 30 fps dalam format BVH. Format BVH menggunakan model tulang yang disusun dalam suatu hierarki untuk mendefinisikan tubuh. Struktur tulang yang digunakan diperlihatkan pada Gbr. 3(b). Dengan format ini, tarian lengkap akan tersimpan sebagai sederetan pose sebanyak 10.500 *frame*. Setiap *frame* mengandung pose yang dinyatakan dalam sehimpunan sudut rotasi Euler untuk setiap tulang pada Gbr. 3(b). Selain sudut rotasi, dalam setiap pose juga tersimpan



(a) (b)

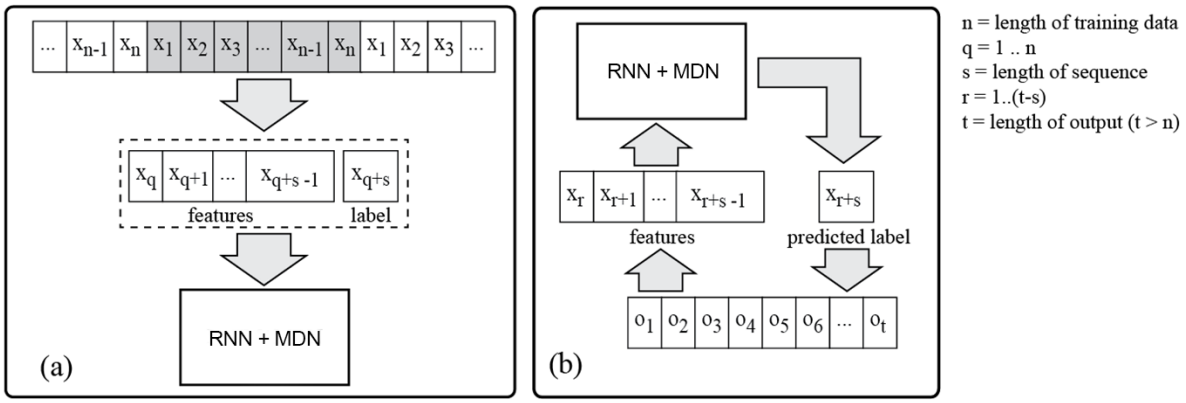
Gbr. 3(a) Perekaman tari dengan *motion capture*, (b) model tulang yang digunakan.

TABEL I
GERAKAN DASAR TARI REMO UNTUK *DATASET* PELATIHAN

Nama Gerakan	*Frame*	Keterangan
Tindak	95	Gerakan berjalan
Gejug kaki, *seblak* tangan	107	Menghentakkan kaki, menyabetkan tangan
Tanjak, gedrug kaki	95	Posisi kuda-kuda dengan menggerakkan tumit
Seblak tangan	145	Melempar tangan
Lawung kanan-kiri	180	Gerakan kedua tangan naik-turun bergantian
Geter kepala	72	Menggetarkan kepala ke kiri dan kanan
Seblak, gejug, tanjak	140	Urutan tiga gerakan diakhiri posisi kuda-kuda
Gejug kaki, *tindak*	137	Menghentakkan kaki lalu berjalan memutar
Selut	87	Menggerakkan kedua tangan naik turun di depan dada
Ukel trap jamang	188	Tangan memutar meniru gerakan memasang hiasan kepala
Iket (sebagian)	86	Bagian dari gerakan meniru tangan yang sedang mengikat sesuatu

sebuah posisi 3D untuk tulang pinggul (*hip*) yang berperan sebagai *root* dari model.

Karena definisi tradisional untuk gerakan dasar tari Remo sangat bervariasi tingkat kompleksitasnya, maka dalam percobaan ini definisi tersebut tidak sepenuhnya digunakan. Sebagai pengganti, gerakan dipilih dan dipotong sesuai gerakan yang akan membantu proses evaluasi. Daftar gerakan yang dijadikan *dataset* disajikan dalam Tabel I. Pada pilihan gerakan ini, bagian-bagian tubuh yang bergerak sedapat mungkin diisolasi dari bagian tubuh yang lain. Gerakan *tindak* misalnya, memfokuskan gerakan kaki, sedangkan *kawung* memiliki fokus pada bagian tangan. Kasus gerakan yang melibatkan seluruh anggota tubuh juga terwakili di gerakan *seblak-gejug-tanjak*. Dengan pemilihan data semacam ini, maka efek gerakan setiap anggota tubuh pada pelatihan lebih mudah diidentifikasi.



(a) (b)

Gbr. 4(a) Arsitektur untuk proses pelatihan, (b) proses *sampling*.

VII. MODEL GENERATIF UNTUK TARI REMO

Dengan menggunakan RNN, yaitu LSTM atau GRU, tiap gerakan dasar tari Remo dapat dimodelkan menjadi sebuah model generatif. Proses pembangunan model ini secara umum adalah sebagai berikut. Gerakan tari Remo yang berupa data sekuensial dilatihkan ke RNN. Agar data tarian dapat dilatihkan, data tari dipotong-potong menjadi sejumlah segmen dengan panjang *frame* tertentu. Potongan ini akan menjadi fitur dalam *dataset* pelatihan. Untuk setiap segmen potongan, satu pose yang terjadi sesudah gerakan segmen tersebut juga dilatihkan sebagai labelnya (deretan x_q pada Gbr. 4(a)). *Dataset* pelatihan yang berupa pasangan fitur (segmen potongan gerakan) dan label (pose sesudahnya) dibuat secara lengkap agar mewakili gerakan tari yang harus diingat oleh RNN. Artinya, dalam sebuah *dataset* yang lengkap, jumlah pasangan fitur dan label adalah sejumlah *frame* dalam segmen yang dipilih.

Pada proses *sampling* atau proses generasi tarian, RNN yang telah terlatih akan mengeluarkan apa yang diingatnya sebagai keluaran. Pada awalnya, RNN dipanasi dulu dengan memberi serangkaian gerakan sebagi masukan. Untuk setiap rangkaian masukan, RNN akan memprediksi sebuah pose sesudahnya sebagai keluaran. Bila keluaran ini digabungkan dengan masukan RNN selanjutnya, maka berikutnya RNN akan menebak pose selanjutnya lagi (deretan o_t pada Gbr. 4(b)). Proses umpan balik ini dilakukan terus menerus dan rangkaian keluaran yang dihasilkan akan ditulis sebagai gerakan tarian yang dihasilkan RNN. Panjang gerakan tarian yang dapat dihasilkan tidak terbatas. Tentu saja gerakan tarian yang dihasilkan tidak persis sama dengan gerakan pada *dataset* pelatihan. Variasi tarian yang tampak di keluaran ini akan dibahas pada pembahasan uji coba.

Hal penting dalam desain arsitektur ini adalah penambahan modul *Mixture Density Model* (MDN). MDN adalah model probabilistik yang memberi kemampuan pada RNN untuk menghasilkan keluaran yang berbeda dari masukan yang ambigu [30]. Contoh sederhana untuk masukan yang ambigu adalah pada fungsi invers dari *sinus*. Nilai *sinus* 0,5 dapat dihasilkan dari sudut 30^0 atau 150^0. Dalam data tarian, ambiguitas semacam ini sering terjadi. Dalam tarian terdapat gerakan yang diulang beberapa kali sebelum berganti ke gerakan yang lain. RNN harus dapat memprediksi, gerakan berikutnya harus melanjutkan perulangan atau berhenti dan berganti gerakan. MDN akan memberi keputusan dalam prediksi sesuai probabilitas data yang dilatihkan. Tanpa MDN, RNN akan menghasilkan nilai rata-rata dari *dataset* pelatihan sebagai keluaran [23].

VIII. UJI COBA DAN PEMBAHASAN

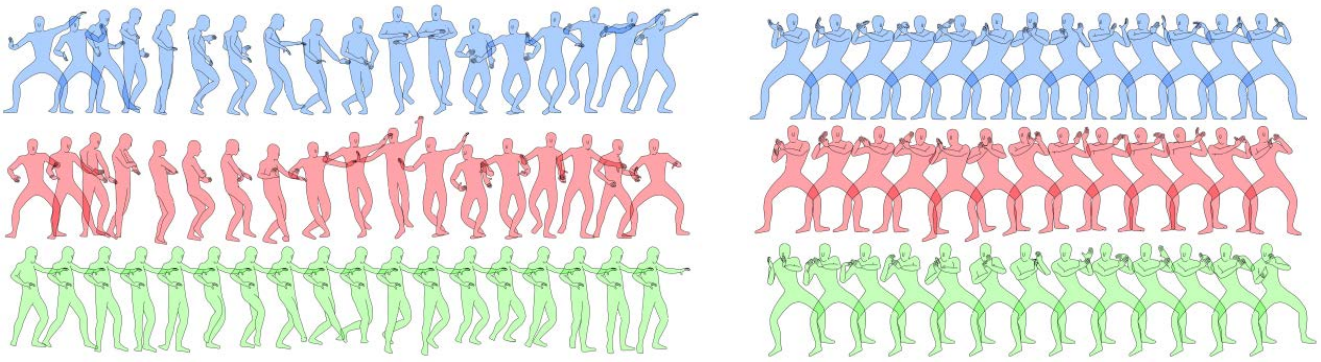
Proses pelatihan dilakukan untuk semua gerakan yang terpilih. Setiap gerakan dilatih pada model yang terpisah. Strategi dan parameter untuk pelatihan yang optimal menggunakan rujukan dari [31]. Hal-hal yang menjadi bahan evaluasi adalah tingkat keberhasilan, yaitu perbandingan antara jumlah pelatihan yang sukses dan yang gagal, dan kualitas gerakan yang dihasilkan.

Kriteria pelatihan yang gagal adalah model yang telah dilatih tetapi tidak dapat menghasilkan gerakan tarian. Gerakan yang dihasilkan oleh model yang gagal terlatih ada dua macam, yaitu gerakan acak atau pose diam. Di lain pihak, pelatihan yang berhasil akan menghasilkan gerakan yang mirip dengan tarian yang dilatihkan.

A. Evaluasi Secara Visual

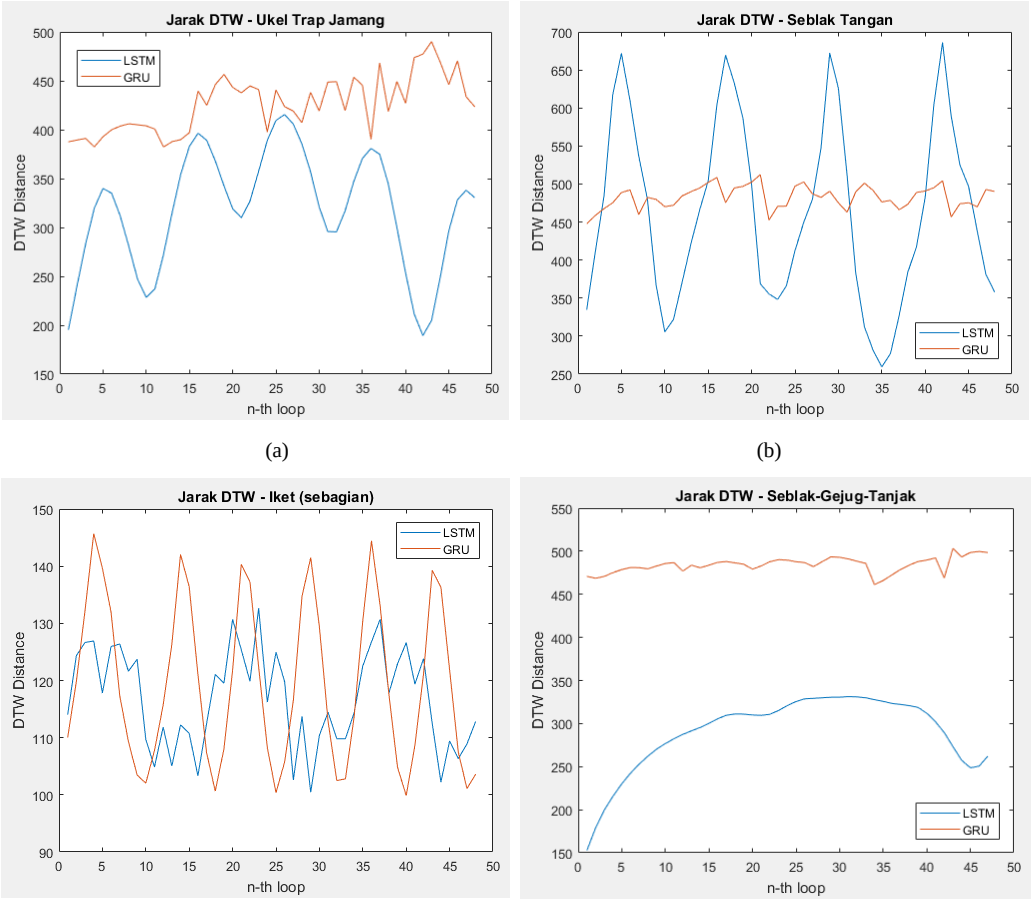
Evaluasi awal untuk menentukan keberhasilan proses pelatihan dilakukan secara visual. Hal ini harus dilakukan karena pemantauan nilai *error* atau *loss* menunjukkan pola yang sama, baik di kasus yang berhasil maupun yang gagal. Semua pelatihan menunjukkan nilai *loss* yang selalu berkurang secara logaritmik di setiap *epoch*-nya. Percobaan juga menunjukkan bahwa setelah sekitar 10.000 *epoch*, bila model terlihat tidak dapat terlatih, maka penambahan *epoch* berapa pun tidak akan membuatnya menjadi lebih baik. Dalam kasus seperti itu, maka pelatihan dianggap gagal. Satu-satunya solusi adalah dengan mengubah *random seed* dan melakukan pelatihan ulang.

Pengamatan visual juga dapat secara cepat menilai gerakan berdasarkan variasinya. Tiap perulangan gerakan tari memiliki variasi yang berbeda-beda. Sedikit variasi adalah hal yang diharapkan, karena variasi tersebut menambah realisme dalam



(a) (b)

Gbr. 5 Visualisasi pelatihan yang dianggap berhasil, (a) gerakan *seblak-gejuk-tanjak*, (b) gerakan *ukel trap jamang*.



(a) (b)

(c) (d)

Gbr. 6 Jarak DTW dari empat contoh gerakan, (a) *ukel trap jamang*, (b) *seblak tangan*, (c) sebagian dari gerakan *iket*, (d) gerakan *seblak-gejug-tanjak*.

animasi (gerakan manusia tidak pernah sama persis). Variasi yang besar membuat gerakan tari yang tidak lagi dianggap valid sebagai gerakan tari Remo. Di lain pihak, gerakan invalid ini dapat dianggap sebagai inspirasi bagi tari yang baru. Variasi yang lebih besar lagi membuat gerakan menjadi tidak manusiawi. Gerakan tidak manusiawi kadang hanya terjadi sekejap dan dapat dianggap sebagai *noise*. Bila gerakan tidak manusiawi ini terjadi sepanjang tarian, maka model ini akan digolongkan sebagai model yang gagal terlatih.

Contoh visualisasi hasil pelatihan yang dianggap berhasil diperlihatkan pada Gbr. 5. Pada gambar tersebut ditampilkan gerakan asli tarian dari *motion capture*, yaitu gambar yang berwarna biru (baris atas). Gerakan ini digunakan sebagai data pelatihan. Gerakan hasil dari LSTM ditampilkan dengan warna merah (baris tengah) dan hasil GRU ditampilkan dengan warna hijau (baris bawah). Pada Gbr. 5(a), diperlihatkan gerakan transisi yang melibatkan pergerakan seluruh tubuh (gerakan ke-8 dari Tabel I). Terlihat bahwa hasil LSTM mirip dengan gerakan asli dengan variasi yang cukup terlihat di gerakan tangan. Namun, hasil dari GRU tidak berhasil memutar badan dan gerakan tangan tidak sebesar yang diharapkan. Pada kasus

TABEL II
JUMLAH KEBERHASILAN DARI EMPAT PELATIHAN DENGAN SGD

Gerakan	*Epoch*								waktu pelatihan LSTM j:mm	waktu pelatihan GRU j:mm
	3.600		6.400		9.600		12.800			
	LSTM	GRU	LSTM	GRU	LSTM	GRU	LSTM	GRU		
1	1	0	4	0	4	0	4	1	2:50	2:54
2	1	0	3	0	3	1	4	2	2:36	2:40
3	1	0	1	0	3	0	4	1	2:42	2:58
4	4	0	4	0	4	1	4	2	4:36	4:36
5	4	0	4	0	4	0	4	1	4:48	5:02
6	2	0	2	0	4	0	4	1	3:48	3:50
7	4	0	4	0	4	0	4	2	3:20	4:05
8	4	0	4	1	4	4	4	4	4:02	4:12
9	4	0	4	1	4	4	4	4	2:28	2:33
10	4	0	4	1	4	4	4	4	5:58	6:05
11	4	0	4	0	4	2	4	2	2:20	2:31

lain di Gbr. 5(b), ditunjukkan gerakan *ukel trap jamang* (gerakan ke-10 dari Tabel I). Gerakan ini hanya menggunakan tangan dan tubuh bagian atas. Hasil LSTM dan GRU pada gerakan ini memiliki variasi yang berbeda, tetapi keduanya mirip dengan gerakan data pelatihan.

B. Evaluasi dengan DTW

Hasil pelatihan yang dianggap berhasil selanjutnya dievaluasi dengan DTW. Metode ini dipilih karena dapat memasangkan gerakan walaupun ada selisih dalam waktu [16]. DTW menghitung jarak perbedaan antara gerakan hasil LSTM/GRU dengan gerakan dari pelatihan untuk setiap tulang pada model. Perbedaan gerakan seluruh badan dihitung dari total perbedaan seluruh tulang.

Gbr. 6 menunjukkan contoh jarak DTW untuk empat macam gerakan. Model terlatih disampel untuk menghasilkan gerakan yang diingatnya sebanyak 50 kali perulangan. Setiap perulangan dihitung jaraknya dengan gerakan pelatihan. Grafik *plot* yang berwarna biru menunjukkan jarak DTW antara gerakan hasil LSTM dibanding gerakan asli, sedangkan grafik yang berwarna jingga adalah hasil dari GRU.

Setiap nilai di sumbu mendatar pada grafik di Gbr. 6 menunjukkan jarak DTW dari tiap perulangan. Sebagai contoh, nilai 25 pada sumbu mendatar menunjukkan jarak DTW dari perulangan ke-25. Nilai rendah pada sumbu tegak menunjukkan jarak DTW kecil. Hal ini berarti gerakan yang dihasilkan pada perulangan tertentu mirip dengan gerakan pelatihan. Sebaliknya, nilai besar menunjukkan perbedaan yang mencolok. Dari grafik ini dapat dilihat, misalnya, untuk gerakan *seblak* tangan pada Gbr. 6(b), perulangan ke-35 dari hasil LSTM sangat mirip dengan gerakan asli (lembah pada grafik), sedangkan perulangan ke-44 lebih berbeda (puncak bukit pada grafik).

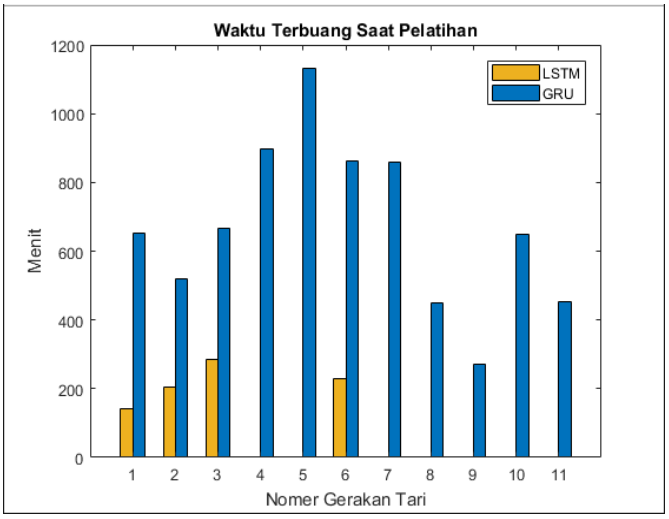
Gbr. 6 juga menunjukkan pola-pola variasi yang kerap terjadi pada gerakan tarian hasil. Seringkali perbedaan gerakan terlihat seperti Gbr. 6(a) dan Gbr. 6(b). Pola ini menunjukkan variasi gerakan hasil LSTM lebih fluktuatif dibanding hasil GRU, yang artinya gerakan hasil LSTM kadang berbeda sekali dengan gerakan pelatihan, tetapi kadang mirip sekali. Sedangkan variasi dari hasil GRU cenderung konstan, walaupun terkadang keduanya menunjukkan fluktuasi, seperti terlihat pada Gbr. 6(c). Dari rata-rata perbedaan seluruh hasil percobaan, dapat dihitung berdasar jarak DTW, gerakan hasil GRU lebih berbeda 116% daripada gerakan hasil LSTM.

C. Perbandingan Kinerja LSTM dan GRU

Walaupun GRU diciptakan untuk mempercepat proses pelatihan relatif terhadap LSTM, tetapi kinerja kedua jenis RNN ini tetap harus dinilai dari percobaan, karena akan berbeda kasus demi kasus [32]. Tabel II menunjukkan hasil LSTM dan GRU untuk pelatihan sampai 12.800 *epoch* dengan menggunakan optimisasi SGD. Setiap pelatihan dilakukan empat kali dengan *random seed* yang berbeda. Gerakan yang dihasilkan dipantau secara bertahap untuk menentukan gerakan sudah berhasil diingat oleh model atau belum. Pemantauan hasil dilakukan setiap *epoch* ke-3.600, 6.400, 9.600, dan 12.800.

Pada dua kolom paling kanan di Tabel II terdapat informasi tentang waktu pelatihan rata-rata untuk 12.800 *epoch* (dibulatkan dalam satuan jam dan menit). Waktu pelatihan tentu saja tergantung pada GPU yang digunakan. Pelatihan di percobaan ini dilakukan dengan menggunakan GPU NVIDIA GeForce 1050 dengan memori 4 GB. Terlihat waktu yang diperlukan untuk melatih setiap gerakan berbeda-beda, tergantung jumlah *frame* dalam gerakan yang bersangkutan. Terlihat pula selisih waktu yang diperlukan untuk LSTM dan GRU tidak banyak berbeda.

Tabel II juga menunjukkan bahwa LSTM lebih mudah dilatih daripada GRU. Hal ini ditunjukkan dalam empat kali pelatihan, LSTM berhasil dilatih seluruhnya. Sering pula saat baru mencapai 3.600 *epoch*, LSTM sudah berhasil dilatih. Di lain pihak, GRU sering mengalami kegagalan dan sebagian baru berhasil dilatih setelah *epoch* ke-12.800. Dari Tabel II, waktu yang terbuang karena kegagalan pelatihan dapat dirangkum dalam grafik batang di Gbr. 7. Ketinggian batang menunjukkan bahwa GRU lebih banyak membuang waktu pelatihan dibanding LSTM. Perbedaan ketinggian juga menunjukkan ada gerakan-gerakan yang lebih mudah dilatih daripada gerakan yang lain. Gerakan-gerakan yang mudah dilatih sudah dapat diingat oleh model pada *epoch-epoch* awal.



Gbr. 7 Waktu yang terbuang dalam empat kali pelatihan.

TABEL III
JUMLAH KEBERHASILAN PADA *EPOCH* 12.800 UNTUK *ADAGRAD* DAN *ADAM*

Gerakan	Adagrad		Adam	
	LSTM	GRU	LSTM	GRU
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	0	0	2	1
5	0	0	2	0
6	0	0	0	0
7	0	0	1	0
8	0	0	2	1
9	0	0	1	0
10	0	0	1	0
11	0	0	0	0

Beberapa gerakan yang sulit dilatih baru dapat diingat setelah epoch 12.800 (terutama oleh GRU). Contoh gerakan yang sulit dilatih adalah gerakan 3, 4, dan 5.

D. Peran Algoritme Optimisasi dan Regularisasi

Algoritme optimasi *AdaGrad* dan *Adam* diciptakan untuk mempercepat proses pelatihan. Namun, pada percobaan di kasus tarian ini kedua algoritme tersebut tidak membantu. Percobaan seperti yang tertera di Tabel II diulang dengan mengganti SGD dengan *AdaGrad* dan *Adam*. Rangkuman jumlah keberhasilan setelah *epoch* 12.800 dapat dilihat pada Tabel III. Pada tabel ini terlihat bahwa penggunaan *Adam* membuat keberhasilan pelatihan pada LSTM menjadi turun. Penggunaan *Adam* untuk GRU lebih rendah lagi. Sedangkan *AdaGrad* sama sekali tidak dapat digunakan karena tidak pernah berhasil dalam semua pelatihan yang dilakukan.

Hal lain yang telah dicoba adalah penambahan regularisasi *drop-out* dengan probabilitas 20%. Tujuan regularisasi sebenarnya untuk meredam parameter-parameter yang terlalu besar pengaruhnya dalam pelatihan [33]. Namun, percobaan di kasus ini menunjukkan bahwa penambahan regularisasi tidak memberikan kontribusi positif. Secara umum, hasil pelatihan dengan penggunaan *drop-out* sama dengan Tabel II dan Tabel III.

Dari hasil percobaan yang telah dilakukan, beberapa hal dapat didiskusikan. Secara umum, percobaan menunjukkan bahwa LSTM lebih mudah dilatih daripada GRU. Tentu saja hal ini spesifik untuk percobaan ini dan spesifik untuk data sejenis. Pada percobaan ini juga ditunjukkan bahwa algoritme optimasi yang lebih kompleks daripada SGD, yaitu *AdaGrad* dan *Adam* semakin menurunkan tingkat keberhasilan proses pelatihan. Demikian pula regularisasi *drop-out* tidak membantu membuat hasil menjadi lebih baik. Karena keberhasilan proses pelatihan terlihat bergantung pada *random seed* yang digunakan untuk MDN, maka dapat ditebak hal ini dapat terjadi karena efek acak dari MDN. Efek acak ini yang meredam segala perhitungan optimisasi pada *Adagrad* dan *Adam*. Efek acak ini juga yang membuat regularisasi *drop-out* (yang juga berdasarkan fungsi acak) tidak dapat memperbaiki hasil.

IX. KESIMPULAN

Analisis terhadap hasil uji coba pelatihan tari Remo menjadi model generatif menunjukkan bahwa dari sisi tingkat keberhasilan, LSTM memiliki kinerja lebih andal daripada GRU. Proses pelatihan dengan GRU memang sedikit lebih cepat, tetapi banyaknya jumlah pelatihan yang gagal membuat waktu yang terbuang lebih besar. Dari tiga algoritme optimisasi yang dicoba, ternyata algoritme paling klasik, yaitu SGD, memberikan hasil yang paling baik. Hal ini disebabkan karena efek acak dari unit MDN membuat teknik optimisasi di *Adam* tidak berhasil. Efek acak ini juga yang membuat *AdaGrad* sama sekali tidak dapat digunakan. Regularisasi *drop-out* juga tidak memberikan efek apa pun dalam keberhasilan.

Untuk pelatihan yang berhasil, penilaian secara visual maupun melalui perhitungan DTW menunjukkan bahwa gerakan tari hasil dari GRU lebih berbeda 116% dibanding hasil LSTM. Perbedaan gerak dari GRU kadang membuat gerakan tidak lagi dapat dianggap gerakan tari Remo. Namun, secara umum, metode pembuatan animasi menggunakan model generatif semacam ini memiliki potensi untuk dikembangkan menjadi suatu koreografer digital yang akan membantu pembuatan animasi tari.

REFERENSI

[1] L.A. Gatys, "Image Style Transfer Using Convolutional Neural Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, hal. 2414–2423.

[2] B. He, F. Gao, D. Ma, B. Shi, dan L. Duan, "ChipGAN : A Generative Adversarial Network for Chinese Ink Wash Painting Style Transfer," in *2018 ACM Multimedia Conference on Multimedia Conference*, 2018, hal. 1172–1180.

[3] T. Zhou, C. Fang, Z. Wang, J. Yang, B. Kim, Z. Chen, J. Brandt, dan D. Terzopoulos, "Learning to Sketch with Deep Q Networks and Demonstrated Strokes," *ArXiv: 1810.05977*, 2018.

[4] P. Isola, J. Zhu, T. Zhou, dan A.A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks.," *arXiv Comput. Vis. Pattern Recognit.*, 2018.

[5] L.-C. Yang, S.-Y. Chou, dan Y.-H. Yang, "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation," *arXiv:1703.10847 [cs.SD]*, 2017.

[6] O. Alemi, J. Françoise, dan P. Pasquier, "GrooveNet : Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks," *Proc. SIGKDD 2017Workshop Mach. Learn. Creat.*, 2017, hal. 1-6.

[7] C. Chan, S. Ginosar, T. Zhou, dan A.A. Efros, "Everybody Dance Now," *arXiv:1808.07371*, Vol. 1, No. 1, hal. 1-9. 2018.

[8] J.L. Elman, "Finding Structure in Time," *Cogn. Sci.*, Vol. 14, No. 2, hal. 179–211, 1990.

[9] S. Hochreiter dan J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, Vol. 9, No. 8, hal. 1735–1780, 1997.

[10] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk, dan Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv:1406.1078 [cs.CL]*, 2014.

[11] J. Wang, L.-C. Yu, K.R. Lai, dan X. Zhang, "Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model," *Proc. 54th Annu. Meet. Assoc. Comput. Linguist*, 2016, hal. 225–230.

[12] I. Sutskever, O. Vinyals, dan Q.V. Le, "Sequence to Sequence Learning with Neural Networks," *Adv. Neural Inf. Process. Syst.*, hal. 3104–3112, 2014.

[13] K. Xu, J.L. Ba, R. Kiros, K.Cho, A. Courville, R. Salakhutdinov, R.S. Zemel, dan Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 2015, Vol. 37 hal. 2048-2057.

[14] A. Graves, "Generating Sequences with Recurrent Neural Networks," *arXiv Prepr. arXiv1308.0850*, hal. 1–43, 2013.

[15] E. Hegarini, Dharmayanti, dan A. Syakur, "Indonesian Traditional Dance Motion Capture Documentation," *International Conference on Science and Technology-Computer (ICST)*, 2016, hal. 1-4.

[16] M. Müller, *Information Retrieval for Music and Motion*, Heidelberg, Germany: Springer-Verlag, 2007.

[17] J. Martens, "Generating Text with Recurrent Neural Networks," *Neural Networks*, Vol. 131, No. 1, hal. 1017–1024, 2011.

[18] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, dan K. Saenko, "Sequence to Sequence - Video to Text," *Proceedings of the IEEE International Conference on Computer Vision*, 2015, hal. 4534–4542.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, dan M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs.LG]*, hal. 1-9, 2013.

[20] T. Reil dan P. Husbands, "Evolution of Central Pattern Generators for Bipedal Walking in a Real-Time Physics Environment," *IEEE Trans. Evol. Comput.*, Vol. 6, No. 2, hal. 159–168, 2002.

[21] D. Holden, J. Saito, dan T. Komura, "A Deep Learning Framework for Character Motion Synthesis and Editing," *ACM Trans. Graph.*, Vol. 35, No. 4, hal. 1–11, 2016.

[22] F. Ofli, E. Erzin, Y. Yemez, dan A.M. Tekalp, "Learn2Dance: Learning Statistical Music-To-Dance Mappings for Choreography Synthesis," *IEEE Trans. Multimed.*, Vol. 14, No. 3, hal. 747–759, 2012.

[23] L. Crnkovic-Friis dan L. Crnkovic-Friis, "Generative Choreography Using Deep Learning," *Proc. of 7th Int. Conf. Comput. Creat.*, 2016, hal. 272-277.

[24] J. Duchi, E. Hazan, dan Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, Vol. 12, hal. 2121–2159, 2011.

[25] D.P. Kingma dan J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs.LG]*, hal. 1–15, 2014.

[26] A.T.R. Sari dan W. Wahyudi, "Rekonstruksi Gerak Pada Tari Remo Tawi Jombang," *Joged*, Vol. 10, No. 2, hal. 577-590, 2018.

[27] T. Wibisono, *Tari Remo di Surabaya: Dari Terob, Tobong, Menuju Kelas*, Surabaya, Indonesia: SatuKata, 2015.

[28] Wahyudianto, "Karakteristik Ragam Gerak dan Tatarias-Busana Tari Ngremo sebagai Wujud Presentasi Simbolis Sosio Kultural," *IMAJI*, Vol. 4, No. 2, hal. 136-156, 2006.

[29] C. Brakel-Papenhuijzen, *Classical Javanese Dance: The Surakarta Tradition and Its Terminology*, Leiden, The Netherlands: KITLV Press, 1995.

[30] C.M. Bishop, "Mixture Density Networks," Aston University, Birmingham, UK, NCRG Report, hal. 1-25, 1994.

[31] L. Zaman, S. Sumpeno, dan M. Hariadi, "Training Strategies for Remo Dance on Long Short- Term Memory Generative Model," *International Conference on Computer Engineering, Network, and Intelligent Multimedia*, 2018, hal. 1-5.

[32] J. Chung, C. Gulcehre, K. Cho, dan Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv:1412.3555v1 [cs.NE]*, 2014.

[33] R. Salakhutdinov, N. Srivastava, G. Hinton, A. Krizhevsky, dan I. Sutskever, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, Vol. 15, hal. 1929–1958, 2014.