# BAB IV

# HASIL DAN PEMBAHASAN

1. **Konfigurasi Hardware dan Software**

Proses *training* dan *testing* pada penelitian ini menggunakan layanan dari *Google Cloud Platform* (GCP) dengan konfigurasi *hardware* dan *software* sebagai berikut:

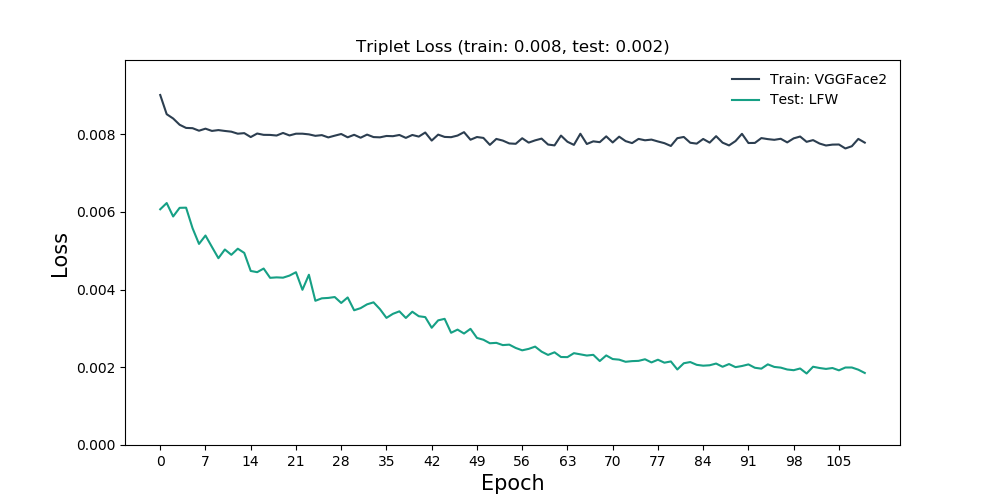
|  |  |
| --- | --- |
| **HARDWARE** | 4 x 12GB NVIDIA Tesla K80 GPU |
| 8 Core vCPU Intel Broadwell |
| 52GB RAM |
| 256GB Boot Disk |
| 200GB Dataset Disk |
| **SOFTWARE** | Debian 4.9.130-2 (2018-10-27) x86\_64 GNU/Linux |
| Python v3.7.1 |
| Pytorch v1.0 |
| CUDA v10.0 |

1. **Proses Training dan Testing**

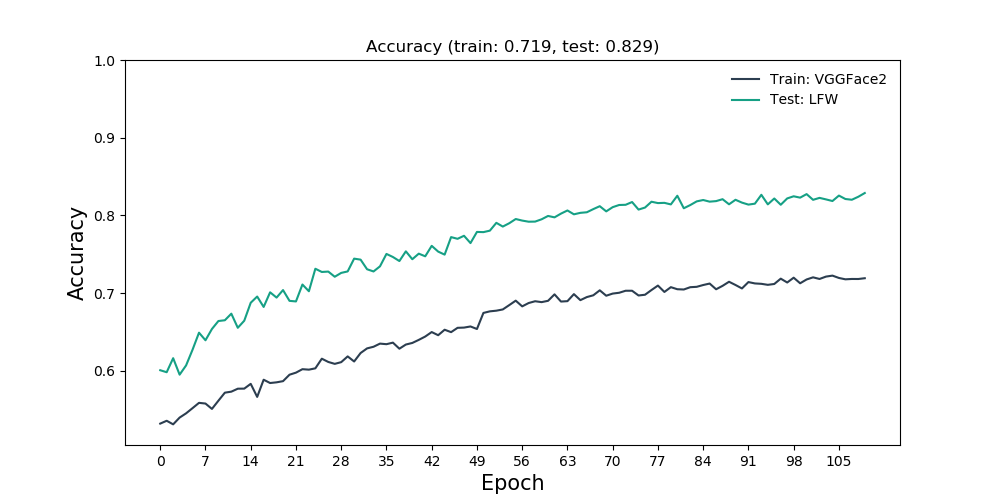
Proses *training* dan *testing* dalam penelitian ini dilakukan dalam beberapa tahap pada setiap layer dimulai dari layer terakhir dilanjutkan sampai layer pertama dengan *learning rate* yang berbeda seperti dijelaskan pada pembahasan berikut ini.

* 1. **Layer FC**

Layer FC (*Fully Connected Layer*) adalah layer yang akan membuat kesimpulan dari fitur yang dipelajari pada sebuah gambar. Dalam hal ini layer FC yang akan mempelajari *embedding* sebuah gambar. *Embedding* yang dipelajari pada tahap ini menggunakan fitur yang dipelajari dari *pretrained* ImageNet, model akan dilatih ulang untuk mempelajari fitur wajah pada dataset *training* melalui layer CNN yang akan dijelaskan pada sub bab 4.2.2. *Embedding* yang dihasilkan adalah vektor berdimensi 1x128. Grafik proses *training* dan *testing* layer ini bisa dilihat pada gambar 4.1 dan 4.2 dibawah.



Gambar 4.1 Grafik *Triplet Loss Training* & *Testing* Hasil Latih Layer FC



Gambar 4.2 Grafik Akurasi *Training* & *Testing* Hasil Latih Layer FC

Pada proses training Layer FC, model berhasil belajar hingga mendapatkan akurasi 82% pada data *testing* dan mendapatkan akurasi 71% pada data *training,* ini bisa diartikan bahwa model dapat belajar dengan baik hingga bisa mendapatkan akurasi yang lebih tinggi pada data *testing* yaitu data yang tidak pernah dilihat sebelumnya. Adapun sekilas proses *training* dan *testing* per *epoch* bisa dilihat pada tabel 4.1 dan tabel 4.2 dibawah.

Tabel 4.1 *Training* Layer FC *Learning Rate* 0.001

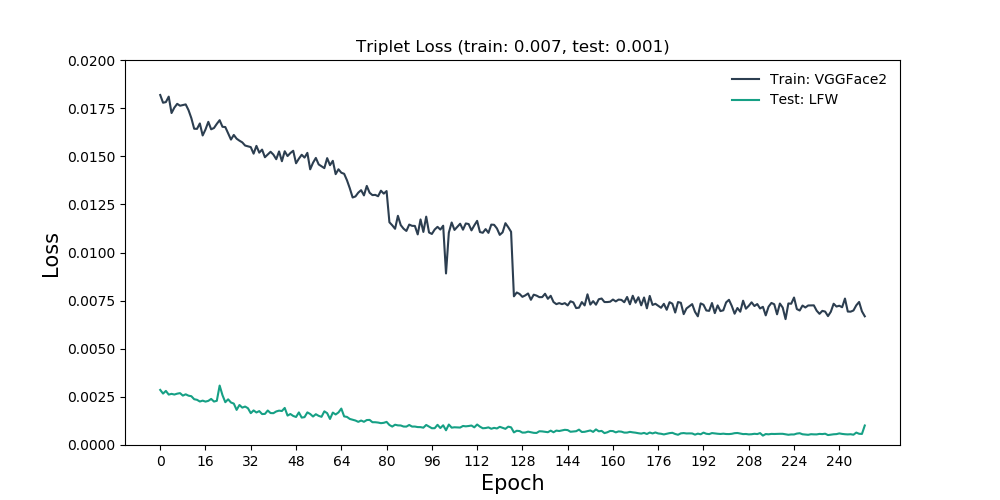
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Acc** | **Train Loss** | **Test Acc** | **Test Loss** |
| 0 | 0.5323 | 0.009 | 0.60095 | 0.00606 |
| 1 | 0.53595 | 0.00851 | 0.5984 | 0.00622 |
| 2 | 0.53135 | 0.0084 | 0.61645 | 0.00588 |
| 3 | 0.54005 | 0.00824 | 0.59525 | 0.0061 |
| 4 | 0.5455 | 0.00816 | 0.60715 | 0.00611 |
| 5 | 0.5522 | 0.00815 | 0.62735 | 0.00558 |
| 6 | 0.559 | 0.00809 | 0.64925 | 0.00517 |
| 7 | 0.5581 | 0.00814 | 0.6395 | 0.00539 |
| 8 | 0.5512 | 0.00808 | 0.654 | 0.0051 |
| 9 | 0.56175 | 0.0081 | 0.66425 | 0.00481 |
| 10 | 0.5721 | 0.00808 | 0.66525 | 0.00503 |
| 11 | 0.5734 | 0.00806 | 0.6737 | 0.00489 |
| 12 | 0.5772 | 0.00801 | 0.65555 | 0.00505 |
| 13 | 0.5773 | 0.00802 | 0.66465 | 0.00494 |
| 14 | 0.58345 | 0.00792 | 0.68775 | 0.00448 |
| 15 | 0.56665 | 0.00801 | 0.69575 | 0.00445 |
| 16 | 0.58875 | 0.00798 | 0.68235 | 0.00454 |
| 17 | 0.5845 | 0.00798 | 0.7012 | 0.0043 |
| 18 | 0.5854 | 0.00796 | 0.6944 | 0.00431 |
| 19 | 0.5869 | 0.00803 | 0.70405 | 0.0043 |
| 20 | 0.5952 | 0.00796 | 0.6902 | 0.00436 |
| 21 | 0.59785 | 0.00801 | 0.68945 | 0.00445 |
| 22 | 0.6023 | 0.00801 | 0.7112 | 0.00399 |
| 23 | 0.6017 | 0.00799 | 0.70255 | 0.00438 |
| 24 | 0.6034 | 0.00795 | 0.7315 | 0.00371 |
| 25 | 0.6158 | 0.00797 | 0.72735 | 0.00377 |
| 26 | 0.6116 | 0.00791 | 0.72785 | 0.00378 |
| 27 | 0.60915 | 0.00796 | 0.7212 | 0.00381 |
| 28 | 0.61135 | 0.008 | 0.72615 | 0.00365 |
| 29 | 0.61875 | 0.00792 | 0.72805 | 0.0038 |
| 30 | 0.6122 | 0.00798 | 0.7445 | 0.00347 |
| 31 | 0.6231 | 0.00791 | 0.7432 | 0.00352 |
| 32 | 0.62905 | 0.00799 | 0.73085 | 0.00362 |
| 33 | 0.63115 | 0.00792 | 0.728 | 0.00367 |
| 34 | 0.6352 | 0.00791 | 0.7346 | 0.00349 |
| 35 | 0.63445 | 0.00795 | 0.75055 | 0.00327 |
| … | … | … | … | … |
| … | … | … | … | … |
| 47 | 0.6558 | 0.00805 | 0.7739 | 0.00287 |
| 48 | 0.65725 | 0.00785 | 0.7645 | 0.00299 |
| 49 | 0.654 | 0.00792 | 0.77885 | 0.00275 |

Tabel 4.2 *Training* Layer FC *Learning* *Rate* 0.0001

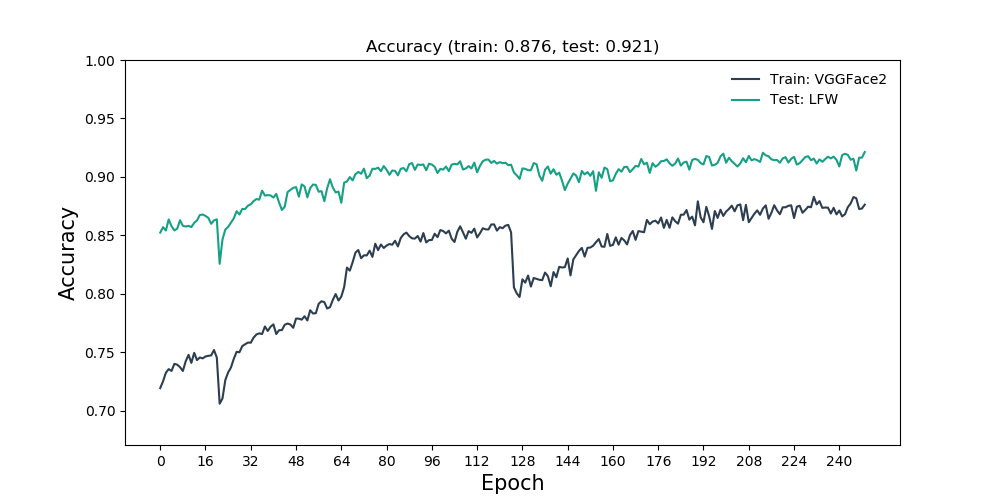
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Acc** | **Train Loss** | **Test Acc** | **Test Loss** |
| 50 | 0.67465 | 0.0079 | 0.7786 | 0.00271 |
| 51 | 0.6767 | 0.00772 | 0.7805 | 0.00262 |
| 52 | 0.67755 | 0.00787 | 0.79055 | 0.00263 |
| 53 | 0.67915 | 0.00783 | 0.78575 | 0.00257 |
| 54 | 0.68475 | 0.00776 | 0.79005 | 0.00258 |
| 55 | 0.69045 | 0.00775 | 0.79545 | 0.0025 |
| 56 | 0.6831 | 0.00789 | 0.79345 | 0.00244 |
| 57 | 0.6874 | 0.00778 | 0.792 | 0.00247 |
| 58 | 0.6897 | 0.00784 | 0.79215 | 0.00253 |
| 59 | 0.68855 | 0.00788 | 0.7951 | 0.0024 |
| 60 | 0.69025 | 0.00773 | 0.7994 | 0.00232 |
| 61 | 0.69855 | 0.00771 | 0.7976 | 0.00238 |
| 62 | 0.6893 | 0.00796 | 0.8024 | 0.00226 |
| 63 | 0.6898 | 0.0078 | 0.80645 | 0.00226 |
| 64 | 0.69885 | 0.00772 | 0.8015 | 0.00236 |
| 65 | 0.6911 | 0.00801 | 0.8034 | 0.00233 |
| 66 | 0.69505 | 0.00774 | 0.80415 | 0.0023 |
| 67 | 0.69735 | 0.00781 | 0.8082 | 0.00232 |
| 68 | 0.70375 | 0.00779 | 0.81195 | 0.00216 |
| 69 | 0.6968 | 0.00794 | 0.8053 | 0.0023 |
| 70 | 0.69955 | 0.00778 | 0.8107 | 0.00221 |
| 71 | 0.7006 | 0.00793 | 0.8135 | 0.0022 |
| 72 | 0.7032 | 0.00782 | 0.81375 | 0.00214 |
| 73 | 0.7032 | 0.00777 | 0.8173 | 0.00216 |
| 74 | 0.6971 | 0.00787 | 0.8076 | 0.00216 |
| 75 | 0.69805 | 0.00784 | 0.8102 | 0.00221 |
| 76 | 0.7041 | 0.00786 | 0.8177 | 0.00212 |
| 77 | 0.70985 | 0.00781 | 0.81595 | 0.00219 |
| 78 | 0.70165 | 0.00777 | 0.81625 | 0.00212 |
| 79 | 0.7078 | 0.00769 | 0.81425 | 0.00215 |
| 80 | 0.70515 | 0.00789 | 0.8254 | 0.00194 |
| 81 | 0.7048 | 0.00792 | 0.8093 | 0.0021 |
| … | … | … | … | … |
| … | … | … | … | … |
| 93 | 0.7121 | 0.0079 | 0.8266 | 0.00196 |
| 94 | 0.7108 | 0.00787 | 0.8143 | 0.00207 |
| 95 | 0.7119 | 0.00785 | 0.82185 | 0.00201 |
| 96 | 0.71875 | 0.00788 | 0.8138 | 0.00199 |
| 97 | 0.7138 | 0.00778 | 0.82195 | 0.00194 |
| 98 | 0.72 | 0.00789 | 0.8247 | 0.00192 |
| 99 | 0.7128 | 0.00794 | 0.82295 | 0.00197 |

* 1. **Layer Ekstraksi Fitur**

Layer ekstraksi fitur yang dimaksud adalah layer sebelum FC seperti yang terlampir pada gambar 3.13. Pada tahap ini model akan dilatih untuk mempelajari dataset baru yaitu dataset VGGv2 untuk mempelajari fitur wajah. Fitur *pretrained* yang dipelajari pada dataset ImageNet pada tahap ini akan disesuaikan ulang agar dapat mempelajari fitur wajah yang ada pada dataset VGGv2. Hasil belajar pada tahap ini bisa dilihat pada gambar 4.3 dan 4.4 dibawah.



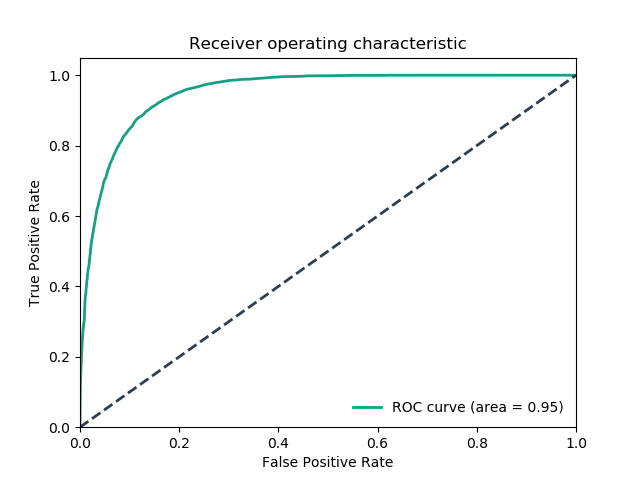
Gambar 4.3 Grafik *Triplet Loss* *Training* & *Testing* Hasil Latih Layer Ekstraksi Fitur



Gambar 4.4 Grafik Akurasi *Training* & *Testing* Hasil Latih Layer Ekstraksi Fitur

Terlihat pada gambar grafik 4.4 diatas, model dapat mencapai akurasi *testing* 90% dan akurasi *training* 83% pada *epoch* ke 70*.* Model kemudian dilatih sampai 100 epoch berikutnya model terlihat masih bertahan pada akurasi 90%. Pada *epoch* ke 250 model dapat mencapai akurasi 92% pada data *testing* dan mendapatkan akurasi 87% pada data *training.* Sampai pada epoch ke 250 proses *training* dihentikan dengan akurasi maksimal yang dicapai adalah 92%. Data sekilas *training* dan *testing* proses ini bisa dilihat pada tabel 4.3 dibawah.

Nilai akurasi *testing* yang lebih tinggi pada grafik diatas menjelaskan bahwa model cukup baik dalam *generalisasi* datasehingga mampu memprediksi data yang tidak pernah dilihat sebelumnya dengan akurasi yang cukup tinggi*.*  Model juga berhasil mendapat nilai AUC 95% seperti terlihat pada gambar 4.5 dibawah.



Gambar 4.5 Grafik ROC Proses Training Terakhir

Nilai AUC mencapai 95% bisa diinterpretasikan bahwa model dapat memprediksi dengan benar gambar wajah orang yang sama dan tidak sama dengan persentase 95% selama proses pengujian.

Tabel 4.3 *Training* Layer Ekstraksi Fitur

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Acc** | **Train Loss** | **Test Acc** | **Test Loss** |
| 100 | 0.7196 | 0.01819 | 0.85245 | 0.00286 |
| 101 | 0.7254 | 0.01778 | 0.8571 | 0.00267 |
| 102 | 0.7327 | 0.01782 | 0.85425 | 0.00281 |
| 103 | 0.7358 | 0.0181 | 0.86375 | 0.00262 |
| 104 | 0.73425 | 0.01725 | 0.8579 | 0.00266 |
| 105 | 0.7403 | 0.01753 | 0.85435 | 0.00262 |
| 106 | 0.7396 | 0.01773 | 0.85585 | 0.00267 |
| 107 | 0.73745 | 0.01763 | 0.86305 | 0.0027 |
| 108 | 0.7342 | 0.01766 | 0.85825 | 0.00257 |
| 109 | 0.74245 | 0.0177 | 0.8577 | 0.00263 |
| 110 | 0.748 | 0.0174 | 0.8582 | 0.00256 |
| 111 | 0.74115 | 0.01698 | 0.8572 | 0.00254 |
| 112 | 0.7497 | 0.01643 | 0.86075 | 0.00238 |
| 113 | 0.7435 | 0.01643 | 0.86295 | 0.00235 |
| 114 | 0.74575 | 0.01671 | 0.8675 | 0.00226 |
| 115 | 0.74495 | 0.01608 | 0.86795 | 0.0023 |
| 116 | 0.7466 | 0.01639 | 0.8667 | 0.00225 |
| 117 | 0.74715 | 0.01679 | 0.86505 | 0.00229 |
| 118 | 0.7475 | 0.0164 | 0.86 | 0.00239 |
| 119 | 0.75215 | 0.01647 | 0.86285 | 0.00226 |
| 120 | 0.74565 | 0.01669 | 0.86385 | 0.0023 |
| 121 | 0.70625 | 0.01688 | 0.8258 | 0.00309 |
| 122 | 0.7107 | 0.01653 | 0.84655 | 0.00259 |
| 123 | 0.72645 | 0.01652 | 0.8551 | 0.00222 |
| 124 | 0.73295 | 0.01619 | 0.8573 | 0.00237 |
| 125 | 0.7372 | 0.01587 | 0.861 | 0.00221 |
| 126 | 0.7446 | 0.01611 | 0.8647 | 0.00215 |
| 127 | 0.7506 | 0.01592 | 0.87075 | 0.00182 |
| 128 | 0.7502 | 0.01581 | 0.86795 | 0.00208 |
| 129 | 0.75545 | 0.01572 | 0.8727 | 0.00194 |
| 130 | 0.75705 | 0.01556 | 0.87265 | 0.002 |
| … | … | … | … | … |
| 340 | 0.8713 | 0.00724 | 0.909 | 0.0006 |
| 341 | 0.86631 | 0.00716 | 0.91875 | 0.00058 |
| 342 | 0.86812 | 0.00761 | 0.9198 | 0.00056 |
| 343 | 0.87413 | 0.00693 | 0.91895 | 0.00055 |
| 344 | 0.87732 | 0.00693 | 0.91475 | 0.00056 |
| 345 | 0.88285 | 0.00699 | 0.9156 | 0.00053 |
| 346 | 0.88187 | 0.00726 | 0.9055 | 0.00064 |
| 347 | 0.87266 | 0.00743 | 0.91655 | 0.00058 |
| 348 | 0.87302 | 0.00694 | 0.9165 | 0.00057 |
| 349 | 0.87636 | 0.00669 | 0.92135 | 0.00101 |

1. **Hasil Belajar**

Pada proses *training,* setiap layer ResNet (gambar 3.13) akan dilatih agar bisa mendapatkan representasi fitur sebuah gambar (dalam hal ini wajah). Setiap layer ResNet memiliki parameter (*weight* dan *bias*) yang akan dilatih agar memenuhi tujuan yang dijelaskan pada sub bab 3.3.1. Setiapmasukan (*input*) gambar akan melalui setiap layer ResNet (gambar 3.13) sebagai bahan latih (*training*) layer ResNet agar berhasil mempelajari fitur yang ada pada wajah dengan akurasi maksimal.

Pada sub bab 4.2 telah dijelaskan proses *training* yang telah dilalui layer ResNet dan mendapatkan akurasi 92% pada data *testing.* Pada sub bab ini akan ditampilkan fitur yang telah berhasil dipelajari layer ResNet sehingga berhasil mendapatkan akurasi 92%. Hasil belajar setiap layer akan dijelaskan pada poin-poin dibawah.

1. Layer Conv1

Layer ini adalah layer konvolusi yang pertama kali dilalui oleh *input.* Layer ini membutuhkan gambar *input* dengan dimensi 224x224 dan memiliki 3 kanal warna, contoh gambar input yang digunakan bisa dilihat pada gambar 4.6. Layer ini memiliki 64 filter/kernel dengan dimensi 7x7 sehingga output dari layer ini adalah 64 gambar dengan dimensi 112x112. Hasil fitur yang dipelajari pada layer ini bisa dilihat pada gambar 4.7 dibawah.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| R | G | B |
| Gambar 4.6 Gambar *Input* Setelah Tahap *Preprocessing* | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.7 *Output* Layer Conv1

1. Layer BN1

Layer BN1 adalah layer *Batch Normalization*, pada layer ini hasil conv1 pada layer sebelumnya akan dinormalisasi sebelum dilanjutkan ke layer aktivasi (layer ReLU). *Batch Normalization* berfungsi untuk mencegah *overfit* pada arsitektur jaringan. Hasil proses dari layer ini bisa dilihat pada gambar 4.8 dibawah.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.8 *Output* Layer BN1

1. Layer ReLU

Layer ReLU adalah layer aktivasi yang akan dilalui oleh *input* setelah melalui layer BN1. Pada layer ini hasil dari BN1 akan diaktivasi dengan fungsi *Rectified Linear Unit* (ReLU). *Output* dari layer ini bisa dilihat pada gambar 4.9 dibawah.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.9 *Output* Layer ReLU

1. Layer Max Pool

Pada layer ini gambar yang telah diaktivasi pada layer sebelumnya akan melalui proses *Max Pool*. Setiap gambar yang melalui layer ini akan memiliki dimensi 56x56 karena proses *Max Pool.* Hasil dari proses *Max* *Pool* bisa dilihat pada gambar 4.10 dibawah.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.10 *Output* Layer *Max* *Pool*

1. Layer1

Layer1 adalah layer CNN pertama yang ada pada arsitektur ResNet. Pada layer ini hasil dari MaxPool akan diproses, *output* dari layer ini adalah fitur wajah yang berhasil dipelajari berjumlah 256 gambar dengan dimensi 56x56. Pada gambar 4.11 ditampilkan 64 gambar pertama hasil dari proses layer1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.11 *Output* Layer1

1. Layer2

Layer2 adalah layer CNN kedua pada arsitektur ResNet. Hasil dari layer1 sebelumnya akan diproses pada layer ini. Layer2 akan menghasilkan fitur wajah berjumlah 512 gambar dengan dimensi 28x28. Pada gambar 4.12 ditampilkan 64 gambar pertama dari hasil belajar pada layer ini.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.12 *Output* Layer2

1. Layer3

Layer3 adalah layer CNN ketiga pada arsitektur ResNet. Layer ini akan mempelajari 512 gambar fitur wajah pada layer2 kemudian menghasilkan 1024 fitur wajah yang berhasil dipelajari dengan dimensi 14x14. Pada gambar 4.13 dibawah ditampilkan 64 dari 1024 gambar fitur wajah yang telah dipelajari.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.13 *Output* Layer3

1. Layer4

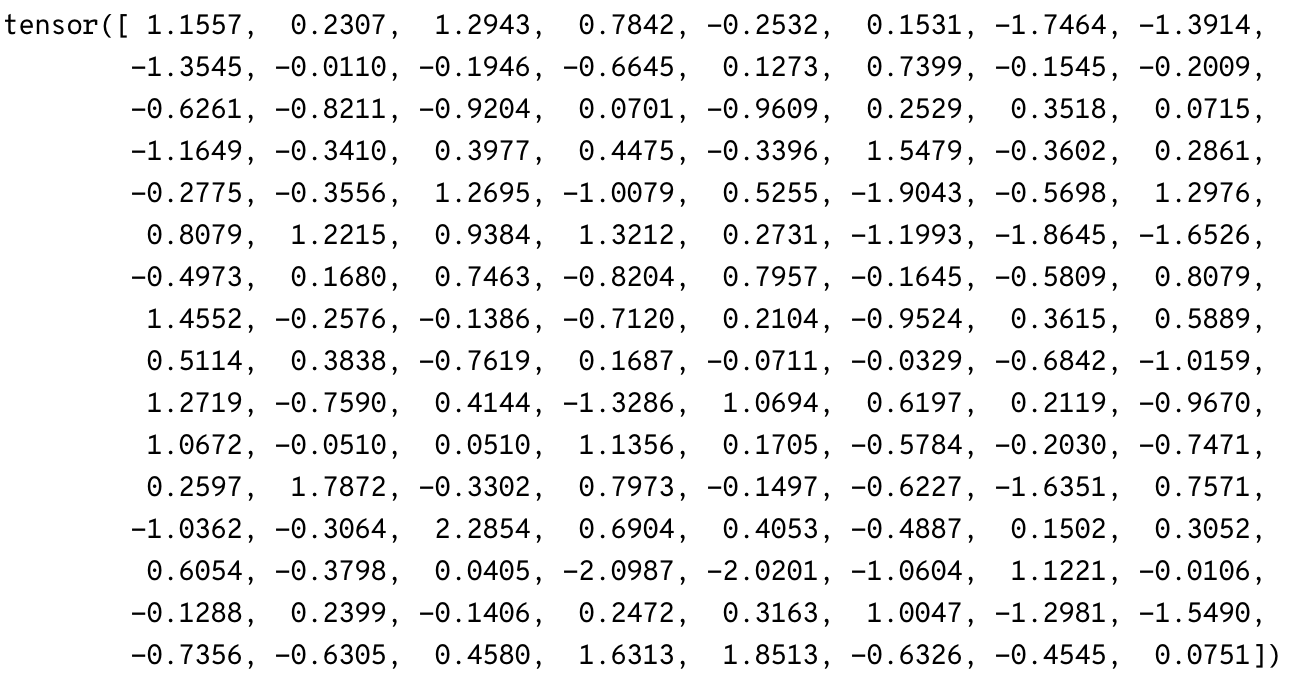
Layer4 pada arsitektur ResNet adalah layer CNN terakhir, layer ini yang mempelajari paling banyak pola yang ada pada gambar wajah. Layer ini menghasilkan 2048 gambar fitur wajah yang telah berhasil dipelajari pada layer CNN sebelumnya (layer3). Gambar fitur wajah pada layer ini berdimensi 7x7. Fitur gambar wajah dengan dimensi 7x7 bisa mempelajari setiap detail yang ada pada wajah dengan sangat rinci. Setiap lekukan dan garis yang ada pada wajah akan dipelajari pada layer ini. Pada gambar 4.14 dibawah ditampilkan 64 gambar pertama hasil proses dari layer4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Gambar 4.14 *Output* Layer4

1. FC

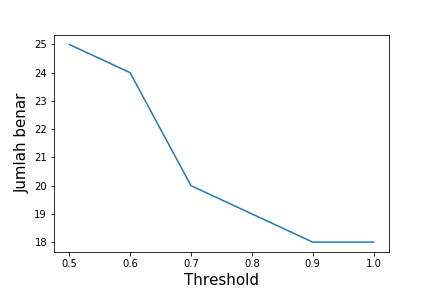
*Fully Connected* *Layer* (FC) adalah layer yang akan mempelajari dan mereduksi fitur yang dihasilkan pada layer3 hingga menghasilkan *embedding* dengan dimensi 1x128 yang memenuhi objektif yang dijelaskan pada sub bab 3.3.1. Fitur yang telah direduksi ini yang disebut dengan *feature vector* yang bisa dibandingkan dengan *feature* *vector* dari input gambar yang lain. Dengan fungsi jarak seperti *Euclidean distance* jarak antar dua *feature vector* bisa dibandingkan kedekatannya. Hasil *embedding* yang berhasil dipelajari dari proses layer3 bisa dilihat pada gambar 4.15 dibawah.



Gambar 4.15 Hasil *Embedding* Layer FC

1. **Percobaan Verifikasi Wajah**

Pada sub bab ini akan ditampilkan hasil percobaan pembandingan 30 pasangan gambar. Untuk mendapatkan *embedding* dari setiap pasang gambar, akan dilakukan proses yang sama seperti yang dijelaskan pada poin-poin pada sub bab 4.3. Jarak antar pasangan *embedding* akan dihitung menggunakan *Euclidean distance.* Jarak yang didapatkan kemudian akan dibandingkan dengan ambang toleransi kedekatan tertentu (*threshold*), jika jarak *embedding* pasangan gambar lebih besar dari nilai *threshold* maka gambar akan dianggap orang yang berbeda dan jika dibawah nilai *threshold* akan dianggap orang yang sama. Pada percobaan ini akan digunakan beberapa nilai *threshold* yaitu 0.5, 0.6, 0.7, 0.8, 0.9 dan 1.0. Hasil dari percobaan ini bisa dilihat pada gambar grafik 4.16 dibawah.



Gambar 4.16 Jumlah Prediksi Benar Setiap Nilai *Threshold*

Pada gambar grafik diatas jumlah prediksi benar setiap nilai threshold berbeda-beda. Terlihat nilai *threshold* 0.5 mendapatkan jumlah prediksi benar terbanyak yaitu 25 dari 30 percobaan dan terlihat semakin tinggi nilai *threshold* yang digunakan semakin kecil jumlah prediksi benar yang didapat. Daftar percobaan pasangan gambar bisa dilihat pada tabel dibawah.

Tabel 4.4 Hasil Perbandingan Pasangan Gambar Threshold 0.5

|  |  |  |
| --- | --- | --- |
| **No** | **Threshold 0.5** | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| … | … | … |
| 30 |  |  |

Tabel 4.5 Hasil Perbandingan Pasangan Gambar Threshold 0.6

|  |  |  |
| --- | --- | --- |
| **No** | **Threshold 0.6** | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| … | … | … |
| 30 |  |  |

Tabel 4.6 Hasil Perbandingan Pasangan Gambar Threshold 0.7

|  |  |  |
| --- | --- | --- |
| **No** | **Threshold 0.7** | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| … | … | … |
| 30 |  |  |

Tabel 4.7 Hasil Perbandingan Pasangan Gambar Threshold 0.8

|  |  |  |
| --- | --- | --- |
| **No** | **Threshold 0.8** | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| … | … | … |
| 30 |  |  |

Tabel 4.8 Hasil Perbandingan Pasangan Gambar Threshold 0.9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Threshold 0.9** | | | |
| 1 |  |  | | |
| 2 |  |  | | |
| 3 |  |  | | |
| 4 |  |  | | |
| 5 |  |  | | |
| … | … | | … |
| 30 |  |  | | |

Tabel 4.9 Hasil Perbandingan Pasangan Gambar Threshold 1.0

|  |  |  |
| --- | --- | --- |
| **No** | **Threshold 1.0** | |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| … | … | … |
| 5 |  |  |
| 30 |  |  |