



UNIVERSITAS INDONESIA

**THIN-CLIENT PIPELINING DARI FAST R-CNN UNTUK PENGENALAN
BATIK**

THESIS

**MUHAMMAD ARIF NASUTION
1306500580**

**FAKULTAS ILMU KOMPUTER
MAGISTER ILMU KOMPUTER
DEPOK
JUNI 2016**



UNIVERSITAS INDONESIA

**THIN-CLIENT PIPELINING DARI FAST R-CNN UNTUK PENGENALAN
BATIK**

THESIS

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Master Ilmu Komputer**

MUHAMMAD ARIF NASUTION

1306500580

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER ILMU KOMPUTER
DEPOK
JUNI 2016**

ABSTRAK

Nama : Muhammad Arif Nasution
Program Studi : Magister Ilmu Komputer
Judul : Thin-Client Pipelining dari Fast R-CNN untuk Pengenalan Batik

Kain batik merupakan salah satu warisan kebudayaan Indonesia yang menjadi salah satu warisan dunia menurut UNESCO. Batik memiliki beberapa motif seperti motif kawung, motif parangkusumo, motif truntum, motif tambal, motif pamiluto, motif parang, motif liris maupun motif udan nitik. Selain banyak variasi motif batik, daerah asal batik juga beragam dan memiliki makna yang memiliki keterikatan dengan daerah asal batik tersebut. Dengan beragamnya daerah motif maupun daerah asal batik, akan menjadi sulit untuk mengetahui daerah asal batik tersebut jika tidak didukungnya informasi maupun pengetahuan terkait batik. Dengan menggunakan metode convolution neural network, diharapkan dapat membantu mempelajari variasi motif batik untuk mendeteksi daerah asal batik tersebut. Convolution neural network akan melakukan pembelajaran image dengan memecah gambar menjadi lebih kecil atau kernel dan diproses kedalam 2 layer utama (layer konvolusi dan subsampling) hingga mencapai output layer dan dilakukan evaluasi terhadap data tes. Untuk melakukan komputasi CNN, digunakan library deeplearning4j sehingga hasil pembelajaran CNN bisa disimpan dalam file biner yang bisa digunakan kembali untuk melakukan evaluasi motif batik.

Kata Kunci:

Android, Deep Learning, Convolution Neural Network

ABSTRACT

Name : Muhammad Arif Nasution
Program : Magister Ilmu Komputer
Title : Thin-Client Pipelining of Fast R-CNN for Batik Recognition

Batik is one of Indonesia traditional fabric which recognized by UNESCO as Masterpiece of Oral and Intangible Heritage of Humanity. Batik has many textures such as kawung, parangkusumo, truntum, tambal, pamiluto, parang liris and udan nitik. Moreover, every province in indonesia has their own texture that represent their culture. The problem of batik classification is appeared caused by variation in texture or province and no useful information and knowledge can support batik recognition. So, CNN is proposed to be used in learning texture for batik recognition. CNN will process input into kernel and processed into common layer called convolution and subsampling and the output layer can be used for batik recognition. For CNN computation, deeplarning4j will be used so learning result can be stored into binary file and can be used to evaluate batik texture for batik recognition.

Keywords:

Android, Deep Learning, Convolution Neural Network

DAFTAR ISI

HALAMAN JUDUL	i
ABSTRAK	ii
Daftar Isi	iv
Daftar Gambar	vi
Daftar Tabel	vii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Penelitian	2
1.4 Batasan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Metodologi Penelitian	3
1.7 Sistematika Penulisan	4
2 LANDASAN TEORI	5
2.1 Penelitian Sebelumnya	5
2.2 <i>Feedforward Neural Network</i>	8
2.3 Convolution Neural Network (CNN)	10
2.3.1 Convolutional Layer	12
2.3.2 Subsampling Layer	13
2.3.3 <i>Fully Connected Layer</i>	14
2.4 <i>Backpropagation</i>	15
2.5 Algoritma Optimasi	16
2.5.1 <i>Stochastic Gradient Descent</i> (SGD)	16
2.5.2 <i>Nesterov's Accelerated Gradient</i> (Nesterov)	16
2.6 Fungsi Galat	17
2.7 Fast R-CNN	17
2.7.1 RoI pooling layer	17
2.7.2 Pre-trained networks	18

2.7.3	<i>Fine-tuning</i> untuk Deteksi	18
2.7.4	Scale invariance	21
2.8	Thin Client Computing	21
2.9	Teknologi Perangkat Mobile	22
2.9.1	Sensor pada mobile	22
2.9.2	Mobile SoC (System On Chip)	22
2.10	Batik	23
3	PERANCANGAN	25
3.1	Tahapan Penelitian	25
3.2	Rencana Implementasi	26
	Daftar Referensi	30

DAFTAR GAMBAR

2.1	Mind map penelitian terdahulu terkait CNN	5
2.2	Contoh <i>feedforward neural network</i>	8
2.3	Interpretasi geometris pada peran <i>hidden layer</i>	9
2.4	<i>Sparse Connectivity</i>	11
2.5	<i>Shared weights</i> untuk <i>filter</i> yang sama	11
2.6	Ilustrasi <i>layer</i> pada <i>convolutional neural network</i>	12
2.7	Ilustrasi <i>max pooling</i>	14
2.8	Contoh arsitektur <i>convolutional neural network</i>	14
2.9	Arsitektur Fast R-CNN	18
2.10	Contoh kain dari berbagai daerah di indonesia	23
2.11	Contoh motif batik indonesia	24
3.1	Metodologi penelitian	25
3.2	Rencana Implementasi	27
3.3	Deep Architecture Decomposition	28
3.4	NVIDIA Tegra K1	29

DAFTAR TABEL

2.1	Matriks literatur penelitian terdahulu yang berhubungan dengan penelitian Convolution Neural Network	7
-----	--	---

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Berdasarkan [4], pada saat ini model statistik yang memberikan akurasi terbaik untuk melakukan pengenalan kebiasaan manusia maupun objek umum dibangun menggunakan metode deep learning, yaitu salah satu area pembelajaran mesin yang sedang berkembang pesat dan memiliki kemampuan untuk melakukan pemodelan data kompleks dari dunia nyata. Area penelitian yang dipengaruhi oleh deep learning mencakup pengenalan wajah [7], aksi [8], gestur [9] maupun integrasi deep learning dengan metode heuristik [12]. Sayangnya, meskipun penelitian terkait deep learning tersebut akan sangat penting dan bermanfaat ketika diimplementasikan pada perangkat mobile, masih sangat sedikit penelitian yang mengadopsi teknik deep learning pada perangkat mobile.

Pendekatan deep learning pada teknologi mobil pada saat ini banyak memiliki kekurangan [4]. Salah satu solusinya adalah melakukan proses deep learning pada cloud, tetapi teknologi cloud sangat tidak praktis untuk jangka panjang menyangkut biaya pelayanan cloud dan konsumsi bandwidth internet. Selain itu, ketika terjadi permasalahan pada jaringan dan server cloud tidak aktif, maka aplikasi tersebut akan menjadi tidak berguna. Untuk melakukan komputasi deep learning pada perangkat CPU lokal memang memungkinkan untuk beberapa skenario, tetapi dibutuhkan usaha dan kemampuan yang mahir, selain itu penggunaan model tersebut tidak bisa digunakan secara umum oleh model deep learning lainnya. Padahal, permasalahan kompleks yang diselesaikan model deep learning adalah hal yang biasanya dibutuhkan pada perangkat mobile.

Batik merupakan kain bergambar yang memiliki gaya, warna dan tekstur dimana proses pembuatannya dilakukan secara manual maupun menggunakan mesin dan merupakan salah satu kain tradisional yang dimiliki oleh negara Indonesia. Batik sudah ditetapkan sebagai Warisan Kemanusiaan untuk Budaya Lisan dan Nonbendawi oleh UNESCO sejak 2 Oktober 2009. Dengan keberagaman suku dan budaya di Indonesia, menyebabkan Indonesia memiliki variasi motif batik yang sangat beraneka ragam dan memiliki makna simbolis berdasarkan daerah asal batik tersebut. Batik memiliki beberapa motif seperti motif kawung, motif parangkusumo, motif truntum, motif tambal, motif pamiluto, motif parang, motif

liris maupun motif udan nitik yang menjadi contoh begitu beragamnya variasi motif batik yang ada di Indonesia.

Kontribusi utama dalam penelitian ini adalah mengusulkan metode pengenalan motif batik menggunakan Fast R-CNN yang merupakan pengembangan dari Convolutional Neural Network yang digabungkan dengan metode Spatial pyramid pooling networks (SPPnets). Fast R-CNN merupakan perbaikan dari R-CNN yang berfokus pada perbaikan waktu komputasi. Diharapkan dengan menggunakan Fast R-CNN untuk pengenalan motif batik, akurasi yang dihasilkan akan lebih baik ditambah waktu komputasi yang lebih cepat dibandingkan R-CNN. Selain itu, kontribusi lainnya adalah penggunaan metode deep learning pada perangkat mobile SoC, dimana Fast R-CNN akan digabungkan dengan metode DAD (Deep Architecture Decomposition) dan RLC (Runtime Layer Compression) untuk menyesuaikan komputasi deep learning dengan sumber daya yang terbatas pada perangkat mobile.

1.2 Rumusan Masalah

Pendahuluan pada butir 1.1 menimbulkan permasalahan-permasalahan yang perlu diselesaikan. Penelitian ini dilakukan untuk menemukan jawaban dari permasalahan-permasalahan tersebut, antara lain sebagai berikut:

1. Bagaimana merancang algoritma Fast R-CNN untuk melakukan proses learning dan deteksi motif batik.
2. Bagaimana integrasi algoritma Fast R-CNN dengan pendekatan Deep Architecture Decomposition & RLS Runtime Layer Compression pada perangkat mobile

1.3 Tujuan Penelitian

Berdasarkan permasalahan pada butir 1.2, tujuan-tujuan yang akan dicapai dalam penelitian ini adalah sebagai berikut.

1. Implementasi Fast R-CNN untuk pengenalan motif batik
2. Implementasi Fast R-CNN dengan integrasi pendekatan DAD & RLC pada perangkat mobile untuk pengenalan motif batik

1.4 Batasan Penelitian

Dalam melakukan penelitian ini, terdapat beberapa batasan-batasan yang digunakan. Batasan-batasan tersebut adalah sebagai berikut.

1. Penelitian ini menggunakan data gambar dengan ukuran 300x300.
2. Penelitian ini menggunakan metode Fast R-CNN untuk membangun model pengenalan motif batik.
3. Penelitian ini hanya mendeteksi 5 motif batik, Ceplok, Kawung, Lereng, Nitik dan Parang.
4. Penelitian ini menggunakan library `faster-rcnn` (<https://github.com/rbgirshick/py-faster-rcnn>) untuk proses Fast R-CNN dan NVIDIA cuDNN untuk implementasi deep learning pada perangkat mobile.
5. Penelitian ini menggunakan perangkat NVIDIA Tegra K-1.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Dari segi keilmuan, penelitian ini akan membantu penggunaan model Fast R-CNN untuk pengenalan objek gambar selain batik. Selain itu, penggunaan Fast R-CNN pada perangkat mobile dapat membantu kontribusi penelitian deep learning pada perangkat mobile.
2. Dari sisi sosial, penelitian ini dapat membantu memberikan informasi pengenalan motif batik secara umum pada masyarakat menggunakan smartphone.

1.6 Metodologi Penelitian

Untuk menjawab masalah yang terdapat pada rumusan masalah dan mencapai tujuan penelitian, penelitian ini dilakukan dengan metode eksperimen dengan langkah-langkah sebagai berikut:

1. Membangun layanan web service untuk membangun model deteksi batik daerah asal batik dengan convolution neural network
2. Membangun aplikasi berbasis android sebagai client yang mampu mengakses layanan web service dan mendapatkan output dari proses CNN

1.7 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN

Bab 1 berisi pendahuluan yang memberi penjelasan mengenai latar belakang penelitian ini dilakukan, masalah-masalah yang akan diselesaikan melalui penelitian ini, tujuan-tujuan penelitian yang akan dicapai, batasan penelitian, metode yang akan digunakan dalam penelitian ini, dan struktur penulisan proposal penelitian ini.

- Bab 2 LANDASAN TEORI

Bab 2 berisi teori-teori yang berkaitan dalam penelitian yang akan dilaksanakan. Teori-teori tersebut antara lain Convolutional Neural Network, Fast Region-CNN, Thin Client dan teknologi perangkat mobile.

- Bab 3 PERANCANGAN

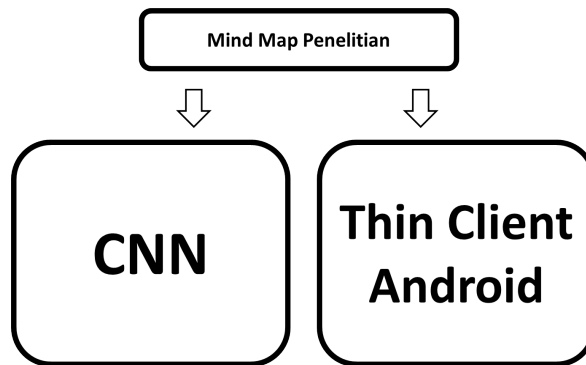
Bab 3 berisi usulan rancangan penelitian yang akan dilaksanakan. Rancangan tersebut terdiri dari langkah-langkah yang akan dilakukan, penjelasan teknis mengenai metode-metode yang akan diterapkan, dan tempat serta waktu penelitian.

BAB 2

LANDASAN TEORI

2.1 Penelitian Sebelumnya

Penelitian-penelitian yang berhubungan dengan penelitian ini adalah penelitian mengenai convolutional neural network dan thin client pada teknologi mobile. Penelitian-penelitian tersebut memiliki keterhubungan seperti yang dapat dilihat pada 2.1.



Gambar 2.1: Mind map penelitian terdahulu terkait CNN

Dalam penelitian [1] yang dilakukan Nicholas D. LanePetko Georgiev, membangun prototipe deep neural network pada lingkungan low-power yang melibatkan CPU dan DSP of a perangkat mobile SoC (System On Chip). Deep Neural Network digunakan melakukan deteksi aktivitas dan dilakukan perbandingan dengan teknik pembelajaran umum lainnya. Penelitian ini menemukan DNN mampu melakukan komputasi tanpa memberikan beban pada perangkat mobile dan mampu meningkatkan akurasi inferensi. Selain itu, ditemukan bahwa Deep Neural Network dapat dengan baik bekerja pada skala kelas inferensi yang lebih besar dan dipartisi secara fleksibel pada sumber daya perangkat mobile.

Dalam penelitian [2] yang dilakukan oleh Nicholas Lane, Petko Georgiev dan Lorena Qendro, menyajikan DeepEar yang merupakan framework perangkat mobile audio pertama dengan metode Deep Neural Network pada sensor audio secara simultan. Penelitian ini membuktikan bahwa DeepEar layak digunakan pada perangkat mobile yang dibangun dengan cloud yang berjalan secara kontinyu dan menghabiskan sumber daya 6% per hari.

Dalam penelitian [3] yang dilakukan oleh Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi dan Fahim Kawsar, meneliti perhitungan model deep learning (misalnya Convolutional Neural Network) pada perangkat mobile dan platform embedded. Penelitian ini bertujuan untuk mempelajari performa, kebutuhan sumber daya dan bottleneck pada perangkat mobile ketika melakukan proses deep learning. Hasil dari penelitian ini memberikan fondasi untuk melakukan optimasi pada metode deep learning agar bisa diintegrasikan dengan teknologi IoT, smartphone dan sistem wearable.

Dalam penelitian [4] yang dilakukan oleh Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, dan Fahim Kawsar, menyajikan desain dan implementasi DeepX, yang merupakan perangkat lunak akselerator untuk melakukan proses deep learning. DeepX menyediakan algoritma untuk melakukan kontrol sumber daya ketika melakukan proses deep learning dengan melakukan dekomposisi model jaringan kedalam beberapa unit. Arsitektur DeepX membuktikan proses deep learning dapat melakukan proses secara efisien pada perangkat prosesor mobile modern.

Dalam penelitian [5] yang dilakukan oleh Sourav Bhattacharya dan Nicholas D. Lane, meneliti metode Restricted Boltzmann Machines (RBM) untuk melakukan deteksi aktivitas pada perangkat smartwatch. Penelitian ini menyertakan variasi aktivitas seperti mode transportasi, aktivitas fisik dan deteksi dalam atau luar ruangan. Penelitian ini menunjukkan Restricted Boltzmann Machines (RBM) dapat menggunakan sumber daya dan memungkinkan penggunaan deep learning pada perangkat smartwatch.

Dalam penelitian [6] yang dilakukan oleh Ross Girshick, menciptakan metode deep learning Fast R-CNN yang merupakan penggabungan dari Region Convolutional Neural Network dan Spatial pyramid pooling networks (SPPnets). Penelitian ini bertujuan untuk memperbaiki kekurangan waktu proses R-CNN dan SPPnets pada saat training dan deteksi objek yang disebabkan multi-stage pipeline. Penelitian ini memberikan kontribusi perbaikan R-CNN dengan memberikan State-of-the-art mAP untuk VOC07, 2010, and 2012, waktu komputasi pembelajaran dan training yang lebih cepat dibandingkan R-CNN dan SPPnet dan fine-tune layer konvolusi.

Penelitian-penelitian terdahulu yang telah dijelaskan di atas dapat dituliskan dalam sebuah matriks literatur yang dapat dilihat pada 2.1 sebagai berikut.

Tabel 2.1: Matriks literatur penelitian terdahulu yang berhubungan dengan penelitian Convolution Neural Network

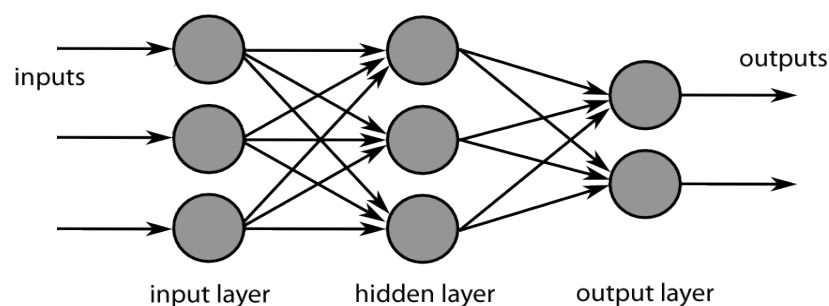
Judul	Can Deep Learning Revolutionize Mobile Sensing
Pengarang	Nicholas D. Lane & Petko Georgiev
Tahun	2015
Metode	Implementasi Deep Neural Network pada soc mobile dan DSP (Digital Signal Processor)
Kontribusi	Implementasi Deep Learning dengan memanfaatkan sensor mobile
Pekerjaan Mendatang	Inovasi sensor mobile dengan deep learning
Judul	DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning
Pengarang	Nicholas D. Lane, Petko Georgiev, Lorena Qendro
Tahun	2015
Metode	Implementasi Deep Neural Network untuk sensor audio pada mobile
Kontribusi	Implementasi deep learning untuk sensor audio pada mobile dan arsitektur yang memberikan efisiensi pada konsumsi power mobile
Pekerjaan Mendatang	Pengembangan sensor audio mobile untuk pengenalan aktivitas
Judul	An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices
Pengarang	Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Fahim Kawsar
Tahun	2015
Metode	Implementasi deep learning pada mobile dan platform embedded untuk meneliti performansi dan kebutuhan sumber daya mobile
Kontribusi	Metode fondasi metode deeplearning untuk teknologi IoT, smartphone and sistem wearable
Pekerjaan Mendatang	Pengembangan lanjut deeplearning pada IoT dan sistem wearable
Judul	DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices
Pengarang	Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro dan Fahim Kawsar
Tahun	2016
Metode	Dekomposisi model arsitektur deep learning menjadi beberapa blok unit dan pengaturan sumber daya model deep learning
Kontribusi	Optimasi sumber daya model deep learning untuk mobile
Pekerjaan Mendatang	Implementasi arsitektur DeepX pada mobile dengan variasi metode deep learning

Judul	Fast R-CNN
Pengarang	Ross Girshick
Tahun	2015
Metode	Integrasi dan optimasi Region-CNN dan Spatial pyramid pooling networks
Kontribusi	optimasi Region-CNN dan Spatial pyramid pooling networks
Pekerjaan Mendatang	Peningkatan deteksi objek dengan perbaikan metode sparse object

2.2 Feedforward Neural Network

Di dalam *machine learning*, *artificial neural network* (ANN) adalah model yang terinspirasi oleh *neural network* biologis. *Artificial neural network* dapat melakukan estimasi atau aproksimasi fungsi non-linear dengan nilai *output* yang dapat berupa nilai *real*, diskret, atau vektor. Pada alamnya, neuron menerima sinyal melalui sinaps yang terletak pada dendrit. Neuron akan teraktivasi dan dapat meneruskan sinyal apabila sinyal yang diterima memenuhi batas tertentu. Sinyal tersebut dapat diterima sinaps lain dan dapat mengaktivasi neuron lainnya. Beberapa permasalahan yang dapat diselesaikan dengan ANN diantaranya adalah klasifikasi, kategorisasi, aproksimasi fungsi, dan prediksi.

Feedforward neural network adalah salah satu jenis ANN. *Feedforward neural network* terdiri dari beberapa unit yang dikelompokkan dalam *layer* di mana setiap *layer* yang bersebelahan saling terhubung melalui suatu unit. Pada *feedforward neural network*, koneksi antar *unit* tidak membentuk *cycle* seperti yang ada pada *recurrent neural network*. Kalkulasi *feedforward neural network* dari *input* ke *output* berjalan ke satu arah sesuai dengan strukturnya yang tampak seperti graf berarah. Contoh *feedforward neural network* dengan satu *hidden layer* dapat dilihat pada Gambar 2.2 berikut.





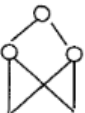
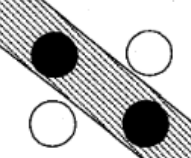

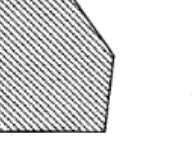

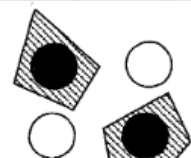




Gambar 2.2: Contoh *feedforward neural network*

Sumber gambar:

<http://technobium.com/stock-market-prediction-using-neuroph-neural-networks/>

Feedforward neural network terdiri dari *input layer*, *output layer*, dan beberapa *hidden layer* apabila dibutuhkan. Jumlah *hidden layer* berperan pada seberapa kompleks batas keputusan yang dapat dibentuk oleh *network* seperti pada Gambar 2.3.

Structure	Description of decision regions	Exclusive-OR problem	Classes with meshed regions	General region shapes
 Single layer	Half plane bounded by hyperplane			
 Two layer	Arbitrary (complexity limited by number of hidden units)			
 Three layer	Arbitrary (complexity limited by number of hidden units)			

Gambar 2.3: Interpretasi geometris pada peran *hidden layer*

Sumber gambar: [?]

Pada *input layer*, jumlah unit atau neuron yang ada sesuai dengan jumlah elemen yang ada pada vektor *input*. Begitu juga dengan *output layer*, jumlah unit yang ada sesuai dengan banyaknya elemen yang diinginkan pada vektor *output*. Berbeda dengan *input layer* dan *output layer*, jumlah unit dalam *hidden layer* sifatnya bebas. Setiap unit menghasilkan suatu *output* dari kombinasi linear beberapa nilai *input* dengan bobotnya masing-masing. Hasil dari kombinasi linear tersebut kemudian diaktivasi dengan fungsi aktivasi. Proses ini dapat dilihat pada Rumus 2.1.

$$y(x) = g \left(\sum_{i=1}^n w_{ij}x_i + w_{0j} \right) \quad (2.1)$$

- x merupakan vektor *input*
- w merupakan vektor bobot
- w_{ij} merupakan bobot yang menghubungkan unit ke- i pada *layer* sebelumnya dan unit ke- j pada *layer* saat itu
- w_{0j} merupakan bobot untuk bias

- g merupakan fungsi aktivasi

Pada Rumus 2.1 vektor *input* pada suatu *layer* bergantung pada *output* di *layer* sebelumnya, kecuali untuk *layer* pertama yaitu *input layer*. Fungsi aktivasi g dapat berbeda untuk setiap *layer*, tapi pada umumnya sama untuk setiap unit pada *layer* yang sama. Pada penelitian ini fungsi aktivasi yang digunakan adalah *rectified linear unit* (ReLU) yang dapat dilihat pada Rumus 2.2, dan *sigmoid* pada Rumus 2.3.

$$g(x) = \max(0, x) \quad (2.2)$$

$$g(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

Pada Rumus 2.1 terdapat w_{0j} yang merupakan bobot untuk bias. Bias bersifat seperti konstanta yang selalu bernilai +1, sehingga pengaruh bias tergantung pada bobotnya.

2.3 Convolution Neural Network (CNN)

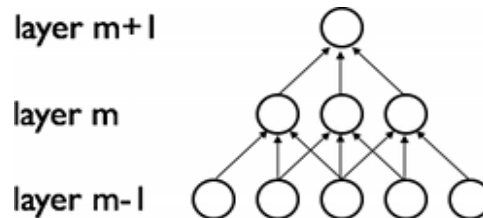
CNN merupakan salah satu variasi dari multilayer perceptron (MLP). Keuntungan dari metode CNN, khususnya untuk kasus pengenalan pola dibandingkan pendekatan konvensional adalah kemampuan untuk mengurangi dimensi data, ekstraksi fitur secara sekuensial, dan mengklasifikasi salah satu struktur jaringan. Arsitektur dasar dari model CNN terinspirasi dari visual cortex yang dikenalkan oleh hubel dan wiesel pada tahun 1962.

Pada tahun 1980, neocognitron fukushima membuat pertama kali komputasi menggunakan model CNN, dan kemudian pada tahun 1989, dengan menggunakan model yang ditemukan fukushima, LeCun menemukan performa dari beberapa proses untuk pengenalan pola menggunakan metode error gradient.

Model CNN yang digunakan oleh LeCun merupakan pengembangan dari MLP tradisional yang didasari 3 ide, local receptive field, weight sharing dan spatial subsampling. Ide dasar ini diorganisir kedalam 2 tipe layer, yaitu convolution dan subsampling layer. Seperti yang ditunjukkan gambar 1, digunakan 3 convolution layer dengan kombinasi layer subsampling dan 1 output layer. Layer convolution dan subsampling tersusun kedalam plane atau disebut feature maps.

Convolutional neural network (CNN) adalah salah satu tipe *feedforward neural network* yang terinspirasi dari cara kerja visual korteks. Berdasarkan penelitian, korteks visual terdiri dari banyak sel kompleks. Sel-sel ini disebut *receptive field* dan sensitif terhadap bagian kecil bidang visual. Salah satu sifat yang dimiliki CNN

adalah *sparse connectivity*. *Sparse connectivity* memungkinkan suatu *layer* saling terhubung hanya dengan unit-unit terdekat. Dengan kata lain, *input* dari suatu unit merupakan *subset* dari unit pada *layer* sebelumnya. Ilustrasi *sparse connectivity* dapat dilihat pada Gambar.

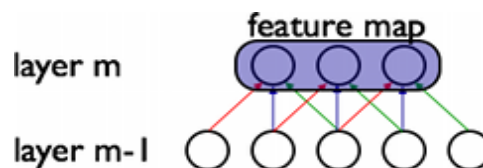


Gambar 2.4: *Sparse Connectivity*

Sumber gambar: <http://deeplearning.net/tutorial/lenet.html>

Tumpukan *layer* pada Gambar 2.4 hanya memiliki satu unit pada *layer m + 1*. Hal ini merepresentasikan unit atau bagian-bagian kecil yang ada pada *layer m – 1* bagaikan disaring atau di-filter hingga *layer m + 1* memahami bidang visual secara utuh.

Selain *sparse connectivity*, CNN juga memiliki sifat *shared weights*. Setiap *layer* kecuali *output layer* memiliki *filter* yang fungsinya untuk menyaring bagian kecil dari bidang visual. *Filter* tersebut direplikasi dengan bobot yang sama ke seluruh bidang visual dengan untuk menyaring unit atau bagian-bagian kecil sehingga membentuk *feature map*. Pada Gambar 2.5 berikut, lima unit pada *layer m – 1* disaring dengan *filter* berukuran tiga, sehingga membentuk tiga unit *feature map*. Garis yang berwarna sama menandakan bobot yang sama.



Gambar 2.5: *Shared weights* untuk *filter* yang sama

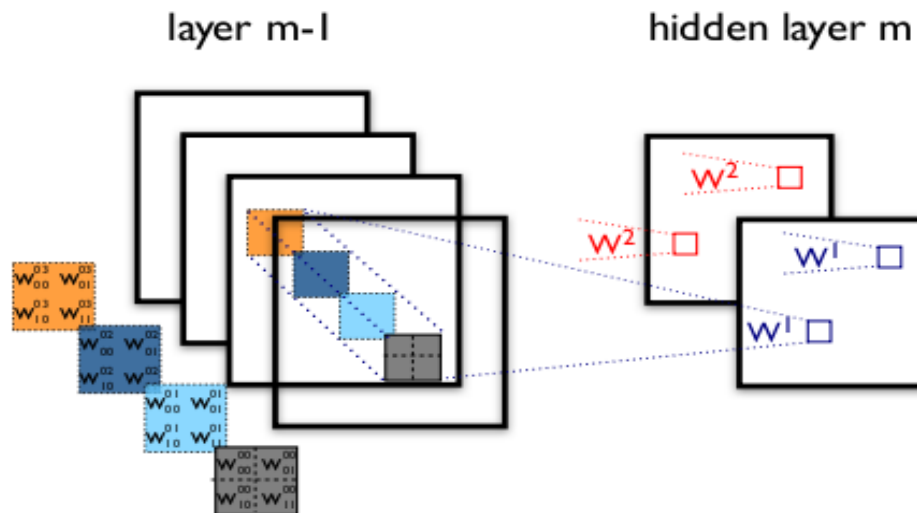
Sumber gambar: <http://deeplearning.net/tutorial/lenet.html>

Untuk merepresentasikan data atau bidang visual yang kompleks, setiap *layer* pada CNN dapat mempunyai beberapa *feature map* atau *channel*. Rumus untuk menghitung *feature map* ke- k pada *pixel* koordinat (i, j) yang ditentukan oleh bobot W^k dan bias b_k adalah sebagai berikut.

$$y_{ij}^k = g \left((W^k * x)_{ij} + b_k \right) \quad (2.4)$$

Setiap *feature map* pada *layer m – 1* berkontribusi dalam menentukan nilai *feature map k* pada *layer m*. Dengan kata lain, setiap *pixel* yang berada dalam *feature map*

$map\ k$ nilainya akan ditentukan oleh seluruh *pixel* yang terhubung, pada *feature map* di *layer* $m - 1$. Pada ilustrasi Gambar 2.6, *layer* m memiliki dua *feature map* y^0 dan y^1 . *Pixel* (kotak berwarna biru) pada y^0 ditentukan dari 2×2 *pixel* yang terhubung dengan bobot W_{ij}^{kl} . W_{ij}^{kl} menyatakan bobot yang menghubungkan *pixel* koordinat (i, j) pada *feature map* ke- l di *layer* $m - 1$ ke *pixel* yang terhubung pada *feature map* ke- k di *layer* m .



Gambar 2.6: Ilustrasi *layer* pada *convolutional neural network*

Sumber gambar: <http://deeplearning.net/tutorial/lenet.html>

Convolutional Neural Network biasanya terdiri dari beberapa *filter* atau disebut juga dengan *convolutional layer*, sering kali diikuti dengan *subsampling layer* dan diakhiri dengan *fully connected layers*. Terdapat juga jenis CNN yang tidak diakhiri dengan *fully connected layers* melainkan *convolutional layer* lainnya yang mempertahankan dimensi data dalam bentuk citra 2D. CNN jenis ini disebut dengan *Fully Convolutional Neural Network*.

2.3.1 Convolutional Layer

Pada *layer convolution*, tiap neuron terhubung secara local kepada area yang lebih kecil (local receptive field) pada *layer* sebelumnya. Semua neuron yang memiliki *feature maps* yang sama memperoleh data dari input area yang berbeda hingga semua input plan tersaring tetapi saling berbagi bobot (weight sharing).

Convolutional layer memiliki tiga *hyperparameter* yang mengatur transformasi *feature map* yang dihasilkan: *depth* D , *stride* S , dan *zero-padding* P . Dalam tahap *feed-forward*, *convolutional layer* berukuran F akan bergeser pada gambar berukuran W ,

dan menghasilkan *feature map* sebanyak D dengan ukuran W' yang dapat dihitung menggunakan Rumus 2.5 berikut.

$$W' = (W - F + 2P)/S + 1 \quad (2.5)$$

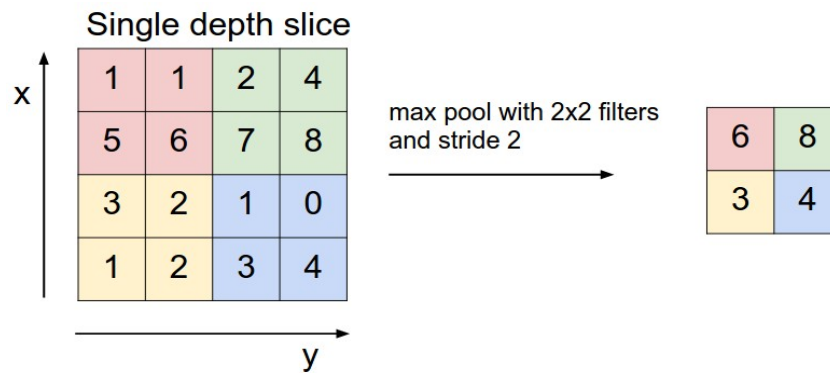
Dalam praktiknya, *zero-padding* sering kali digunakan untuk menyesuaikan ukuran hasil *feature map*. *Zero padding* dengan ukuran $P = (F - 1)/2$ dipastikan menghasilkan *feature map* dengan ukuran yang sama dengan *feature map* sebelumnya apabila *stride* $S = 1$. *Deep learning framework* seperti *theano* menyebut *padding* dengan sifat tersebut sebagai *same padding*. Terdapat juga *full padding* yang menghasilkan *feature map* dengan ukuran $W' = W + F - 1$.

2.3.2 Subsampling Layer

Pada layer subsampling, feature maps didownsampling secara spasial, dimana ukuran map dikurangi berdasarkan 2 faktor. Contohnya, feature map pada layer C3 dengan ukuran 10x10 dilakukan subsampling untuk menyesuaikan feature map dengan ukuran 5x5 pada layer selanjutnya. Layer terakhir adalah F6 yang merupakan proses klasifikasi. Dalam arsitektur *convolutional neural network*, *pooling layer* berfungsi untuk mereduksi ukuran spasial gambar secara bertahap sehingga mengurangi parameter dan komputasi pada *network*. *Pooling layer* menerima *input feature map* berukuran W_1 sebanyak D_1 dan parameter ukuran *pooling* F dan *stride* S . Sama halnya dengan *convolutional layer*, *pooling layer* juga bergeser terhadap input dan menghasilkan *feature map* berjumlah $D_2 = D_1$ dengan ukuran $W_2 = (W_1 - F)/S + 1$. Terdapat beberapa jenis *pooling layer*, dan yang paling sering digunakan adalah:

1. Max Pooling

Dalam *pooling* berukuran F , *Max pooling* akan mengambil hasil aktivasi yang terbesar dalam daerah yang termasuk pada pergeseran saat itu. Ilustrasi ini dapat dilihat pada Gambar 2.7 berikut dengan $F = 2$ dan $S = 2$.



Gambar 2.7: Ilustrasi *max pooling*

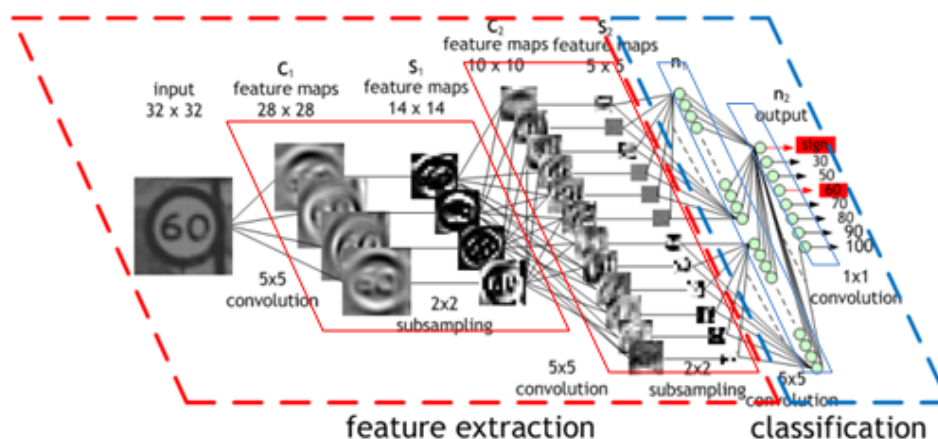
Sumber gambar: <http://cs231n.github.io/convolutional-networks>

2. Average Pooling

Berbeda dengan *max pooling*, *average pooling* mengambil hasil aktivasi pada daerah *filter* dan menghitung rata-rata dari seluruh hasil aktivasi dalam daerah tersebut.

2.3.3 Fully Connected Layer

Fully connected layer adalah *feed forward neural network* yang dijelaskan pada subbab 2.2. Biasanya, *fully connected layer* berada pada *layer* akhir pada arsitektur *convolutional neural network* dan bertujuan untuk memproses hasil *feature extraction* yang dilakukan pada *layer* sebelumnya menjadi suatu *output* tertentu. Gambar 2.8 berikut merupakan contoh arsitektur dengan susunan *convolutional layer*, *pooling layer*, dan *fully connected layer*.



Gambar 2.8: Contoh arsitektur *convolutional neural network*

Sumber gambar: <https://devblogs.nvidia.com>

2.4 Backpropagation

Tujuan dari melatih model *feedforward neural network* adalah membuat model tersebut dapat memprediksi nilai dengan galat seminimal mungkin. *Feedforward neural network* menggunakan metode *backpropagation* untuk melatih nilai bobot-bobot pada *network* tersebut. Untuk melatih suatu bobot, metode ini melakukan *propagation* dari galat pada *output layer* sehingga informasi ini tersebar ke *layer* sebelumnya untuk digunakan dalam mengubah bobot.

Untuk $\delta^{(l+1)}$ sebagai galat pada *layer* $l + 1$ dengan fungsi galat $J(W, b; x, y)$ di mana W, b adalah parameter, dan x, y adalah pasangan *input* dan label, apabila *layer* l dan *layer* $l + 1$ saling terhubung maka galat pada *layer* l adalah sebagai berikut.

$$\begin{aligned}\delta^{(l)} &= ((W^{(l)})^T \delta^{(l+1)}) \cdot f'(z^{(l)}) \\ f'(z^{(l)}) &= a^{(l)} \cdot (1 - a^{(l)})\end{aligned}\tag{2.6}$$

$$\begin{aligned}\Delta_{w^{(l)}} J(W, b; x, y) &= \delta^{(l+1)} (a^{(l)})^T \\ \Delta_{b^{(l)}} J(W, b; x, y) &= \delta^{(l+1)}\end{aligned}\tag{2.7}$$

Pada *convolutional layer* dan *pooling layer*, galat pada *layer* l dihitung dengan fungsi *upsample* $g(x)$ tergantung pada *layer* yang digunakan.

$$\delta_k^{(l)} = \text{upsample}((W_k^{(l)})^T \delta_k^{(l+1)}) \cdot f'(z_k^{(l)})\tag{2.8}$$

$$g(x) \begin{cases} \frac{\sum_{k=1}^m x_k}{m}, \frac{\delta g}{\delta x} = \frac{1}{m}, & \text{average pooling} \\ \max(x), \frac{\delta g}{\delta x} = \begin{cases} 1, & \text{if } x_i = \max(x) \\ 0, & \text{otherwise} \end{cases} & \text{max pooling} \end{cases}$$

Sumber rumus: <http://www.slideshare.net/kuwajima/cnnbp>

Rumus 2.7 digunakan untuk mengubah bobot pada suatu *layer*. Salah satu cara untuk mengubah bobotnya adalah dengan *stochastic gradient descent* yang menggunakan Rumus 2.9 berikut.

$$\theta = \theta - \alpha \Delta_{\theta} J(\theta; x^{(i)}, y^{(i)})\tag{2.9}$$

Sumber rumus: <http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork>

2.5 Algoritma Optimasi

Algoritma optimasi digunakan untuk permasalahan meminimalisasi *loss*. Untuk *dataset* D , objektif dari optimasi adalah rata-rata *loss* $|D|$ dari seluruh anggota *dataset* D . Dengan $f_w(X^{(i)})$ adalah galat pada data ke $X^{(i)}$, dan $r(W)$ adalah regularisasi dengan bobot λ , maka galat rata-rata dapat dihitung menggunakan Rumus 2.10 berikut .

$$L(W) = \frac{1}{|D|} \sum_i^{|D|} f_w(X^{(i)}) + \lambda r(W) \quad (2.10)$$

Karena $|D|$ pada kenyataannya dapat bernilai sangat besar, dalam praktiknya setiap iterasi menggunakan aproksimasi secara stokastik dengan memilih $N \ll |D|$ sehingga galat rata-rata dihitung dengan Rumus 2.11 berikut .

$$L(W) \approx \frac{1}{N} \sum_i^N f_w(X^{(i)}) + \lambda r(W) \quad (2.11)$$

2.5.1 Stochastic Gradient Descent (SGD)

Stochastic gradient descent mengubah bobot W dengan kombinasi linear dari *gradient* negatif $\Delta L(W)$ dan perbaharuan bobot V_t pada iterasi sebelumnya. Untuk menghitung nilai perbaharuan V_{t+1} dan bobot W_{t+1} pada iterasi $t + 1$ digunakan Rumus 2.12 dan 2.13 berikut .

$$V_{t+1} = \mu V_t - \alpha \Delta L(W_t) \quad (2.12)$$

$$W_{t+1} = W_t + V_{t+1} \quad (2.13)$$

2.5.2 Nesterov's Accelerated Gradient (Nesterov)

Nesterov's accelerated gradient (Nesterov) diusulkan oleh Nesterov sebagai metode optimal untuk optimasi, dengan laju konvergensi mencapai $O(1/t^2)$. Dalam praktiknya, Nesterov dapat menjadi metode yang cukup efektif untuk mengoptimasi arsitektur *deep learning* tertentu, seperti yang didemonstrasikan oleh dalam membuat *autoencoders*. Perubahan bobot dalam algoritma Nesterov menggunakan Rumus berikut .

$$V_{t+1} = \mu V_t - \alpha \Delta L(W_t + \mu V_t) \quad (2.14)$$

$$W_{t+1} = W_t + V_{t+1} \quad (2.15)$$

Perbedaan Nesterov dengan SGD terdapat pada penambahan momentum dalam menghitung *gradient* $\Delta L(W_t + \mu V_t)$. Dalam SGD, perhitungan *gradient* $\Delta L(W_t)$ hanya mengambil bobotnya saja.

2.6 Fungsi Galat

Fungsi galat adalah fungsi yang menghitung galat dari hasil prediksi \hat{y} dengan nilai y yang sebenarnya. Pada penelitian ini, penulis menggunakan dua jenis fungsi galat, yakni *cross-entropy loss* (Rumus 2.16) dan *euclidean loss* (L2) (Rumus 2.17). Dilihat dari fungsinya, *cross entropy* bekerja lebih agresif dalam memperbaiki dibandingkan dengan *euclidean error*.

$$crossentropy(y, \hat{y}) = -(y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})) \quad (2.16)$$

$$euclideanloss(y, \hat{y}) = \frac{1}{2} \|y - \hat{y}\|_2^2 \quad (2.17)$$

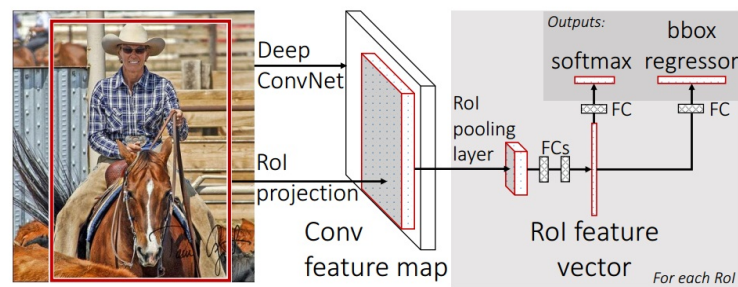
Sumber rumus: <http://caffe.berkeleyvision.org>

2.7 Fast R-CNN

Arsitektur Fast R-CNN 2.9 memiliki input objek gambar dan gambar yang sudah diberikan RoI (Object Proposal). Input akan diproses pertama kali kedalam beberapa layer konvolusi dan max pooling untuk mendapatkan feature map. Kemudian tiap objek proposal dengan Region of Interest (RoI) akan melakukan ekstraksi fitur dari feature map. Tiap vektor dari fitur akan diproses secara berurutan kedalam Fully Connected Layer dan dipecah ke dalam 2 output layer. Layer pertama menggunakan probabilitas softmax untuk melakukan estimasi pada K kelas objek dan background. Layer kedua memberikan 4 output dalam bentuk bilangan real untuk tiap K kelas objek. Untuk setiap 4 nilai dilakukan encoding posisi bounding-box pada salah satu posisi dari kelas K.

2.7.1 RoI pooling layer

RoI merupakan suatu area yang berbentuk segi empat dan berada didalam feature map konvolusi. Layer RoI pooling bertujuan melakukan proses max pooling dengan



Gambar 2.9: Arsitektur Fast R-CNN

tujuan konversi fitur didalam RoI kedalam feature map yang lebih kecil dengan ukuran yang sudah didefinisikan sebelumnya. RoI didefinisikan menggunakan 4 tuple (r, c, h, w) dimana (r, c) mendefinisikan *top-left corner* dan (h, w) mendefinisikan nilai tinggi dan lebar.

Proses max pooling pada ROI membagi area dengan $h \times w$ menjadi $H \times W$ sub-window dengan rumus $h/H \times w/W$. Nilai sub-windows tersebut akan memiliki korespondensi dengan output grid cell. Pooling dilakukan secara independen kepada tiap channel feature map yang merupakan standard dari max pooling.

2.7.2 Pre-trained networks

Pada [6], pra pembelajaran jaringan menggunakan 5 layer max pooling dan layer konvolusi dengan jumlah antara 5 hingga 13 layer dengan data training. Pra pembelajaran jaringan bertujuan memberikan inisialisasi Fast R-CNN dengan melakukan 3 transformasi:

1. Layer max pooling terakhir dirubah menjadi layer RoI pooling yang dikonfigurasi dengan H dan W agar sesuai dengan layer Fully Connected pertama.
2. Layer Fully Connected dan softmax terakhir dirubah menjadi 2 layer output
3. Network dimodifikasi kedalam 2 input data: input gambar dan RoI pada tiap input gambar tersebut.

2.7.3 Fine-tuning untuk Deteksi

Fast R-CNN memiliki metode training yang efisien dengan memanfaatkan sharing feature. Minibatch Stochastic Gradient Descent (SGD) dilakukan proses sampling secara hirarki, yaitu N gambar dan dilakukan secara iteratif R/N dari tiap gambar. RoI dari gambar yang sama akan menggunakan komputasi dan memori pada proses forward dan backward. Contohnya, ketika $N=2$ dan $R=128$, skema training

akan meningkat 64x kali lebih cepat dibandingkan menggunakan 1 RoI dengan 128 gambar. Salah satu permasalahan pada skema training ini adalah proses training akan berjalan lambat karena RoI dari gambar yang sama akan saling berkorelasi. Berdasarkan [6], masalah ini tidak menjadi isu pada saat pembelajaran dan tetap memberikan hasil yang baik karena jumlah iterasi SGD yang lebih sedikit dibandingkan R-CNN biasa.

Sebagai tambahan pada proses sampling yang dilakukan secara hirarki, Fast R-CNN menggunakan proses training yang lebih singkat dengan fase *fine-tuning* yang menggabungkan softmax dan bounding-box regressor, dibandingkan R-CNN biasa yang melakukan training dengan softmax, SVM dan regressor pada fase yang berbeda. Detail komponen *fine-tuning* sbb:

- Multi-task loss

Fast R-CNN memiliki 2 output layer: layer pertama memberikan output distribusi probabilitas diskret, $p = (p_0, \dots, p_K)$, per $K + 1$ kategori. Kemudian p dihitung menggunakan softmax pada output $K + 1$ dari layer Fully Connected. Layer kedua memberikan output bounding-box regression offset, $tk = t_x^k, t_y^k, t_w^k, t_h^k$, untuk tiap K kelas objek, menggunakan index k . tk didefinisikan menggunakan translasi scale-invariant dan log-space pergeseran tinggi atau lebar menyesuaikan proposal objek.

Setiap melakukan training RoI akan diberi label kelas u dan target bounding-box regression v . Fast R-CNN menggunakan multi-task loss L pada tiap RoI yang sudah memiliki label untuk dilatih proses klasifikasi dan regression bounding-box

$$L(p, u, t^u, v) = L_{cls}(p, u) + [u1]L_{loc}(t^u, v) \quad (2.18)$$

dimana $L_{cls}(p, u) = -\log p_u$ merupakan log loss dari kelas u true. L_{loc} didefinisikan pada tuple dari true target bounding-box regression pada kelas $u, v = (v_x, v_y, v_w, v_h)$, dan tuple $tu = (t_x^u, t_y^u, t_w^u, t_h^u)$ untuk kelas u . Indikator kurung iverson $[u1]$ dievaluasi menjadi 1 ketika $u1$ dan 0 sebagainya. Dengan syarat pengambilan semua kelas background diberikan label 0. Untuk background RoI tidak ada gagasan sehingga L_{loc} diabaikan. Untuk bounding-box regression, digunakan fungsi loss

$$L_{loc}(t^u, v) = \sum_{ix,y,w,h} \text{smooth}_{L_1}(t_i^u v_i) \quad (2.19)$$

dimana

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{jika } |x| \leq 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (2.20)$$

adalah loss L_1 yang tidak terlalu sensitif pada outlier dibandingkan loss L_2 yang digunakan R-CNN dan SPPnet. Ketika target regresi tidak memiliki batas, training L_2 akan membutuhkan perbaikan pada learning rate dengan tujuan mencegah perubahan gradien. Rumus 2.20 menghilangkan sensitifitas ini.

- Mini-batch sampling

Selama melakukan fine-tuning, tiap mini-batch SGD dibangun dengan gambar $N = 2$, yang dipilih secara acak. Ukuran mini-batch yang digunakan $R = 128$, dengan melakukan sampling 64 RoI dari tiap gambar. RoI yang diambil hanya 25% dari objek proposal ketika terjadi saling tumpang dengan batas 0.5 pada bounding box. RoI mengandung contoh yang diberikan label dengan kelas object foreground yaitu $u \geq 1$. RoI sisanya akan dilakukan sampel dari objek proposal yang memiliki maksimum IoU dengan interval $[0.1, 0.5]$.

- Back-propagation through RoI pooling layers

Backpropagation mengarah pada penurunan pada layer pooling RoI. Untuk memperjelas, diasumsikan hanya 1 gambar per batch ($N = 1$), perluasan pada $N > 1$ akan terang-terangan karena proses forward memperlakukan gambar secara independen.

$x_i \in R$ menjadi input aktivasi i -th kedalam layer RoI pooling dan y_{rj} menjadi bagian j -th memberikan output r -th RoI. Layer RoI pooling menghitung $y_{rj} = x_{i^*(r,j)}$ dimana $i^*(r,j) = \operatorname{argmax}_{i' \in R(r,j)} x_{i'}$. $R(r,j)$ adalah kumpulan indeks dari input dalam sub-window dimana pada unit output y_{rj} max pool. Satu x_i bisa diberikan kepada beberapa output y_{rj} .

Proses backward Layer RoI pooling menghitung turunan parsial dari loss function dengan untuk tiap variabel input x_i sbb:

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}} \quad (2.21)$$

Dengan kata lain, untuk setiap mini-batch RoI r dan untuk setiap unit output pooling y_{rj} , dilakukan akumulasi penurunan parsial $\partial L / \partial y_{rj}$ jika i adalah

argmax yang terpilih untuk y_{rj} dengan max pooling. pada backpropagation, penurunan parsial $\partial L / \partial y_{rj}$ sudah dihitung dengan fungsi backward dari layer teratas dari layer RoI pooling.

- SGD hyper-parameters

Layer Fully Connected yang digunakan untuk klasifikasi softmax dan regression bounding-box merupakan hasil inisialisasi dari distribusi zero-mean gaussian dengan standard deviasi 0.01 dan 0.01. Nilai bias diinisialisasi menjadi 0. Semua layer menggunakan learning rate bobot 1 dan 2 untuk bias dan learning rate global sejumlah 0.001. Momentum 0.9 dan parameter decay digunakan.

2.7.4 Scale invariance

Fast R-CNN mengeksplorasi 2 cara untuk memperoleh invarian ska deteksi objek: brute force dan menggunakan image piramid. Pendekatan Brute Forse, tiap gambar diproses dengan ukuran pixel yang belum didefinisikan selama proses training maupun testing. Network harus mempelajari langsung deteksi objek scale-invariant dari data training. Pendekatan Multi-scale, berbanding terbalik, menyediakan pendekatan scala-invariance pada network pada image piramid. pada waktu tes, image piramid digunakan scale-normalize dari tiap objek proposal. selama multi-scale training, dipili secara acak skala piramida ketika image dijadikan sampel.

2.8 Thin Client Computing

Berdasarkan [15], Thin client merupakan istilah kepada komputer yang bersifat ringan dengan tujuan melakukan akses kepada server yang bersifat cloud maupun virtual. Kemunculan thin client disebabkan penggunaan komputer desktop yang memberikan beban dalam proses manajemen maupun maintenance. Terminal services atau remote desktop services adalah salah satu software yang menyediakan fasilitas komunikasi dengan thin client, dimana biasanya pada suatu perusahaan akan membeli mesin thin client, melakukan instalasi microsoft windows server, kemudian dilakukan konfigurasi peraturan akses antara terminal service dan client dengan memberikan session untuk tiap komunikasi. Secara umum, penggunaan terminal service adalah salah satu solusi terbaik dan cocok untuk penggunaan aplikasi berskala kecil maupun penggunaan internal, harga terjangkau dan kemudahan proses instalasi dan maintenance. Selain terminal services, teknologi yang mampu mengadopsi thin client adalah CITRIX XENAPP.

Thin client berbasis desktop memiliki kekurangan yaitu dibutuhkan mobilitas dari pengguna, pembaharuan pada hardware client, keamanan data dan beberapa kondisi tertentu yang perlu diperhatikan seperti konsumsi listrik dan pendinginan. Selain itu, perkembangan thin client saat ini mulai mengarah ada embedded desktop systems, menggunakan RAM dengan kapasitas tinggi, mendukung multi monitor, bahkan menggunakan teknologi smartphone yang saat ini perkembangannya sangat pesat dengan didukung komposisi hardware yang semakin mumpuni.

2.9 Teknologi Perangkat Mobile

2.9.1 Sensor pada mobile

Meskipun sensor pada perangkat mobile memiliki variasi dalam beberapa bentuk dan target untuk penggunaan secara umum, penggabungan elemen antara sensor adalah mereka semua melibatkan kumpulan dan interpretasi dari data. Untuk menyelesaikan hal ini, mobile sensor akan ditanamkan algoritma deep learning kedalam aplikasi mereka. DeepX bertujuan untuk digunakan sebagai black-box oleh developer dari aplikasi mobile dan menyediakan pengganti media eksekusi inferensi untuk model deep learning yang diadopsi. Kunci permasalahan ini adalah frekuensi sensor data yang akan digunakan dan dilakukan proses. Aplikasi sensor yang secara kontinyu menginterpretasi data memberikan tantangan karena memungkinkan untuk melakukan inferensi beberapa kali dalam 1 menit. Oleh karena itu penggunaan sumber daya tiap 1 inferensi harus rendah jika aplikasi harus memiliki waktu hidup baterai yang baik. Sensor yang melakukan proses tidak terlalu kontinyu bisa memiliki beban sumber daya yang lebih besar. Bagaimanapun, model deep learning pada perangkat mobile perlu dilakukan optimasi sumber daya sebelum dieksekusi pada perangkat mobile. Banyak model deep learning yang memiliki kebutuhan sumber daya yang terlalu besar pada perangkat mobile SoC. Waktu eksekusi bisa melebihi batas kemampuan sumber daya perangkat mobile dan memberikan masalah ketika inferensi aktif secara bersamaan yang dilakukan oleh user pada satu hari.

2.9.2 Mobile SoC (System On Chip)

Perangkat Soc pada saat ini telah berevolusi dan meningkat ke dalam beberapa area yang lebih luas pada unit komputasional (GPUs, low-power CPU cores, multi-core CPUs). Meskipun LG G Watch R yang berteknologi android memiliki snapdragon 400 yang mengandung pasangan DSP dan CPU Dual-core. Tiap prosesor mem-

berikan profile dari sumber dayanya masing-masing ketika menunjukkan beberapa tipe komputasi. Hal ini memberikan timbal balik pada saat melakukan eksekusi arsitektur deep learning, tergantung pada jumlah layer maupun parameter lainnya. Perbedaan ini merupakan permasalahan yang umum untuk perangkat mobile dan dilakukan pendekatan partisi layer-wise ditambah menyelesaikan rumus optimisasi (rumus) untuk menentukan bagaimana heterogen ini harus memberikan hasil terbaik pada berbagai kondisi.

2.10 Batik

Indonesia merupakan negara kepulauan yang memiliki banyak suku dan ras dan sehingga memiliki keanekaragaman bahasa, budaya dan kebiasaan. Salah satu keunikan yang dimiliki Indonesia adalah memiliki keanekaragaman jenis kain yang berasal dari tiap daerah diantaranya batik, tok wie, kain dodot, kain panjang, dll.



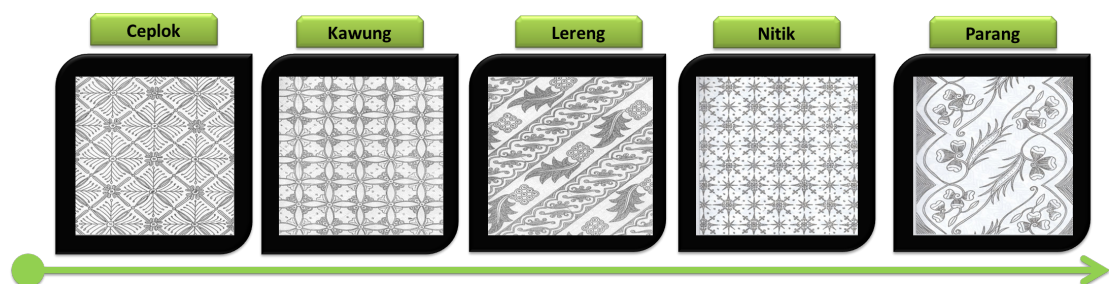
Gambar 2.10: Contoh kain dari berbagai daerah di Indonesia

Salah satu kain tradisional Indonesia adalah batik. Batik merupakan kain bergambar yang memiliki gaya, warna dan tekstur dimana proses pembuatannya dilakukan secara manual maupun menggunakan mesin. Batik memiliki beberapa motif seperti motif kawung, motif parangkusumo, motif truntum, motif tambal, motif pamiluto, motif parang, motif liris maupun motif udan nitik seperti pada gambar 2.11.

Tiap motif batik memiliki makna tersendiri berdasarkan daerah asal batik tersebut. Beberapa makna motif batik (sumber : <http://batik-tulis.com/blog/batik-yogyakarta>):

- Motif ceplok dengan model grompol melambangkan harapan orang tua akan semua hal yang baik terkumpul, yaitu rejeki, kerukunan hidup, kebahagiaan, dan ketentraman untuk kedua mempelai dan keluarga pengantin.
- Motif kawung jogja melambangkan segala sesuatu yang bersifat murni, suci, dari putih kembali ke putih.

- Motif batik lereng dari yogyakarta melambangkan kesuburan, harapan untuk kemakmuran, tekad, untuk memiliki keberanian untuk melaksanakan apa yang penting bagi bangsa dan rakyat.
- Motif nitik dengan "motif cakar ayam biasa dikenakan pada acara perkawinan dengan tujuan agar pasangan yang menikah dapat mencari dengan halal sependai ayam mencari makan dengan cakarnya".
- Motif parang menjadi "pedoman utama untuk menentukan derajat kebangsawanan seseorang dan menjadi pedoman yang termaktub dalam pranatan dalem asmanipun panganggo keprabon wonten kraton nagari ngayogyakarta hadiningrat pada tahun 1927".



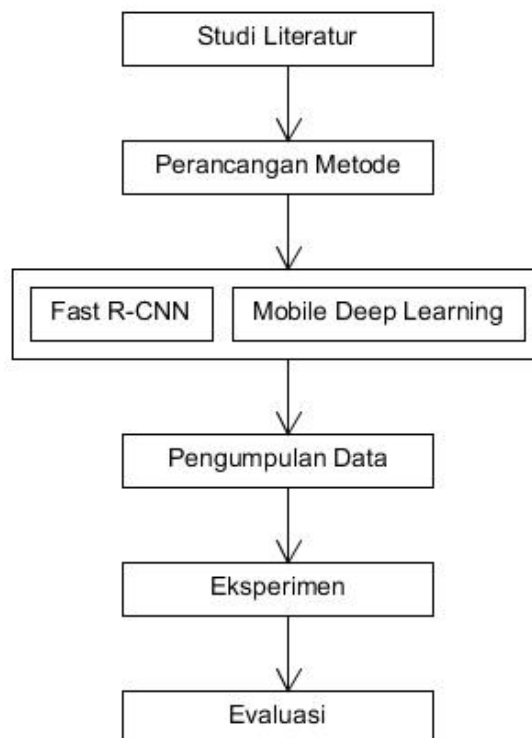
Gambar 2.11: Contoh motif batik indonesia

BAB 3

PERANCANGAN

3.1 Tahapan Penelitian

Secara garis besar, tahapan penelitian yang hendak dilaksanakan dalam penelitian ini terdiri dari beberapa tahapan utama yaitu studi literatur, perancangan metode, pengumpulan data, implementasi program, evaluasi dan analisis hasil. Bagan tahapan penelitian dapat dilihat pada 3.1 berikut:



Gambar 3.1: Metodologi penelitian

Adapun penjelasan jenis kegiatan yang hendak dilaksanakan adalah sebagai berikut:

1. Studi literatur

Pada tahap ini meliputi pengumpulan berbagai informasi tentang dasar teori serta penelitian-penelitian sebelumnya yang berkaitan dengan penelitian yang akan dilakukan. Tahapan ini bertujuan untuk mengetahui the state of the

art dari penelitian yang berkaitan dengan pengenalan convolutional neural network dan deep learning pada teknologi mobile. Hasil kajian tersebut dijadikan sebagai acuan untuk mencari suatu pembaruan terhadap metode yang telah ada.

2. Perancangan Metode

Pada perancangan metode, ada 2 kegiatan umum yang akan dilakukan yaitu perancangan metode Fast R-CNN untuk melakukan deteksi batik dan metode Mobile Deep Learning untuk melakukan implementasi deep learning pada SoC mobile.

3. Pengumpulan Data

Pada penelitian menggunakan data batik yang berasal dari penelitian sebelumnya [12] dan terdiri dari 5 motif:

- Ceplok
- Kawung
- Lereng
- Nitik
- Parang

4. Eksperimen

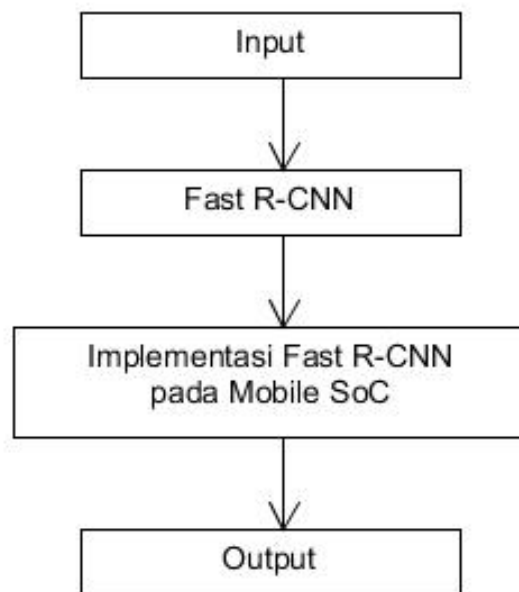
Eksperimen berdasarkan rancangan metode yang telah dibuat akan dilakukan menggunakan library Fast R-CNN pada <https://github.com/rbgirshick/py-faster-rcnn> dan NVIDIA Deep Neural Network library (cuDNN) untuk implementasi pada platform mobile SoC.

5. Evaluasi dan Analisis Hasil

Pengujian hasil eksperimen meliputi pengujian performa dan akurasi. Selain itu, dalam penelitian ini juga akan dilakukan analisis terhadap waktu komputasi yang dibutuhkan oleh sistem dalam melakukan proses prapengolahan dokumen formulir, segmentasi tulisan dan pengenalan karakter.

3.2 Rencana Implementasi

Secara umum, rencana eksperimen yang akan dilakukan dalam penelitian ini dapat dilihat pada 3.2 berikut:



Gambar 3.2: Rencana Implementasi

Berdasarkan gambar 3.2, terdapat dua eksperimen utama yang akan dilakukan pada penelitian ini. Eksperimen pertama fokus kepada Fast R-CNN untuk pengenalan batik sedangkan pada eksperimen kedua fokus kepada implementasi Fast R-CNN pada perangkat mobile. Penjelasan untuk masing-masing proses adalah sebagai berikut:

1. Tahap pertama

Tahap pertama akan melakukan implementasi Fast R-CNN untuk mendeteksi batik sesuai dengan arsitektur pada gambar 2.9. Arsitektur Fast R-CNN 2.9 memiliki input gambar secara keseluruhan dan kumpulan objek tertentu. Input akan diproses pertama kali kedalam beberapa layer konvolusi dan max pooling untuk mendapatkan feature map. Kemudian tiap objek proposal atau Region of Interest (RoI) akan melakukan ekstraksi fitur dari feature map. Tiap vektor fitur akan diproses secara berurutan kedalam Fully Connected Layer dan membagi ke dalam 2 output layer. Layer pertama menggunakan probabilitas softmax melakukan estimasi pada K kelas object dan kelas background. Layer kedua memberikan 4 output dalam bentuk bilangan real untuk tiap K kelas objek. Untuk setiap 4 nilai dilakukan encoding posisi bounding-box pada salah satu posisi dari kelas K.

2. Tahap Kedua

Tahap kedua akan berfokus pada implementasi Fast R-CNN pada perangkat

mobile NVIDIA Tegra K-1 berdasarkan penelitian [4]. Implementasi pada perangkat mobile akan mengoptimasi penggunaan sumber daya maupun waktu eksekusi dengan 2 pendekatan:

(a) Runtime Layer Compression (RLC)

RLC memiliki 2 komponen utama, komponen pertama melakukan pengurangan dimensi untuk merendahkan kebutuhan komputasi tiap layer. komponen kedua berfungsi untuk mengatur level pengerungan dimensi yang akan diaplikasikan sebelum model akurasi dipengaruhi penggunaan DeepX. Input kepada RLC adalah

- i. sepasang adjacent layer dari model untuk dieksekusi
- ii. Batas error yang digunakan oleh yang menggambarkan observasi dari rekonstruksi error setelah pengurangan dimensi dilakukan

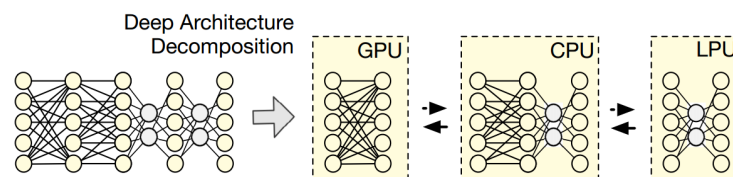
2 input tersebut disediakan oleh DAD yang juga menerima output dari RLS. secara spesifik, penggantian matriks bobot antara layer L dan L+1 yang membutuhkan parameter dan komputasi yang lebih sedikit.

(b) Deep Architecture Decomposition (DAD)

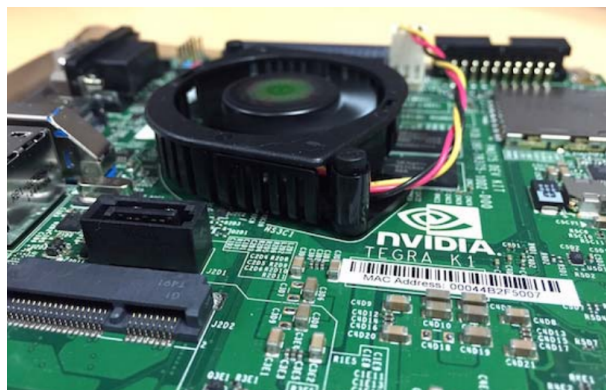
DAD memiliki 2 komponen diantaranya Decomposition Search dan Recomposition Inference. Komponen pertama bertujuan untuk memperhitungkan proses dekomposisi dari arsitektur deep secara efisien, kemudian dihitung performa estimasi sesuai dengan tujuan dari pengguna. RLS memperluas area pencarian DAD hingga kompresi layer dari beberapa rencana. komponen kedua melakukan inferensi hingga hasil model klasifikasi dengan melakukan rekomposisi hasil dekomposisi dari unit-block yang dialokasikan pada unit komputasi terpisah. input DAD terdiri dari:

- i. model deep yang akan dieksekusi
- ii. kumpulan tujuan performansi

Output dari DAD adalah inferensi dari model input.



Gambar 3.3: Deep Architecture Decomposition



Gambar 3.4: NVIDIA Tegra K1

DAFTAR REFERENSI

- [1] Nicholas D. Lane, Petko Georgiev. Can Deep Learning Revolutionize Mobile Sensing?.
- [2] Nicholas D. Lane, Petko Georgiev, Lorena Qendro. DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning.
- [3] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Fahim Kawsar. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices.
- [4] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices.
- [5] Sourav Bhattacharya and Nicholas D. Lane. From Smart to Deep: Robust Activity Recognition on Smartwatches using Deep Learning. The Second IEEE International Workshop on Sensing Systems and Applications Using Wrist Worn Smart Devices, 2016
- [6] Ross Girshick. Fast R-CNN. 2015 IEEE International Conference on Computer Vision
- [7] Yizhang Xia, Bailing Zhang, Frans Coenen. Face Occlusion Detection Based on Multi-task Convolution Neural Network. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)
- [8] Earnest Paul Ijjina, C Krishna Mohan. Human Action Recognition based on Motion Capture Information using Fuzzy Convolution Neural Networks.
- [9] Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen. Human Hand Gesture Recognition Using a Convolution Neural Network. 2014 IEEE International Conference on Automation Science and Engineering (CASE) Taipei, Taiwan, August 18-22, 2014
- [10] Nanik Suciati, Agri Kridanto, Mohammad Farid Naufal, Muhammad Machmud & Ardian Yusuf Wicaksono. Fast Discrete Curvelet Transform And HSV

Color Features For Batik Image Classification. 2015 International Conference on Information, Communication Technology and System (ICTS)

- [11] Xinyan Yu¹, Guohao Lyu, Siwei Luo, Junbo Liu. A Convolution Neural Network Based Variational Restoration Model
- [12] L. M. Rasdi Rere, Mohamad Ivan Fanany, Aniati Murni Arymurthy. Meta-heuristic Algorithms for Convolution Neural Network. Hindawi Publishing Corporation Computational Intelligence and Neuroscience
- [13] yejun Tang, Liangrui Peng, Qian Xu, Yanwei Wang dan Akio Furuhashi. CNN based Transfer Learning for Historical Chinese Character Recognition. 2016 12th IAPR Workshop on Document Analysis Systems
- [14] Giuseppe Attardi. DeepNL: a Deep Learning NLP pipeline. Proceedings of NAACL-HLT 2015, Association for Computational Linguistics
- [15] Thin Client Computing. 2010. Info World Deep Dive
- [16] Machine Learning in the Cloud. 2010. Info World Deep Dive
- [17] Ynn LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. PROCEEDINGS OF THE IEEE, VOL. 86, NO. 11, NOVEMBER 1998.