1. *Build a convolutional network with pre-processing on the input data (jittering, normalization). Also add dropout regularization, batch normalization, and at least one additional convolutional layer which achieves at least 90% test accuracy (for any training epoch) on MNIST dataset. Your part 1 network should train under 10 minutes, without GPUs.*

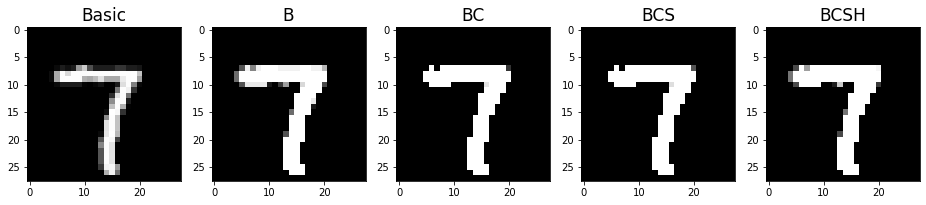
Jawab:

Eksperimen dilakukan dengan menggunakan *framework* Pytorch dengan dokumentasi yang disajikan pada link: ………..

* Gambar pada dataset MNIST berukuran 28x28 dengan channel sebesar 1. Dataset MNIST terdiri dari 60,000 data training dan 10,000 data testing. Data tersebut akan dipreprocess dengan menggunakan color jittering dan normalization. Visualisasi terhadap data yang telah melalui tahap preprocessing disajikan pada Gambar 1. Pada visualisasi yang disajikan pada Gambar 1, dilakukan uji coba nilai Brightness (B), Contrast (C), Saturation (S), dan Hue (H).

Berdasarkan pengertiannya, secara umum:

1. Brightness merujuk pada kecerahan atau kegelapan keseluruhan gambar. Brightness dengan nilai tinggi terlihat lebih terang sedangkan brightness rendah terlihat lebih gelap.
2. Contrast, merujuk pada perbedaan antara area terang dan gelap dalam gambar. Kontras dengan nilai tinggi memiliki perbedaan yang lebih besar antara area terang dan gelap, sedangkan kontras dengan nilai rendah memiliki perbedaan yang lebih kecil.
3. Hue merupakan istilah yang digunakan untuk menggambarkan nama dari warna-warna yang ada, sebagai contoh merah, biru, atau hijau.
4. Saturation merujuk pada intensitas warna dalam gambar (kepekatan warna atau kuat/lemahnya kemurnian sebuah warna).



(e)

(d)

(c)

(b)

(a)

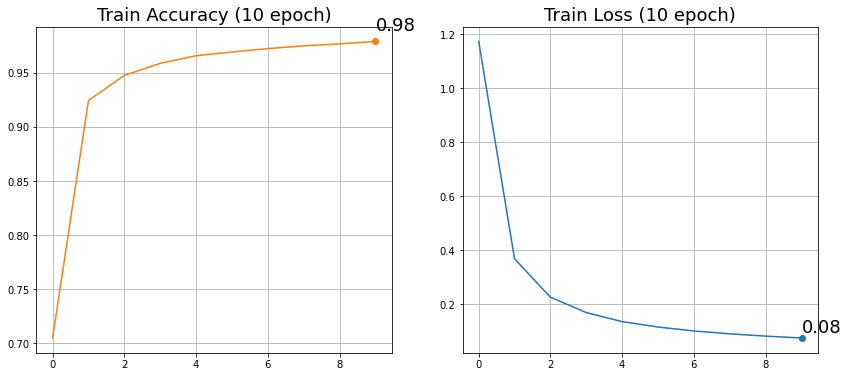
Gambar 1 Visualisasi data pada dataset MNIST (a) gambar tanpa dilakukan preprocessing (b) preprocessing dengan Brightness sebesar 10, (c) preprocessing dengan Brightness sebesar 10 dan Contrast sebesar 10 (d) preprocessing dengan Brightness sebesar 10, Contrast sebesar 10, dan Saturation sebesar 10 (e) preprocessing dengan Brightness sebesar 10, Contrast sebesar 10, Saturation sebesar 10, dan Hue sebesar 10

* Eksperimen dilakukan dengan menggunakan arsitektur CNN (RegularCNN) dengan total 4 buah layer Convolution dan 4 buah layer BatchNormalization. Eksperimen juga menggunakan MaxPooling, dropout, dan fungsi aktivasi Leaky Relu dengan nilai alpha 0.1. Setelah proses konvolusi selesai dilakukan, maka dilanjutkan dengan layer Linear dengan jumlah output sebesar label dari MNIST (10). Secara detail layer yang digunakan pada arsitektur disajikan pada Tabel 1.

Tabel 1 Detail arsitektur CNN (RegularCNN)

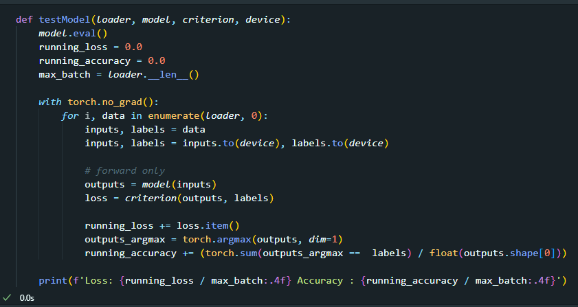
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Layer** | **Detail** | **Output shape [*batch x channel x height x width*]** | **Param#** |
| 1 | Convolution 2D | In\_channel = 1 out\_channel = 8 kernel = 3x3 stride = 1 padding = same bias = False | [-1 x 8 x 28 x 28] | 72 |
| 2 | BatchNorm 2D | Num\_features=8 epsilon = 1e-5  Momentum = 0.1 | [-1 x 8 x 28 x 28] | 16 |
| 3 | Leaky Relu | Alpha = 0.1 | [-1 x 8 x 28 x 28] | 0 |
| 4 | Convolution 2D | In\_channel = 8 out\_channel = 8 kernel = 3x3 stride = 1 padding = same bias = False | [-1 x 8 x 28 x 28] | 572 |
| 5 | BatchNorm 2D | Num\_features=8 epsilon = 1e-5  Momentum = 0.1 | [-1 x 8 x 28 x 28] | 16 |
| 6 | Leaky Relu | Alpha = 0.1 | [-1 x 8 x 28 x 28] | 0 |
| 7 | Dropout | Prob = 0.2 | [-1 x 8 x 28 x 28] | 0 |
| 8 | Maxpool | Kernel = 2x2  Stride = 2 | [-1 x 8 x 14 x 14] | 0 |
| 9 | Convolution 2D | In\_channel = 8 out\_channel = 16 kernel = 3x3 stride = 1 padding = same bias = False | [-1 x 16 x 14 x 14] | 1152 |
| 10 | BatchNorm 2D | Num\_features = 16 epsilon = 1e-5  Momentum = 0.1 | [-1 x 16 x 14 x 14] | 32 |
| 11 | Leaky Relu | Alpha = 0.1 | [-1 x 16 x 14 x 14] | 0 |
| 12 | Convolution 2D | In\_channel = 16 out\_channel = 16 kernel = 3x3 stride = 1 padding = same bias = False | [-1 x 16 x 14 x 14] | 2304 |
| 13 | BatchNorm 2D | Num\_features = 16  epsilon = 1e-5  Momentum = 0.1 | [-1 x 16 x 14 x 14] | 32 |
| 14 | Leaky Relu | Alpha = 0.1 | [-1 x 16 x 14 x 14] | 0 |
| 15 | Dropout | Prob = 0.2 | [-1 x 16 x 14 x 14] | 0 |
| 16 | Maxpool | Kernel = 2x2  Stride = 2 | [-1 x 16 x 7 x 7] | 0 |
| 17 | Linear | In\_features = 784  Out\_features = 10  Bias = True | [-1 x 10] | 7850 |

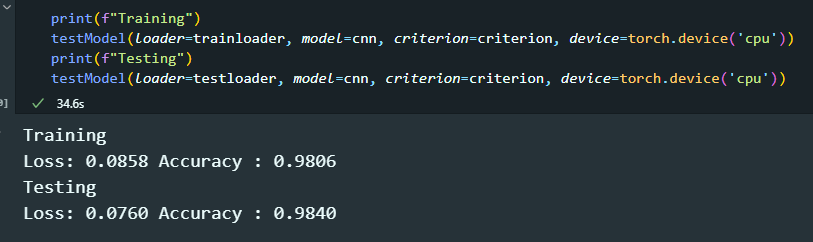
* Total jumlah parameter dari arsitektur tersebut sebesar 12050.
* Eksperimen dilakukan dengan menggunakan fungsi loss CrossEntropy dan fungsi optimasi Adam dengan nilai learning rate sebesar 1e-4. Jumlah epoch yang digunakan pada proses pelatihan sebesar 10 dan ukuran batch yang digunakan sebesar 256. Hasil proses pelatihan diilustrasikan pada Gambar 2. Proses pelatihan dilakukan dengan menggunakan CPU selama 477.19 detik (7.95 menit).



Gambar 2 Hasil training arsitektur CNN pada dataset MNIST

* Berdasarkan hasil eksperimen yang telah dilakukan pada dataset MNIST, model RegularCNN mampu memperoleh nilai akurasi sebesar 98% dengan nilai loss sebesar 0.08. Selanjutnya model akan dievaluasi dengan data training dan data testing, hasil evaluasi disajikan pada Tabel 2. Kode untuk evaluasi disajikan pada Gambar 3.





(b)

(a)

Gambar 3 Evaluasi model yang telah dilatih (a) fungsi untuk melakukan evaluasi (b) cuplikan hasil evaluasi

Tabel 2 Hasil evaluasi RegularCNN pada dataset MNIST

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Waktu Training**  **(menit)** | **Loss** | | **Accuracy** | |
| Train | Test | Train | Test |
| RegularCNN | 7.95 | 0.0858 | 0.0760 | 0.9806 | 0.9849 |

* Berdasarkan hasil evaluasi yang disajikan pada Tabel 2, model RegularCNN mampu memperoleh nilai akurasi yang baik pada data train ataupun test yaitu dengan nilai akurasi sebesar 0.9806 dan 0.9849 dengan nilai loss yang kecil yaitu sebesar 0.0858 dan 0.0760 pada data train dan test.

1. Fine-tune AlexNet to achieve at least 80% test accuracy on the MNIST dataset. Your network should train under 10 minutes, without GPUs.

Jawab:

* Eksperimen dilakukan untuk mengklasifikasikan digit pada dataset MNIST menggunakan arsitektur AlexNet. Eksperimen dilakukan dengan menggunakan bobot ImageNet pada arsitektur AlexNet dan tidak menggunakan bobot ImageNet.

1. **Without Pre-trained**

Model AlexNet memiliki ukuran kernel dan stride yang cukup besar untuk dilatih pada dataset MNIST yang memiliki ukuran tinggi dan lebar sebesar 28x28. Oleh karena itu pada eksperimen pertama dilakukan pemangkasan layer pada AlexNet yaitu dengan menggunakan 5 layer pertama (tanpa menggunakan bobot dari ImageNet). Dikarenakan AlexNet menerima masukan channel sebesar 3 sedangkan MNIST hanya memiliki 1 channel, maka pada bagian awal arsitektur AlexNet digunakan layer convolutional untuk menghasilkan 3 channel dengan padding same sehingga ukuran tinggi dan lebar gambar tidak berubah. Secara detail, implementasi model AlexNet yang termodifikasi disajikan pada Tabel 3.

Tabel 3 Detail arsitektur AlexNet 5 Layer [Tanpa bobot ImageNet]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Layer** | **Detail** | **Output shape [*batch x channel x height x width*]** | **Param#** |
| 1 | Convolution 2D | In\_channel = 1 out\_channel = 3 kernel = 1x1 stride = 1 padding = same bias = True | [-1, 3, 28, 28] | 6 |
| 2 | Convolution 2D (AlexNet) | In\_channel = 3 out\_channel = 64 kernel = 11x11 stride = 4 padding = 2 bias = True | [-1, 64, 6, 6] | 23,296 |
| 3 | ReLU (AlexNet) | - | [-1, 64, 6, 6] | 0 |
| 4 | Maxpool (AlexNet) | Kernel = 3  Stride = 2 | [-1, 64, 2, 2] | 0 |
| 5 | Convolution 2D (AlexNet) | In\_channel = 64 out\_channel = 192 kernel = 5x5 stride = 1 padding = 2 bias = True | [-1, 192, 2, 2] | 307,392 |
| 6 | ReLU (AlexNet) | - | - | 0 |
| 7 | AdaptiveAvgPool  (AlexNet) | Output\_size = 6 x 6 | [-1, 192, 6, 6] | 0 |
| 8 | Flatten | - | [-1, 6192] | 0 |
| 9 | Linear | In\_features = 6912  Out\_features = 32  Bias = False | [-1, 32] | 221,184 |
| 10 | BatchNorm 1D | Num\_features=32 epsilon = 1e-5  Momentum = 0.1 | [-1, 32] | 64 |
| 11 | Leaky Relu | Alpha = 0.1 | [-1, 32] | 0 |
| 12 | Linear | In\_features = 32  Out\_features = 10  Bias = True | [-1, 10] | 330 |

* Total jumlah parameter dari arsitektur AlexNet 5 Layer sebesar 552,272.
* Eksperimen dilakukan dengan menggunakan fungsi loss CrossEntropy dan fungsi optimasi Adam dengan nilai learning rate sebesar 1e-4. Jumlah epoch yang digunakan pada proses pelatihan sebesar 10 dan ukuran batch yang digunakan sebesar 256. Proses pelatihan dilakukan dengan menggunakan CPU selama 355.82 detik (5.93 menit). Hasil pengujian dan perbandingan dengan arsitektur RegularCNN disajikan pada Tabel 4. Seluruh bobot dilakukan pelatihan **tanpa ada** freeze layer.

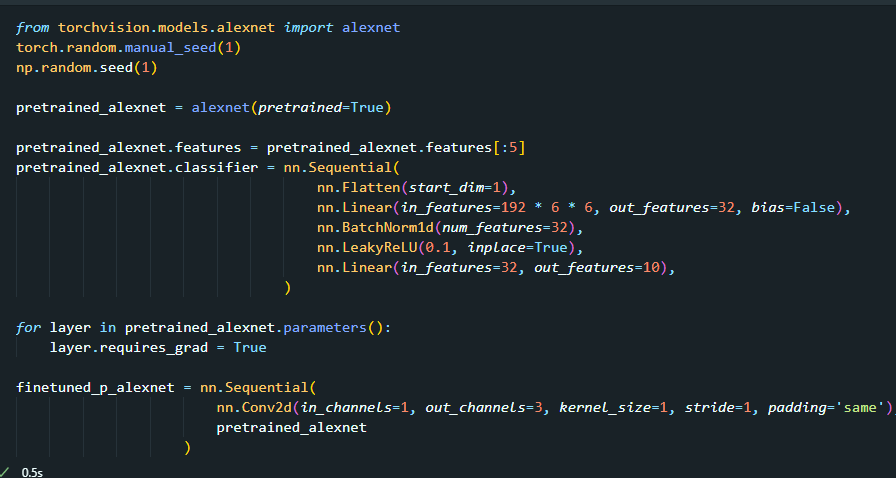
Tabel 4 Hasil evaluasi 5 layer pertama AlexNet tanpa bobot ImageNet pada dataset MNIST

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Waktu Training**  **(menit)** | **Loss** | | **Accuracy** | |
| Train | Test | Train | Test |
| RegularCNN | 7.95 | 0.0858 | **0.0760** | 0.9806 | **0.9849** |
| AlexNet 5 Layer [Tanpa bobot ImageNet] | 5.93 | **0.0791** | 0.1023 | **0.9929** | 0.9839 |

* Berdasarkan hasil pengujian, model AlexNet 5 Layer tanpa menggunakan bobot ImageNet memperoleh nilai loss dan akurasi yang lebih baik pada tahap training dibandingkan dengan RegularCNN, sedangkan pada tahap pengujian model RegularCNN mampu memperoleh nilai yang sedikit lebih baik dibandingkan dengan AlexNet.

1. **With Pre-trained**

* Eksperimen dilanjutkan dengan menggunakan model arsitektur AlexNet yang sama seperti yang disajikan pada Tabel 3. Namun pada eksperimen ini, kelima layer tersebut menggunakan bobot ImageNet dan dilakukan pelatihan pada dataset MNIST tanpa freeze layer. Kode untuk menggunakan bobot ImageNet pada AlexNet disajikan pada Gambar 4.



Gambar 4 Source code untuk menggunakan bobot ImageNet pada AlexNet

* Total jumlah parameter dari arsitektur tersebut sebesar 552,272.
* Eksperimen dilakukan dengan menggunakan fungsi loss CrossEntropy dan fungsi optimasi Adam dengan nilai learning rate sebesar 1e-4. Jumlah epoch yang digunakan pada proses pelatihan sebesar 10 dan ukuran batch yang digunakan sebesar 256. Proses pelatihan dilakukan dengan menggunakan CPU selama 349.36 detik (5.82 menit). Hasil pengujian dan perbandingan dengan arsitektur sebelumnya disajikan pada Tabel 5. Seluruh bobot dilakukan pelatihan **tanpa ada** freeze layer.

Tabel 5 Hasil evaluasi 5 layer pertama AlexNet dengan bobot ImageNet pada dataset MNIST

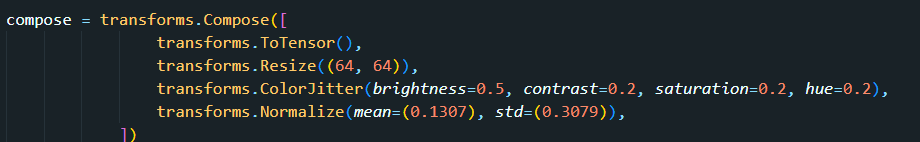
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Waktu Training**  **(menit)** | **Loss** | | **Accuracy** | |
| Train | Test | Train | Test |
| RegularCNN | 7.95 | 0.0858 | **0.0760** | 0.9806 | **0.9849** |
| AlexNet 5 Layer [Tanpa bobot ImageNet] | 5.93 | **0.0791** | 0.1023 | **0.9929** | 0.9839 |
| AlexNet 5 Layer [Dengan bobot ImageNet] | 5.82 | 0.1030 | 0.1136 | 0.9871 | 0.9811 |

* Berdasarkan hasil pengujian, model AlexNet 5 Layer dengan menggunakan bobot ImageNet memperoleh nilai loss dan akurasi yang kurang baik jika dibandingkan dengan arsitektur sebelumnya. Oleh karena itu eksperimen dilanjutkan dengan menggunakan seluruh layer pada arsitektur AlexNet dengan bobot ImageNet.
* Arsitektur AlexNet secara lengkap disajikan pada Tabel 6.

Tabel 6 Detail arsitektur AlexNet [Dengan bobot ImageNet]

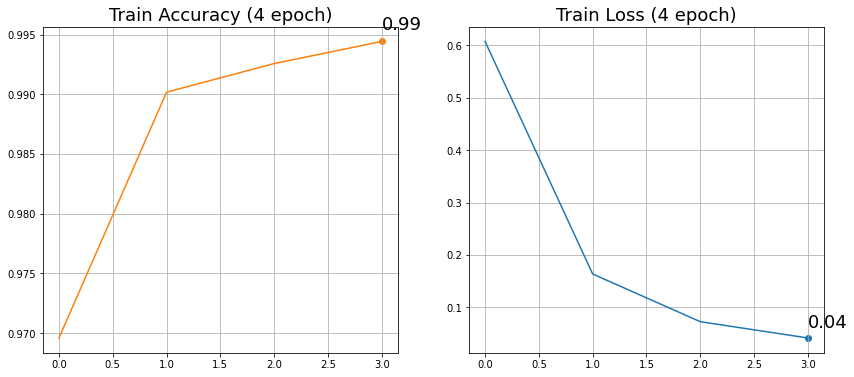
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Layer** | **Detail** | **Output shape [*batch x channel x height x width*]** | **Param#** |
| 1 | Convolution 2D | In\_channel = 1 out\_channel = 3 kernel = 1x1 stride = 1 padding = same bias = True | [-1, 3, 64, 64] | 6 |
| 2 | Convolution 2D (AlexNet) | In\_channel = 3 out\_channel = 64 kernel = 11x11 stride = 4 padding = 2 bias = True | [-1, 64, 15, 15] | 23,296 |
| 3 | ReLU (AlexNet) | - | [-1, 64, 15, 15] | 0 |
| 4 | Maxpool (AlexNet) | Kernel = 3  Stride = 2 | [-1, 64, 7, 7] | 0 |
| 5 | Convolution 2D (AlexNet) | In\_channel = 64 out\_channel = 192 kernel = 5x5 stride = 1 padding = 2 bias = True | [-1, 192, 7, 7] | 307,392 |
| 6 | ReLU (AlexNet) | - | [-1, 192, 7, 7] | 0 |
| 7 | Maxpool (AlexNet) | Kernel = 3x3  Stride = 1  Padding = 1 | [-1, 192, 3, 3] | 0 |
| 8 | Convolution 2D (AlexNet) | In\_channel = 192 out\_channel = 384 kernel = 3x3 stride = 1 padding = 1 bias = True | [-1, 384, 3, 3] | 663,936 |
| 9 | ReLU (AlexNet) | - | [-1, 384, 3, 3] | 0 |
| 10 | Convolution 2D (AlexNet) | In\_channel = 384 out\_channel = 256 kernel = 3x3 stride = 1 padding = 1 bias = True | [-1, 256, 3, 3] | 884,992 |
| 11 | ReLU (AlexNet) | - | [-1, 256, 3, 3] | 0 |
| 12 | Convolution 2D (AlexNet) | In\_channel = 256 out\_channel = 256 kernel = 3x3 stride = 1 padding = 1 bias = True | [-1, 256, 3, 3] | 590,080 |
| 13 | ReLU (AlexNet) | - | [-1, 256, 3, 3] | 0 |
| 14 | Maxpool (AlexNet) | Kernel = 3x3  Stride = 2  Padding = 0 | [-1, 256, 1, 1] | 0 |
| 15 | AdaptiveAvgPool  (AlexNet) | Output\_size = 6 x 6 | [-1, 256, 6, 6] | 0 |
| 16 | Flatten |  | [-1, 9216] | 0 |
| 17 | Linear | In\_features = 9216  Out\_features = 32  Bias = False | [-1, 32] | 294,912 |
| 18 | BatchNorm 1D | Num\_features=32 epsilon = 1e-5  Momentum = 0.1 | [-1, 32] | 64 |
| 19 | Leaky Relu | Alpha = 0.1 | [-1, 32] | 0 |
| 20 | Linear | In\_features = 32  Out\_features = 10  Bias = True | [-1, 10] | 330 |
| 21 | Linear | In\_features = 32  Out\_features = 10  Bias = True | [-1, 10] | 0 |

* Untuk menggunakan seluruh arsitektur AlexNet maka diperlukan untuk memperbesar ukuran gambar agar data gambar dapat diproses oleh arsitektur AlexNet. Pada eksperimen kali ini, ukuran gambar diubah menjadi 64x64 dengan proses transform yang disajikan pada Gambar 5.



Gambar 5 Proses transform memperbesar ukuran dataset MNIST

* Eksperimen dilakukan dengan menggunakan fungsi loss CrossEntropy dan fungsi optimasi Adam dengan nilai learning rate sebesar 1e-4. Jumlah epoch yang digunakan pada proses pelatihan sebesar 4 dan ukuran batch yang digunakan sebesar 64. Proses pelatihan model AlexNet disajikan pada Gambar 6.



Gambar 6 Proses pelatihan model AlexNet dengan bobot ImageNet pada dataset MNIST

* Proses pelatihan dilakukan dengan menggunakan CPU selama 495.65 detik (8.26 menit). Hasil pengujian dan perbandingan dengan arsitektur sebelumnya disajikan pada Tabel 7. Seluruh bobot dilakukan pelatihan **tanpa ada** freeze layer.

Tabel 7 Hasil evaluasi AlexNet dengan bobot ImageNet pada dataset MNIST

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Waktu Training**  **(menit)** | **Loss** | | **Accuracy** | |
| Train | Test | Train | Test |
| RegularCNN | 7.95 | 0.0858 | 0.0760 | 0.9806 | 0.9849 |
| AlexNet 5 Layer [Tanpa bobot ImageNet] | 5.93 | 0.0791 | 0.1023 | 0.9929 | 0.9839 |
| AlexNet 5 Layer [Dengan bobot ImageNet] | 5.82 | 0.1030 | 0.1136 | 0.9871 | 0.9811 |
| AlexNet [Dengan bobot ImageNet] | 8.26 | **0.0265** | **0.0386** | **0.9963** | **0.9919** |

1. **Kesimpulan**

* Berdasarkan hasil eksperimen yang telah dilakukan pada arsitektur RegularCNN dan AlexNet dalam melakukan klasifikasi digit pada dataset MNIST, model tersebut berhasil melakukan pembelajaran dengan baik dengan menerapkan layer konvolusi dalam pembelajaran data gambar.
* Dengan menggunakan konsep arsitektur yang sudah ada, proses pelatihan model dapat dipercepat dan cukup memberikan hasil metrik evaluasi yang baik. Pada kasus ini model AlexNet dengan 5 layer mampu memperoleh akurasi sebesar 98% dengan waktu 2 menit lebih sedikit jika dibandingkan dengan RegularCNN.
* Berdasarkan hasil eksperimen, arsitektur AlexNet dengan bobot ImageNet dapat mempercepat proses pelatihan dan model menjadi konvergen. Hal ini dibuktikan dengan nilai Loss yang diperoleh pada saat pelatihan dan pengujian lebih kecil dibandingkan dengan arsitektur pembanding sebelumnya yang disajikan pada Tabel 7.