Jurnal Penelitian Sistem Informasi
Vol. 1 No. 4 November 2023
e-ISSN :2985-7759, p-ISSN :2985-6310, Hal 215-238
DOI: https://doi.org/10.54066/jpsi.v1i4.1619

Penggunaan Model Deep Learning Untuk Meningkatkan Efisiensi Dalam Aplikasi Machine Learning

Siswanto Siswanto[1], Maya Utami Dewi[2], Siti Kholifah[3], Greget Widhiati[4], Widya Aryani[5]
[1-5]Universitas Sains dan Teknologi Komputer
Koresppondensi Penulis: siswanto@stekom.ac.id

Abstract. *The use of deep learning models has become a major focus in optimizing the efficiency of machine learning applications. This research discusses various deep learning models that can be applied to improve efficiency in the context of machine learning applications. These models are designed to handle the complexity of machine learning tasks with a high level of accuracy while still considering aspects of computational efficiency. This article involves an in-depth look at several deep learning models that have proven effective in various application domains. Discussion includes the use of convolutional neural network (CNNs) models for image processing, recurrent neural networks (RNNs) for sequential data, and transformer-based models for natural language processing tasks. In addition, deep learning model tuning and optimization strategies, such as pruning and quantization, are also discussed to improve the efficient use of computing resources. This research identifies challenges and opportunities in integrating these deep learning models into machine learning applications with maximum efficiency. By considering the need for accuracy and limited computational resources, this research provides a holistic view of the approaches that can be applied to deal with complexity in diverse machine learning scenarios. The results are expected to provide a significant contribution to the development of efficient and effective machine learning applications.*

Keywords: *Artificial Intelligence, Deep Learning, Machine Learning.*

Abstrak. Penggunaan model deep learning telah menjadi fokus utama dalam mengoptimalkan efisiensi aplikasi machine learning. Penelitian ini membahas berbagai model deep learning yang dapat diterapkan untuk meningkatkan efisiensi dalam konteks aplikasi machine learning. Model-model tersebut dirancang untuk menangani kompleksitas tugas-tugas machine learning dengan tingkat akurasi yang tinggi sambil tetap mempertimbangkan aspek efisiensi komputasional. Artikel ini melibatkan tinjauan mendalam terhadap beberapa model deep learning yang telah terbukti efektif dalam berbagai domain aplikasi. Diskusi mencakup penggunaan model convolutional neural networks (CNNs) untuk pengolahan citra, recurrent neural networks (RNNs) untuk data berurutan, dan transformer-based models untuk tugas-tugas pemrosesan bahasa alami. Selain itu, strategi penyesuaian dan optimasi model deep learning, seperti pruning dan quantization, juga dibahas untuk meningkatkan efisiensi penggunaan sumber daya komputasional. Penelitian ini mengidentifikasi tantangan dan peluang dalam mengintegrasikan model-model deep learning ini ke dalam aplikasi machine learning dengan efisiensi yang maksimal. Dengan mempertimbangkan kebutuhan akan akurasi dan keterbatasan sumber daya komputasional, penelitian ini memberikan pandangan yang holistik terhadap pendekatan-pendekatan yang dapat diterapkan untuk menghadapi kompleksitas dalam skenario machine learning yang beragam. Hasilnya diharapkan dapat memberikan kontribusi signifikan pada pengembangan aplikasi machine learning yang efisien dan efektif.

Keywords*: Artificial Intelligent, Deep Learning, Machine Learning.*

PENDAHULUAN

Perkembangan pesat dalam bidang machine learning, terutama dengan penerapan model deep learning, telah membawa transformasi mendalam dalam pemrosesan data dan pengambilan keputusan. Seiring dengan itu, semakin kompleksnya tugas-tugas yang dihadapi oleh aplikasi machine learning menimbulkan tantangan baru terkait efisiensi penggunaan sumber daya komputasional. Oleh karena itu, penting untuk mengidentifikasi dan menerapkan

Revised September 20, 2023, Revised Oktober 20, 2023; Accepted November 30, 2023
*Siswanto Siswanto : siswanto@stekom.ac.id

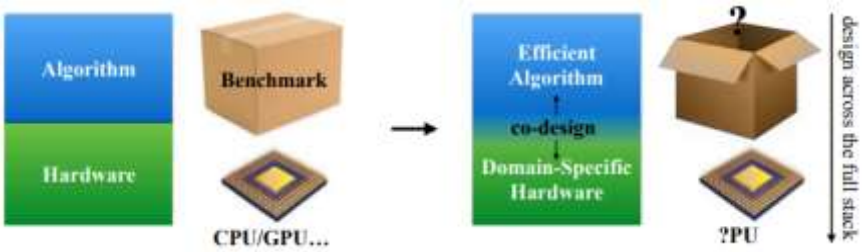
model deep learning yang dapat meningkatkan efisiensi dalam konteks aplikasi machine learning.

Model deep learning, seperti convolutional neural networks (CNNs), recurrent neural networks (RNNs), dan transformer-based models, telah membuktikan keunggulannya dalam menangani berbagai jenis data, termasuk citra, teks, dan data berurutan. Meskipun demikian, penggunaan model-model ini seringkali memerlukan sumber daya komputasional yang signifikan. Oleh karena itu, penelitian ini bertujuan untuk mengkaji cara-cara untuk meningkatkan efisiensi penggunaan model deep learning dalam aplikasi machine learning.

Dengan melibatkan penelitian-penelitian terkini dalam domain ini, kami berusaha untuk menyajikan pandangan yang komprehensif terhadap berbagai model deep learning dan strategi-strategi optimasi yang dapat diterapkan untuk meningkatkan efisiensi. Dengan memahami kompleksitas tugas-tugas machine learning dan keterbatasan sumber daya, diharapkan penelitian ini dapat memberikan landasan bagi pengembangan aplikasi machine learning yang efisien, tanpa mengorbankan tingkat akurasi.Menjalankan model yang lebih besar membutuhkan lebih banyak referensi memori, dan setiap referensi memori membutuhkan dua kali lipat lebih banyak energi daripada operasi aritmatika. Model DNN besar tidak cocok dengan penyimpanan on-chip sehingga memerlukan akses DRAM yang lebih mahal. Terlepas dari tantangan dan kendala yang terjadi dilapangan, berbagai kemajuan dibidang perangkat keras deep learning berkembang dengan cepat dan efisien. Desainer telah merancang akselerator perangkat keras khusus untuk jaringan saraf (Jiantao et al., (2016); Mahendra et al., (2021)). Dengan mengadopsi pendekatan yang sangat spesifik, akselerator ini dapat mengoptimalkan arsitektur perangkat keras sesuai dengan pola komputasi deep learning, sehingga mencapai tingkat efisiensi yang lebih tinggi daripada CPU dan GPU. Gelombang pertama akselerator secara efisien mengimplementasikan primitif komputasi untuk jaringan saraf (Mahendra et al., (2021)).

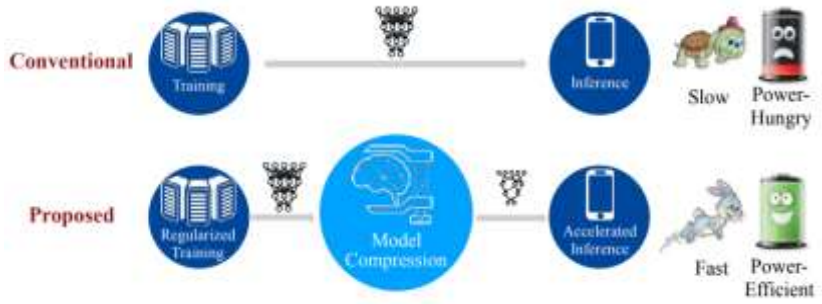
Kedua generasi akselerator deep learning memperlakukan algoritme sebagai black box dan hanya berfokus pada pengoptimalan arsitektur perangkat keras. Dalam penelitian ini ditemukan bahwa, model DNN dapat dikompresi dan disederhanakan secara signifikan sebelum menyentuh perangkat keras. Namun, akselerator perangkat keras yang ada dioptimalkan untuk model DNN yang tidak dikompresi, yang mengakibatkan pemborosan siklus komputasi dan bandwidth memori yang sangat besar dibandingkan dengan menjalankan model DNN yang dikompresi. Oleh karena itu, dalam penelitian ini algoritme dan perangkat keras dirancang bersama untuk deep learning agar berjalan lebih cepat dan lebih hemat energi. Selain itu penelitian ini juga mengembangkan teknik untuk membuat beban kerja deep learning

lebih efisien dan ringkas untuk memulai, lalu merancang arsitektur perangkat keras khusus untuk beban kerja DNN yang dioptimalkan.



Gambar 1. Ini merupakan rancangan hardware terbaik dan metode yang dapat membuat algoritma deep learning tetap efisien yang digunakan dalam penelitian ini.

Sebuah neuron dalam jaringan saraf menerima banyak input dari neuron pendahulunya dan menghasilkan satu output dengan mengikuti fungsi aktivasi yang biasanya nonlinier. Neuron-neuron dikelompokkan dalam lapisan, dimana neuron dalam lapisan yang sama tidak terhubung. Neuron input tidak memiliki pendahulu, sementara neuron output tidak memiliki penerus. Jika jumlah lapisan antara input dan output banyak, maka jaringan disebut sebagai deep neural network. Neuron terhubung melalui koneksi dengan bobot yang akan disesuaikan selama proses pelatihan. Pelatihan dilakukan menggunakan turunan gradien, metode pengoptimalan orde pertama yang menghitung gradien fungsi kerugian dan memindahkan variabel ke arah negatif gradien. Gradien dihitung dengan melakukan feed-forward pass dari input ke output neuron, menghitung gradien fungsi kerugian, dan menghitung gradien secara iteratif dari lapisan output ke lapisan input menggunakan aturan rantai.



Gambar 2 Model dari kerangka penelitian pada umumnya yang diubah dengan model penelitian yang diusulkan dalam penelitian

TINJAUAN PUSTAKA

Arsitektur Neural Network (NN)

Multi-layer perceptron (MLP) terdiri dari banyak lapisan yang terhubung penuh, masing-masing diikuti oleh fungsi non-linear. Dalam MLP, setiap neuron dari layeri terhubung ke layeri+1, dan perhitungan bermuara pada perkalian matriks-vektor. MLP menyumbang lebih dari 61% beban kerja Google TPU (Norman et al., (2017). Convolutional Neural Network

(CNN) mengambil keuntungan dari lokalitas spasial dari sinyal input (seperti gambar) dan berbagi bobot dalam ruang, yang membuatnya tidak berubah untuk terjemahan input. Pembagian bobot seperti itu membuat jumlah bobot jauh lebih kecil dibandingkan dengan lapisan yang terhubung penuh dengan dimensi input/output yang sama.

Dataset

Dataset yang digunakan dalam penelitian adalah dataset yang berbeda untuk berbagai tugas machine learning guna menguji kinerja kompresi model dan teknik regularisasi meliputi MNIST, Cifar-10, dan ImageNet untuk klasifikasi gambar.

Kerangka Deep learning

Memprogram CPU multi-core atau GPU secara langsung bisa jadi sulit, tetapi untungnya perhitungan jaringan saraf dapat diabstraksi menjadi beberapa operasi dasar seperti konvolusi, perkalian matriks, dll. Menggunakan operasi yang sudah sangat dioptimalkan untuk perangkat keras berperforma tinggi, pengguna hanya perlu fokus pada arsitektur jaringan saraf tingkat tinggi daripada implementasi tingkat rendah. Penggunaan kerangka kerja deep learning seperti Caffe dan PyTorch memungkinkan pemrogram untuk fokus pada deskripsi perhitungan jaringan saraf tingkat tinggi dan secara efisien memetakan perhitungan ke perangkat keras berkinerja tinggi. Selain itu, Caffe dan PyTorch menyediakan pilihan model terlatih yang populer dan menggunakan pustaka cuDNN untuk akselerasi GPU. Penggunaan perangkat keras pelatihan seperti kotak pengembangan NVIDIA-DIGITS dengan GPU NVIDIA Maxwell TitanX, memori DDR4 64GB, dan prosesor Core i7-5930K, serta konfigurasi RAID5 dan cache SSD PCI-E M.2 512GB, memungkinkan pencapaian bandwidth I/O disk yang tinggi untuk memuat data pelatihan.

Mengompresi Neural Network

Jaringan saraf biasanya over-parameter, dan ada redundansi yang signifikan untuk model pembelajaran yang mendalam (Misha et al., (2013), ini membuat komputasi dan memori menjadi overload. Emily et al., (2014) mengeksploitasi struktur linear dari jaringan saraf dengan menemukan perkiraan peringkat rendah yang sesuai untuk mengurangi jumlah parameter. Dekomposisi Nilai Singular (SVD) dan Dekomposisi Tucker juga dapat menurunkan jumlah bobot (Alexander et al., (2015). Ada inovasi arsitektural untuk mengurangi jumlah parameter jaringan saraf, seperti mengganti lapisan yang terhubung sepenuhnya dengan lapisan konvolusional atau mengganti lapisan yang terhubung penuh dengan penyatuan rata-rata global. Jaringan dalam Arsitektur jaringan (Min et al., (2013) Sajid et al., (2015) mengkuantisasi jaringan saraf menggunakan minimalisasi kesalahan L2. Kyuyeon et al., (2014) mengusulkan metode optimasi untuk jaringan saraf dengan bobot ternary dan aktivasi

3-bit. Yunchao et al., (2014) mengkompresi jaringan syaraf dalam menggunakan kuantisasi vektor. HashedNets (Wenlin et al., (2015) mengurangi lebar bit parameter dengan menggunakan fungsi hash untuk mengelompokkan bobot koneksi secara acak ke dalam tabel hash. Dengan teknik di atas, presisi bobot dapat dikurangi, dan setiap bobot dapat direpresentasikan dengan bit yang lebih sedikit.

Perangkat Keras Khusus untuk Neural Networks

Meskipun akselerator DNN perangkat keras merupakan area yang relatif baru, sudah ada tiga gelombang desain. Gelombang pertama dan kedua memperlakukan aplikasi sebagai black box. Gelombang ketiga adalah usulan dari penelitian ini yang melihat pengoptimalan perangkat keras/perangkat lunak bersama. Gelombang akselerator pertama termasuk DC-CNN (Han et al., 2016) Convolution Engine (Wajahat et al., (2013) yang mengusulkan logika yang disesuaikan secara efisien untuk memetakan konvolusi ke perangkat keras dengan lebih banyak paralelisme dan fleksibilitas.

Di dunia akademis, banyak akselerator jaringan saraf muncul yang juga memanfaatkan sparsity. Cambricon-X (Shijin et al., (2016) mengeksploitasi ketersebaran bobot dengan menemukan aktivasi yang sesuai hanya dengan bobot bukan nol. SCNN (Angshuman et al., (2017) mengeksploitasi bobot dan sparsity aktivasi dengan melakukan produk luar dari keduanya, dan juga mendukung lapisan konvolusi yang jarang. Cnvlutin2 (Patrick et al., (2017) ditingkatkan di atas Cnvlutin (Jorge et al., (2016) dan mendukung bobot dan sparsity aktivasi untuk melewati komputasi yang tidak efektif. ESE (Song et al., (2017) menggunakan EIE sebagai blok bangunan dasar untuk mendukung RNN dan LSTM yang jarang.

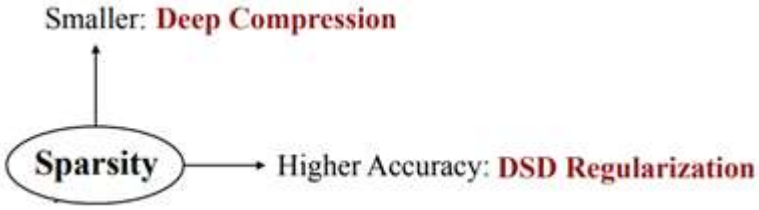
Pemangkasan Deep Neural Network

Jaringan saraf dalam modern memiliki banyak parameter untuk menyediakan kapasitas model yang cukup, menjadikannya intensif secara komputasi dan memori. Selain itu, jaringan saraf konvensional memperbaiki arsitektur sebelum pelatihan dimulai; dengan demikian, pelatihan tidak dapat meningkatkan arsitektur. Selain itu, sejumlah besar parameter dapat menyebabkan overfitting. Mengidentifikasi kapasitas model yang tepat dan menghilangkan redudansi sangat penting untuk efisiensi dan akurasi komputasi. Untuk mengatasi masalah ini, metode pruning dikembangkan untuk menghilangkan redundansi dan mempertahankan koneksi jaringan saraf yang dapat mengurangi kebutuhan komputasi dan penyimpanan untuk melakukan inferensi. Pemangkasan mengubah jaringan saraf yang padat menjadi jaringan saraf yang jarang, mengurangi jumlah parameter dan komputasi sambil mempertahankan akurasi prediksi sepenuhnya. Pemangkasan meningkatkan kecepatan inferensi dan juga mengurangi energi yang diperlukan untuk menjalankan jaringan besar tersebut, memungkinkan

penggunaan pada perangkat seluler dengan baterai terbatas. Pemangkasan juga memfasilitasi penyimpanan dan transmisi aplikasi seluler yang menggabungkan jaringan saraf dalam.

DSD: Pelatihan Padat-Jarang-Padat

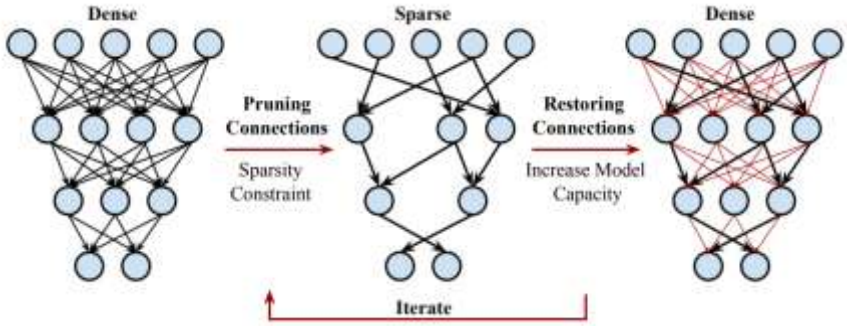
Perangkat keras modern berperforma tinggi memudahkan untuk melatih model DNN kompleks dengan Kapasitas model besar. Keuntungan dari model yang kompleks adalah bahwa mereka sangat ekspresif dan dapat menangkap hubungan yang sangat non-linear antara fitur dan keluaran. Kelemahan dari model besar seperti itu adalah mereka cenderung menangkap kebisingan, daripada pola yang dimaksudkan, dalam dataset pelatihan. Kebisingan ini tidak digeneralisasikan untuk menguji dataset, yang menyebabkan overfitting dan varians yang tinggi. Namun, hanya dengan mengurangi kapasitas model akan mengarah ke ekstrem lainnya: sistem machine learning yang kehilangan hubungan yang relevan antara fitur dan keluaran target, yang mengarah ke underfitting dan bias yang tinggi. Dengan demikian, bias dan varian sulit untuk dioptimalkan secara bersamaan. Untuk mengatasi masalah ini, alur pelatihan padat-jarang-padat (DSD) diusulkan untuk mengatur jaringan saraf dalam, mencegah overfitting dan mencapai akurasi yang lebih baik.



Gambar 3. Mengeksploitasi sparsity untuk meningkatkan efisiensi jaringan neural dari berbagai aspek dalam penelitian ini

Pelatihan DSD mengatur jaringan dengan memangkas dan memulihkan koneksi secara berkala. Jumlah koneksi efektif pada waktu pelatihan berubah secara dinamis. Pemangkasan koneksi memungkinkan melakukan pengoptimalan dalam ruang berdimensi rendah dan menangkap fitur yang kuat; memulihkan koneksi memungkinkan peningkatan kapasitas model. Tidak seperti pelatihan konvensional di mana semua bobot diinisialisasi hanya sekali pada awal pelatihan, DSD memungkinkan koneksi memiliki lebih dari satu kesempatan untuk diinisialisasi melalui pemangkasan dan pemulihan berulang. Keuntungan dari pelatihan DSD adalah bahwa model jaringan saraf final masih memiliki arsitektur dan dimensi yang sama dengan model padat aslinya, sehingga pelatihan DSD tidak memerlukan overhead inferensi apa pun. Tidak diperlukan perangkat keras khusus atau kerangka deep learning khusus untuk melakukan inferensi pada model DSD. Eksperimen menunjukkan bahwa pelatihan DSD meningkatkan kinerja berbagai CNN, RNN, dan LSTM pada tugas klasifikasi gambar, pembuatan teks, dan pengenalan ucapan. Di ImageNet, DSD meningkatkan akurasi Top1

GoogleNet sebesar 1,1%, VGG-16 sebesar 4,3%, ResNet-18 sebesar 1,2%, dan ResNet-50 sebesar 1,1%. Pada dataset WSJ'93, DSD meningkatkan tingkat kesalahan kata (WER) DeepSpeech dan DeepSpeech2 masing-masing sebesar 2,0% dan 1,1%. Pada dataset Flickr-8K, DSD meningkatkan skor NeuralTalk BLEU lebih dari 1,7.



Gambar 4. Pemangkasan berulang DSD dan pemulihan bobot.

DSD mudah diimplementasikan dalam praktiknya: pada waktu pelatihan, DSD hanya menimbulkan satu hiperparameter tambahan: rasio ketersebaran dalam langkah pelatihan jarang. Pada waktu pengujian, DSD tidak mengubah arsitektur jaringan atau mengeluarkan overhead inferensi apa pun. Keuntungan kinerja yang konsisten dan signifikan dari percobaan DSD menyoroti ketidakcukupan metode pelatihan saat ini untuk menemukan optimal global. Pelatihan DSD, berbeda dengan pelatihan konvensional, secara efektif mencapai akurasi yang lebih tinggi.

METODE

Metode Pemangkasan

Pemangkasan jaringan diterapkan pada Caffe (Yangqing et al., (2014) yang telah dimodifikasi untuk menambahkan topeng yang mengabaikan parameter yang dipangkas selama operasi jaringan untuk setiap tensor bobot. Eksperimen ini dilakukan pada GPU Nvidia TitanX dan GTX980.

Pemangkasan untuk MNIST

Eksperimen pertama dilakukan pada dataset MNIST dengan jaringan LeNet-300-100 dan LeNet-5 (Yann et al., (1998). Setelah pemangkasan, jaringan dilatih ulang dengan 1/10 dari kecepatan pembelajaran awal jaringan asli. Hasilnya menunjukkan bahwa pemangkasan mengurangi jumlah sambungan sebanyak 12× pada jaringan ini.

Pemangkasan untuk ImageNet

Di luar dataset MNIST kecil, kinerja pemangkasan pada dataset ImageNet ILSVRC-2012 diepriksa lebih lanjut, yang jauh lebih besar dan karenanya hasilnya lebih meyakinkan. Model AlexNet Caffe digunakan sebagai model referensi, yang mencapai akurasi top-1 sebesar

57,2% dan akurasi top-5 sebesar 80,3%. Setelah pemangkasan, seluruh jaringan dilatih ulang dengan 1/100 dari kecepatan pembelajaran awal jaringan asli. Mengingat hasil yang menjanjikan di AlexNet, jaringan yang lebih besar dan lebih akurat yakni VGG-16 (Karen et al., (2014) juga digunakan pada dataset ILSVRC-2012 yang sama. VGG-16 memiliki jauh lebih banyak lapisan konvolusional tetapi masih hanya tiga lapisan yang terhubung sepenuhnya. Mengikuti metodologi serupa, secara agresif memangkas lapisan konvolusional dan lapisan yang terhubung penuh untuk mewujudkan pengurangan yang signifikan dalam jumlah bobot. Lapisan konvolusi dipangkas menjadi sekitar 30% bukan nol. Jaringan VGG-16 secara keseluruhan telah dikurangi menjadi 7,5% dari ukuran aslinya (13× lebih kecil). Secara khusus, perhatikan bahwa dua lapisan terbesar yang terhubung sepenuhnya masing-masing dapat dipangkas hingga kurang dari 4% dari ukuran aslinya. Pengurangan ini sangat penting untuk pemrosesan gambar real-time, di mana ada sedikit penggunaan kembali lapisan yang terhubung sepenuhnya di seluruh gambar (tidak seperti pemrosesan batch selama pelatihan).

Pemangkasan RNN dan LSTM

Setelah menunjukkan bahwa teknik pemangkasan bekerja dengan baik pada CNN, teknik pemangkasan pada RNN dan LSTM dievaluasi. Pemangkasan model diterapkan ke NeuralTalk (Andrej et al., (2014) menggunakan CNN sebagai ekstraktor fitur gambar dan LSTM untuk menghasilkan teks. Untuk menunjukkan bahwa LSTM dapat dipangkas, bobot CNN diperbaiki dnegan memangkas bobot LSTM. Pada langkah pemangkasan ini semua lapisan dipangkas kecuali W_s, kata yang menyematkan tabel pencarian, menjadi hanya 10% bukan nol. Jaringan jarang yang tersisa di pangkas ulang menggunakan peluruhan berat dan ukuran batch yang sama seperti kertas aslinya. Skor BLEU juga diukur sebelum dan sesudah pelatihan ulang untuk mengukur kesamaan teks yang dihasilkan dengan teks kebenaran dasar.

Metode Percepatan dan Efisiensi Energi

Setelah pemangkasan, persyaratan penyimpanan DNN lebih kecil, yang mengurangi akses DRAM off-chip, bahkan sepenuhnya menghindari akses DRAM off-chip jika semua bobot dapat disimpan pada chip. Memori on-chip membutuhkan lebih sedikit energi dan jumlah siklus yang lebih sedikit untuk diakses, yang memungkinkan inferensi yang lebih cepat dan lebih hemat energi. Performa DNN, terutama layer yang terhubung penuh, bergantung pada apakah banyak gambar dapat dijalankan melalui DNN pada saat yang sama (pemrosesan batch) atau dijalankan satu per satu. Dalam hal ini sejumlah 3 jenis perangkat keras siap pakai digunakan: NVIDIA GeForce GTX Titan X dan Intel Core i7 5930K sebagai prosesor desktop dan NVIDIA Tegra K1 sebagai prosesor seluler. Untuk menjalankan tolok ukur pada GPU, cuBLAS GEMV digunakan untuk lapisan padat asli. Sedangkan lapisan jarang yang dipangkas,

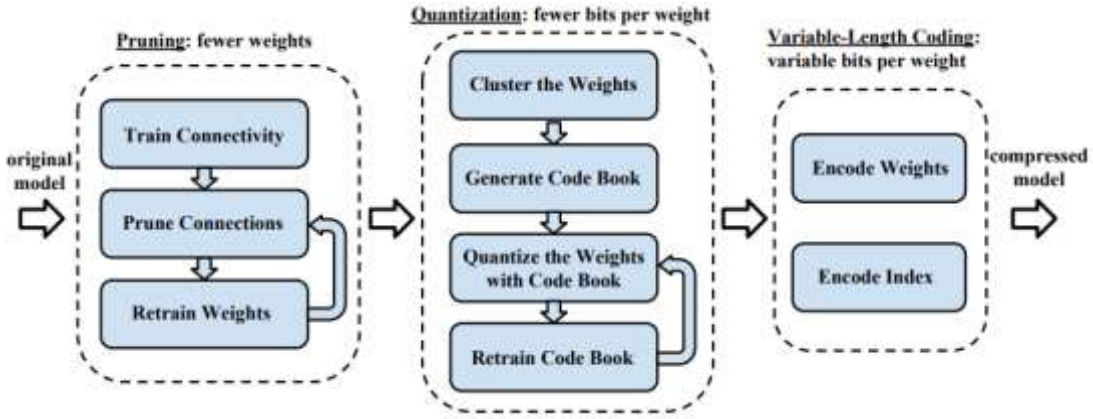
digunakan untuk menyimpan matriks jarang dalam format CSR dan menggunakan kernel cuSPARSE CSRMV, yang dioptimalkan untuk perkalian matriks-vektor jarang pada GPU. Untuk menjalankan benchmark pada CPU, MKL CBLAS GEMV diguanakan untuk model padat asli dan MKL SPBLAS CSRMV untuk model jarang yang dipangkas. Untuk menghindari variasi saat mengukur latensi, waktu yang dihabiskan diukur di setiap lapisan untuk 4096 sampel masukan, dan menghitung rata-rata waktu untuk setiap sampel masukan. Untuk GPU, waktu yang digunakan oleh cudaMalloc dan cudaMemcpy tidak dihitung.

Dalam analisis data penelitian ini, kekuatan pra-regulasi dari gabungan seluruh prosesor aplikasi (AP) / SOC dan DRAM. Pada CPU, benchmark berjalan pada satu soket dengan satu prosesor kelas Haswell-E Core i7-5930K. Soket CPU dan daya DRAM disediakan oleh Intel, sedangkan GPU menggunakan utilitas nvidia-smi untuk melaporkan kekuatan Titan X. Untuk GPU seluler menggunakan papan pengembangan Jetson TK1 dan mengukur konsumsi daya total dengan pengukur daya. Ketika ukuran batch sama dengan 1, lapisan jaringan yang dipangkas memperoleh rata-rata 3× sampai 4× kecepatan di atas jaringan padat karena memiliki jejak memori yang lebih kecil dan mengurangi overhead transfer data, terutama untuk matriks besar yang tidak dapat masuk ke dalam chace jaringan. Dalam aplikasi toleransi latensi seperti pemrosesan gambar off-line, pengelompokan meningkatkan lokalitas memori saat bobot dapat digunakan kembali dalam perkalian matriks-matriks. Dalam skenario ini, jaringan yang dipangkas tidak lagi memiliki keunggulan dan mengakibatkan pelambatan 2× hingga 4× untuk kasus batch = 64. Pemangkasan sadar keseimbangan beban dapat mempertahankan akurasi prediksi dan mencapai percepatan yang lebih baik. Saat parameter dipangkas lebih banyak, tingkat kesalahan meningkat. Kedua metode pruning dapat memangkas 90% parameter sebelum tingkat kesalahan meningkat. Pemangkasan sadar keseimbangan beban dapat meningkatkan pemanfaatan perangkat keras dan mencapai kecepatan yang lebih tinggi daripada metode pemangkasan dasar. Dengan pemangkasan sadar-keseimbangan-beban, model jarang yang dipangkas hingga 10% bukan nol mencapai percepatan 6.2× di atas baseline yang padat, sementara tanpa pemangkasan sadar-keseimbangan-beban hanya percepatan 5.5× yang dicapai. Keuntungan dari load-balance-aware pruning adalah dapat meningkatkan efisiensi perangkat keras dari perspektif beban kerja tanpa mengubah arsitektur perangkat keras.

Model Deep Compression

Deep Compression (DC) terdiri dari pemangkasan, kuantisasi terlatih, dan pengkodean dengan panjang variabel, dan dapat mengompres jaringan saraf dalam dengan urutan besarnya tanpa kehilangan akurasi prediksi. Kompresi besar ini memungkinkan aplikasi machine

learning berjalan di perangkat seluler. Deep Compression membuat penyimpanan yang diperlukan sangat kecil (megabita) sehingga semua bobot dapat di-cache di dalam chip alih-alih masuk ke DRAM di luar chip, yang lambat dan menghabiskan energi.



Gambar 5. Pipa Deep Compression: pemangkasan, kuantisasi, dan pengkodean panjang variabel.

Kuantisasi Terlatih dan Berbagi Berat

Kuantisasi terlatih dan pembagian bobot memampatkan jaringan yang dipangkas dengan mengurangi jumlah bit yang diperlukan untuk mewakili setiap bobot. Jumlah bobot efektif yang perlu disimpan dengan memiliki beberapa koneksi yang memiliki bobot yang sama dibatasi dan disempurnakan. Pengemlompokan k-means dalam penelitian ini sigunakan untuk mengidentifikasi bobot bersama untuk setiap lapisan jaringan yang dilatih, sehingga semua bobot yang termasuk dalam cluster yang sama akan memiliki bobot yang sama. Inisialisasi centroid memengaruhi kualitas pengelompokan sehingga memengaruhi akurasi prediksi jaringan. Terdapat tiga metode yang inisialisasi diantaranya adalah Forgy(acak), berbasis kepadatan, dan inisialisasi linier. Inisialisasi Forgy (acak) secara acak memilih k observasi dari dataset dan menggunakannya sebagai centroid awal. Centroid yang diinisialisasi ditunjukkan dengan warna kuning. Karena ada dua puncak dalam distribusi bimodal, metode Forgy cenderung berkonsentrasi di sekitar dua puncak tersebut. Inisialisasi berbasis kepadatan secara linear meruangkan CDF bobot pada sumbu y, kemudian menemukan perpotongan horizontal dengan CDF, dan terakhir menemukan perpotongan vertikal pada sumbu x, yang menjadi pusat massa (titik biru). Metode ini membuat centroid lebih rapat di sekitar dua puncak, tetapi lebih tersebar daripada metode Forgy. Inisialisasi linier secara linier menempatkan sentroid antara [min, maks] dari bobot asli. Metode inisialisasi ini invarian terhadap distribusi bobot dan paling tersebar dibandingkan dengan dua metode sebelumnya. Bobot besar memainkan peran yang lebih penting daripada bobot kecil (Song et al., (2015), tetapi bobot besar ini lebih sedikit. Jadi, untuk inisialisasi Forgy dan inisialisasi berbasis kepadatan, sangat sedikit centroid yang

memiliki nilai absolut besar, yang menghasilkan representasi yang buruk dari beberapa bobot besar ini. Inisialisasi linier tidak mengalami masalah ini.

Menyimpan Data Meta

Pemangkasan dan kuantisasi terlatih menghasilkan data meta. Pemangkasan membuat matriks bobot jarang, dan dengan demikian diperlukan ruang ekstra untuk menyimpan indeks bobot bukan nol. Kuantisasi membutuhkan penyimpanan ekstra untuk buku kode. Setiap lapisan memiliki buku kodenya sendiri. Dilakukan analisis secara kuantitatif overhead dari data meta lalu struktur jarang yang dihasilkan dari pemangkasan dengan bobot itu sendiri dan indeks disimpan. Jumlah bit yang digunakan diminimalkan untuk mewakili indeks dengan menyimpan indeks relatif (perbedaannya) alih-alih indeks absolut, sehingga perbedaan dalam bit yang lebih sedikit dapat dibandingkkan. Dalam praktiknya, empat bit sudah cukup, yang dapat mewakili perbedaan indeks maksimum 16. Mempertimbangkan sparsity umum 10%, rata-rata setiap sepuluh bobot memiliki bobot bukan nol, jadi 16 adalah batas atas yang baik.

Pengodean Panjang Variabel

Untuk kompresi model lebih lanjut dapat menggunakan pengkodean panjang variabel. Idenya, lebih sedikit bit dapat digunakan untuk merepresentasikan bobot yang lebih sering muncul, dan menggunakan lebih banyak bit untuk merepresentasikan bobot yang lebih jarang muncul. Pengkodean Huffman (Jan, (1976) digunakan untuk Deep Compression. Ini merupakan kode awalan optimal yang biasa digunakan untuk kompresi data lossless menggunakan kata kode dengan panjang variabel untuk menyandikan simbol sumber. Pipa kompresi menghemat penyimpanan jaringan sebesar 17× hingga 49× di berbagai jaringan tanpa kehilangan akurasi. Deep Compression diimplementasikan dengan framework Caffe (Yangqing et al., (2014) dan framework Pytorch. Kuantisasi terlatih dan pembagian bobot diimplementasikan dengan mempertahankan struktur buku kode yang menyimpan bobot bersama, dan kelompokkan berdasarkan indeks setelah menghitung gradien setiap lapisan. Setiap bobot bersama diperbarui dengan semua gradien yang termasuk dalam keranjang itu. Selanjutnya Deep Compression diterapkan pada dataset ImageNet ILSVRC-2012 menggunakan model Caffe AlexNet (Alex et al., (2012) sebagai model referensi, yang memiliki 61 juta parameter dan mencapai akurasi top-1 sebesar 57,2% dan akurasi top-5 sebesar 80,3%. Pemangkasan mengurangi jumlah parameter menjadi 11%. Setelah kuantisasi terlatih, ada 256 bobot bersama di setiap lapisan konvolusional, yang dikodekan dengan 8 bit, dan 32 bobot bersama di setiap lapisan yang terhubung sepenuhnya, yang dikodekan hanya dengan 5 bit. Indeks jarang relatif dikodekan dengan 4 bit. Pengkodean Huffman mengompresi tambahan 22%, menghasilkan total 35× kompresi. Untuk lapisan konvolusi dan terhubung penuh,

kuantisasi 8/5 bit tidak kehilangan akurasi. Kuantisasi 8/4 bit, yang lebih ramah perangkat keras untuk dikodekan dalam mode selaras byte, memiliki kehilangan akurasi yang dapat diabaikan sebesar 0,01%; untuk mencapai kompresi yang lebih agresif, kuantisasi 4/2 bit mengakibatkan hilangnya akurasi Top-1 dan Top-5 sebesar 1,99% dan 2,60%.

Ppada dataset ILSVRC-2012 yang sama, VGG-16 (Karen et al., (2014) memiliki jauh lebih banyak lapisan konvolusional tetapi masih hanya tiga lapisan yang terhubung sepenuhnya. Mengikuti metodologi serupa, secara agresif lapisan convolutional dikompresi. Jaringan VGG-16 secara keseluruhan telah dikompresi sebesar 49×. Pemangkasan mengurangi jumlah parameter menjadi 7,5%. Setelah kuantisasi terlatih, bobot pada lapisan konvolusi diwakili dengan 8 bit, dan lapisan yang terhubung sepenuhnya menggunakan 5bit, yang tidak memengaruhi akurasi. Dua lapisan terbesar yang terhubung sepenuhnya masing-masing dapat dipangkas menjadi kurang dari 1,6% dari ukuran aslinya. Pengurangan ini sangat penting untuk pemrosesan gambar waktu nyata, di mana ukuran batch adalah satu dan tidak seperti pemrosesan batch, penggunaan ulang bobot lebih sedikit. Lapisan yang dikurangi akan cocok dengan SRAM on-chip dan memiliki persyaratan bandwidth sederhana. Tanpa pengurangan, persyaratan bandwidth memori menjadi penghalang.

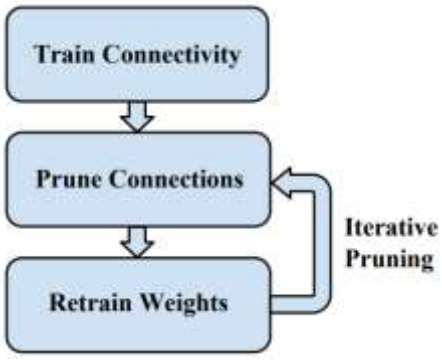
Selanjutnya Deep Compression diterapkan pada jaringan saraf yang sepenuhnya convolutional, yang jauh lebih efisien dalam parameter. 2 perwakilan jaringan saraf yang sepenuhnya konvolusional dipilih, yakni Inception-V3 dan ResNet-50. Hasil kompresi Inception-V3 (Christian et al., (2016) telah dikompresi menjadi 4:6% dari ukuran aslinya, dari 91MB menjadi 4,2MB, yang dapat dengan mudah masuk ke dalam cache SRAM. Kernel 7x7, 7x1 dan 1x7 dapat dipangkas hingga 10%-20% bukan nol, kernel 3x3, 3x1 dan 1x3 dapat dipangkas hingga 20%-30% bukan nol, dan kernel 1x1 dapat dipangkas hingga 20% -60% bukan nol. Semua lapisan dikuantisasi menjadi hanya 4 bit, kecuali beberapa lapisan pertama yang mengekstraksi fitur tingkat rendah dan lapisan pertama dari setiap blok awal. Pengkodean Huffman selanjutnya menurunkan representasi bobot dari 4 bit menjadi 3,8 bit, dan indeks dari 4 bit menjadi 3,1 bit, mendorong rasio kompresi dari 18× menjadi 22×.

Jaringan ResNet-50 (Kaiming et al., (2015) secara keseluruhan telah dikompresi hingga 5.95% dari ukuran aslinya, dari 100MB menjadi 5,8MB. Bobot di lapisan konvolusional dan lapisan yang terhubung sepenuhnya dapat dipangkas menjadi 30% bukan nol, dan dikuantisasi menjadi hanya 4 bit, kecuali untuk beberapa lapisan pertama yang mengekstraksi fitur tingkat rendah. Eksperimen dengan menghitung semua lapisan ResNet-50 yang telah dipangkas menjadi 4-bit juga dilakukan, dan akurasi yang dicapai sebesar [75,91%, 92,84%]. Ini hanya berbeda [0,24%, 0,03%] dari akurasi dasar. Dalam praktiknya semua lapisan pengkodean

Huffman selanjutnya menurunkan representasi bobot dari 4bit menjadi 3,5 bit, dan indeks dari 4 bit menjadi 2,8 bit, mendorong rasio kompresi dari 13× menjadi 17×.

Metode Pemangkasan DNN

Metode pemangkasan menggunakan proses tiga langkah: konektivitas pelatihan, koneksi pemangkasan, dan pelatihan ulang bobot yang tersisa. Proses dimulai dengan mempelajari konektivitas melalui pelatihan jaringan normal, (blok Train Connectivity di Algoritma 1) dengan tujuan untul mempelajari koneksi mana yang penting. Kemudiamn heuristik sederhana digunakan untuk menentukan pentingnya suatu bobot, yaitu nilai absolut: jika nilai absolutnya kecil, maka bobot ini dianggap tidak penting. Langkah kedua adalah memangkas koneksi dengan nilai absolut rendah (blok Prune Connectivity di Algoritma 1). Semua koneksi dengan bobot di bawah ambang batas dihapus dari jaringan, mengubah jaringan padat menjadi jaringan jarang. Dengan mengambil perkalian titik antara tensor bobot asli dan tensor topeng, koneksi yang tidak penting disetel ke nol, artinya koneksi tersebut dipangkas. Langkah terakhir melatih ulang jaringan untuk mempelajari bobot akhir untuk koneksi jarang yang tersisa (blok Retrain Weights di Algoritma 1), jika jaringan yang dipangkas digunakan tanpa pelatihan ulang, akurasi akan terpengaruh secara signifikan.



Gambar6: Jalur pemangkasan DNN secara iteratif yang dilakukan pada penelitian ini

Metode Pelatihan DSD

Pelatihan DSD menggunakan proses tiga langkah: padat, jarang, padat ulang. Langkah pelatihan padat awal mempelajari bobot koneksi dan kepentingannya. Dalam pendekatan penelitian ini, heuristik sederhana digunakan untuk mengukur pentingnya bobot menggunakan nilai absolutnya. Bobot dengan nilai absolut yang besar dianggap penting. Mengoptimalkan jaringan saraf yang dalam adalah masalah yang sangat non-cembung, dan saat ini umumnya diselesaikan menggunakan stochastic gradient descent (SGD), metode pengoptimalan cembung. Akibatnya, redundansi diperlukan untuk membantu konvergensi. Selama pengoptimalan, parameter redundan menyediakan banyak jalur yang memungkinkan konvergensi lebih mudah ke minima lokal yang baik. Jika tidak memiliki langkah pelatihan

yang padat atau memangkas terlalu dini, jaringan gagal untuk menyatu. Setelah pelatihan padat awal menemukan minima lokal yang bagus, aman untuk menghapus koneksi yang berlebihan dan melakukan pengoptimalan di ruang dimensi rendah. Langkah pelatihan yang jarang memangkas koneksi berbobot rendah dan melatih jaringan yang jarang. Sparsity yang sama diterapkan ke semua layer. Jadi ada satu parameter hiper: sparsity, yang merupakan persentase bobot yang dipangkas menjadi 0. Untuk setiap lapisan W dengan parameter N, k-th terbesar $\lambda = S_k$ dipilih sebagai ambang batas di mana k = N*(1-sparsity), dan hasilkan topeng biner untuk menghapus semua bobot yang lebih kecil.

Algorithm 2: DSD training

Initialization: $W^{(0)}$ *with* $W^{(0)} \sim N(0, \Sigma)$
Output : $W^{(t)}$.

Initial Dense Phase

while *not converged* **do**
  $W^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
  $t = t + 1$;
end

Sparse Phase

// initialize the mask by sorting and keeping the Top-k weights.
$S = sort(|W^{(t-1)}|)$;
$\lambda = S_{k_i}$;
$Mask = \mathbb{1}(|W^{(t-1)}| > \lambda)$;
while *not converged* **do**
  $W^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
  $W^{(t)} = W^{(t)} \cdot Mask$;
  $t = t + 1$;
end

Final Dense Phase

while *not converged* **do**
  $W^{(t)} = W^{(t-1)} - 0.1\eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
  $t = t + 1$;
end
goto *Sparse Phase* for iterative DSD;

DSD mengubah jaringan padat menjadi jaringan jarang yang memiliki dukungan ketersebaran yang diketahui. Baik kompresi model (Song et al., (2015); Song et al., (2016) dan pelatihan DSD menggunakan pemangkasan jaringan. Bedanya, fokus pelatihan DSD adalah meningkatkan akurasi, bukan mengurangi ukuran model. Akibatnya, pelatihan DSD tidak memerlukan pemangkasan yang agresif. ditemukan jaringan yang dipangkas sederhana (30% -50% jarang) berfungsi dengan baik. Namun, kompresi model memerlukan pemangkasan jaringan secara agresif untuk mencapai tingkat kompresi yang tinggi. langkah terakhir mengembalikan dan menginisialisasi ulang koneksi yang dipangkas, membuat jaringan kembali padat. Pada langkah ini, koneksi yang sebelumnya dipangkas diinisialisasi dari nol, dan seluruh jaringan dilatih ulang dengan 1/10 laju pembelajaran asli (karena jaringan jarang sudah berada pada minima lokal yang baik, diperlukan laju pembelajaran yang lebih kecil).

Parameter hiper seperti putus sekolah dan penurunan berat badan tetap tidak berubah. Dengan memulihkan koneksi yang dipangkas, pelatihan padat akhir meningkatkan kapasitas model jaringan dan memungkinkan untuk mencapai minima lokal yang lebih baik dibandingkan dengan model jarang. Proses pemulihan ini juga memberikan kesempatan kedua bagi koneksi yang dipangkas untuk diinisialisasi.

Distribusi bobot asli berpusat pada nol dengan ekor yang turun dengan cepat. Pemangkasan didasarkan pada nilai absolut, jadi, setelah pemangkasan, bagian tengah yang besar dipotong. Parameter jaringan yang tidak dipangkas menyesuaikan diri selama pelatihan ulang, distribusi bobot yang tidak dipangkas tetap sama, sedangkan bobot yang dipulihkan didistribusikan lebih jauh di sekitar nol. Nilai absolut rata-rata keseluruhan dari distribusi bobot jauh lebih kecil.

Eksperimen DSD

Pelatihan DSD diterapkan ke berbagai jenis jaringan saraf di berbagai domain dan ditemukan bahwa pelatihan DSD meningkatkan akurasi untuk semua jaringan ini dibandingkan dengan jaringan dasar yang tidak dilatih dengan DSD. Jaringan saraf dipilih dari CNN, RNN, dan LSTM; dataset mencakup klasifikasi gambar, pengenalan ucapan, dan pembuatan teks.

DSD untuk CNN

Model BVLC GoogleNet (Christian et al., (2015) yang diperoleh dari Caffe Model Zoo digunakan dalam eksperimen penelitian ini. Terdapat 13 juta parameter dan 57 lapisan konvolusional, dengan setiap lapisan dipangkas menjadi 30% ketersebaran. Setelah pelatihan berakhir, tingkat kesalahan GoogleNet berkurang sebesar 1,12% dan 0,62% dari garis dasar. Selanjutnya pelatihan DSD dibandingkan dengan pelatihan konvensional dengan jumlah periode pelatihan yang sama dengan menurunkan kecepatan pembelajaran di atas model pra-pelatihan setelah konvergensi. Hasilnya menunjukkan bahwa LLR (learning rate) turun. Masa pelatihan untuk LLR sama dengan DSD sebagai perbandingan yang adil. LLR tidak dapat mencapai akurasi yang sama dengan pelatihan DSD.

Tabel 1. Hasil DSD di GoogleNet

GoogleNet	Top-1 Err	Top-5 Err	Sparsity	Epochs	LR
Baseline	31.14%	10.96%	0%	250	1e-2
Sparse	30.58%	10.58%	30%	11	1e-3
DSD	**30.02%**	**10.34%**	0%	22	1e-4
LLR	30.20%	10.41%	0%	33	1e-5
Improve (abs)	1.12%	0.62%	-	-	-
Improve (rel)	**3.6%**	**5.7%**	-	-	-

Pelatihan DSD pada VGG-16 (Karen et al., (2014) yang banyak digunakan dalam tugas klasifikasi, deteksi, dan segmentasi dijelajahi. Mirip dengan GoogleNet, setiap lapisan VGG-16 dipangkas menjadi 30% ketersebaran (70% bukan nol). Pelatihan DSD sangat mengurangi

kesalahan sebesar 4,31% (Top-1) dan 2,65% (Top-5), DSD juga menunjukkan margin peningkatan akurasi yang besar dibandingkan hasil LLR.

Tabel 2. Hasil DSD pada VGG-16

VGG-16	Top-1 Err	Top-5 Err	Sparsity	Epochs	LR
Baseline	31.50%	11.32%	0%	74	1e-2
Sparse	28.19%	9.23%	30%	1.25	1e-4
DSD	**27.19%**	**8.67%**	0%	18	1e-5
LLR	29.33%	10.00%	0%	20	1e-7
Improve (abs)	4.31%	2.65%	-	-	-
Improve (rel)	**13.7%**	**23.4%**	-	-	-

Pelatihan DSD diterapkan pada ResNet-18 dan ResNet-50 (Kaiming et al., (2015). Model dasar ResNet-18 dan ResNet-50 disediakan oleh Facebook (Facebook, (2016). 30% sparsity digunakan selama pelatihan DSD. Pass pelatihan DSD tunggal untuk jaringan ini mengurangi kesalahan top-1 sebesar 1,26% (ResNet-18) dan 1,12% (ResNet-50), ditunjukkan pada Tabel 3. Sebagai perbandingan yang adil, model asli terus dilatih dengan menurunkan laju pembelajaran selama satu dekade lagi, tetapi tidak dapat mencapai akurasi yang sama seperti DSD. Di bawah waktu pelatihan yang sama, hanya menurunkan tingkat pembelajaran pelatihan konvensional tidak dapat mencapai akurasi yang sama dengan pelatihan DSD.

Tabel 3. Hasil DSD pada ResNet-18 dan ResNet-50

	ResNet-18		ResNet-50				
	Top-1 Err	Top-5 Err	Top-1 Err	Top-5 Err	Sparsity	Epochs	LR
Baseline	30.43%	10.76%	24.01%	7.02%	0%	90	1e-1
Sparse	30.15%	10.56%	23.55%	6.88%	30%	45	1e-2
DSD	**29.17%**	**10.13%**	**22.89%**	**6.47%**	0%	45	1e-3
LLR	30.04%	10.49%	23.58%	6.84%	0%	90	1e-5
Improve (abs)	1.26%	0.63%	1.12%	0.55%	-	-	-
Improve (rel)	**4.14%**	**5.86%**	**4.66%**	**7.83%**	-	-	-

DSD untuk RNN

Pelatihan DSD di RNN dan LSTM dievaluasi di luar CNN. DSD diterapkan padaNeuralTalk (Andrej et al., (2014), sebuah LSTM untuk menghasilkan deskripsi gambar. Ini menggunakan CNN sebagai ekstraktor fitur gambar dan LSTM untuk menghasilkan teks. Untuk memverifikasi bahwa pelatihan DSD berfungsi pada LSTM, bobot CNN diperbaiki dan hanya bobot LSTM yang dilatih. Pada langkah pemangkasan, semua lapisan dipangkas kecuali W_s, tabel pencarian penyematan kata, hingga 80% jarang. Jaringan jarang yang tersisa dilatih secara berulang menggunakan peluruhan berat dan ukuran batch yang sama seperti kertas aslinya. Kecepatan pembelajaran disetel berdasarkan set validasi. Melatih ulang jaringan jarang meningkatkan skor BLUE sebesar [1.2, 1.1, 0.9, 0.7]. Setelah menghilangkan batasan sparsity dan memulihkan bobot yang dipangkas, hasil akhir DSD semakin meningkatkan skor BLEU sebesar [2.0, 2.1, 2.0, 1.7] di atas baseline. Skor BLEU bukan satu-satunya metrik kualitas dari sistem teks gambar. Menarik untuk diperhatikan bahwa model jarang terkadang bekerja lebih

baik daripada model DSD: pada gambar terakhir, model jarang menangkap genangan lumpur dengan benar, sedangkan model DSD hanya menangkap hutan dari latar belakang.

Tabel 4. Hasil DSD pada NeuralTalk

NeuralTalk	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Sparsity	Epochs	LR
Baseline	57.2	38.6	25.4	16.8	0	19	1e-2
Sparse	58.4	39.7	26.3	17.5	80%	10	1e-3
DSD	**59.2**	**40.7**	**27.4**	**18.5**	0	6	1e-4
Improve(abs)	2.0	2.1	2.0	1.7	-	-	-
Improve(rel)	**3.5%**	**5.4%**	**7.9%**	**10.1%**	-	-	-

Pelatihan DSD dijelajahi pada tugas-tugas pengenalan suara menggunakan jaringan Deep Speech 1 (DS1) dan Deep Speech 2 (DS2) (Awni et al. (2014); Dario et al., (2015). Pelatihan DSD meningkatkan akurasi relatif model DS1 dan DS2 pada set tes Wall Street Journal (WSJ) sebesar 2,1% ~ 7,4%. Model DS1 adalah jaringan lima lapis dengan satu lapis Berulang Model Birectional dilatih menggunakan dataset Wall Street Journal (WSJ) yang berisi 81 jam bicara. Set validasi terdiri dari 1 jam pidato. Set tes berasal dari WSJ'92 dan WSJ'93 dan berisi gabungan 1 jam pidato.

Pelatihan DSD membutuhkan 150 epoch, 50 epoch untuk setiap langkah pelatihan D-S-D. Pelatihan DSD meningkatkan WER sebesar 0,13 (WSJ'92) dan 1,35 (WSJ'93) dengan jumlah epoch yang sama dengan pelatihan konvensional. Dengan iterasi DSD kedua (DSDSD), akurasi dapat lebih ditingkatkan. Mirip dengan iterasi pertama, model jarang dan model padat selanjutnya dilatih ulang selama 50 zaman. Tingkat pembelajaran diperkecil untuk setiap langkah pelatihan ulang. Untuk lebih meningkatkan kinerja dapat dilakukan banyak iterasi DSD (DSDSD). Iterasi DSD kedua meningkatkan WER sebesar 0,58 (WSJ'92) dan 1,96 (WSJ'93), peningkatan relatif sebesar 2,07% (WSJ'92) dan 5,84% (WSJ'93).

Tabel 5. Hasil DSD pada Deep Speech 1

DeepSpeech 1	WSJ '92	WSJ '93	Sparsity	Epochs	LR
Dense Iter 0	29.82	34.57	0%	50	8e-4
Sparse Iter 1	27.90	32.99	50%	50	5e-4
Dense Iter 1	27.90	32.20	0%	50	3e-4
Sparse Iter 2	27.45	32.99	25%	50	1e-4
Dense Iter 2	**27.45**	**31.59**	0%	50	3e-5
Baseline	28.03	33.55	0%	150	8e-4
Improve(abs)	0.58	1.96	-	-	-
Improve(rel)	**2.07%**	**5.84%**	-	-	-

Untuk menunjukkan cara kerja DSD pada jaringan yang lebih dalam, DSD pada jaringan Deep Speech 2 (DS2) dievaluasi. Jaringan ini memiliki tujuh lapisan berulang dua arah dengan sekitar 67 juta parameter, sekitar delapan kali lebih besar dari model DS1. Subset dari set pelatihan bahasa Inggris internal digunakan. Set pelatihan terdiri dari 2.100 jam pidato. Set validasi terdiri dari 3,46 jam pidato. Set tes berasal dari WSJ'92 dan WSJ'93, yang berisi gabungan 1 jam pidato. Model DS2 dilatih menggunakan Nesterov SGD selama 20 epoch

untuk setiap langkah pelatihan. Mirip dengan eksperimen DS1, kecepatan pembelajaran dikurangi dengan urutan besarnya dengan setiap pelatihan ulang. Parameter hiper lainnya tetap tidak berubah. Untuk pelatihan ulang jarang pertama, mirip dengan DS1, 50% parameter dari lapisan berulang dua arah dan lapisan yang terhubung sepenuhnya dipangkas menjadi nol. Model Baseline dilatih selama 60 zaman untuk memberikan perbandingan yang adil dengan pelatihan DSD. Model dasar menunjukkan tidak ada perbaikan setelah 40 zaman. Dengan satu iterasi pelatihan DSD, WER meningkat sebesar 0,44 (WSJ'92) dan 0,56 (WSJ'93) dibandingkan dengan baseline yang terlatih penuh.

Tabel 6. Hasil DSD pada Deep Speech 2

DeepSpeech 2	WSJ '92	WSJ '93	Sparsity	Epochs	LR
Dense Iter 0	11.83	17.42	0%	20	3e-4
Sparse Iter 1	10.65	14.84	50%	20	3e-4
Dense Iter 1	9.11	13.96	0%	20	3e-5
Sparse Iter 2	**8.94**	14.02	25%	20	3e-5
Dense Iter 2	9.02	**13.44**	0%	20	6e-6
Baseline	9.55	14.52	0%	60	3e-4
Improve(abs)	0.53	1.08	-	-	-
Improve(rel)	**5.55%**	**7.44%**	-	-	-

Di sini ditunjukkan lagi bahwa DSD dapat diterapkan berkali-kali atau secara iteratif untuk mendapatkan kinerja lebih lanjut. Iterasi kedua pelatihan DSD mencapai akurasi yang lebih baik. Untuk iterasi jarang kedua, 25% parameter di lapisan yang terhubung sepenuhnya dan lapisan berulang dua arah dipangkas. Secara keseluruhan, pelatihan DSD mencapai peningkatan relatif sebesar 5,55% (WSJ'92) dan 7,44% (WSJ'93) pada arsitektur DS2. Hasil ini sejalan dengan eksperimen DSD pada jaringan DS1 yang lebih kecil. Dapat disimpulkan bahwa pelatihan ulang DSD terus menunjukkan peningkatan akurasi dengan lapisan yang lebih besar dan jaringan yang lebih dalam.

Signifikansi Perbaikan DSD

Pelatihan DSD meningkatkan kinerja model dasar dengan memangkas dan memulihkan bobot jaringan secara berurutan. Eksperimen yang lebih intensif dilakukan untuk memvalidasi bahwa peningkatan tersebut signifikan dan bukan karena keacakan dalam proses pengoptimalan. Untuk mengevaluasi signifikansinya, pelatihan dasar, pelatihan DSD, dan penyempurnaan konvensional diulangi sebanyak 16 kali. Signifikansi statistik peningkatan DSD dihitung pada dataset Cifar-10 menggunakan ResNet-20.

Tabel 7. Hasil DSD untuk ResNet-20 pada Cifar-10. Eksperimen diulang 16 kali untuk menghilangkan kebisingan.

ResNet-20	AVG Top-1 Err	STD Top-1 Err	Sparsity	Epochs	LR
Baseline	8.26%	-	0%	164	1e-1
Direct Finetune (First half)	8.16%	0.08%	0%	45	1e-3
Direct Finetune (Second half)	7.97%	0.04%	0%	45	1e-4
DSD (Fist half, Sparse)	8.12%	0.05%	50%	45	1e-3
DSD (Second half, Dense)	**7.89%**	**0.03%**	0%	45	1e-4
Improve from baseline(abs)	0.37%	-	-	-	-
Improve from baseline(rel)	**4.5%**	-	-	-	-

Pelatihan pada Cifar-10 cukup cepat sehingga layak untuk melakukan percobaan intensif dalam waktu yang wajar untuk mengevaluasi kinerja DSD. Eksperimen juga menunjukkan bahwa pelatihan DSD dapat mengurangi variasi pembelajaran: model yang dilatih setelah pelatihan jarang dan pelatihan DSD akhir keduanya memiliki standar deviasi kesalahan yang lebih rendah dibandingkan dengan rekan mereka yang menggunakan penyetelan halus konvensional. Uji-T dilakukan untuk membandingkan tingkat kesalahan dari DSD dan pelatihan konvensional. Hasil T-test menunjukkan bahwa pelatihan DSD mencapai peningkatan yang signifikan dibandingkan dengan model baseline (dengan $p<0,001$) dan fine tuning konvensional (dengan $p<0,001$).

Mengurangi Waktu Pelatihan

Pelatihan DSD melibatkan pemangkasan model yang sudah terlatih sepenuhnya. Ini lebih memakan waktu daripada metode pelatihan umum. Eksperimen pada AlexNet dan DeepSpeech dilakukan, dan ditemukanbahwa pemangkasan awal sebelum model benar-benar konvergen dapat dilakukan, tetapi langkah pelatihan padat pertama tidka dapat dihilangkan. Dalam dua percobaan berulang pada pelatihan jaringan yang jarang, satu model menyimpang setelah iterasi pelatihan 80k, dan sisanya menyatu pada akurasi top-1 hanya 33:0% dan akurasi top-5 41:6%. Jadi pengamatannya adalah bahwa pelatihan yang jarang dari awal menyebabkan akurasi yang lebih rendah. Pendekatan yang kurang agresif adalah melatih jaringan untuk waktu yang singkat dan kemudian memangkasnya.

HASIL

Saat bekerja secara individual, akurasi jaringan yang dipangkas mulai turun secara signifikan saat dikompresi di bawah 8% dari ukuran aslinya; keakuratan jaringan terkuantisasi juga mulai turun secara signifikan saat dikompresi di bawah 8% dari ukuran aslinya. Ini berarti pemangkasan dan kuantisasi dapat secara independen mengurangi ukuran model, tetapi rasio kompresinya tidak signifikan. Namun, saat menggabungkan pemangkasan dan kuantisasi, jaringan dapat dikompresi hingga 3% dari ukuran aslinya tanpa kehilangan akurasi, rasio kompresi yang sangat signifikan saat menggabungkan teknik ini. Kuantisasi bekerja dengan

baik di atas jaringan yang dipangkas karena model yang tidak dipangkas memiliki bobot 100% untuk dikuantisasi, sedangkan model yang dipangkas hanya memiliki 10% hingga 30% dari bobot yang perlu dikuantisasi (yang lainnya semuanya nol). Mengingat jumlah centroid yang sama, yang terakhir memiliki lebih sedikit kesalahan kuantisasi. Kuantisasi seragam dapat ditangani oleh aritmatika titik tetap (Jiantao et al., (2016). Namun, kuantisasi seragam berkinerja lebih buruk daripada kuantisasi tidak seragam dalam hal akurasi. Untuk kuantisasi non-seragam (pekerjaan ini), semua lapisan garis dasar ResNet-50 dapat dikompresi menjadi 4-bit tanpa kehilangan akurasi. Namun, untuk kuantisasi seragam, semua lapisan garis dasar ResNet-50 dapat dikompresi menjadi 8bit tanpa kehilangan akurasi (pada 4 bit, ada sekitar 1,6% kehilangan akurasi top-1 saat menggunakan kuantisasi seragam). Keuntungan dari kuantisasi tidak seragam adalah dapat menangkap distribusi bobot yang tidak seragam dengan lebih baik. Semakin tinggi distribusi probabilitasnya, maka jarak antara setiap centroid akan semakin dekat. Namun, kuantisasi seragam tidak dapat mencapai hal ini.

Selama fine-tuning, salah satu strateginya adalah hanya memperbarui centroid; strategi lainnya adalah memperbarui centroid dan label. Secara intuitif, kasus terakhir memiliki tingkat kebebasan yang lebih besar dalam proses pembelajaran dan seharusnya memberikan kinerja yang lebih baik. Namun, percobaan menunjukkan peningkatan yang tidak signifikan. Jadi banyaknya bobot yang mengubah labelnya dapat dilihat selama proses pelatihan ulang dan menemukan bahwa sebagian besar bobot tetap berada di dalam grup tempat mereka berada selama proses pengelompokan Kmeans pertama. Gradiennya tidak cukup besar untuk mendorong mereka ke centroid tetangga. Ini terutama berlaku untuk bit yang lebih sedikit karena jarak antara centroid yang berdekatan lebih besar daripada ketika memiliki lebih banyak bit. Kcepatan pembelajaran dengan urutan besarnya dicoba untuk dilakukan, kemudian bobot mulai mengubah label. Dengan meningkatkan kecepatan pembelajaran dengan dua kali lipat, bobot yang lebih banyak mulai mengubah label. Namun, model terkuantisasi yang sudah memiliki akurasi > 50% top-1 tidak dapat mentolerir laju pembelajaran sebesar itu, dan proses pengoptimalannya gagal untuk mencapai konvergensi. Selanjutnya masalah Trained Ternary Quantization (Chenzhuo et al., (2016) dapat dipecahkan dengan memperkenalkan faktor penskalaan terpisah untuk centroid yang berbeda, yang memungkinkan pembelajaran yang berbeda untuk centroid yang berbeda. Selama operasi SGD, dua gradien (gradien bobot laten menyesuaikan label, dan gradien faktor penskalaan menyesuaikan pusat massa) dikumpulkan. Dengan mempelajari label dan centroid secara terpisah, kuantisasi ResNet-18 dapat diubah menjadi ternary (positif, negatif, dan nol) sambil kehilangan 3% akurasi Top-1.

Inisialisasi linier mengungguli inisialisasi kerapatan dan inisialisasi acak dalam semua kasus kecuali pada 3 bitBaik inisialisasi acak maupun berbasis kepadatan tidak mempertahankan centroid besar. Dengan metode inisialisasi ini, bobot besar dikelompokkan ke centroid kecil karena bobot besar lebih sedikit daripada bobot kecil. Sebaliknya, inisialisasi linier memberikan bobot yang besar peluang yang lebih baik untuk membentuk pusat massa yang besar, dan menghasilkan akurasi prediksi yang lebih baik. Pemangkasan tanpa data (Suraj et al., (2015) menghemat parameter 1.5× dengan banyak kehilangan akurasi. Deep Fried Convnets (Zichao et al., (2014) bekerja pada lapisan yang terhubung sepenuhnya dan mengurangi parameter sebesar 2 – 3.7×. Maxwell et al., (2014) mengurangi parameter AlexNet sebanyak 4×. Pemotongan ukuran lapisan secara naif dan model pelatihan yang lebih kecil menghemat parameter tetapi kehilangan akurasi sebesar 4%. SVD menyimpan parameter tetapi mengalami kehilangan akurasi yang besar, sebanyak 2% kehilangan akurasi Top-1 di Imagenet. Di jaringan lain struktur linier jaringan saraf konvolusional mengkompresi jaringan dengan lapisan 2.4× hingga 13.4×, dengan kehilangan akurasi 0,9% pada kompresi satu lapisan.

Pelatihan Padat-Jarang-Padat mengubah proses pengoptimalan dan meningkatkan akurasi dengan margin yang signifikan. Selama pelatihan, DSD menyusun ulang jaringan dengan memangkas dan memulihkan koneksi. Berdasarkan penelitian sebelumnya, salah satu kesulitan terbesar dalam mengoptimalkan jaringan dalam adalah proliferasi titik pelana (Yann et al, (2014). Metode DSD yang diusulkan membantu melepaskan diri dari titik sadel dengan memangkas dan memulihkan koneksi. Pemangkasan model konvergensi mengganggu dinamika pembelajaran dan memungkinkan jaringan untuk melompat jauh dari titik pelana, yang memberi kesempatan pada jaringan untuk berkumpul pada minimum lokal yang lebih baik. Ide ini mirip dengan anil simulasi (Chii-Ruey, (1988). Baik simulasi anil dan pelatihan DSD dapat lolos dari solusi suboptimal, dan keduanya dapat diterapkan secara iteratif untuk mencapai peningkatan kinerja lebih lanjut. Perbedaan antara anil simulasi dan pelatihan DSD adalah bahwa anil simulasi melompat secara acak dengan probabilitas yang menurun pada grafik pencarian, DSD secara deterministik menyimpang dari solusi konvergen yang dicapai pada fase pelatihan padat pertama dengan menghilangkan bobot kecil dan menerapkan dukungan sparsity.

Setelah keluar dari titik sadel, DSD dapat mencapai minima lokal yang lebih baik. Penelitian ini juga telah memvalidasi pentingnya perbaikan dibandingkan dengan fine-tuning konvensional menggunakan uji-T. Regularisasi sparsitas dalam langkah pelatihan jarang memindahkan pengoptimalan ke ruang berdimensi lebih rendah di mana permukaan kerugian lebih halus dan cenderung lebih tahan terhadap noise. Eksperimen numerik memverifikasi

bahwa jarang pelatihan mengurangi varian kesalahan pelatihan. Inisialisasi bobot memainkan peran penting dalam deep learning (Dmytro et al., (2015). Pelatihan konvensional memberi setiap koneksi hanya satu kesempatan untuk inisialisasi. DSD memberi koneksi peluang tambahan untuk menginisialisasi ulang selama langkah pemulihan, di mana bobot yang dipulihkan tersebut diinisialisasi ulang dari nol.

Simetri permutasi dari unit tersembunyi membuat bobot menjadi simetris, yang rentan terhadap ko-adaptasi. Di DSD, dengan memangkas dan mengembalikan bobot, proses pelatihan mematahkan simetri unit tersembunyi yang terkait dengan bobot, dan bobot tidak lagi simetris pada fase akhir pelatihan padat. Ada banyak pekerjaan sebelumnya pada teknik regularisasi untuk mencegah over-fitting, seperti penurunan berat badan, dropout, dan normalisasi batch.

KESIMPULAN

Penelitian ini berfokus pada peningkatan efisiensi deep learning dari dua aspek: ukuran model yang lebih kecil dengan Deep Compression dan akurasi prediksi yang lebih tinggi dengan regularisasi DSD. Metode Deep Compression yang mengompresi model jaringan syaraf dalam dengan urutan besarnya tanpa mempengaruhi akurasi prediksi yang diusulkan dalam penelitian ini beroperasi dengan memangkas koneksi yang tidak penting, mengkuantisasi jaringan menggunakan pembagian bobot, dan kemudian menerapkan pengkodean panjang variabel. Deep Compression mengarah pada persyaratan penyimpanan yang lebih kecil dari jaringan saraf dalam dan membuatnya lebih mudah untuk mengimplementasikan jaringan saraf dalam pada aplikasi seluler. Dengan Deep Compression, ukuran jaringan ini sesuai dengan cache SRAM on-chip (5pJ/akses) daripada membutuhkan memori DRAM off-chip (640pJ/akses). Ini berpotensi membuat jaringan saraf dalam lebih hemat energi untuk berjalan di platform seluler. Sedangkan metode pelatihan padat-jarang-padat (DSD) yang mengatur jaringan saraf dengan memangkas dan kemudian memulihkan koneksi yang diusulkan ini digunakan untuk mempelajari koneksi mana yang penting selama pelatihan padat awal. DSD kemudian mengatur jaringan dengan memangkas koneksi yang tidak penting dan melatih ulang ke solusi yang lebih tipis dan lebih kuat dengan akurasi yang sama atau lebih baik. Akhirnya, koneksi yang dipangkas dipulihkan, dan seluruh jaringan dilatih kembali. Ini meningkatkan dimensi parameter dan kapasitas model. Pelatihan DSD mencapai akurasi prediksi yang lebih tinggi. Eksperimen yang dilakukan menggunakan GoogleNet, VGGNet, dan ResNet di ImageNet; NeuralTalk di Flickr-8K; DeepSpeech dan DeepSpeech-2 pada dataset WSJ menunjukkan bahwa akurasi CNN, RNN, dan LSTM dapat

memperoleh manfaat signifikan dari pelatihan DSD. Uji-T dalam penelitian ini dilakukan untuk memverifikasi bahwa peningkatan pelatihan DSD signifikan secara statistik. Hasil eksperimen menunjukkan keefektifan pelatihan DSD dalam meningkatkan akurasi.

DAFTAR PUSTAKA

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

Chii-Ruey Hwang. Simulated annealing: theory and applications. Acta Applicandae Mathematicae, 12(1):108–111, 1988.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint arXiv:1609.04802, 2016.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2818–2826, 2016.

Christopher A Walsh. Peter huttenlocher (1931-2013). Nature, 502(7470):172–172, 2013.

Convex optimization. In Advances in neural information processing systems, pages 2933–2941, 2014.

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In International Conference on Learning Representations (ICLR).

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710, 2016.

Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. Exploring the regularity of sparse structure in convolutional neural networks. arXiv preprint

Kyuyeon Hwang and Wonyong Sung. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In Signal Processing Systems (SiPS), 2014 IEEE Workshop on, pages 1–6. IEEE, 2014.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In ICML, pages 1058–1066, 2013.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in Neural Information Processing Systems, pages 3105–3113, 2015.

Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. arXiv preprint arXiv:1412.1442, 2014.

Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. arXiv preprint arXiv:1707.09405, 2017.

Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. Cambricon-x: An accelerator for sparse neural networks. In Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on, pages 1–12. IEEE, 2016.

Sicheng Li, Wei Wen, Yu Wang, Song Han, Yiran Chen, and Hai Li. An fpga design framework for cnn sparsification and acceleration. In Field-Programmable Custom Computing Machines (FCCM), 2017 IEEE 25th Annual International Symposium on, pages 28–28. IEEE, 2017.

Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In Proceedings of the 19th international conference on Architectural support for programming languages and operating systems, pages 269–284. ACM, 2014.

Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. arXiv preprint arXiv:1412.7149, 2014.