Rancangan Pelatihan Paralel Jaringan Saraf *Deep Learning* Berbasis *Map-Reduce*

Moh Edi Wibowo
Departemen Ilmu Komputer dan Elektronika, FMIPA, Universitas Gadjah Mada
FMIPA Sekip Utara, Bulaksumur, Sleman, Yogyakarta, 55281
e-mail: mediw@ugm.ac.id

Abstrak

Jaringan saraf deep learning telah menjadi model pembelajar dengan unjuk kerja yang tinggi pada beragam persoalan pengenalan pola. Meskipun demikian, pelatihan model ini seringkali terkendala oleh keterbatasan memori serta oleh kecepatan pengolahan yang rendah ketika data pelatihan yang digunakan berukuran besar. Untuk menyelesaikan persoalan tersebut, penelitian ini mengusulkan suatu rancangan pelatihan paralel jaringan saraf deep learning berdasarkan kerangka kerja map-reduce pada klaster komputer. Map-reduce diadopsi sebagai kerangka kerja pelatihan paralel karena memiliki dukungan implementasi yang kuat dan beragam.

Kata kunci: *jaringan saraf, deep learning, paralel, map-reduce*

1. Pendahuluan

Jaringan saraf *deep learning* telah dikenal sebagai model pembelajar dengan tingkat memorisasi dan tingkat generalisasi yang sangat baik. Model ini telah menjadi solusi terdepan pada beragam persoalan. Klasifikasi wajah, sebagai contoh, telah berhasil dilakukan secara akurat dengan menggunakan model tersebut pada suatu koleksi ribuan wajah yang dikumpulkan tanpa pengkondisian tertentu [1, 2]. Jaringan saraf *deep learning* juga telah berhasil melakukan pengenalan ucapan manusia (transkripsi) yang lebih baik dibanding model-model pembelajar yang telah lama mapan seperti model hidden Markov [3]. Jaringan saraf tersebut juga mulai mendapatkan perhatian yang besar di beragam bidang ilmu seperti interpretasi citra aerial [4], bioinformatika [5, 6], dan pengenalan pola secara umum.

Dibanding jaringan saraf konvensional (*multilayer perceptron*), jaringan saraf *deep learning* memiliki arsitektur koneksi yang lebih sederhana (konvolusional), jumlah lapisan (kedalaman) yang jauh lebih banyak, serta karakteristik-karakteristik empirik yang lebih menarik. Arsitektur konvolusional telah menolong jaringan saraf *deep learning* menghindari jebakan *local optima*. Hal ini membantu proses pelatihan mendapatkan model yang generik dari data berukuran besar. Lapisan yang banyak memberikan kemampuan kepada model tersebut untuk menemukan secara otomatis ciri-ciri diskriminatif pada data yang seringkali lebih baik daripada ciri-ciri yang dikembangkan dengan teliti secara manual. Dalam prakteknya, hal-hal di muka, sayangnya, seringkali sulit dicapai karena keterbatasan sumber daya komputasi yang ada.

Ukuran data pelatihan yang dapat mencapai puluhan bahkan ratusan gigabytes mengakibatkan adanya kebutuhan sumber daya yang tinggi serta menimbulkan sejumlah persoalan. Pertama, memori yang disediakan oleh sebuah komputer tidak lagi mampu menampung seluruh data pelatihan. Kedua, waktu yang diperlukan untuk menyelesaikan satu iterasi pelatihan seringkali menjadi sangat panjang karena item-item yang ada pada koleksi data berukuran besar tersebut pada umumnya dievaluasi secara sekuensial.

Untuk menangani dua persoalan di muka, penelitian ini mengusulkan suatu rancangan untuk melakukan pelatihan jaringan saraf *deep learning* secara paralel. Pelatihan paralel ini dibuat untuk berjalan pada klaster komputer dengan mengadopsi kerangka kerja *map-reduce*. *Map-reduce* sendiri telah menjadi kerangka kerja yang banyak digunakan untuk mengolah koleksi data tak terstruktur di web, misalnya untuk membangun dan memperbaharui indeks pada mesin pencari. Kerangka kerja ini memiliki dukungan implementasi yang kuat dengan jenis yang beraneka ragam. Sementara *map-reduce* telah pula digunakan untuk mengolah data semi terstruktur, seperti untuk membuat rangkuman informasi dari data historis atau log transaksi, penggunaannya pada pembelajaran jaringan saraf berskala besar belum dieksplorasi secara luas. Rancangan komputasi yang diusulkan, dengan demikian, menjadi suatu upaya

untuk menemukan solusi yang lebih optimal bagi pelatihan jaringan saraf *deep learning* dengan nilai kebaruan yang strategis.

2. Tinjauan Pustaka

Dengan masifnya data yang perlu ditangani pada pengenalan pola, pelatihan *backpropagation* dalam mode *batch* yang umum digunakan pada jaringan saraf tidak lagi dapat dieksekusi pada kebanyakan komputer dengan jangka waktu yang dapat diterima. *Stochastic gradient descent* [7, 8] merupakan suatu pendekatan klasik yang telah diusulkan untuk mengatasi persoalan tersebut. Pendekatan ini tidak lagi meng-*update* jaringan saraf berdasarkan nilai-nilai gradien dari seluruh data, melainkan berdasarkan gradien dari sebagian kecil data yang disebut *mini-batch*. *Mini-batch*, dalam hal ini, diperoleh dari data pelatihan yang ada dengan mengaplikasikan *random sampling*. Meskipun *stochastic gradient descent* dapat menghasilkan model yang cukup baik, pendekatan ini diyakini tetap mengakibatkan penurunan unjuk kerja model, dengan kata lain, tidak memberikan hasil yang maksimal.

High performance computing (HPC) telah menjadi solusi alternatif yang cukup efektif bagi pelatihan jaringan saraf *deep learning* [9]. Pendekatan ini dibangun dengan memanfaatkan model komputasi paralel *message passing interface* (MPI) yang didukung oleh kajian teoritis yang mendalam dan terstruktur. Sementara beragam komputasi saintifik intensif telah ditangani secara memuaskan menggunakan pendekatan ini, kebergantungan HPC pada ketersediaan memori primer masih menjadi sumber kerawanan. Terdapat kemungkinan yang cukup besar bahwa data yang masif tidak dapat ditampung seluruhnya pada memori tersebut. Persoalan serupa juga ditemui pada komputasi paralel berbasis pengolahan larik, yang umum dilakukan menggunakan Graphical Processing Unit (GPU) [10].

Pemanfaatan memori sekunder pada pengolahan data yang terdistribusi menjadi keunikan dan keuntungan dari komputasi paralel berbasis klaster komputer. *Map-reduce* telah menjadi kerangka kerja yang umum digunakan pada komputasi semacam ini. Lebih jauh lagi, *map-reduce* telah dimanfaatkan untuk mengeksekusi algoritma-algoritma pembelajaran mesin, antara lain *k-means* dan *spectral clustering*, klasifikasi *naive* Bayes, dan *singular value decomposition*, seperti pada Apache Mahout (http://mahout.apache. org). Pada penelitian ini, kerangka kerja tersebut digunakan untuk mengembangkan suatu pendekatan yang belum banyak dieksplorasi, yaitu komputasi paralel untuk pelatihan jaringan saraf *deep learning*. Pengembangan ini meliputi perancangan objek jaringan saraf, penentuan skema *map-reduce*, penentuan format data untuk pelatihan, serta penyusunan langkah-langkah pelatihan (*epoch*). *Platform* Hadoop (http://hadoop. apache.org) dipilih pada penelitian ini sebagai target dari *deployment* untuk rancangan yang dibuat.

3. Hasil dan Pembahasan

Pada bagian ini, dijelaskan rancangan pelatihan paralel dari jaringan saraf *deep learning* yang diusulkan.

3.1. Komponen-Komponen *Map-Reduce*

Terdapat 3 komponen utama yang terlibat pada komputasi paralel *map-reduce* seperti ditunjukkan oleh Gambar 1. Objek **Driver** merupakan koordinator dari keseluruhan komponen. Objek ini terletak pada *node* master dan bertugas mengendalikan rangkaian proses pelatihan. Objek ini memegang nilai-nilai parameter (bobot dan bias) dari jaringan saraf yang sedang dilatih, mengirimkan nilai-nilai tersebut ke Mapper, serta mengubah mereka dengan nilai-nilai (rerata) *update* yang diperoleh dari Reducer. Objek **Mapper** merupakan objek yang terduplikasi di seluruh *node* (*slave*) pada klaster. Kumpulan objek Mapper melakukan *parsing* data pelatihan secara paralel, mengolah item-item data yang diperolehnya melalui proses *backpropagation*, menghasilkan nilai-nilai *update* untuk seluruh parameter jaringan saraf, serta mengirimkan nilai-nilai tersebut Reducer-Reducer yang ada. Objek **Reducer** juga merupakan objek yang yang terduplikasi di seluruh *node* (*slave*) pada klaster. Objek-objek ini bertugas melakukan *pooling* serta menghitung rerata nilai-nilai *update* untuk setiap parameter jaringan saraf, dan mengurutkan nilai-nilai rerata tersebut untuk kemudian diberikan kepada Driver.

3.2. Objek Jaringan Saraf

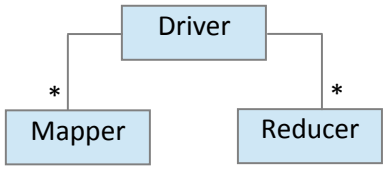
Jaringan saraf pada penelitian ini dirancang sebagai kelas (objek) dengan mengadopsi struktur yang dikembangkan pada `convnet-min.js` (http://cs.stanford.edu/people/karpathy/convnetjs/build/convnet-min.js). Beberapa antarmuka yang utama pada kelas ini antara lain

[1] `void Create(String specification)`: Untuk mengkonstruksi jaringan saraf berdasarkan spesifikasi yang diberikan;
[2] `Vol Forward(Vol input)`: Untuk mengeksekusi langkah *forward* pada suatu *input* dalam *backpropagation* sehingga menghasilkan *output*;
[3] `void Backward(Vol target)`: Untuk mengeksekusi langkah *backward* setelah *forward* selesai dilakukan untuk menghasilkan nilai-nilai *update* berdasarkan target;
[4] `List<double> getUpdates()`: Untuk mendapatkan nilai-nilai *update* yang dihasilkan *backward*;
[5] `List<double> getParams()`: Untuk mendapatkan nilai-nilai parameter (bobot dan bias) yang sedang dimiliki jaringan saraf;
[6] `void setParams(List<double> params)`: Untuk menge-set nilai-nilai parameter (bobot dan bias) dari jaringan saraf.

Antarmuka-antarmuka di atas berfungsi untuk menjembatani interaksi antara objek jaringan saraf dengan Driver, Mapper, dan Reducer. Objek `Vol` merupakan suatu struktur data penting berupa matriks 3 dimensi yang digunakan untuk merepresentasikan *input*, *output*, target, serta bagian-bagian tertentu dari jaringan saraf, seperti filter konvolusi. Adapun spesifikasi jaringan saraf dijabarkan dalam bentuk *string* dengan mengadopsi format JSON, sebagai contoh:

```
[{type:'input', out_sx:1, out_sy:1, out_depth:2},
 {type:'fc', num_neurons:20, activation:'relu'},
 {type:'fc', num_neurons:20, activation:'relu'}].
```

Objek jaringan saraf perlu didistribusikan (disalin) ke seluruh *node* (*slave*) pada klaster untuk digunakan Mapper pada saat eksekusi map.



Gambar 1. Komponen-komponen *map-reduce*

3.3. Skema *Map-Reduce*

Map-reduce pada rancangan yang diusulkan diwujudkan sebagai sebuah iterasi dari pelatihan jaringan saraf *deep learning*. Setiap iterasi melibatkan dua langkah pengolahan data: *map* dan *reduce*.

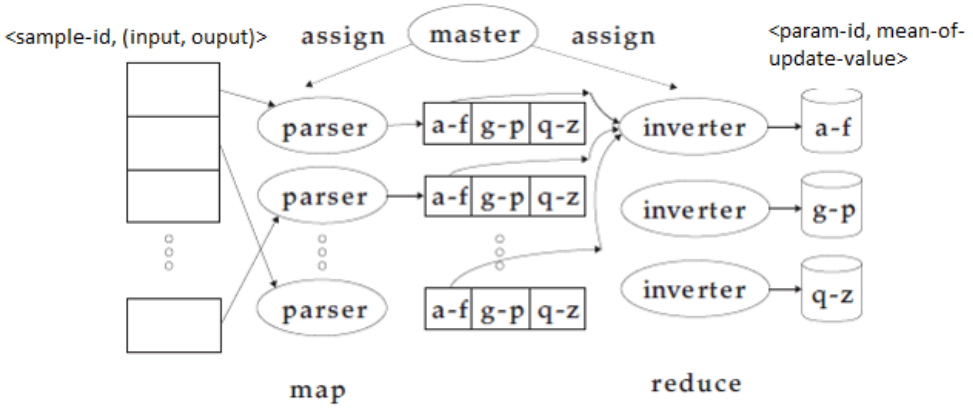
Map merupakan langkah yang dijalankan oleh Mapper dan berfungsi untuk memecah data pelatihan menjadi partisi-partisi (*splits*) kecil, mengolah item-item pada partisi melalui *backpropagation* (*forward* dan *backward*), serta menghasilkan nilai-nilai *update* untuk dikirimkan ke Reducer. Data pelatihan, dalam hal ini, dibaca oleh Mapper kedalam bentuk `<sample-id, (input, target)>`. Untuk mengeksekusi *map*, setiap Mapper perlu menyimpan salinan objek jaringan saraf yang terkini. Driver, karenanya, perlu mendistribusikan spesifikasi jaringan saraf beserta nilai-nilai terkini dari parameternya dalam bentuk *string* ketika menginisiasi setiap

ISSN (Printed) : 2579-7271
ISSN (Online) : 2579-5406

iterasi pelatihan (misalnya melalui objek konfigurasi `job` pada Hadoop). Hasil dari *map* adalah nilai-nilai *update* yang disimpan dalam bentuk `<param-id, update-value>`.

Reduce merupakan langkah yang dieksekusi oleh Reducer dan berfungsi untuk mem-*pool* dan menghitung rerata `update-value` berdasarkan `param-id` yang dihasilkan oleh *map* serta mensortir hasilnya. Reducer, dalam hal ini, membaca *input* dalam bentuk `<param-id, List(update-value)>` dan menghasilkan *output* dalam bentuk `<param-id, mean-of-update-value>`. *Output* yang dihasilkan ini selanjutnya akan dibaca oleh Driver dan digunakan untuk meng-*update* nilai-nilai parameter jaringan saraf *deep learning* (*gradient descent*).

Gambar 2 menyajikan ilustrasi dari skema *map-reduce* yang dirancang. Terlihat bahwa Reducer-Reducer yang ada melakukan *reduce* untuk kelompok-kelompok `param-id` yang berbeda (yaitu a-f, g-p, dan q-z).



Gambar 2. Skema *map-reduce*

3.4. Format Data untuk Pelatihan

Seperti disampaikan pada uraian di muka, Mapper membaca data dalam bentuk `<sample-id, (input, target)>` sedangkan Reducer menghasilkan *ouput* dalam bentuk `<param-id, mean-of-update-value>`. Contoh dari berkas data pelatihan adalah seperti berikut

0001: 4.0, 0.2, 0.1, 0.05, 8.2, 3.6: 0
0002: 3.0, 0.3, 0.7, 0.01, 9.4, 4.6: 0
0003: 2.3, 0.4, 0.5, 0.11, 6.6, 3.0: 1
...

dan contoh dari berkas *output* adalah seperti berikut:

001: 0.0268
002: 0.1342
003: 0.0031
... .

Perlu diingat bahwa data pelatihan dan hasil *map-reduce* dapat didistribusikan atau dibaca sebagai berkas HDFS (*Hadoop file system*). Lebih jauh lagi, hasil akhir dari pelatihan jaringan saraf *deep learning* (spesifikasi, nilai-nilai bobot dan bias) dapat pula disimpan sebagai berkas HDFS.

3.5. Langkah-Langkah Pelatihan

Pelatihan jaringan saraf *deep learning* merupakan proses iteratif yang dikendalikan oleh objek Driver dimana setiap iterasi melibatkan langkah-langkah *map* dan *reduce*. Dengan demikian pelatihan jaringan saraf *deep learning* lebih kompleks daripada pembentukan indeks untuk mesin pencari dimana *map* dan *reduce* hanya perlu dieksekusi dalam satu putaran. Rangkaian proses pelatihan ini dapat dirangkum dalam kode semu berikut:

```
[1] Driver membaca spesifikasi jaringan saraf dan nama berkas data
    pelatihan;
[2] Driver menginisiasi nilai-nilai parameter awal jaringan saraf dan
    menge-set konfigurasi job yang sesuai;
[3] Selama iterasi maksimal belum tercapai: (a) Driver mendistribusikan
    nilai-nilai parameter jaringan saraf terkini, (b) Driver
    menjalankan map-reduce, (c) Driver mengaplikasikan nilai-nilai
    update yang dihasilkan untuk memperbaharui nilai-nilai parameter
    jaringan saraf.
```

Perlu diperhatikan bahwa paralelisasi dilakukan pada evaluasi item-item data pelatihan di dalam iterasi-iterasi pelatihan (paralelisasi untuk *loop* level 2). Adapun proses *update* dari parameter jaringan saraf *deep learning* tetap dilakukan secara sekuensial.

4. Kesimpulan

Pada penelitian ini, telah dirancang suatu pendekatan untuk melakukan pelatihan jaringan saraf *deep learning* secara paralel dengan menggunakan kerangka kerja *map-reduce*. Rancangan yang dibuat meliputi deskrisi komponen-komponen *map-reduce*, objek jaringan saraf, skema *map-reduce*, format data untuk pelatihan, serta langkah-langkah pelatihan jaringan saraf.

Penelitian ini perlu dilanjutkan dengan mengimplementasikan rancangan yang telah dibuat menjadi suatu program yang dapat dieksekusi pada klaster komputer. Program ini selanjutnya dapat dibandingkan dengan program pelatihan jaringan saraf *stochastic gradient descent* untuk diamati perbedaan waktu eksekusinya serta perbedaan unjuk kerja dari model-model jaringan saraf yang dihasilkan.

Daftar Pustaka

[1] Sun, Y., Wang, X. & Tang, X., 2014. Deep learning face representation from predicting 10,000 classes. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. pp. 1891–1898.

[2] He, K. et al., 2015. Deep residual learning for image recognition. Arxiv.Org, 7, pp.171–180. Available at: http://arxiv.org/pdf/1512.03385 v1.pdf.

[3] LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning. Nature, 521, pp.436–444.

[4] Mnih, V., 2013. Machine learning for aerial image labeling. Ph. D. Univeristy of Toronto.

[5] Spencer, M., Eickholt, J. & Cheng, J., 2015. A deep learning network approach to ab initio protein secondary structure prediction. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 12, pp.103–112.

[6] Zhang, S. et al., 2016. A deep learning framework for modeling structural features of RNA-binding protein targets. Nucleic acids research, 44, p.e32.

[7] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp.177–186.

[8] Kingma, D. & Ba, J., 2014. Adam: A method for stochastic optimization. In: International conference on learning representations, pp.1–13.

[9] Coates, A. et al., 2013. Deep learning with COTS HPC systems. In: Proceedings of the 30th international conference on machine Learning, pp.1337–1345.

[10] Zhang, H. et al., 2015. Poseidon: A system architecture for efficient GPU-based deep learning on multiple machines. arXiv. Available at: http://arxiv.org/abs/1512.06216.