Verifikasi Kemiripan Wajah Menggunakan *Deep Learning* Dengan Arsitektur Jaringan *Siamese*

Hairul Imam[1], Jian Budiarto, S.T., M.Eng[2], Kartarina, S.Kom., M.Kom[3]

[1,3] *Jurusan Ilmu Komputer Universitas Bumigora*
Jln. Ismail Marzuki, Kota Mataram, Nusa Tenggara Barat. 83127, INDONESIA
[2] *1310520075@stmikbumigora.ac.id*

INFORMASI ARTIKEL
Received:
Received in revised:
Accepted:

KEYWORD
***Siamese, Triplet Loss,* Verifikasi Wajah, *Face Embedding, Dimensionality Reduction*.**

INTISARI

Verifikasi wajah adalah masalah yang cukup populer dalam bidang *computer vision.* Banyak pendekatan yang telah dilakukan untuk menyelesaikan masalah tersebut baik menggunakan model matematika murni dengan mempelajari pola geometri pada wajah secara manual maupun cara otomatis menggunakan pendekatan pembelajaran mesin.

Penelitian ini mencoba memecahkan masalah tersebut dengan pendekatan *deep learning*, dimana model dilatih menggunakan *triplet loss* yang didefinisikan pada paper *FaceNet.* Rancangan model yang digunakan adalah *Siamese* dengan menerapkan ResNet-50 yang telah dimodifikasi untuk mempelajari fitur yang ada pada gambar sehingga mampu mereduksi dimensi gambar yang tinggi menjadi vektor baris yang rendah berdimensi 1x128 yang disebut sebagai *embedding*.

Setelah model berhasil mempelajari *embedding* yang baik pada gambar maka masalah verifikasi wajah bisa diselesaikan dengan membandingkan jarak *embedding* antar gambar dimana jarak yang dekat dapat diartikan sebagai wajah yang mirip (*genuine*) dan jarak yang jauh dapat diartikan sebagai wajah yang berbeda (*impostor*).

Pada penelitian ini, model berhasil dilatih pada dataset VGG Face v2 (*Visual Geometry Group*) dengan nilai akurasi 92% pada dataset LFW (*Labeled Faces in the Wild*) sebagai data *testing* dan mendapatkan nilai AUC (*Area Under the Curve*) 97%. Nilai AUC yang tinggi dapat diartikan bahwa model dapat memverifikasi dengan baik gambar wajah orang yang sama sebagai *genuine* dan gambar wajah orang yang berbeda sebagai *impostor.*

I. PENDAHULUAN

Masalah verifikasi kemiripan wajah adalah masalah pencocokkan antar dua gambar wajah untuk memperoleh suatu kesimpulan apakah kedua gambar wajah merupakan orang yang sama (*genuine*) atau berbeda (*impostor*), kesimpulan tersebut diambil dari hasil membandingkan ciri-ciri wajah yang ada pada dua gambar individu tersebut. Kebergantungan sebuah kesimpulan verifikasi berdasarkan perbandingan ciri wajah dari dua individu menyebabkan tantangan utama dalam masalah ini adalah seberapa handal sebuah metode atau *algoritma* dalam mengenali ciri-ciri (*feature*) pada gambar wajah. Mengenai hal ini, telah banyak metode atau pendekatan yang dilakukan oleh peneliti sebelumnya (lihat bagian II) seperti metode pembelajaran pola geometry wajah secara manual menggunakan pendekatan matematis, menggunakan pendekatan semacam ini membutuhkan keahlian khusus pada bidang tertentu oleh karena itu sangat sulit untuk diterapkan.

Pendekatan yang lain yaitu menggunakan pendekatan *deep learning.* Dengan metode *deep learning* pembelajran fitur pada wajah bisa dilakukan secara otomatis dengan mempelajari pola pada sekumpulan data dalam jumlah besar tanpa memerlukan pengetahuan tambahan dari lingkup luar (*prior knowledge*), cukup hanya dengan analisa otomatis pada data yang diberikan (*feature extraction*).

Berdasarkan hal tersebut penelitian ini menggunakan pendekatan *deep learning.* Dimana arsitektur jaringan yang digunakan adalah *Siamese* dengan badan jaringan menggunakan ResNet-50 yang telah dimodifikasi, lalu jaringan akan dilatih dengan fungsi *triplet loss*. Fungsi *triplet loss* digunakan dengan objektif mereduksi dimensi gambar wajah yang tinggi menjadi vektor baris dengan dimensi lebih rendah yang dapat merepresentasikan fitur gambar wajah (*feature vector/embedding*).

Setelah model berhasil dilatih dengan optimal, maka model bisa digunakan untuk mendapatkan representasi *feature vector* dalam dimensi lebih

rendah dari gambar wajah yang dapat langsung digunakan untuk menyelesaikan masalah verifikasi wajah dengan membandingkan jarak *feature vector* antar gambar. Adapun cara melatih jaringan dengan *triplet loss* agar mendapatkan hasil *embedding* yang optimal akan dijelaskan pada bagian III

II. Topik Berkaitan

Ada beberapa penelitian dalam menyelesaikan masalah serupa yang telah dilakukan, seperti yang dilakukan oleh Tri Mulyono dkk [1] dimana pada penelitian tersebut pendekatan yang digunakan adalah menggabungkan metode *eigenface* dengan JST (Jaringan Syarat Tiruan). Dimana fitur dari citra wajah diekstrak terlebih dahulu menggunakan metode PCA (*Principal Component Analysis*) lalu hasil fitur yang telah diekstrak tersebut akan digunakan sebagai masukkan (*input)* ke JST. Dalam kasus seperti ini jaringan JST hanya dilatih untuk mengenali fitur yang didapatkan melalui ekstraksi fitur manual pada proses PCA. JST tidak dilatih untuk mengenali dan mengambil informasi fitur secara otomatis, sehingga tingkat kebaikan PCA dalam mengekstrak dan mengenali fitur wajah menjadi penentu keberhasilan utama dari metode ini. Pada penelitian tersebut JST berhasil dilatih pada 120 dataset dengan 10 individu dan dapat mengenali rata-rata sebanyak 84,6% dari data evaluasi.

Pendekatan serupa juga telah dilakukan oleh Dimas Achmad Akbar Kusuma [2] dimana dalam mengekstrak fitur wajah metode yang digunakan adalah *Discrete Cosine Transform* lalu hasil dari proses ini akan menjadi masukkan ke JST. JST akan dilatih berdasarkan fitur yang didapat pada proses *Discrete Cosine Transform.* Pada penelitian tersebut jumlah gambar wajah yang digunakan adalah 100 dimana terdapat 10 individu dengan 10 pose yang berbeda-beda dan berhasil mendapat akurasi sebanyak 90% hingga 100% pada proses evaluasi dengan individu yang sama pada saat proses latihan.

Melakukan ekstraksi fitur secara manual seperti dua metode diatas masuk keruang lingkup *classical learning* dimana fitur diekstrak terlebih dahulu menggunakan algoritma tertentu lalu hasil fitur yang telah diekstrak akan menjadi masukkan (*input*) ke algoritma yang menyelesaikan masalah terkait. Pendekatan yang demikian memiliki hasil yang baik dalam ruang lingkup data yang relatif sedikit, belum diuji apakah metode yang sama dapat mengekstrak fitur pada gambar wajah yang lebih beragam dan dengan jumlah dataset yang lebih besar lagi.

III. Metodologi

Pada penelitian ini fungsi yang akan digunakan untuk melatih jaringan dalam mempelajari *embedding* yang baik adalah fungsi *triplet loss* seperti yang didefinisikan dalam paper Facenet oleh Florian Schroff dkk [3]. Dimana komponen utama dari triplet loss adalah adanya 3 pasangan gambar yang disebut pasangan triplet yang terdiri dari gambar *anchor* yaitu gambar acuan, gambar *positive* yaitu gambar individu yang sama dengan gambar *anchor* dan gambar *negative* yaitu gambar individu yang berbeda dari gambar *anchor*.

Gambar 1. **Contoh pasangan triplet**. Gambar disebut pasangan *triplet* yang valid ketika sudah memenuhi kriteria yaitu gambar *anchor* dan *positive* terdiri dari individu yang sama dan gambar *negative* merupakan individu yang berbeda.

Adapun objektif dari fungsi ini adalah agar jarak antar embedding *anchor* dengan *positive* lebih kecil dari jarak *embedding* antar *anchor* dengan *negative*, seperti terlihat dalam persamaan (1) dibawah.

$$\left\|x_i^a - x_i^p\right\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2 , \forall \left(x_i^a, x_i^p, x_i^n\right) \in \mathrm{T} \tag{1}$$

Dimana nilai α adalah sebuah margin yang memastikan adanya perbedaan nilai antar jarak gambar *positive* dan *negative* dengan *anchor*. Sehingga fungsi *triplet loss* yang harus diminimalkan adalah seperti persamaan (2) dibawah.

$$L = \sum_i^N \left[\left\|f(x_i^a) - f\left(x_i^p\right)\right\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+ \tag{2}$$

Secara intuisi persamaan diatas bisa dibaca bahwa nilai jarak antar gambar *positive* harus seminimal mungkin hingga lebih kecil dari jarak gambar *negative*, nilai α pada persamaan (2) diatas untuk memastikan bahwa jarak gambar *positive* harus lebih kecil dari gambar *negative* namun tidak melebihi

nilai margin sehingga nilai *loss* tetap berada pada nilai *positive*. Ilustrasi objektif pembelajaran *triplet loss* bisa dilihat pada gambar 2[1] dibawah.

Gambar 2. **Ilustrasi objektif *triplet loss***. Model harus dilatih sehingga jarak antar gambar *positive* lebih kecil dari jarak gambar *negative*.

Melihat persamaan (2) diatas, tidak semua pasangan *triplet* bisa digunakan untuk melatih jaringan. Pada bagian dibawah ini akan dijelaskan beberapa pilihan *triplet* yang bisa digunakan untuk melatih jaringan.

A. Pemilihan Triplet

Agar dapat melatih jaringan dengan optimal dibutuhkan pasangan triplet yang baik, adapun beberapa jenis *triplet* seperti berikut:

1) Easy Triplet

Easy triplet adalah jenis *triplet* yang sudah memenuhi fungsi kendala pada persamaan (1). Jenis triplet seperti ini tidak perlu digunakan untuk melatih jaringan karena memiliki nilai loss 0 dan dapat menyebabkan *convergent* yang lebih lama. Secara matematis jenis triplet seperti ini bisa diekspresikan seperti persamaan (3) dibawah.

$$d(f(x^a), f(x^p)) + \alpha < d(f(x^a), f(x^n)) \tag{3}$$

Dimana d adalah fungsi jarak, sehingga persamaan diatas bisa diartikan bahwa jarak gambar *anchor* dan *positive* ditambah margin lebih kecil dari jarak gambar *anchor* dengan *negative*.

2) Hard Triplet

Hard triplet adalah pasangan triplet yang dimana jarak antar gambar *negative* dengan *anchor* lebih kecil dari jarak gambar *positive* dengan *anchor*. Jenis triplet ini bisa diekspresika seperti persamaan (4) dibawah.

$$d(f(x^a), f(x^n)) < d(f(x^a), f(x^p)) \tag{4}$$

3) Semi-hard Triplet

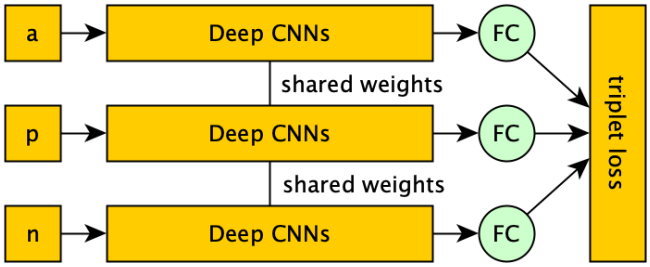
Semi-hard triplet adalah jenis *triplet* yang jarak gambar *negative* dengan *anchor* lebih kecil dari jarak antar gambar *positive* dengan anchor ditambah margin namun tidak lebih kecil dari jarak gambar *anchor* dan *positive* tanpa margin, sehingga bisa diartikan selisih jarak gambar *negative* dengan *positive* pada *triplet* ini tidak melebihi nilai margin. Jenis *triplet* ini bisa diekspresikan seperti persamaan (5) dibawah.

$$d(f(x^a), f(x^p)) < d(f(x^a), f(x^n)) < d(f(x^a), f(x^p)) + \alpha \tag{5}$$

Setelah mengetahui jenis-jenis *triplet* diatas, adapun yang bisa digunakan untuk melatih jaringan adalah jenis *hard triplet* dan *semi-hard triplet*. Pada penelitian ini jenis *triplet* yang dipilih adalah *semi-hard triplet*.

B. Arsitektur Jaringan

Secara garis besar arsitektur jaringan yang digunakan terlihat seperti gambar 3 dibawah.



Gambar 3. **Gambaran besar struktur jaringan**. Setiap pasangan gambar triplet akan melewati jaringan dengan parameter yang sama (*shared weights*) fitur dari gambar akan dipelajari secara otomatis pada lapisan *Deep* CNNs lalu hasil fitur yang dipelajari akan dimasukkan ke lapisan Fully Connected Layer (FC). Lapisan FC akan dilatih agar dapat menghasilkan *feature vector* (*embedding*) yang baik menggunakan fungsi *triplet loss*.

1) Siamese

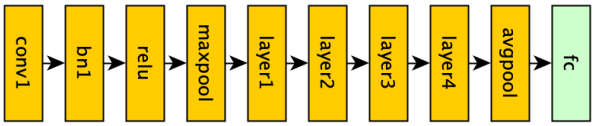
Ide utama dari *Siamese* adalah melewatkan sebuah *input* melalui dua (*twine*) atau tiga (*triplet*) arsitektur jaringan yang identik yang berbagi nilai parameter yang sama sehingga pada saat *training* masukkan yang dilewatkan pada arsitektur jaringan tersebut diproses dengan nilai parameter yang sama pada saat bersamaan. Pada gambar 3, arsitektur jaringan yang identik terdapat berjumlah tiga dengan berbagi nilai

[1] Sumber gambar paper FaceNet, referensi nomor 3.

parameter yang sama, adapun arsitektur *Deep* CNNs pada arsitektur tersebut menggunakan blok layer ResNet-50 yang telah dimodifikasi seperti dijelaskan pada bagian berikut.

2) Modifikasi ResNet-50

Adapun blok layer ResNet-50 yang utuh dapat dilihat seperti gambar 4 dibawah.



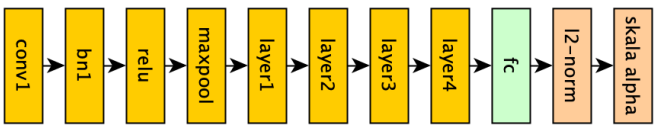
Gambar 4. **Blok layer ResNet-50 yang utuh**. Adapun blok layer ResNet-50 yang utuh tanpa modifikasi terdiri dari layer conv1, bn1, relu, maxpool, layer1, layer2, layer3, layer4, avgpool dan layer fc.

Modifikasi layer ResNet-50 yang dilakukan pada penelitian ini berdasarkan penelitian yang dilakukan oleh Hermans dkk [4] dan Rajeev Ranjan dkk [5]. Modifikasi layer ResNet-50 yang dilakukan yaitu pembuangan layer avgpool dan penambahan operasi L2-Normalize dan operasi perkalian skala *alpha* pada layer FC. Adapun operasi L2-Normalize dan perkalian skala *alpha* yang dilakukan pada layer FC bisa dilihat pada persamaan (6) dan persamaan (7) dibawah.

$$y = \frac{x}{\|x\|_2} \tag{6}$$

$$z = y \cdot \alpha \tag{7}$$

Pada proses L2-Normalize persamaan (6) setiap titik nilai pada input akan dinormalkan dengan norm L2 dari vektor input, lalu hasil dari proses ini akan dikalikan dengan skala *alpha* persamaan (7). Nilai skala alpha dapat dilatih selama proses *training* seperti parameter yang lain, namun pada paper [5] Rajeev Ranjan menyebutkan bahwa pemberian nilai tetap pada variabel *alpha* dengan nilai yang rendah memberikan performa yang lebih baik, pada penelitian ini nilai *alpha* yang digunakan adalah 10.



Gambar 5. **Blok layer ResNet-50 setelah modifikasi**. Modifikasi layer ResNet-50 dengan pembuangan layer avgpool dan penambahan operasi l2-norm dan skala alpha pada layer FC.

Rincian masukkan dan keluaran setiap blok layer ResNet-50 disajikan pada tabel 1 dibawah.

TABEL I. RINCIAN MASUKKAN DAN KELUARAN BLOK RESNET

Layer	Input	Proses	Output
Conv1	1x3x224x224	Conv2D	1x64x112x112
BN1	1x64x112x112	BatchNorm	1x64x112x112
ReLU	1x64x112x112	ReLU	1x64x112x112
MaxPool	1x64x112x112	MaxPool	1x64x56x56
Layer1	1x64x56x56	Bottleneck1	1x256x56x56
	1x256x56x56	Bottleneck2	1x256x56x56
	1x256x56x56	Bottleneck2	1x256x56x56
Layer2	1x256x56x56	Bottleneck3	1x512x28x28
	1x512x28x28	Bottleneck4	1x512x28x28
	1x512x28x28	Bottleneck4	1x512x28x28
	1x512x28x28	Bottleneck4	1x512x28x28
Layer3	1x512x28x28	Bottleneck5	1x1024x14x14
	1x1024x14x14	Bottleneck6	1x1024x14x14
	1x1024x14x14	Bottleneck6	1x1024x14x14
	1x1024x14x14	Bottleneck6	1x1024x14x14
	1x1024x14x14	Bottleneck6	1x1024x14x14
Layer4	1x1024x14x14	Bottleneck7	1x2048x7x7
	1x2048x7x7	Bottleneck8	1x2048x7x7
	1x2048x7x7	Bottleneck8	1x2048x7x7
FC	1x100,352	Linear	1x128

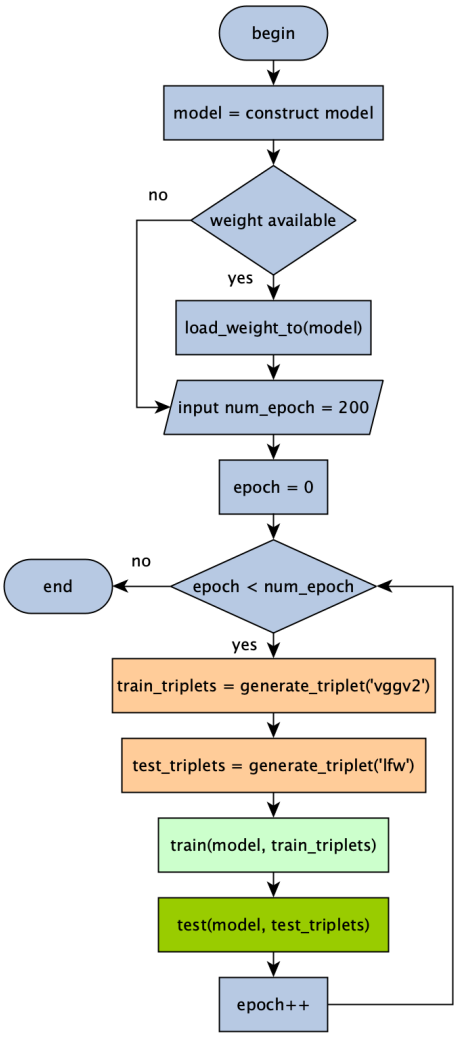
Setiap masukkan gambar akan melalui setiap blok layer ResNet seperti tabel 1 diatas. Layer conv1 adalah layer pertama pada layer ResNet yang akan dilalui input, layer conv1 menerima masukkan gambar berupa 3 kanal warna dengan ukuran 224×224. Angka 1 pada dimensi tensor tersebut menunjukkan jumlah gambar yang dimasukkan dalam satu waktu (*batch size*). Jika pada saat training menggunakan 64 *batch size* maka dimensi tensor menjadi $64 \times 3 \times 224 \times 224$ begitu juga dengan dimensi tensor pada layer yang lain akan berubah.

Lapisan conv1 sampai lapisan Layer4 adalah yang dimaksud blok *Deep* CNNs pada gambar 3, hasil dari blok *Deep* CNNs pada layer ResNet adalah tensor dengan dimensi $1 \times 2048 \times 7 \times 7$ untuk setiap masukkan gambar. Fitur yang berhasil dipelajari pada blok Deep CNNs lalu dimasukkan ke lapisan FC sebagai fitur yang akan dipelajari untuk mendapatkan *embedding* yang baik, fitur yang masuk ke layer FC berjumlah 100,352 didapat dari hasil proses *flatten* (perataan) fitur pada layer Deep CNNs jika dimensi $(1 \times 2048 \times 7 \times 7)$ dikalikan akan menghasilkan 100,352. Keluaran dari proses layer FC adalah vektor baris dengan dimensi

1×128 yang akan menjadi dimensi dari *embedding* yang dipelajari. Sehingga semua gambar dengan dimensi tinggi cukup bisa diwakili hanya dengan representasi vektor baris dengan dimensi 1×128.

C. Proses Training dan Testing

Secara garis besar proses melatih dan menguji model dilakukan seperti gambar 6 dibawah.



Gambar 6. **Gambaran besar proses *training* dan *testing***. Pada tahap pertama model akan dibangun, jika sudah ada *weight* yang tersimpan dari proses *training* sebelumnya maka *weight* akan dimuat ke model. Proses *training* dan *testing* akan dilakukan sebanyak *epoch* yang diinginkan atau sehingga model dianggap belajar dengan baik dengan akurasi *testing* yang memuaskan. Dalam setiap iterasi *epoch* akan dilakukan pembuatan pasangan triplet secara acak untuk data training dan data testing, lalu data training diproses untuk training dan data testing dimasukkan ke proses testing. Proses ini berulang hingga objektif tercapai.

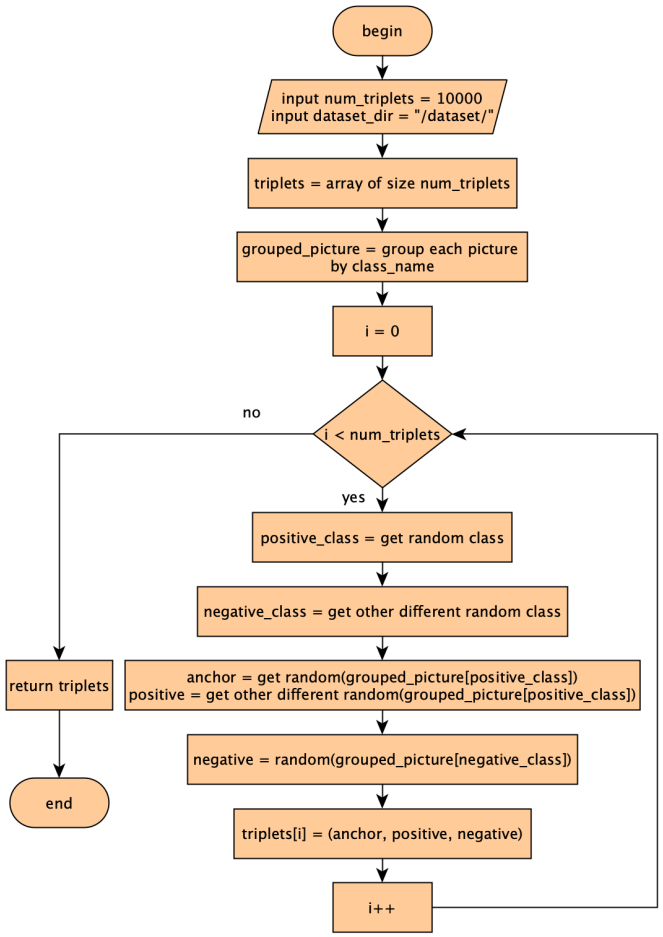
1) Dataset

Dataset yang digunakan dalam penelitian ini ada dua yaitu dataset VGG (*Visual Geometry Group*) *Face* v2 sebagai dataset untuk latihan dan dataset *Labeled Face in the Wild* (LFW) sebagai data uji coba. Kedua dataset ini merupakan dataset yang sangat terkenal dalam melakukan penelitian face recognition atau computer vision dengan deep learning secara umum karena keberagaman data dan jumlah data dimiliki.

Dataset VGG *Face* v2 memiliki 9000 jenis identitas dengan variasi etnis, aksen, profesi dan umur yang berbeda dan memiliki 3,3 juta gambar. Dataset ini didistribusi secara bebas dengan lisensi Creative Common Attribution-ShareAlike 4.0.

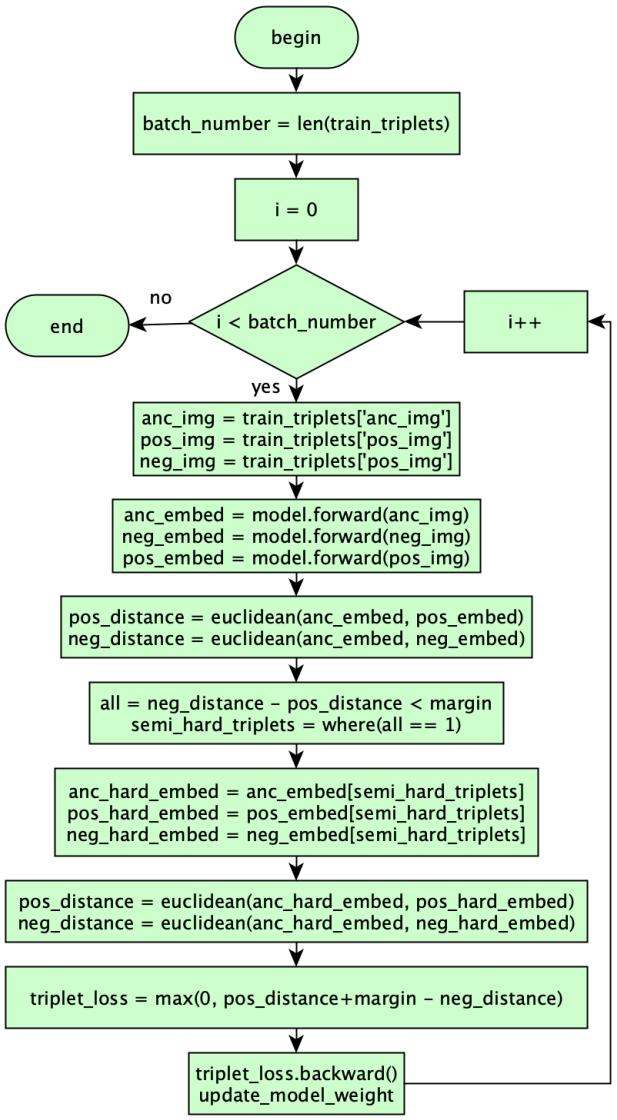
Dataset LFW disediakan oleh University of Massachusetts. Dataset ini merupakan acuan pengujian seluruh peneliti *computer vision* dalam bidang pengenalan wajah saat ini. Dataset ini berisi 13,233 gambar dengan 5749 jenis individu dimana 1680 diantaranya memiliki lebih dari satu gambar wajah dan 4096 sisanya masing-masing hanya memiliki 1 gambar wajah.

2) Generate Triplet



Gambar 7. **Flowchart pembuatan pasangan triplet acak**. Pertama tentukan jumlah pasangan triplet yang valid yang diinginkan. Lalu lakukan perulangan, pada setiap perulangan ambil class (individu) secara acak lalu ambil dua gambar dari individu tersebut sebagai gambar *anchor* dan *positive*, lalu ambil satu gambar dari individu yang berbeda secara acak sebagai gambar negative.

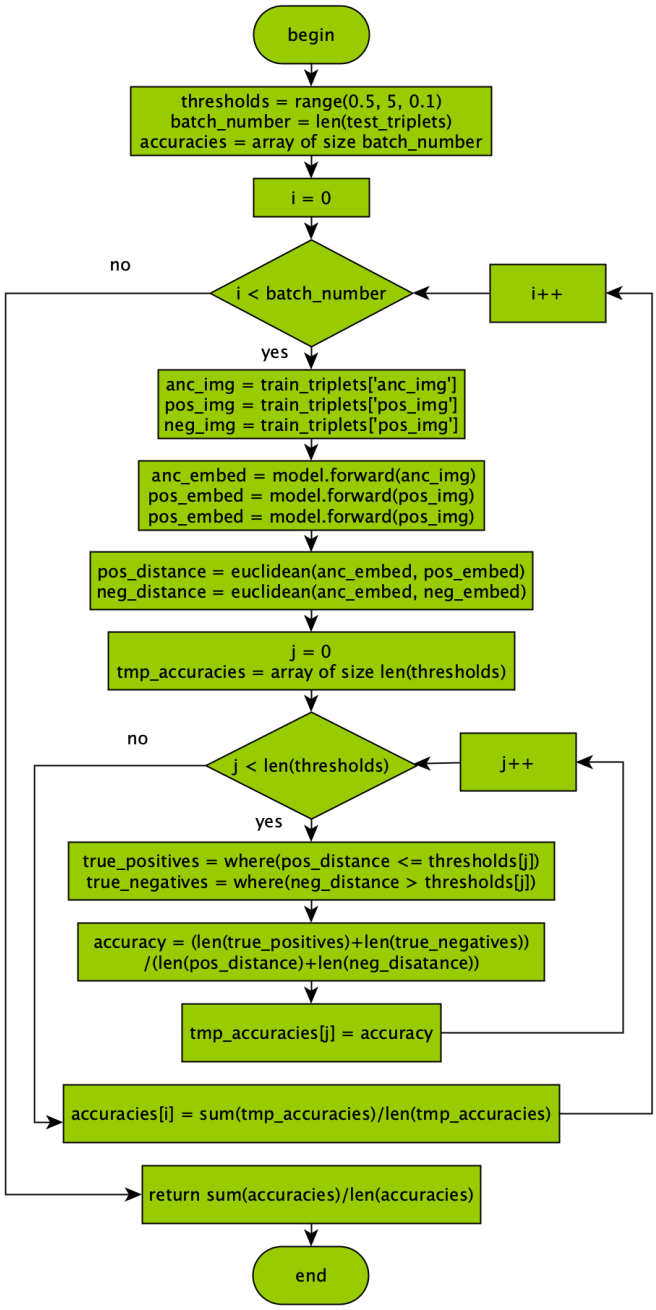
3) Proses Training



Gambar 8. **Flowchart proses training**. Jumlah iterasi pada proses training adalah jumlah bagian pasangan triplets acak setelah dibagi dengan *batch size*.

Bagian penting pada proses training adalah pada proses pemilihan triplet yang baik untuk melatih model. Pada flowchart gambar 8 diatas, proses pemilihan triplet dilakukan pada blok proses ke 4 dalam lingkup perulangan. Hasil perhitungan jarak *positive* dan *negative* dari *embedding* triplet acak akan dilakukan proses pengurangan, pada tahap ini jika nilai *neg_distance* lebih besar dari nilai *pos_distance+margin* maka tidak ada pasangan *semi hard triplets* yang didapat karena tidak ada yang memenuhi kondisi. Tapi jika nilai *neg_distance* lebih besar namun selisihnya tidak melebihi nilai margin maka pasangan *semi_hard_triplets* bisa didapat. Pasangan *semi_hard_triplets* pada proses tersebut yang akan digunakan untuk melatih model pada proses perhitungan *triplet_loss* sehingga *weight* model bisa diperbarui berdasarkan nilai *gradient* yang didapat dari operasi *backpropagation* nilai *loss semi_hard_triplets*.

4) Proses Testing



Gambar 9. **Flowchart proses testing**. Nilai akurasi pada proses testing didapat berdasar rata-rata akurasi pada setiap iterasi.

Pada proses testing untuk mendapatkan nilai akurasi model akan dihitung jumlah verifikasi *true positive* dan *true negative* berdasarkan rentang nilai *threshold* tertentu. Rentang nilai threshold yang

digunakan yaitu nilai 0,5 sampai 5,0 dengan selisih masing-masing 0,1.

IV. HASIL DAN PEMBAHASAN

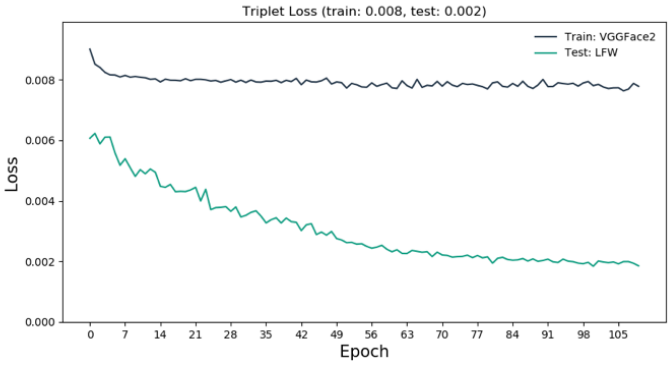
Untuk melakukan proses *training* dan *testing* pada penelitian ini digunakan layanan GCP (Google Cloud Platform) dengan konfigurasi hardware dan software sebagai berikut:

HARDWARE	4 x 12GB NVIDIA Tesla K80 GPU
	8 Core vCPU Intel Broadwell
	52GB RAM
	256GB Boot Disk
	200GB Dataset Disk
SOFTWARE	Debian 4.9.130-2 (2018-10-27) x86_64 GNU/Linux
	Python v3.7.1
	Pytorch v1.0
	CUDA v10.0

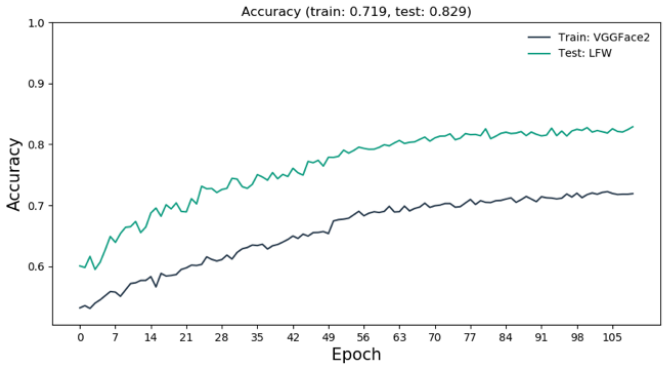
Adapun proses training dan testing dibagi menjadi dua tahap, pertama training dan testing pada layer FC kedua training dan testing pada layer ekstraksi fitur (Deep CNNs).

1) Training dan Testing Layer FC

Pada proses ini masukkan dari layer FC adalah hasil proses layer ekstraksi fitur (Deep CNNs) dengan weight dari *pretrained* ResNet-50. Layer ekstraksi fitur akan dilatih kembali untuk mempelajari ciri wajah pada dataset training. Hasil latihan pada proses ini terlihat pada gambar 10 dan gambar 11 grafik dibawah.



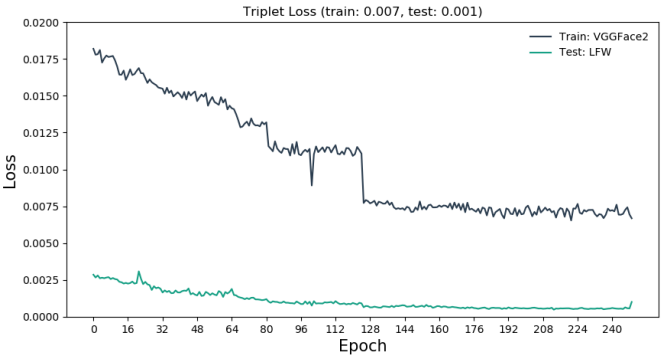
Gambar 10. **Hasil loss setiap epoch layer FC**. Pada proses melatih layer FC model dapat dilatih hingga mendapat loss 0.008 pada data train dan 0.002 pada data testing.



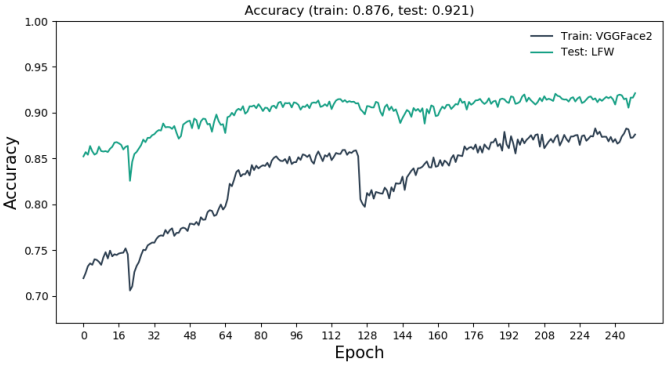
Gambar 11. **Hasil akurasi setiap epoch layer FC**. Pada proses melatih layer FC model dapat dilatih hingga mendapat akurasi 71% pada data train dan 82% pada data testing.

2) Training dan Testing Layer Ekstraksi Fitur

Pada proses ini layer ekstraksi fitur (*Deep* CNNs) akan dilatih kembali agar dapat mempelajari ciri wajah pada data training. Adapun hasil latih dan uji pada proses ini bisa dilihat pada gambar dibawah.



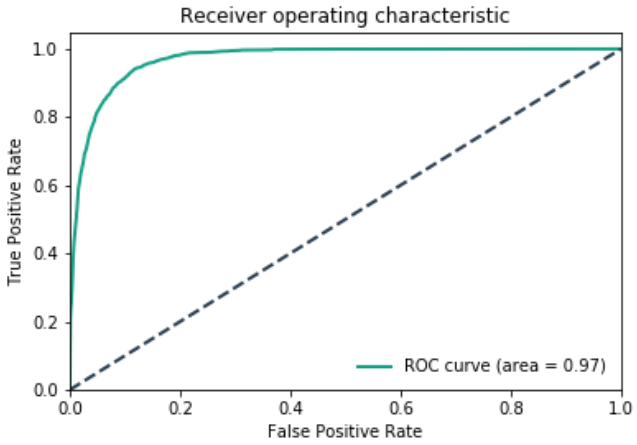
Gambar 12. **Hasil loss setiap epoch layer ekstraksi fitur**. Pada proses melatih layer ekstraksi fitur model dapat dilatih hingga mendapat nilai loss 0.007 pada data training dan nilai 0.001 pada data testing.



Gambar 13. **Hasil akurasi setiap epoch layer ekstraksi fitur**. Pada proses melatih layer ekstraksi fitur model dapat dilatih hingga mendapat nilai akurasi 87% pada data training dan nilai 92% pada data testing.

Bisa dilihat pada gambar 13, model telah berhasil dilatih hingga mendapat nilai akurasi 87% pada training dan 92% pada testing. Nilai akurasi testing yang lebih tinggi dari nilai akurasi training

menunjukkan bahwa model telah belajar dengan baik dan tidak terjebak *overfit* hingga mampu melakukan generalisir pada data baru. Grafik ROC (*Receiver Operating Characteristic*) pada proses training dan testing terakhir bisa dilihat pada gambar 14 dibawah.

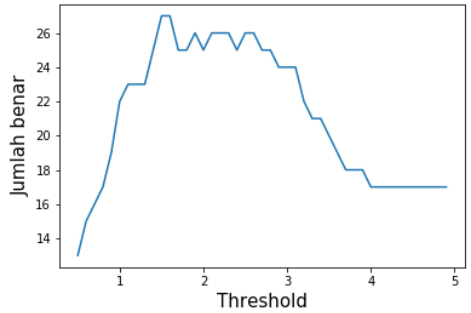


Gambar 14. **Grafik ROC dan nilai AUC**. Pada proses training terakhir model berhasil mendapat nilai AUC (*Area Under the Curve*) sebanyak 97%.

Nilai AUC mencapai 97% bisa diinterpretasikan bahwa model dapat memverifikasi dengan benar gambar wajah orang yang sama dan orang yang berbeda dengan persentase 97% selama proses pengujian.

3) Percobaan Verifikasi Wajah

Setelah model berhasil dilatih hingga mendapat nilai akurasi 92% pada data testing, pada bagian ini akan dilakukan percobaan verifikasi wajah 30 pasangan individu secara acak dengan nilai *threshold* yang berbeda-beda seperti dijelaskan pada sub sub bab Proses Testing. Lalu akan dihitung jumlah true positive dan true negative setiap nilai *threshold*. Hasil dari percobaan ini bisa dilihat pada gambar 15 dibawah.



Gambar 15. **Grafik jumlah benar setiap nilai threshold**. Pada grafik diatas bisa dilihat nilai threshold 1,5 dan 1,6 mendapat jumlah benar terbanyak.

Terlihat pada grafik gambar 15 jumlah benar terbanyak didapat ketika nilai threshold berada pada nilai 1,5 dan 1,6 denga mendapat jumlah benar sebanyak 27 dari 30 percobaan. Daftar lengkap jumlah benar setiap nilai threshold pada percobaan ini bisa dilihat pada tabel 2 dibawah.

TABEL 2. RINCIAN JUMLAH BENAR SETIAP NILAI THESHOLD

Nilai Threshold	Jumlah Benar
0,5	13
0,6	15
0,7	16
0,8	17
0,9	19
1,0	22
1,1 – 1,3	23
1,4	25
1,5 – 1,6	27
1,7 – 1,8	25
1,9	26
2,0	25
2,1 – 2,3	26
2,4	25
2,5 – 2,6	26
2,7 – 2,8	25
2,9 – 3,1	24
3,2	22
3,3 – 3,4	21
3,5	20
3,6	19
3,7 – 3,9	18
4,0 – 4,9	17

V. KESIMPULAN

Berdasarkan pemaparan hasil penelitian pada bagian IV bisa disimpulkan beberapa hal yaitu:

1. Arsitektur jaringan Siamese berhasil mempelajari fitur wajah pada dataset VGGv2 menggunakan ResNet-50 sehingga mampu menghasilkan *embedding* yang baik.
2. Arsitektur jaringan Siamese berhasil dalam mempelajari *embedding* gambar menggunakan ResNet-50 yang telah dimodifikasi dengan akurasi 92% pada data *testing*.
3. Model berhasil mempelajari fitur pada data training dan telah diuji pada data testing dan mendapatkan nilai AUC sebanyak 97%.

REFERENSI

[1] T. Mulyono, K. Adi, and R. Gernowo, "Sistem Pengenalan Wajah Dengan Metode Eigenface Dan Jaringan Syaraf Tiruan (Jst)," *Berk. Fis.*, vol. 15, no. 1, pp. 15–20, 2012.

[2] D. A. A. Kusuma, F. Ardilla, and B. S. B. Dewantara, "Verifikasi Citra Wajah Menggunakan Metode Discrete Cosine Transform Untuk Aplikasi Login," *Ind. Electron. Semin.*, vol. 8, no. 5, p. 55, 2011.

[3] B. Huang, "FaceNet: A Unified Embedding for Face Recognition and Clustering," pp. 1–2, 2015.

[4] A. Hermans, L. Beyer, and B. Leibe, "In Defense of the Triplet Loss for Person Re-Identification," 2017.

[5] R. Ranjan, C. D. Castillo, and R. Chellappa, "L 2 -constrained Softmax Loss for Discriminative Face Verification," 2017.