**PENGGUNAAN *DEEP LEARNING* UNTUK MENGIDENTIFIKASI DIGIT ANGKA DAN JENIS PAKAIAN BERDASARKAN GAMBAR**

Dibuat utuk memenuhi tugas PR 1 mata kuliah MA4072 Pembelajaran Mendalam

Tahun Akademik 2019/2020

****

Disusun oleh :

Marcello Susanto – 10116017

Suparjo – 10116031

**PROGRAM STUDI MATEMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

1. **Latar Belakang**

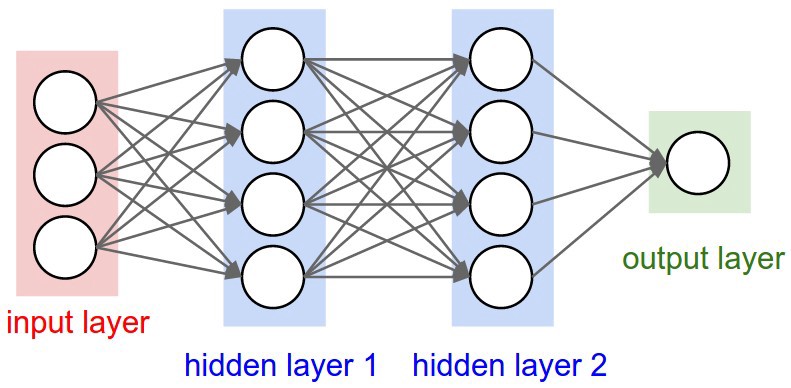
PR ini terdiri dari dua buah paket eksperimen. Eksperimen bagian pertama (disingkat E1) relatif mudah, eksperimen bagian kedua (disingkat E2) agak lebih sulit dan memakan waktu lebih lama.

Untuk E1 akan digunakan Python code yang sudah dipelajari di kuliah yang lalu untuk memecahkan masalah klasi.kasi angka. Di E1 ini akan digunakan 1.797 observasi dengan 64 features (variables) untuk model Arti.cial Neural Network (ANN) dengan dua buah hidden layers dengan tiap hidden layer terdiri dari 128 neurons (bila perlu neurons boleh ditambah). Di model ini akan ditaksir 26.122 buah parameter. Pada eksperimen E1 ini belum digunakan komponen validasi hasil prediksi. Target yang ingin dicapai di E1 ini berupa hasil training dengan prediksi yang mempunyai akurasi minimal 95%.

Sedangkan untuk E2 akan digunakan Python code tentang klasifikasi jenis pakaian yang akan segera dibagikan. Di E2 ini akan digunakan 60.000 observasi untuk training dengan 784 features (variables) untuk model Artificial Neural Network (ANN) dengan dua buah hidden layer dengan tiap hidden layer terdiri dari 784 neurons (bila perlu neurons boleh ditambah). Di model ini akan ditaksir lebih dari sejuta parameter, tepatnya 1.238.730 buah parameter. Pada eksperimen E2 ini sudah digunakan komponen validasi hasil prediksi di tahap training berupa penggunaan data test sebanyak 10.000 observasi untuk prediksi. Data untuk test ini berada diluar dari data sebanyak 60.000 observasi yang digunakan untuk training. Langkah validasi dilakukan dengan mengamati akurasi hasil dari penggunaan data untuk training yang akan dibandingkan dengan akurasi hasil penggunaan data untuk test. Target yang ingin dicapai di E2 ini berupa hasil training dengan prediksi yang mempunyai akurasi minimal 85%.

1. **Dasar Teori**

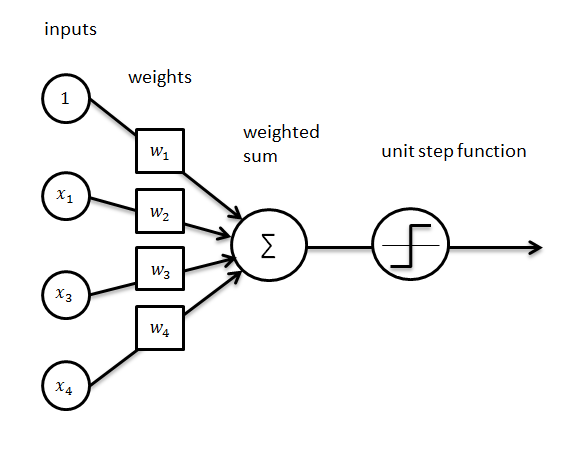
*Artificial Neural Network* (ANN) adalah sebuah model komputasional yang terinspirasi dari cara kerja sistem jarinan saraf pada manusia. Parameter yang ada pada model ini sangat dipengaruhi oleh *input* dan *output* yang dihasilkan oleh model dimana tiap iterasinya parameter ini akan diperbaharui agar mencapai hasil prediksi yang paling akurat. ANN ini sendiri dapat dikelompokkan pada jenis model linear maupun non-linear bergantung pada penggunaan fungsi aktivasinya. Kemampuan ANN dalam mempelajari dataset yang terobservasi terjamin oleh *Universal Approximation Theorem* yang menyatakan bahwa sebuah model ANN yang melakukan *forward propagation* dengan menggunakan sebuah *hidden layer* yang mengandung sejumlah neuron yang terbatas dapat mengaproksimasi sebuah fungsi kontinu yang digunakan untuk melakukan prediksi. Berikut adalah arsitektur sederhana dari sebuah model ANN



Pada sebuah model ANN kumpulan dar neuron diorganisir pada sebuah layer yang biasanya terdiri dari :

* ***Input Layer*** : Layer dimana data inputan dimasukkan ke dalam model ANN dan diproses untuk layer berikutnya
* ***Hidden Layer*** : Layer antara *Input* dan *Output Layer* dimana data dari *Input Layer* akan dihitung menggunakan *weight, bias,¸*dan fungsi aktivasi agar kompleksitas model meninkat
* ***Output Layer*** : Layer terakhir yang memproduksi hasil prediksi model

Komputasi pada model ANN dilakukan menggunakan *weight, bias,* dan fungsi aktivasi yang telah dipilih untuk tiap layer. Berikut akan diberikan contoh gambaran komputasi yang terjadi pada satu neuron



Weighted sum

weight

Fungsi aktivasi

bias

Input

Jadi pada sebuah proses komputasi pada model ANN terdapat beberapa komponen penting yaitu :

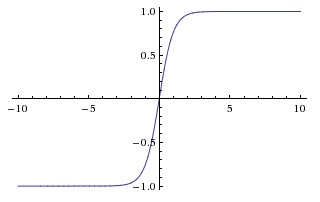
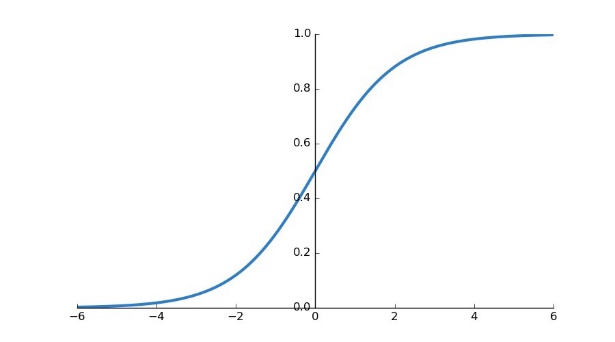
* ***Weight*** : Sebuah nilai yang digunakan untuk membobotkan nilai inputan yang digunakan
* ***Bias*** : Sebuah nilai yang digunakan untuk menanggulangi adanya galat pada prediksi
* **Fungsi Aktivasi** : Fungsi yang digunakan untuk memberikan aspek non-linear pada model

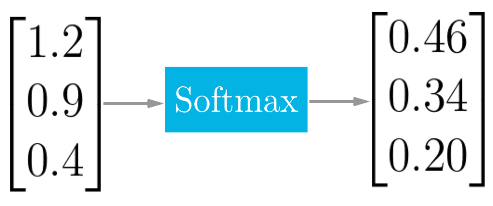
Sehingga formula untuk menghitung nilai dari sebuah neuron adalah sebagai berikut

Dengan sebagai fungsi aktivasi, sebagai *bias*, dan sebagai *weight*. Formula ini akan digunakan untuk tiap neuron pada tiap layer hingga mencapai layer terakhir yaitu *Output Layer.* Proses ini disebut ***forward propagation.***

Banyak sekali fungsi aktivasi yang dapat digunakan pada eksperimen kali ini ada kita membatasi diri dengan hana menggunakan tiga fungsi aktivasi yaitu **Sigmoid**, **Tanh**, dan **Softmax**. Berikut akan diperlihatkan formula dan grafik dari tiap fungsi aktivasi

* **Sigmoid** : Fungsi yang biasa digunakan pada kasus klasifikasi dikarenakan *outp*ut yang dihasilkannya memiliki *range* [0,1] seperti probabilitas pada umumya
* **Tanh** : Fungsi yang memiliki *range output* yang cukup kecil yaitu pada kisaran [-1.1]
* **Softmax** : Fungsi yang biasa digunakan pada kasus multi klasifikasi dikarenakan sifatnya yang dapat membuat jumlah dari tiap *input* nya menjadi 1





Dimana menandakan nilai neuron ke i pada *output layer*, **M** menandakan banyaknya *class* yang ingin diklasifikasi. Untuk mengevaluasi hasil prediksi model yang didapatkan pada proses *forward propagation* diperlukan sebuah tolak ukur. Pada model ANN tolak ukur ini disebut ***loss function*** dan fungsi ini akan mempengaruhi proses ***backward propagation*** dimana pada proses ini semua parameter yang digunakan pada model akan diperbaharui dengan menggunakan metode *gradient descent* untuk meminimalisir nilai *loss function* yang digunakan. Pada eksperimen ini *loss function* yang digunakan ada dua yaitu **Log-loss** dan **Mean Squared Error (MSE)** dengan detail formula sebagai berikut

Dimana **N** menandakan jumlah observasi, **M** menandakan jumlah *class* yang diprediksi, **y** menandakan variabel respon aktual, **p** menandakan variabel respon prediksi. Proses *backward propagation* model ANN menggunakan metode *gradient descent* dengan menggunakan turunan dari fungsi aktivasi dan *loss function*. Berikut adalah turunan fungsi yang akan digunakan pada eksperimen ini

**Turunan Tanh**

**Turunan Sigmoid**

**Turunan MSE berdasarkan Sigmoid**

**Turunan Logloss berdasarkan Softmax**

Dimana **p** menandakan variabel respon prediksi dan **y** menandakan variabel respon aktual. Selanjutnya akan dijelaskan proses *backward propagation* secara lebih rinci.

, Menyatakan besar nilai pada layer ke-i

, Menyatakan fungsi aktivasi keluar dari layer i

, Menyatakan besar nilai weight keluar dari layer ke-i

, Menyatakan besar nilai bias keluar dari layer ke-i

Permasalahan optimisasi dari model ini adalah

(MSE)

(Log – loss)

, Menyatakan nilai asli

, Menyatakan hasil prediksi

Apabila ingin mencari gradient decent untuk dan , maka

Dengan, , menyatakan fungsi log-loss atau MSE

Diketahui bahwa

Dengan metode yang sama diperoleh

Pada model ANN kita juga dapat menerapkan metode *Feature Scaling* pada data input yang kita gunakan. *Feature Scaling* ini penting agar setiap variabel yang kita gunakan pada modelling memiliki *scale* yang sama sehingga pada tahap *backward propagation* nilai *gradient descent* yang dihasilkan juga memiliki *scale* yang sama. Jika kita tidak melakukan scaling pada dataset maka variabel yang memiliki jangkauan maksimum yang lebih besar akan memiliki nilai *gradient descent* yang lebih besar juga dibandingkan dengan variabel lain yang jangkauan maksimumnya tidak besar. Alasan lain untuk melakukan *Feature Scaling* adalah agar metode *gradient descent* kita mencapai nilai konvergen lebih cepat. Metode *Feature Scaling* yang kita gunakan pada eksperimen kali ini adalah sebagai berikut :

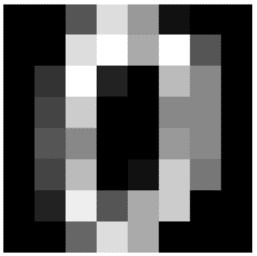
* ***Normalization*** : Mengubah *range* dari variabel input menjadi skala [0, 1] dengan cara mengurangi nilai variabelnya dengan nilai minimum terus dibagi dengan *range* variabelnya
* ***Standarization*** : Mengubah distribusi dari variabel input menjadi normal baku

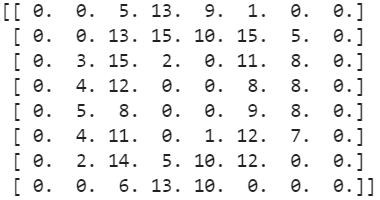
1. **Eksperimen Identifikasi Digit Angka**

Eksperimen 1 adalah suatu eksperimen untuk mengidentifikasi tulisan tangan berupa angka dari 0 sampai 9 yang berukuran 64 pixel. Pada eksperimen ini, akan dilakukan perbandingan dari parameter-parameter pada deep learning. Parameter-parameter ini berupa jumlah epoch, learning rate, normalisasi, fungsi aktivasi, dan fungsi kerugian. Akurasi adalah akurasi dari training, sesuai instruksi. Dimensi dari data yang diberikan adalah sebagai berikut

Tabelx. Dimensi dari data untuk eksperimen 1

|  |  |
| --- | --- |
| Jenis | Ukuran |
| Ukuran Feature | 8x8 |
| Ukuran Target | 10x1 |
| Jumlah Sample | 1797 |





Gambarx. Gambar dari sampel dan konversi tensornya

Kita akan melakukan eksperimen pada tiap kombinasi fungsi aktivasi dan *loss function* yang diberikan.

* **Kombinasi I**

Pada kombinasi ini akan digunakan **fungsi sigmoid** pada *hidden layer*, **fungsi softmax** pada *output layer* dan **fungsi log-loss** sebagai *loss function.* Pada eksperimen ini akan dicari learning rate yang cocok dengan jumlah epoch yang kecil terlebih dahulu. Pada eksperimen ini digunakan sigmoid sebagai fungsi aktivasi di hidden layer dan softmax sebagai fungsi aktivasi di output layer.

|  |  |  |
| --- | --- | --- |
| Learning Rate | Akurasi (epoch 3) | Akurasi (epoch 5) |
| 1.4 | 20.923 | 18.697 |
| 1.2 | 17.306 | 32.220 |
| 1 | 17.918 | 33.277 |
| 0.8 | 24.763 | 45.575 |
| 0.6 | 32.943 | 43.183 |

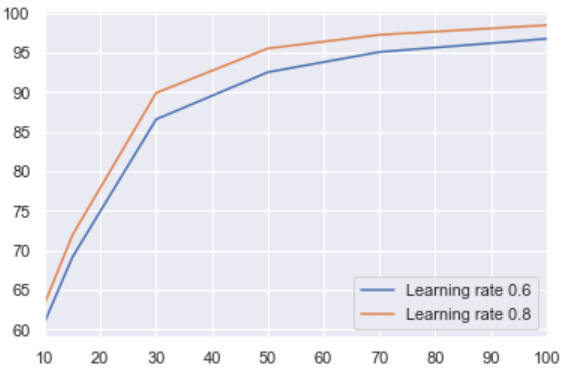
Grafikx. Perbandingan Learning rate dengan epoch kecil, 3 dan 5

Dari tabelx, diperoleh informasi bahwa learning rate tertinggi pada epoch 3 adalah 0.6 sedangkan pada epoch 5 adalah 0.8. akan dilakukan uji untuk kedua learning rate dengan nilai epoch yang lebih tinggi.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| 0.6 | 60.934 | 69.170 | 86.588 | 92.543 | 95.102 | 96.772 |
| 0.8 | 63.272 | 71.953 | 89.927 | 95.548 | 97.273 | 98.497 |

Tabelx. Tabel learning rate 0.6 dan 0.8 dengan nilai epoch yang tinggi

Grafikx. perbandingan rate 0.6 dan 0.8 dengan nilai epoch yang tinggi



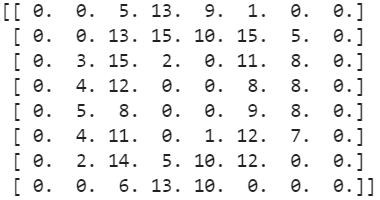
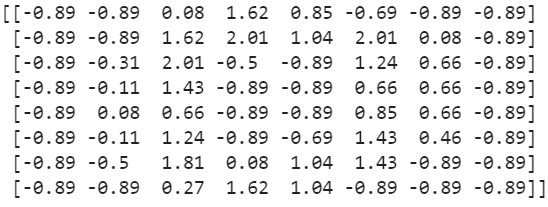
Perbandingan rate 0.6 dan 0.8

Akurasi

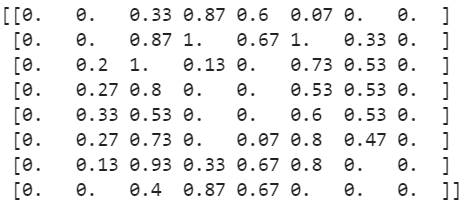
Epochs

Dapat dilihat bahwa learning rate 0.8 melampaui 0.6 dalam akurasi di semua epoch bernilai tinggi yang diujikan. Maka, akan digunakan learning rate 0.8 sebagai learning rate optimal

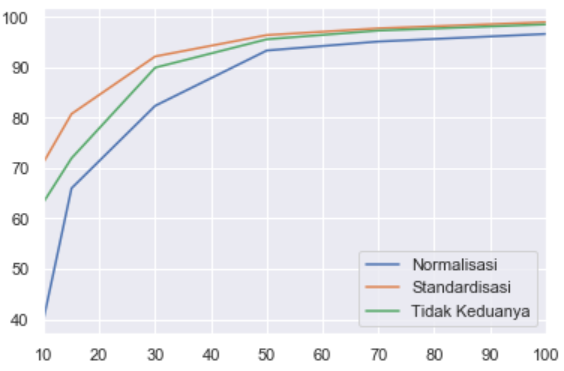
Normalisasi atau standardisasi dilakukan pada kumpulan data input (feature). Hasil dari normalisasi dapat dilihat pada gambarx. Operasi dari normalisasi dilakukan di semua 1797 sampel gambar yang diberikan.



Gambarx. Gambar input sebelum dinormalisasi (kiri) dan setelah distandardisasi (kanan) serta normalisasi (bawah)



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| Normalisasi | 40.066 | 65.998 | 82.359 | 93.322 | 95.102 | 96.605 |
| Standardisasi | 71.174 | 80.745 | 92.153 | 96.382 | 97.718 | 98.942 |
| Tidak | 63.272 | 71.953 | 89.927 | 95.548  Tabelx. Tabel learning akurasi dari feature dengan normalisasi, standardisasi, dan tidak keduanya | 97.273 | 98.497 |



Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

Akurasi

Epochs

Grafikx. Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

* **Kombinasi II**

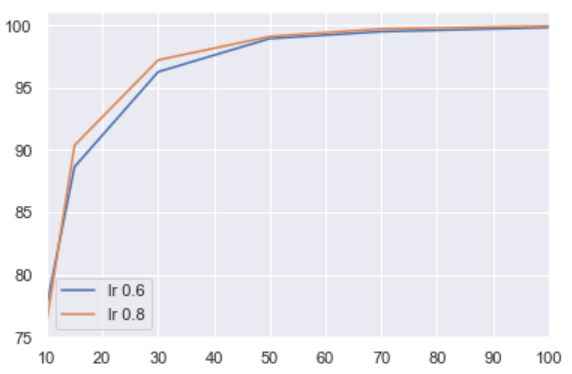
Pada kombinasi ini akan digunakan **fungsi tanh** pada *hidden layer*, **fungsi softmax** pada *output layer* dan **fungsi log-loss** sebagai *loss function.* Pada eksperimen ini akan dicari learning rate yang cocok dengan jumlah epoch yang kecil terlebih dahulu. Pada eksperimen ini digunakan tanh sebagai fungsi aktivasi di hidden layer dan softmax sebagai fungsi aktivasi di output layer.

|  |  |  |
| --- | --- | --- |
| Learning Rate | Akurasi (epoch 3) | Akurasi (epoch 5) |
| 1.4 | 19.810 | 30.050 |
| 1.2 | 27.434 | 34.891 |
| 1 | 31.274 | 37.673 |
| 0.8 | 34.947 | 47.746 |
| 0.6 | 34.891 | 44.351 |

Grafikx. Perbandingan Learning rate dengan epoch kecil, 3 dan 5

Dari tabelx, diperoleh informasi bahwa learning rate tertinggi pada epoch 3 adalah 0.6 sedangkan pada epoch 5 adalah 0.8. akan dilakukan uji untuk kedua learning rate dengan nilai epoch yang lebih tinggi.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| 0.6 | 68.391 | 73.511 | 87.256 | 94.657 | 98.775 | 99.443 |
| 0.8 | 70.172 | 77.017 | 87.534  Tabelx. Tabel learning rate 0.6 dan 0.8 dengan nilai epoch yang tinggi | 94.769 | 99.332 | 99.888 |



Perbandingan rate 0.6 dan 0.8

Akurasi

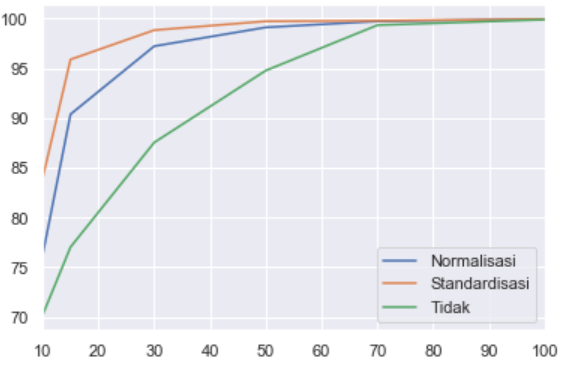
Epochs

Epochs

Grafikx. perbandingan rate 0.6 dan 0.8 dengan nilai epoch yang tinggi

Normalisasi atau standardisasi dilakukan pada kumpulan data input (feature).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| Normalisasi | 76.182 | 90.372 | 97.217 | 99.109 | 99.721 | 99.944 |
| Standardisasi | 83.917 | 95.882 | 98.831 | 99.721 | 99.777 | 99.944 |
| Tidak | 70.172 | 77.017 | 87.534  Tabelx. Tabel learning rate 0.6 dan 0.8 dengan nilai epoch yang tinggi | 94.769 | 99.332 | 99.888 |



Epochs

Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

Akurasi

Grafikx. Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

* **Kombinasi III**

Pada kombinasi ini akan digunakan **fungsi sigmoid** pada *hidden layer*, **fungsi sigmoid** pada *output layer* dan **fungsi MSE** sebagai *loss function***.** Pada eksperimen ini akan dicari learning rate yang cocok dengan jumlah epoch yang kecil terlebih dahulu.

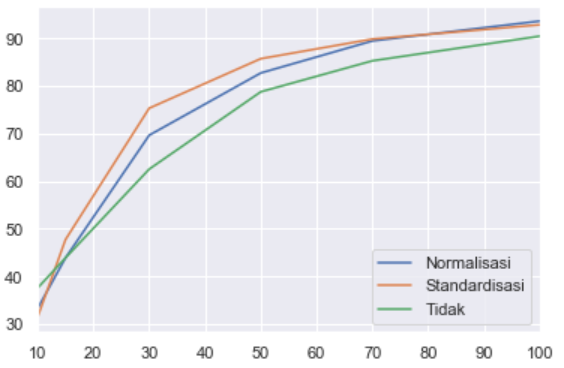
|  |  |  |
| --- | --- | --- |
| Learning Rate | Akurasi (epoch 3) | Akurasi (epoch 5) |
| 4.5 | 20.311 | 23.316 |
| 4 | 22.036 | 26.432 |
| 3.5 | 21.869 | 26.544 |
| 3 | 21.981 | 26.265 |
| 2.5 | 22.148 | 26.711 |
| 2.0 | 20.812 | 24.763 |
| 1 | 18.864 | 20.367 |
| 0.8 | 18.363 | 19.143 |

Grafikx. Perbandingan Learning rate dengan epoch kecil, 3 dan 5

Dari tabel yang disajikan, dapat dilihat bahwa untuk fungsi aktivasi sigmoid di output layer menghasilkan learning rate optimal di 2.5. Berbeda dengan kombinasi sebelumnya, learning rate optimal lebih kecil dari 1. Selanjutnya akan dilakukan Normalisasi atau standardisasi pada kumpulan data input (feature).

Tabelx. Tabel learning rate 0.6 dan 0.8 dengan nilai epoch yang tinggi

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| Normalisasi | 33.277 | 43.906 | 69.671 | 82.749 | 89.482 | 93.656 |
| Standardisasi | 31.441 | 47.746 | 75.347 | 85.754 | 89.872 | 92.877 |
| Tidak | 37.562 | 43.906 | 62.548 | 78.797 | 85.308 | 90.484 |



Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

Akurasi

Grafikx. Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

Epochs

* **Kombinasi IV**

Pada kombinasi ini akan digunakan **fungsi tanh** pada *hidden layer*, **fungsi sigmoid** pada *output layer* dan **fungsi MSE** sebagai *loss function***.**  Pada eksperimen ini akan dicari learning rate yang cocok dengan jumlah epoch yang kecil terlebih dahulu.

|  |  |  |
| --- | --- | --- |
| Learning Rate | Akurasi (epoch 3) | Akurasi (epoch 5) |
| 4 | 19.031 | 21.535 |
| 3.7 | 19.866 | 23.316 |
| 3.5 | 19.031 | 25.876 |
| 3.3 | 18.030 | 22.871 |
| 3 | 19.143 | 22.148 |
| 1.5 | 15.859 | 18.419 |
| 1 | 13.021 | 16.082 |
| 0.5 | 10.795 | 12.353 |

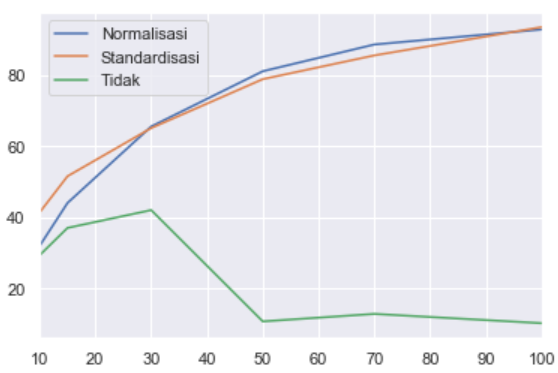
Grafikx. Perbandingan Learning rate dengan epoch kecil, 3 dan 5

Dari tabel yang disajikan, dapat dilihat bahwa learning rate optimal di 3.5. Kemudian akan dilakukan Normalisasi atau standardisasi pada kumpulan data input (feature).

Tabelx. Tabel learning rate 0.6 dan 0.8 dengan nilai epoch yang tinggi

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Akurasi (epoch 10) | Akurasi (epoch 15) | Akurasi (epoch 30) | Akurasi (epoch 50) | Akurasi (epoch 70) | Akurasi (epoch 100) |
| Normalisasi | 31.942 | 44.017 | 65.498 | 81.023 | 88.536 | 92.765 |
| Standardisasi | 41.346 | 51.530 | 65.052 | 78.797 | 85.475 | 93.433 |
| Tidak | 29.326 | 37.006 | 42.014 | 10.740 | 12.854 | 10.239 |

Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya



Akurasi

Grafikx. Perbandingan Hasil Normalisasi, Standardisasi, dan tidak keduanya

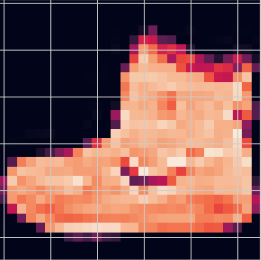
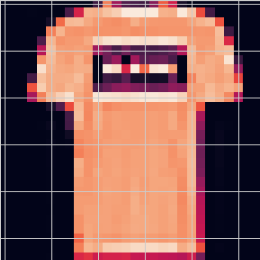
Epochs

1. **Eksperimen Identifikasi Jenis Pakaian**

Model ANN yang digunakan pada kasus ini memiliki struktur *deep learning* yang mirip dengan model identifikasi digit angka. Hanya saja sekarang tiap *Hidden Layer* memiliki 784 neuron yaitu jumlah pixel yang dimiliki tiap observasi (ukuran 28x28) dan kita menggunakan dataset train yang lebih besar yaitu sebanyak 60000 observasi. Selain itu kita juga memiliki dataset test sebanyak 10000 observasi yang akan digunakan untuk mengevaluasi model. Pada permodelan ini kita akan melihat pengaruh dari perubahan learning rate dan jumlah epoch. Kemudian akan dlihat juga pengaruh dari standarisasi dan normalisasi data untuk learning rate dan jumlah epoch yang sesuai berdasarkan hasil sebelumya. *Random Seed* yang akan digunakan adalah 10116017 dan berikut adalah informasi mengenai dataset yang kita gunakan

|  |  |
| --- | --- |
| Ukuran Feature | 28x28 |
| Ukuran Target | 10x1 |
| Ukuran Dataset Train | 60000 |
| Ukuran Dataset Test | 10000 |

Tabel dimensi dataset untuk eksperimen 2



Contoh data gambar baju

Contoh data gambar sepatu

Kita akan melakukan eksperimen pada tiap kombinasi fungsi aktivasi dan *loss function* yang diberikan.

* **Kombinasi I**

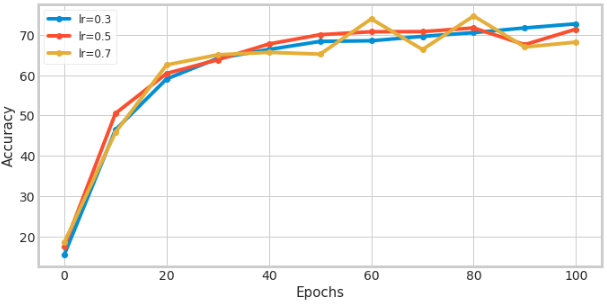
Pada kombinasi ini akan digunakan **fungsi sigmoid** pada *hidden layer*, **fungsi softmax** pada *output layer* dan **fungsi log-loss** sebagai *loss function*. Pertama kita akan mencari learning rate yang terbaik untuk kombinasi ini dengan melihat nilai dan perubahan pada akurasi dataset training dengan menggunakan epoch 1, 5, dan 10. Setelah melakukan modelling didapat hasil sebagai berikut

|  |  |  |  |
| --- | --- | --- | --- |
| Learning Rate | Epoch 1 (Akurasi Train) | Epoch 5 (Akurasi Train) | Epoch 10 (Akurasi Train) |
| 0.01 | 5.71 | 7.05 | 8.16 |
| 0.1 | 9.48 | 21.27 | 30.89 |
| 0.3 | 15.56 | 38.63 | 46.46 |
| 0.5 | 17.56 | 35.87 | 50.50 |
| 0.7 | 18.59 | 32.13 | 45.82 |
| 1 | 19.72 | 10.00 | 10.00 |
| 1.5 | 22.86 | 10.00 | 10.00 |

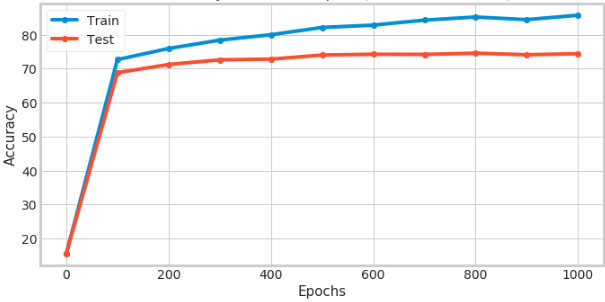
Tabel perbandingan learning date kombinasi I

Dari tabel di atas dapat dilihat bahwa terlihat bahwa dengan menggunakan *learning rate* sebasar 0.5 pada epoch 10 kita telah bisa mendapatkan nilai akurasi sebesar 50%. Walau begitu *learning rate* 0.3 dan 7 juga mengalami peningkatan sehingga akan kita tinjau lebih lanjut ketiga *learning rate* ini sampai epoch ke 100. Didapat grafik seperti di bawah

Grafik perubahan nilai akurasi berdasakan epoch pada kombinasi I

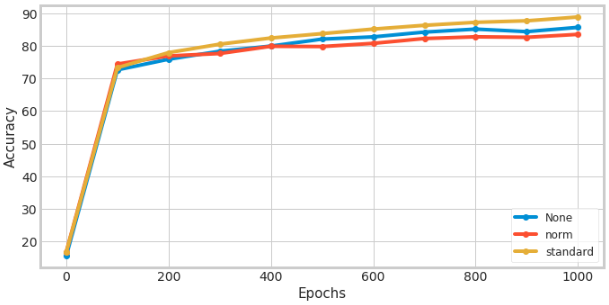


Dapat dilihat pada grafik di atas bahwa dengan menggunakan *learning rate* 0.3 kita dapat mencapai nilai akurasi 71% secara stabil dan *learning rate* 0.5 dan 0.7 terlihat tidak stabil .Oleh karena itu agar kita bisa mencapai nilai akurasi yang tinggi dengan tetap menjaga kestabilan dari modelnya akan kita coba menggunakan *learning rate* 0.3 hingga epoch ke 1000. Hasilnya didapat grafik seperti di bawah



Grafik perubahan nilai akurasi dengan *learning rate* 0.3 pada kombinasi I

Terlihat bahwa dengan menggunakan *learning rate* 0.3 kita dapat mencapai nilai akurasi train sebesar 85.7%, mencapai target akurasi yang diinginkan. Maka selanjutnya kita akan menggunakan ***learning rate* 0.3** untuk melihat pengaruh melakukan normalisasi dan standarisasi

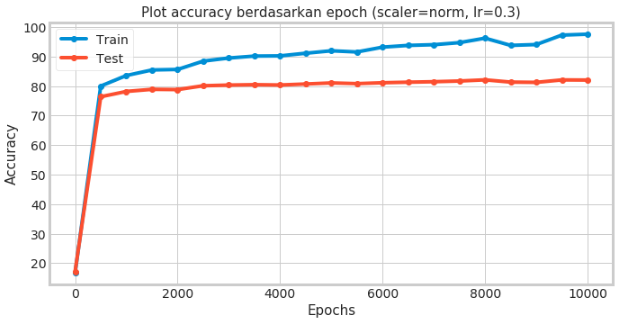


Grafik perubahan nilai akurasi dengan *learning rate* 0.3 berdasarkan scaler

|  |  |  |  |
| --- | --- | --- | --- |
| Scaler | Epoch 100 (Akurasi Test) | Epoch 500 (Akurasi Test) | Epoch 1000  (Akurasi Test) |
| None | 68.82 | 73.99 | 74.38 |
| Norm | 68.78 | 74.25 | 75.62 |
| Standard | 72.54 | 76.27 | 78.19 |

Tabel perubahan nilai akuras dataset test

Dapat dilihat bahwa dengan melakukan standarisasi pada nilai variabel input X kita bisa meningkatkan nilai akurasi hingga mencapai 88.89% (+3%). Walau begitu pada kasus normalisasi nilai akurasi model malah berkurang menjadi 83.5% (-2%). Walau begitu jika kita tinjau nilai akurasi dari dataset testnya terlihat bahwa nilai akurasi terbesar kita dapatkan saat kita melakukan normalisasi. Hal ini mengindikasikan bahwa variabel yang di nomalisasi membuat model kita bagus secara umum dan nilai akurasinya terlihat masih bisa meningkat. Maka kedepanya kita akan gunakan **normalisasi** dengan memperbesar nilai epochnya



Dengan meningkatkan epoch sampai 10000 kita dapatkan model dengan nilai akurasi train sebesar 97.53% dan nilai akurasi test sebesar 82.04%. Dari grafik terlihat bahwa nilai akurasi train masih berpotensi untuk meningkat tapi nilai akurasi test terlihat sudah cukup stagnan maka dirasa menambah epoch tidak akan membuat nilai akurasi test meningkat lagi

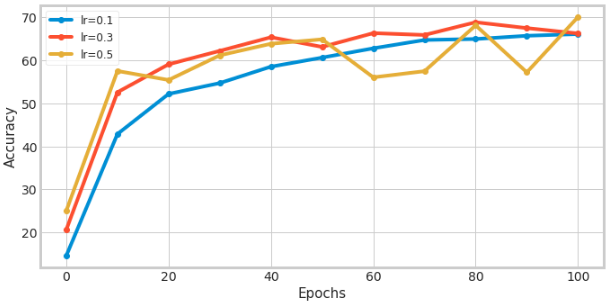
* **Kombinasi II**

Pada kombinasi ini akan digunakan **fungsi tanh** pada *hidden layer*, **fungsi softmax** pada *output layer* dan **fungsi log-loss** sebagai *loss function*. Pertama kita akan mencari learning rate yang terbaik untuk kombinasi ini dengan melihat nilai dan perubahan pada akurasi dataset training dengan menggunakan epoch 1, 5, dan 10. Setelah melakukan modelling didapat hasil sebagai berikut

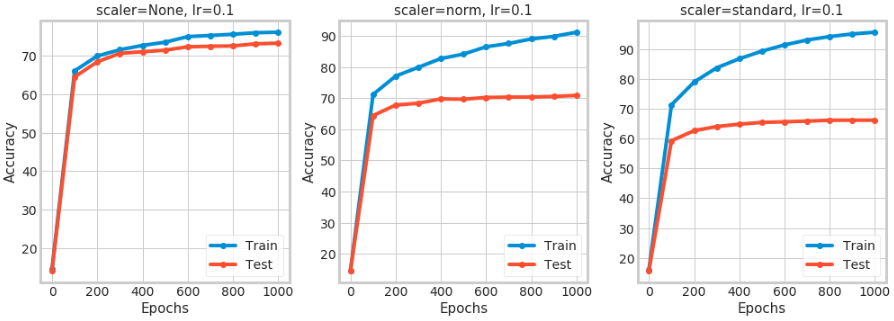
|  |  |  |  |
| --- | --- | --- | --- |
| Learning Rate | Epoch 1 (Akurasi Train) | Epoch 5  (Akurasi Train) | Epoch 10 (Akurasi Train) |
| 0.01 | 11.37 | 12.86 | 14.39 |
| 0.1 | 14.60 | 31.94 | 42.86 |
| 0.3 | 20.64 | 47.30 | 52.56 |
| 0.5 | 25.03 | 44.59 | 57.5 |
| 0.7 | 27.22 | 38.23 | 28.99 |
| 1 | 28.61 | 26.11 | 22.48 |
| 1.5 | 28.07 | 26.71 | 10.41 |

Tabel perbandingan learning rate kombinasi II

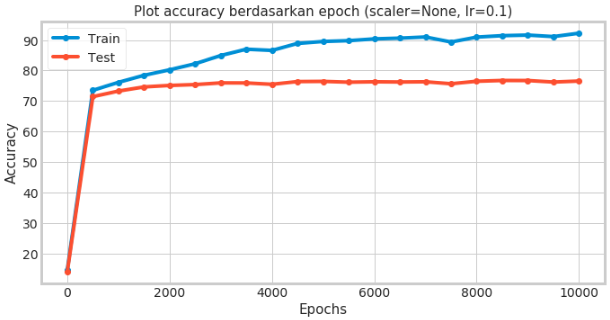
Dari tabel di atas dapat dilihat bahwa *learning rate* 0.3 dan 0.5 memiliki nilai akurasi train yang cukup mirip pada kisaran 52-57%. Walau begitu *learning rate* 0.1 juga terlihat cukup stabil. Oleh karena itu kita akan meninjau ketiga *learning rate* ini dengan menggunakan epoch sampai ke 100. Hasilnya didapat grafik sebagai berikut



Dari grafik di atas dapat dilihat bahwa *learning rate* 0.5 mencapai nilai akurasi train paling besar yaitu 70% tetapi terlihat pergerakan perubahan nilai akurasinya tidak stabil. Berbeda dengan *learning rate* 0.1 yang lebih stabil walaupun nilai akurasinya di bawah *learning rate* 0.5. Dirasa dengan epoch yang lebih besar *learning rate* 0.1 akan memberikan performa yang lebih baik. Oleh karena itu untuk selanjutnya akan digunakan ***learning rate* 0.1** untuk melihat pengaruh dari normalisasi dan standarisasi



Dapat dilihat bahwa dengan melakukan standarisasi dan normalisasi kita bisa mendapatkan nilai akurasi train lebih dari 90%. Tetapi dari grafik di atas dapat disimpulkan bahwa modelnya *overfitting* ke train karena nilai akurasi test hanya sekitar 70% saja. Oleh karena untuk selanjutnya kita **tidak akan melakukan scaling**. Akan dicoba membuat model dengan epoch yang lebih besar lagi yaitu 10000



Dengan meningkatkan epoch sampai 10000 kita dapatkan model dengan nilai akurasi train sebesar 92.23% dan nilai akurasi test sebesar 76.55%. Dari grafik terlihat bahwa nilai akurasi train masih berpotensi untuk meningkat tapi nilai akurasi test terlihat sudah cukup stagnan maka dirasa menambah epoch tidak akan membuat nilai akurasi test meningkat lagi

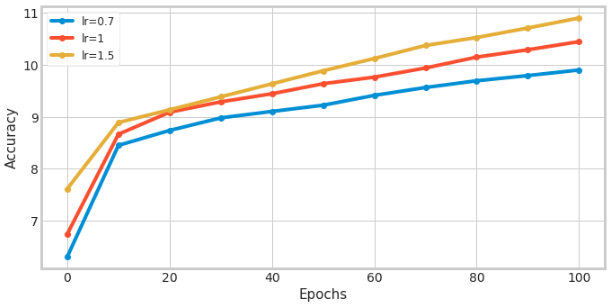
* **Kombinasi III**

Pada kombinasi ini akan digunakan **fungsi sigmoid** pada *hidden layer*, **fungsi sigmoid** juga pada *output layer* dan **fungsi *mean squared error (MSE*** sebagai *loss function*. Pertama kita akan mencari learning rate yang terbaik untuk kombinasi ini dengan melihat nilai dan perubahan pada akurasi dataset training dengan menggunakan epoch 1, 5, dan 10. Setelah melakukan modelling didapat hasil sebagai berikut

|  |  |  |  |
| --- | --- | --- | --- |
| Learning Rate | Epoch 1 (Akurasi Train) | Epoch 5  (Akurasi Train) | Epoch 10 (Akurasi Train) |
| 0.01 | 5.46 | 5.51 | 5.59 |
| 0.1 | 5.57 | 6.09 | 6.48 |
| 0.3 | 5.81 | 7.06 | 7.67 |
| 0.5 | 6.07 | 7.69 | 8.14 |
| 0.7 | 6.29 | 8.05 | 8.44 |
| 1 | 6.73 | 8.36 | 8.66 |
| 1.5 | 7.60 | 8.66 | 8.88 |

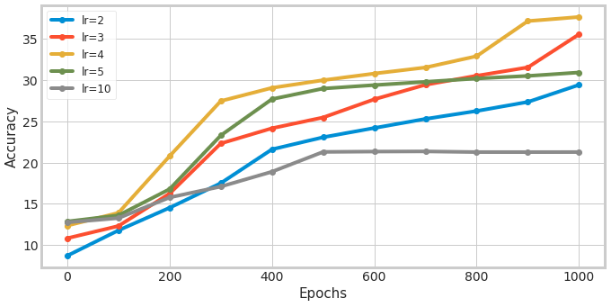
Tabel perbandingan learning date kombinasi III

Dari tabel di atas dapat dilihat bahwa semakin besar nilai *learning rate* yang digunakan semakin besar juga nilai akurasi yang didapat pada epoch 10. Walau nilai akurasi yang didapat sangat kecil bahkan di bawah 10%. Selanjutnya akan dilihat apakah fenomena ini tetap bertahan pada epoch yang lebih besar, dan juga apakah model tetap stabil pada nilai *learning rate* yang besar. Hasilnya didapat grafik seperti di bawah



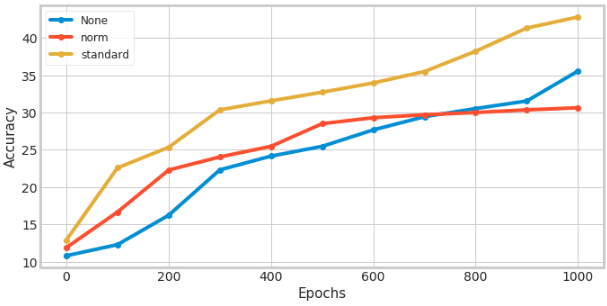
Grafik perubahan nilai akurasi dengan *learning rate* lebih besar

Pada grafik di atas dapat dilihat bahwa *learning rate* yang besar tetap menghasilkan nilai akurasi yang besar juga dengan stabil. Karenanya akan dicoba beberapa nilai *learning rate* yang lebih besar lagi untuk mendapatkan nilai akurasi yang diinginkan dengan menggunakan epoch hingga 1000

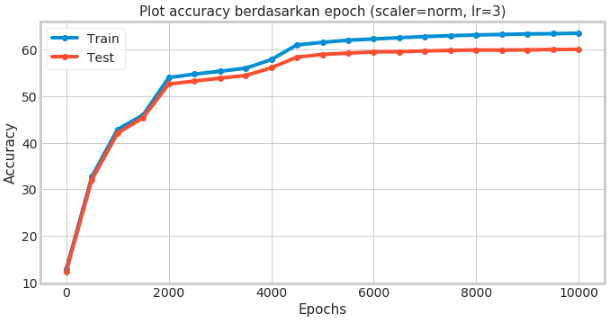


Grafik perubahan nilai akurasi berdasakan epoch pada kombinasi III

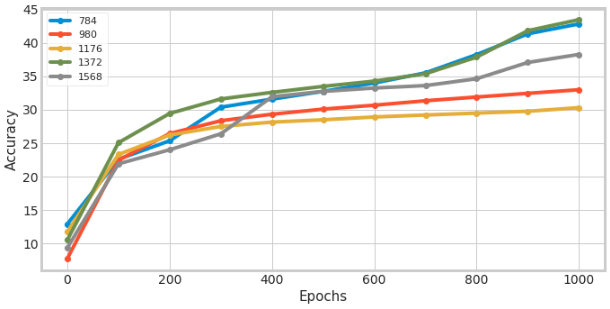
Dapat dilihat bahwa *learning rate* 4 memberikan nilai akurasi terbesar yaitu sekitar 37%, tetapi gradien perubahannya sudah menipis. Pada *learning rate* 3 dapat dilihat bahwa nilai akurasinya di bawah *learning rate 4* tetapi gradien perubahannya masih besar. Karenanya untuk selanjutnya kita akan mnggunakan ***learning rate 3*** untuk melihat pengaruh dari normalisasi dan standarisasi



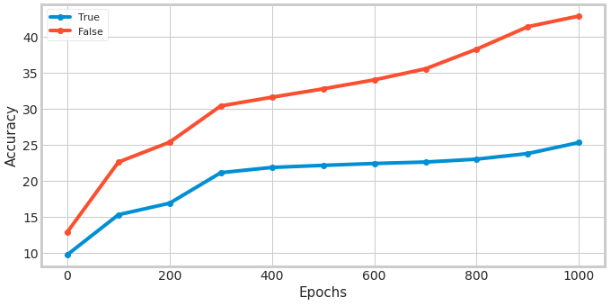
Pada grafik di atas dapat dilihat bahwa input yang distandarisasi dapat mencapai nilai akurasi training sekitar 42.8% (+7%) jauh melebihi akurasi input yang tidak discale. Pada kasus input yang dinormalisasi nilai akurasi trainng malah berkuang menjadi 30.65% (-5%). Nilai akurasi test pada tiap kasus tidak berbeda jauh dengan nilai akurasi trainingnya jadi untuk selanjutnya akan digunakan input yang **distandarisasi**. Selanjutnya akan dicoba menambah epoch untuk medapatkan nilai akurasi yang diinginkan



Dengan menggunakan *learning rate* 3 dan melakukan standarisasi, pada epoch ke 10000 kita mendapatkan hasil akurasi training sebesar 63.45% dan testing sebesar 59.98% dimana dapat dilihat pada grafik di atas bahwa gradien perubahannya sudah sangat tipis. Jadi menambah epoch sepertinya sudah tidak memberikan pengaruh signifikan lagi. Oleh karena itu kita akan mencoba menambah jumlah neuron pada *hidden layer*



Dapat dilihat pada grafik di atas nilai akurasi train terbesar (43%) kita dapatkan dengan menggunakan jumlah neuron sebanyak 1372, walau nilai akurasinya tidak berbeda jauh dengan jumlah neuron awal yaitu 784. Sehingga dirasa menambah jumlah neuron tidak memberikan hasil akurasi yang lebih baik. Selanjutnya dengan menggunakan jumlah neuron awal yaitu 784 akan kita lihat pengaruh dari menambah jumlah layer



Dari grafik di atas dapat dilihat bahwa penambahan *hidden layer* malah membuat nilai akurasi trainnya lebih kecil daripada sebelumnya.

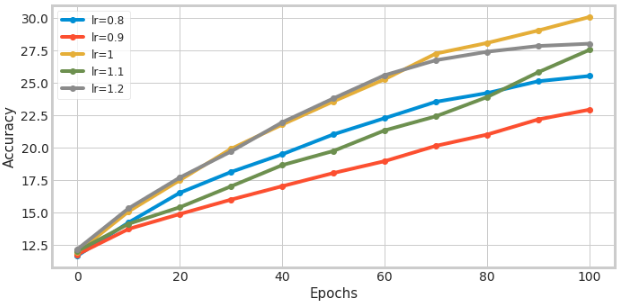
* **Kombinasi IV**

Pada kombinasi ini akan digunakan **fungsi tanh** pada *hidden layer*, **fungsi sigmoid** pada *output layer* dan **fungsi *mean squared error (MSE)*** sebagai *loss function*. Pertama kita akan mencari learning rate yang terbaik untuk kombinasi ini dengan melihat nilai dan perubahan pada akurasi dataset training dengan menggunakan epoch 1, 5, dan 10. Setelah melakukan modelling didapat hasil sebagai berikut

|  |  |  |  |
| --- | --- | --- | --- |
| Learning Rate | Epoch 1 (Akurasi Train) | Epoch 5  (Akurasi Train) | Epoch 10 (Akurasi Train) |
| 0.01 | 11.24 | 11.29 | 11.29 |
| 0.1 | 11.28 | 11.65 | 11.93 |
| 0.3 | 11.39 | 12.19 | 12.90 |
| 0.5 | 11.29 | 12.55 | 13.75 |
| 0.7 | 11.58 | 12.68 | 13.08 |
| 1 | 11.89 | 13.88 | 15.05 |
| 1.5 | 12.28 | 14.05 | 14.71 |

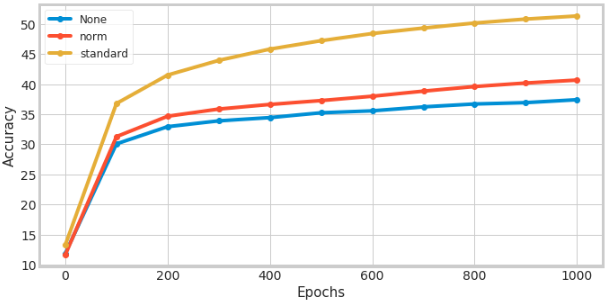
Tabel perbandingan learning date kombinasi IV

Dari tabel di atas dapat dilihat bahwa *learning rate* 1 memberikan nilai akurasi train paling besar dimana dengan *learning rate* 1.5 nilai akurasi train malah mengalami penurunan walau nilainya lebih besar daripada *learnin rate* 0.7 . Maka selanjunya akan kita tes beberapa nilai *learning rate* yang mendekati 1 hingga epoch ke 100. Hasilnya didapat grafik seperti di bawah

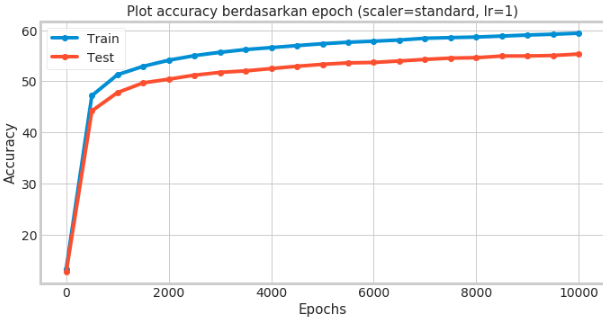


Grafik perubahan nilai akurasi berdasakan epoch pada kombinasi IV

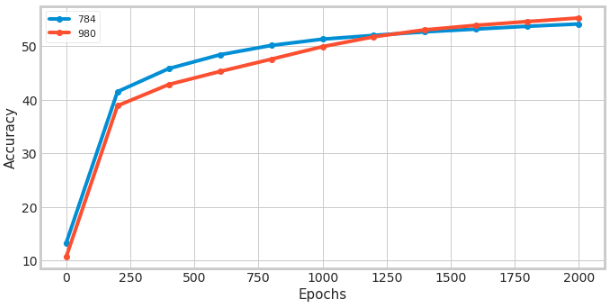
Dari grafik di atas terlihat bahwa memang *learning rate* 1 memberikan nilai akurasi train terbaik bahkan pada epoch ke 100. Nilai akurasi trainnya mencapai 30.07%. Maka selanjutnya kita akan menggunakan ***learning rate* 1** untuk melihat pengaruh dari normalisasi dan standarisasi



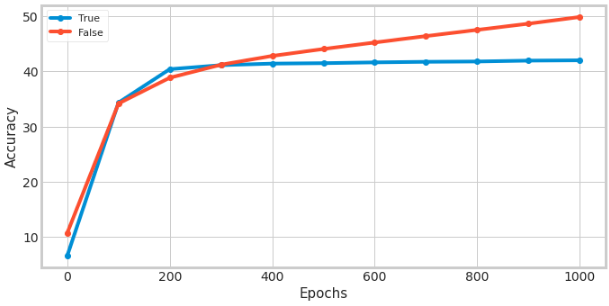
Pada grafik di atas dapat dilihat bahwa input yang distandarisasi dapat mencapai nilai akurasi training sekitar 51.29% (+14%) jauh melebihi akuras input yang tidak discale. Pada kasus input yang dinormalisasi nilai akurasi trainng malah juga bertambah walau hanya menjadi 40.65% (+5%). Nilai akurasi test pada tiap kasus tidak berbeda jauh dengan nilai akurasi trainingnya jadi untuk selanjutnya akan digunakan input yang **distandarisasi**. Selanjutnya kita akan mencoba menggunakan epoch yang lebih tinggi lagi yaitu sampai epoch ke 10000



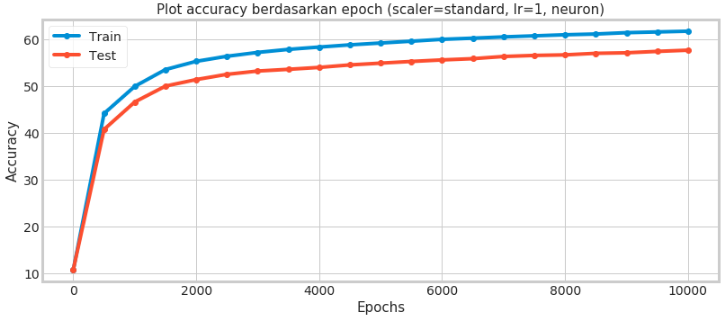
Dengan menggunakan *learning rate* 1 dan melakukan standarisasi, pada epoch ke 10000 kita mendapatkan hasil akurasi training sebesar 59.42% dan testing sebesar 55.33% dimana dapat dilihat pada grafik di atas bahwa gradien perubahannya masih memberikan indikasi bahwa menambah jumlah epoch masih akan meningkatkan nilai akurasi. Sebelum itu kita akan coba menambahkan neuron pada *hidden layer* dan membandingkan hasilnya dengan struktur original



Terlihat bahwa dengan menambah jumlah neuron menjadi 980 nilai akurasi trainingnya akhirnya bisa melewati nilai akurasi training jumlah neuron awal. Walau begitu untuk jumlah neuron yang lebih dari 980 nilai akurasi trainnya malah berkurang. Sehingga kedepannya kita akan menggunakan jumlah neuron 980 untuk menguji pengaruh dari penambahan *hidden layer*



Dari grafik di atas dapat dilihat bahwa penambahan *hidden layer* malah membuat nilai akurasi trainnya lebih kecil daripada sebelumnya. Terlihat bahwa nilai akurasinya konvergen ke sekitar 41% tanpa ada peningkatan lagi. Maka sebagai usaha terakhir kita akan menggunakan *learning rate* 1 dengan standarisasi dan jumlah neuron 980 sampai epoch ke 10000



Dengan menggunakan semua modifikasi yang telah ditetapkan pada epoch ke 10000 kita mendapatkan hasil akurasi training sebesar 61.68% dan testing sebesar 57.62%, meningkat dari hasil sebelumnya.

1. **Kesimpulan**

Dari eksperimen mengindentifikasi digit angka dengan menggunakan jumlah epoch hingga 100 kita dapatkan hasil sebagai berikut

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kombinasi | *Learning Rate* | Scaler | Total Neuron | Total *Hidden Layer* | Train Accuracy |
| I | 0.8 | Standard | 128 | 2 | 98.94% |
| II | 0.8 | Any | 128 | 2 | 99.94% |
| III | 2.5 | Norm | 128 | 2 | 93.65% |
| IV | 3.5 | Standard | 128 | 2 | 93.43% |

Dari eksperimen mengindentifikasi jenis pakaian dengan menggunakan jumlah epoch hingga 10000 kita dapatkan hasil sebagai berikut

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kombinasi | *Learning Rate* | Scaler | Total Neuron | Total *Hidden Layer* | Train Accuracy | Test Accuracy |
| I | 0.3 | Norm | 784 | 2 | 97.53% | 82.04% |
| II | 0.1 | None | 784 | 2 | 92.23% | 76.55% |
| III | 3 | Standard | 784 | 2 | 63.45% | 59.98% |
| IV | 1 | Standard | 980 | 2 | 61.68% | 57.62% |

Dari eksperimen ini dapat disimpulkan bahwa *learning rate* dan *feature scaling* sangat berpengaruh pada kualtas prediksi dari model. Hal ini terbukti dari perbedaan *learing rate* dan *feature scaling* yang optimal untuk tiap kombinasi fungsi aktivasi yang kita gunakan. Pada kasus kombinasi dimana *output layer* nya menggunakan fungsi aktivasi Softmax, dapat dilihat bahwa *learning rate*nya menggunakan nilai yang cukup kecil (0.3 dan 0.1). Tetapi pada kasus kombinasi dimana *output layer*nya menggunakan fungsi aktivasi Sigmoid, *learning rate* yang digunakan cukup besar yaitu 3 dan 1. Hal ini wajar jika kita meninjau turunan dari fungsi Sigmoid itu sendiri yaitu . Kita tahu bahwa nilai berada pada *range* [0,1] yang berarti juga berada pada range [0,1]. Hal ini menyebabkan nilai turunan Sigmoid sangatlah kecil sehingga lebih bagus menggunakan *learning rate* yang besar. Secara umum penggunaan *learning rate* yang terlalu besar dapat menyebabkan fluktuasi nilai akurasi seiring bertambahnya epoch. Dari sini dapat disimpulkan bahwa penggunaan *learning rate* yang terlalu besar dapat menyebabkan training model menjadi tidak stabil.

Selain itu secara umum *feature scaling* standarisasi dapat meningkatkan nilai akurasi train lebih baik dibandingkan dengan normalisasi, bahkan kadang normalisasi memberikan nilai akurasi yang lebih buruk daripada tanpa scaling. Tetapi terdapat kasus khusus dimana saat kita melakukan standarisasi ada indikasi bahwa terjadi *overfitting*. Contohnya saja pada kasus identifikasi jenis pakaian kombinasi II dimana nilai akurasi dataset train dan test berbeda jauh sehingga pada akhirnya kami memutuskan untuk tidak melakukan scaling. Dari sini dapat disimpulkan bahwa standarisasi merupakan sebuah metode yang cukup baik untuk dicoba. Selain itu jika melihat hasil eksperimen identifikasi digit angka, dapat disimpulkan juga bahwa scaling dapat mempercepat kekonvergenan dari model dengan epoch yang lebih sedikit, kecuali pada kombiasi IV

Target akurasi 85% dapat dicapai oleh kombinasi I dan II yang menggunakan fungsi aktivasi Softmax pada *output layer*nya. Berbeda dengan kombinasi III dan IV yang hanya bisa mencapai nilai akurasi kisaran 62%. Kedua kombinasi ini menggunakan fungsi aktivasi Sigmoid pada *output layer*nya. Dari sini dapat disimpulkan bahwa fungsi Softmax lebih baik daripada fungsi Sigmoid pada kasus multiklasifikasi. Secara teori memang biasanya fungsi Sigmoid digunakan pada kasus klasifikasi biner sehingga dirasa wajar bahwa performanya lebih buruk

Secara intuisi menambahkan jumlah neuron pada tiap layer seharusnya dapat meningkatkan performa dari model karena media pembelajarannya bertambah. Intuisi yang sama juga berlaku pada strategi menambahkan *hidden layer*.. Kami mengimplementasikan kedua strategi ini pada kombinasi III dan IV dengan harapan dapat mencapai target akurasi 85%. Tapi pada kenyataanya menurut hasil eksperimen kedua strategi ini tidak memberikan hasil yang memuaskan. Hal ini bisa terjadi karena parameter seperti *learning rate* dan scaling yang didapatkan sebelumnya tidak sesuai dengan struktur model yang telah berubah sehingga dirasa perlu mencari kembali nilai parameter yang optimal terhadap struktur model yang baru ini. Dapat disimpulkan bahwa memang model ANN ini sangat kompleks, tidak serta merta dengan menambah media pembelajaran di dalamnya kita bisa mendapatkan nilai akurasi yang lebih baik.