Nama	Hugo Alfedo Putra	**NIM**	225150201111013
Kelas	DL-B	**Tanggal Tugas**	3 September 2024
Judul Tugas		Backpropagation Menggunakan PyTorch	

Laporan ini ditulis sesuai dengan urutan kemunculan cuplikan-cuplikan kode pada contoh feed-forward neural-network PyTorch oleh junjey. Kode yang telah dimodifikasi dapat diakses melalui link Google Colab ini atau melalui Github.

Arsitektur Jaringan

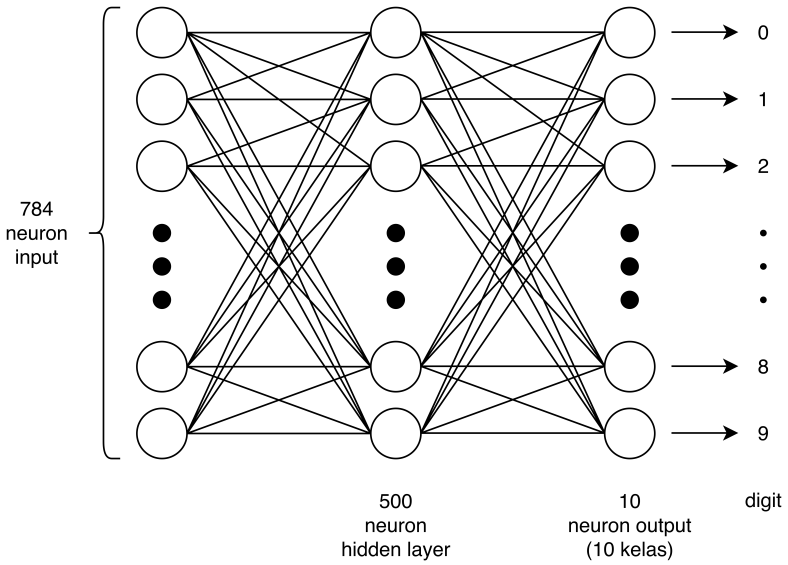
```
# Fully connected neural network with one hidden layer
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        ...

    def forward(self, x):
        ...

model = NeuralNet(input_size, hidden_size, num_classes).to(device)
```

Pertama, `class NeuralNet(nn.Module)` merupakan deklarasi kelas `NeuralNet` yang meng-extend kelas nn.Module milik PyTorch, sebagaimana pula direkomendasikan dalam dokumentasi PyTorch sendiri.

```
self.fc1 = nn.Linear(input_size, hidden_size)
self.relu = nn.ReLU()
self.fc2 = nn.Linear(hidden_size, num_classes)
```

merupakan arsitektur jaringan `NeuralNet` yang terdiri dari dua lapisan berupa Fully Connected (FC) layer yang mengapit fungsi aktivasi ReLU (Rectified Linear Unit). FC layer sendiri berarti tiap neuron input memetakan ke tiap output neuron (M:N).

Gambar arsitektur jaringan seperti berikut:



Implementasi Fungsi Aktivasi Lain

Digunakan fungsi aktivasi lain berupa:

- Untuk hidden layer: (1) Sigmoid dan (2) LeakyReLU untuk menghindari permasalahan pada ReLU di mana saat weighted sum ≤ 0 akan menyebabkan neuron itu mati atau hanya bernilai 0.
- Untuk output layer: Softmax untuk menghasilkan distribusi probabilitas sebagai input Cross Entropy Loss.

Hasil Eksperimen

Hasil RandomizedSearch

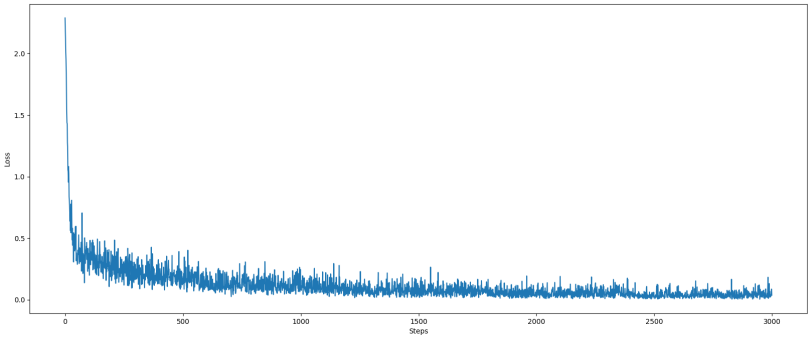
Dari hasil RandomizedSearch dipilih tiga parameter-parameter terbaik, dirincikan:

Nama Model		
`Rank_1`	`Rank_2`	`Rank_3`

`'batch_size': 600,`
`'lr': 0.01,`
`'max_epochs': 3,`
`'activation': LeakyReLU',`
`'hidden_layers': 2,`
`'hidden_size': 400`
	`'batch_size': 600,`
	`'lr': 0.01,`
	`'max_epochs': 7,`
	`'activation': ReLU',`
	`'hidden_layers': 1,`
	`'hidden_size': 600`
		`'batch_size': 100,`
		`'lr': 0.001,`
		`'max_epochs': 3,`
		`'activation': LeakyReLU',`
		`'hidden_layers': 2,`
		`'hidden_size': 400`

Visualisasi Loss

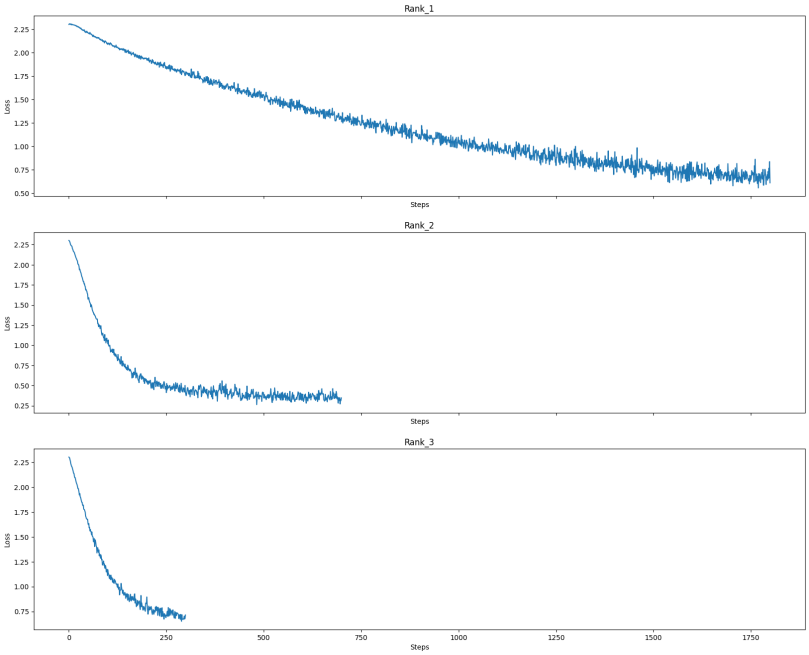
Berikut adalah visualisasi loss dengan hyper-parameter sesuai dengan contoh oleh junjey atau model `yunjey`:



Terlihat bahwa semakin banyak step yang dilakukan, nilai loss mengecil kemudian melandai.

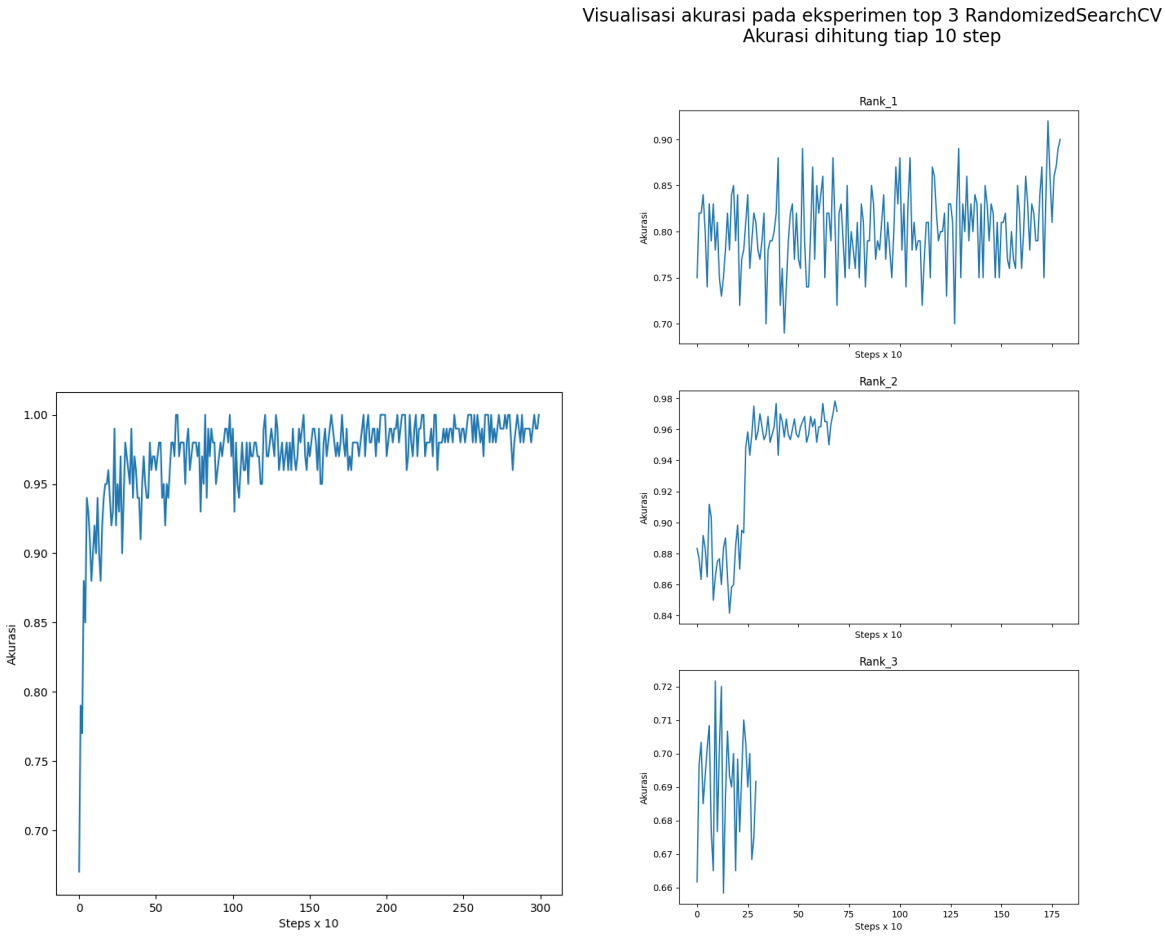
Berikut adalah visualisasi loss pada model `Rank_1`, `Rank_2`, dan `Rank_3`:

Visualisasi loss pada eksperimen top 3 RandomizedSearchCV



Akurasi

Berikut adalah visualisasi akurasi pada model `yunjey` di sisi kiri dan akurasi pada model `Rank_1`, `Rank_2`, dan `Rank_3` di sisi kanan:



Berikut adalah tabel akurasi untuk model `Rank_1`, `Rank_2`, dan `Rank_3`:

Akurasi Model			
`yunjey`	`Rank_1`	`Rank_2`	`Rank_3`
97.86%	69.75%	96.59%	86.14%

Eksperimen Lanjut

Dari visualisasi akurasi, terlihat model `Rank_2` memiliki peningkatan akurasi yang signifikan; berbeda halnya dengan `Rank_3` di mana akurasinya sudah relatif melandai di mana loss-nya pun juga melandai. Hasil tersebut tidak terlihat memiliki potensi karena nilai loss yang sudah melandai dahulu belum tentu akan meningkatkan akurasi (karena dari saat itu model hanya akan belajar terlalu sedikit demi sedikit).

Hasil model `Rank_2`—yang memiliki konvergensi (dalam kasus ini melandainya loss) yang lebih cepat dibanding `Rank_1`—memotivasi eksperimen lebih lanjut yaitu dengan meningkatkan `max_epoch` menjadi 5. Pada `Rank_2` pula akan diimplementasikan fungsi aktivasi sigmoid. Terlihat pula `learning_rate` pada `Rank_1` dan `Rank_2` sama, sehingga konvergensi `Rank_1` yang relatif lamban dapat disebabkan oleh tambahan satu hidden layer.

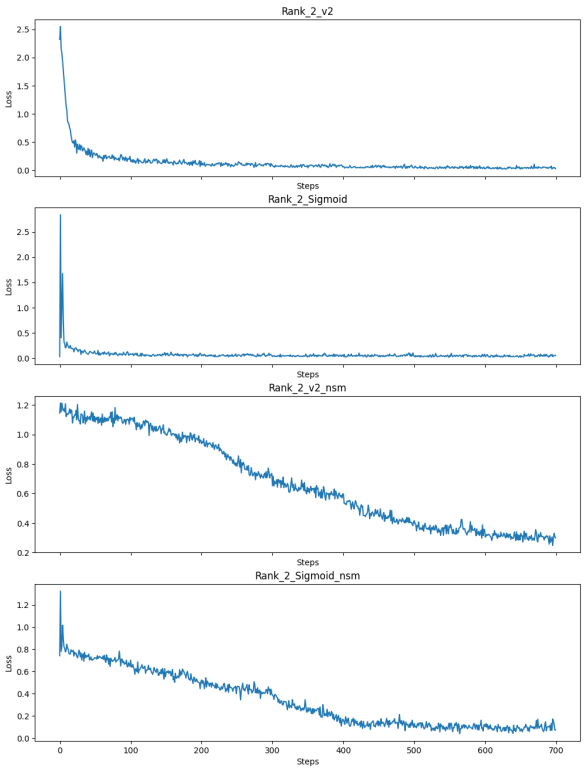
Perlu diketahui model `Rank_1`, `Rank_2`, dan `Rank_3` menggunakan softmax pada neuron output. Hal tersebut memotivasi eksperimen pada model `Rank_2` dengan menerapkan fungsi aktivasi leakyReLU dan sigmoid baik menggunakan softmax maupun tidak.

Model-model modifikasi `Rank_2`			**Model-model modifikasi `Rank_2` tanpa softmax**	
`Rank_2_`	`Rank_2_v2`	`Rank_2_Sigmoid`	`Rank_2_v2_nsm`	`Rank_2_Sigmoid_nsm`
`'batch_size': 600, 'lr': 0.01, 'max_epochs': 7, 'activation': ReLU', 'hidden_layers': 1, 'hidden_size': 600`	`'batch_size': 600, 'lr': 0.01, 'max_epochs': 7, 'activation': LeakyReLU', 'hidden_layers': 1, 'hidden_size': 600`	`'batch_size': 600, 'lr': 0.01, 'max_epochs': 7, 'activation': Sigmoid', 'hidden_layers': 1, 'hidden_size': 600`	`'batch_size': 600, 'lr': 0.01, 'max_epochs': 7, 'activation': LeakyReLU', 'hidden_layers': 1, 'hidden_size': 600`	`'batch_size': 600, 'lr': 0.01, 'max_epochs': 7, 'activation': Sigmoid', 'hidden_layers': 1, 'hidden_size': 600`

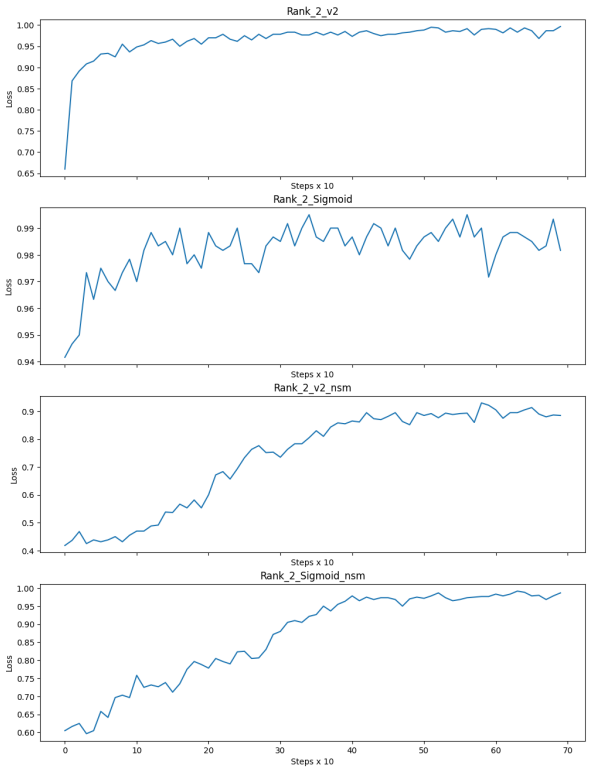
Hasil Eksperimen Lanjut

Berikut adalah visualisasi dan tabel akurasi pada model `yunjey`, `Rank_2`, `Rank_2_v2`, `Rank_2_Sigmoid`, `Rank_2_v2_nsm`, dan `Rank_2_Sigmoid_nsm`:

Visualisasi loss pada eksperimen lanjut



Visualisasi akurasi pada eksperimen lanjut
Akurasi dihitung tiap 10 step



Akurasi Model					
`yunjey`	`Rank_2`	`Rank_2_v2`	`Rank_2_Sigmoid`	`Rank_2_v2_nsm`	`Rank_2_v2_sigmoid`
97.86%	87.11%	96.32%	87.73%	96.95%	97.33%

Dari hasil-hasil tersebut, terlihat bahwa fungsi aktivasi sigmoid lebih cepat konvergen (loss melandai lebih cepat) dibanding dengan ReLU dan LeakyReLU. Terlihat pula loss melandai relatif cukup lamban apabila tidak menggunakan softmax. Namun, penggunaan LeakyReLU pada `Rank_2_v2` menghasilkan akurasi yang lebih landai atau konsisten dibandingkan dengan sigmoid.

Fluktuasi dalam akurasi pada `Rank_2_Sigmoid` dapat disebabkan oleh vanishing gradient problem (VGP) di mana nilai-nilai gradien yang dihasilkan mendekati nol (sangat kecil seakan-akan menghilang). Namun, fluktuasi tersebut hilang atau distabilkan dengan tidak menggunakan softmax seperti pada `Rank_2_Sigmoid_nsm`. Hal tersebut menunjukkan penggunaan softmax pula dapat memperparah VGP karena sejatinya softmax akan bergradien kecil saat nilai x besar (arah positif dan negatif).

Konvergensi Model

Konvergensi model dapat berupa melandainya loss:

1. Pada contoh oleh junjey tampak setelah 2000 step.
2. Pada `Rank_1` terlihat loss menurun cukup lamban dan akhirnya konvergen menjelang step-step terakhir
3. Pada `Rank_2` terlihat loss menurun tajam dan relatif melandai di paling akhir
4. Pada `Rank_3` terlihat loss menurun tajam sama halnya dengan `Rank_2` lalu melandai saat `num_epoch` tercapai
5. Pada `Rank_2_v2` mirip dengan `Rank_2`
6. Pada `Rank_2_Sigmoid` terlihat loss menurun tajam dan bahkan lebih cepat daripada `Rank_2` dan `Rank_2_v2`
7. Pada `Rank_2_v2_nsm` loss menurun cukup lamban; tetapi masih lebih lamban daripada `Rank_1`
8. Pada `Rank_2_Sigmoid_nsm` loss menurun relatif lebih cepat dibanding dengan `Rank_2_v2_nsm`; tetapi masih lebih lamban daripada `Rank_1`

Kesimpulan

Neural network menerapkan forward pass, backward pass, dan optimasi untuk memperbarui nilai-nilai bobot dan bias. Pada dataset MNIST, sebuah neural network memprediksi digit dari gambar. Hasil-hasil latihan memperlihatkan bahwa parameter pada model `yunjey` memberikan hasil yang lebih akurat dibanding model-model eksperimen lainnya. Parameter-parameter yang berbeda pula tentu memberikan pengaruh terhadap konvergensi dan akurasi model. Terdapat parameter yang memberikan konvergensi yang relatif lamban seperti pada `Rank_1` dan juga relatif cepat seperti pada `Rank_2`, `Rank_2_Sigmoid`, `Rank_3`, dan juga `yunjey`. Selain itu, penggunaan fungsi aktivasi juga memainkan peran penting dalam kecepatan konvergensi dan kapan model dapat memprediksi hasil-hasil yang akurat.