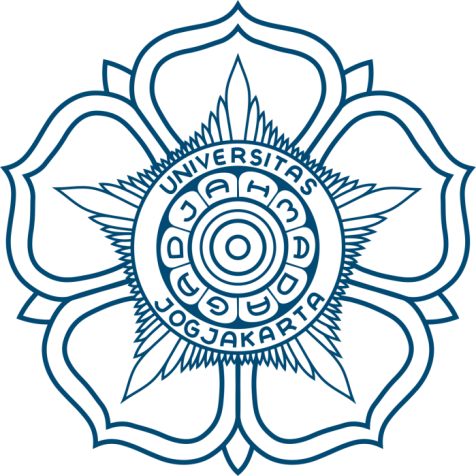
**JARINGAN SARAF TIRUAN:**

**PENGARUH HYPERPARAMETER PADA ARSITEKTUR**

**MULTI LAYER PERCEPTRON**



**Vincent Michael Sutanto**

16/398531/PA/17492

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2019

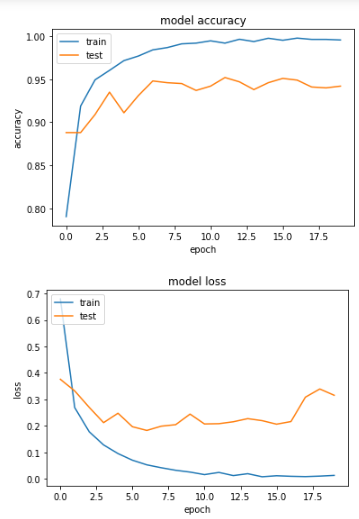
**Pengantar**

Uji coba dilakukan pada data MNIST berukuran 28 x 28 yang diratakan menjadi 1 dimensi berukuran 784 yang kemudian dimasukkan ke dalam multi layer perceptron. Dilakukan *classification task* yang memprediksi data MNIST (6000 data latih dan 1000 data validasi) ke dalam 10 kelas yang berbeda, dengan jumlah epoch sebesar 20 epoch.

Pada percobaan ini dilakukan pengubahan nilai **hyper-parameter** untuk mengetahui pengaruh dari hyper-parameter terhadap performa dari multi-layer perceptron

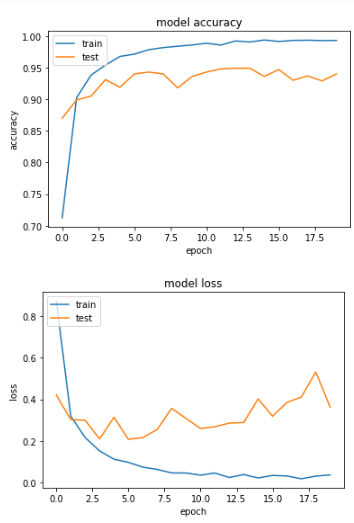
**Hidden Layer**

**1 Hidden Layer**



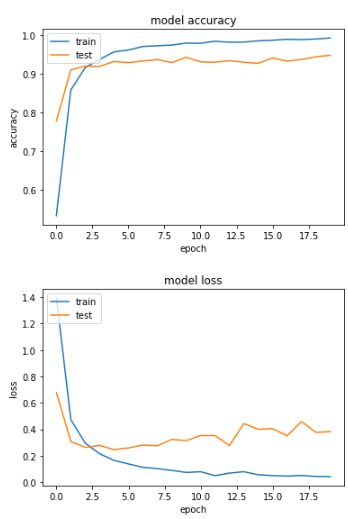
Pada percobaan 1 Hidden Layer didapatkan bahwa tingkat akurasi test mencapai 0.942, namun dari grafik terlihat bahwa akurasi dari train dan test terpaut jarak sekitar 0.06.

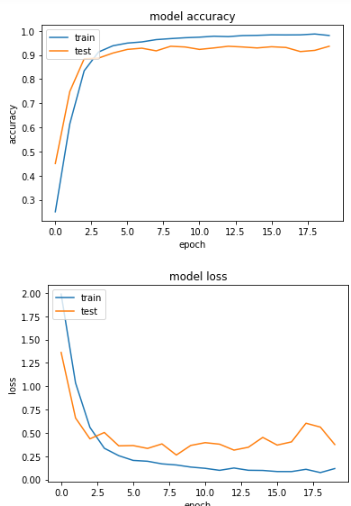
**3 Hidden Layer**



Pada percobaan 3 Hidden Layer didapatkan bahwa tingkat akurasi test mencapai 0.94, namun dari grafik terlihat bahwa akurasi dari train dan test terpaut jarak sekitar 0.06 seperti grafik 1 Hidden Layer, tetapi dengan fitting yang lebih baik (lebih rapat)

**5 Hidden Layer**

Pada percobaan 5 Hidden Layer didapatkan bahwa tingkat akurasi test mencapai 0.948. Dari grafik terlihat bahwa akurasi dari train dan test terlihat lebih rapat jika dibandingkan dengan 1 dan 3 Hidden Layer.

**8 Hidden Layer**

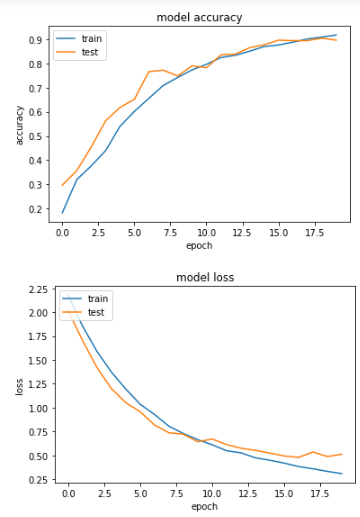
Pada percobaan 8 Hidden Layer didapatkan bahwa tingkat akurasi test mencapai 0.936 (Lebih rendah dari 5 Hidden Layer). Dari grafik juga terlihat bahwa akurasi dari train dan test terlihat lebih rapat jika dibandingkan dengan 1, 3, dan 5 Hidden Layer.

**Kesimpulan Hidden Layer**

Dari percobaan yang dilakukan, jumlah Hidden Layer mempengaruhi fitting dari data training dan data uji, dimana semakin banyak jumlah Hidden Layer, fitting dari data training dan data uji semakin baik (yang paling terlihat berpengaruh adalah Hidden Layer mempengaruhi grafik data uji)

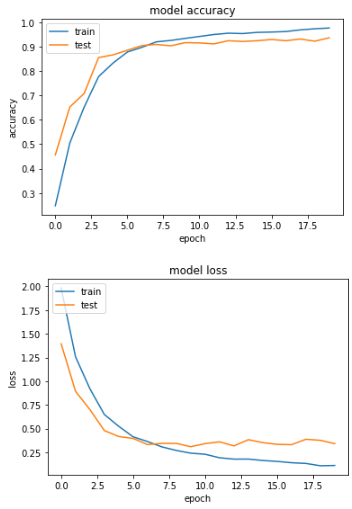
**Node**

**64 Node**



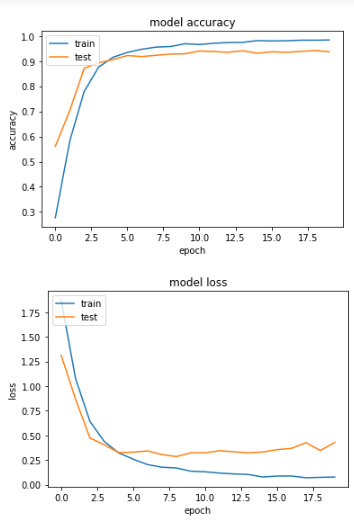
Percobaan dengan menggunakan 64 node menghasilkan tingkat akurasi sebesar 0.896, dengan grafik yang cukup baik (tidak underfit atau overfit). Namun terlihat bahwa penggunaan epoch sebesar 20 belum menghasilkan model yang konvergen untuk jumlah node 64

**128 Node**



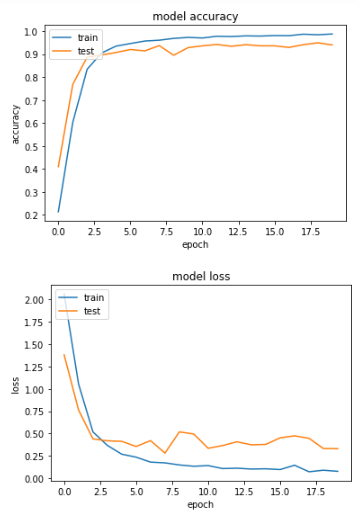
Percobaan dengan menggunakan128 node menghasilkan tingkat akurasi sebesar 0.936, dengan grafik yang cukup baik (tidak underfit atau overfit, selisih training dan validation hanya sebesar 0.05). Terlihat juga dengan menggunakan 20 epoch model dengan jumlah node 128 sudah konvergen.

**256 Node**



Percobaan dengan menggunakan 256 node menghasilkan tingkat akurasi sebesar 0.937, dengan grafik yang cukup baik (tidak underfit atau overfit, selisih training dan validation hanya sebesar 0.05). Terlihat juga dengan menggunakan 20 epoch model dengan jumlah node 256 sudah konvergen dan konvergen lebih cepat dibandingkan dengan 128 node.

**512 Node**



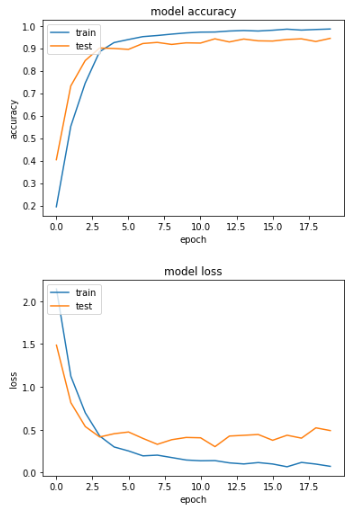
Percobaan dengan menggunakan 512 node menghasilkan tingkat akurasi sebesar 0.94, dengan grafik yang cukup baik (tidak underfit atau overfit, selisih training dan validation hanya sebesar 0.04). Terlihat juga dengan menggunakan 20 epoch model dengan jumlah node 512 sudah konvergen dan konvergen lebih cepat dibandingkan dengan 128 dan 256 node.

**Kesimpulan Node**

Jumlah node pada percobaan yang dilakukan memperlihatkan bahwa semakin banyak node yang digunakan, semakin cepat pula model mengalami konvergensi. Dapat terlihat dimana pada 64 node model belum mengalami konvergensi dalam 20 epoch. Pada 128 node, model mengalami konvergensi pada epoch 8. Pada 256 node, model mengalami konvergensi pada epoch 4. Dan terakhir pada 512 node, model sudah mengalami konvergensi pada epoch 3.

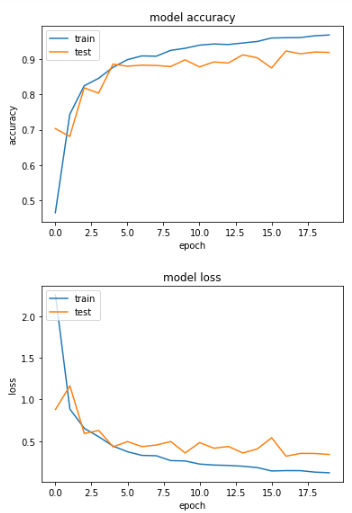
**Activation Function**

**Relu**



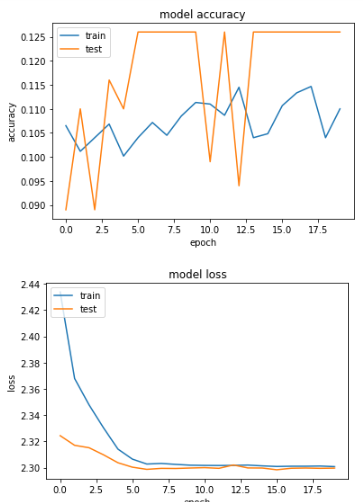
Penggunaan fungsi aktivasi relu menunjukan performa yang bagus pada dataset MNIST dengan arsitektur MLP. Dibuktikan dengan menghasilkan grafik yang konvergen dan nilai akurasi yang cukup tinggi yaitu sebesar 0.944

**TanH**



Penggunaan fungsi aktivasi tanh menunjukan performa yang bagus pada dataset MNIST dengan arsitektur MLP, namun tidak lebih baik jika dibandingkan dengan fungsi aktivasi relu. Grafik yang dihasilkan konvergen dengan selisih training-testing lebih besar daripada relu. Akurasi yang dihasilan sebesar 0.918 pun tidak lebih baik daripada fungsi aktivasi relu

**Sigmoid**



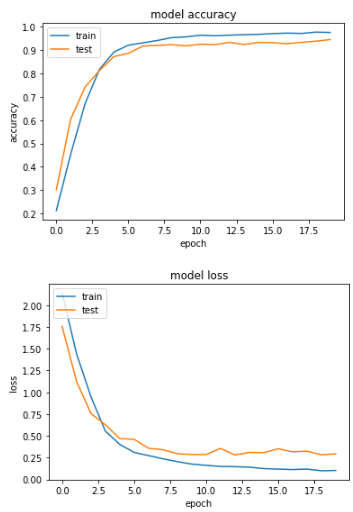
Penggunaan fungsi aktivasi sigmoid bukan pilihan yang tepat pada dataset MNIST dengan arsitektur MLP. Akurasi yang dihasilkan hanya sebesar 0.126 dengan grafik yang tidak konvergen sedikitpun (cenderung overfit).

**Kesimpulan Activation Function**

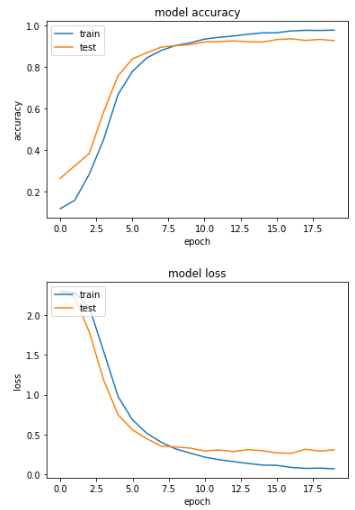
Pada percobaan kali ini fungsi aktivasi relu menunjukan kemampuan yang paling baik dibandingkan dengan tanh, sementara untuk sigmoid bukan pilihan yang tepat jika digunakan dalam arsitektur ini (diasumsikan karena representasi data yang tidak sesuai dengan yang seharusnya diterima oleh fungsi aktivasi sigmoid)

**Learning-rate**

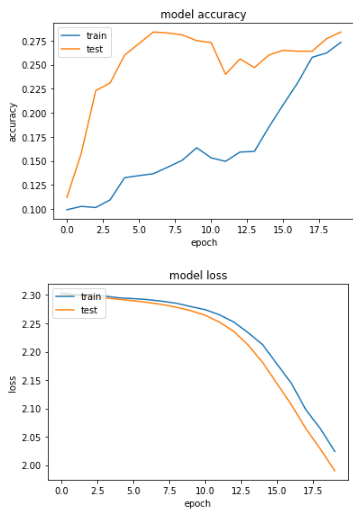
**0.1**



**0.01**



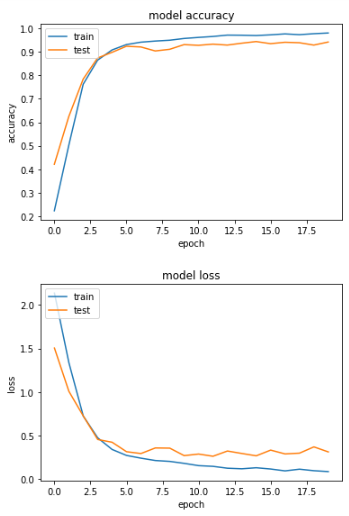
**0.001**



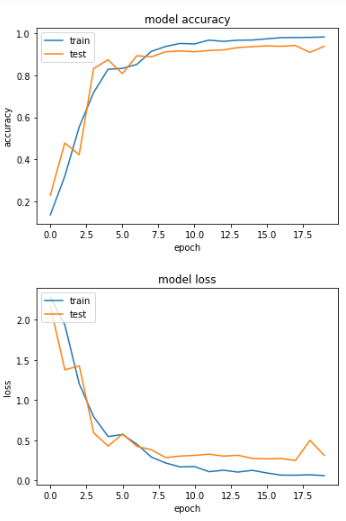
**Kesimpulan Learning Rate**

1. **Momentum**

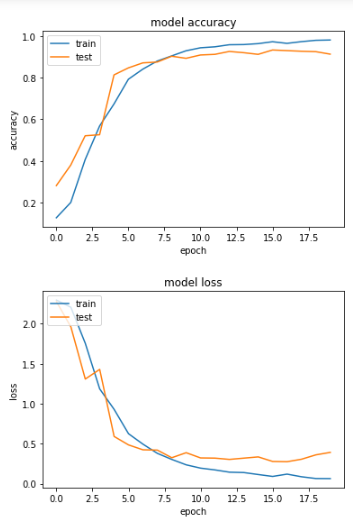
**0.9**



**0.5**



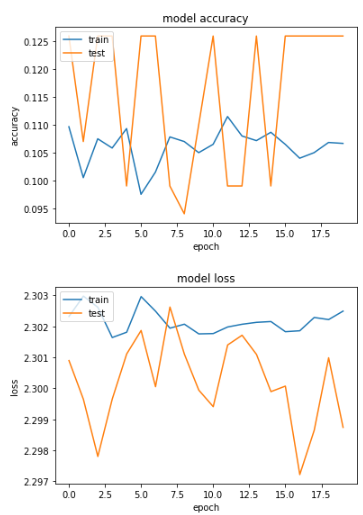
**0.2**



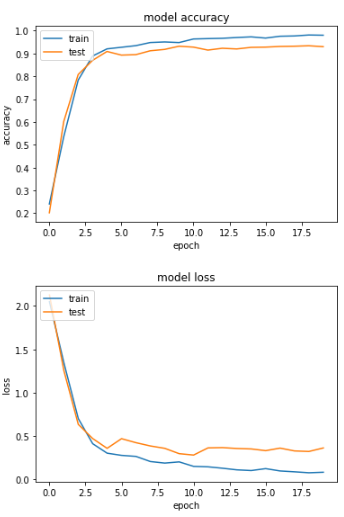
**Kesimpulan Momentum**

1. **Initializer**

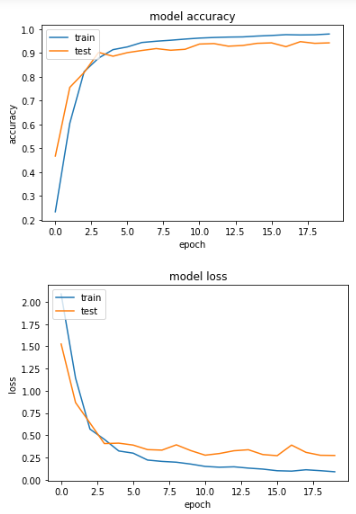
**Zeros**



**Random-Normal**



**Glorot-Uniform**



**Kesimpulan Initializer**