Deteksi Rumus Matematika pada Halaman Dokumen Digital dengan Metode Convolutional Neural Network

Martina Marcelline Taslim, Kartika Gunadi[2], Alvin Nathaniel Tjondrowiguno[3]
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
Email: martinamarcelline@gmail.com[1], kgunadi@petra.ac.id[2], alvin.nathaniel@petra.ac.id[3]

ABSTRAK

Rumus matematika dalam *scientific paper* yang tersedia secara daring seringkali tidak bisa dikenali dengan baik oleh proses *Optical Character Recognition* (OCR). Sulitnya pengenalan ini dikarenakan adanya perbedaan-perbedaan yang dimiliki rumus matematika dibandingkan dengan baris teks biasa. Oleh karena itu, deteksi rumus matematika pada halaman dokumen digital diharapkan dapat membantu proses OCR.

Deteksi rumus matematika pada halaman dokumen digital dilakukan dengan mengubah dokumen menjadi *image,* kemudian memisahkan setiap baris teks dan kata-kata yang ada di dalamnya dan mengklasifikasi baris teks dan kata-kata tersebut menggunakan *Convolutional Neural Network* (CNN). Arsitektur CNN yang digunakan untuk melakukan klasifikasi memiliki 64 *filter* di setiap *convolutional layer*-nya.
Untuk rumus *displayed* (tidak bercampur dengan teks), digunakan 10 kumpulan *layer Convolutional*-ReLU-*Max Pooling*, sedangkan untuk rumus *inline* (bercampur dengan teks) digunakan 12 kumpulan *layer*. Dengan menggunakan arsitektur-arsitektur CNN yang telah disebutkan, didapat F1 *score* sebesar 0,980 untuk klasifikasi rumus *displayed* pada dokumen dengan *layout* 1 kolom; F1 *score* sebesar 0,940 untuk klasifikasi rumus *displayed* pada dokumen dengan 2 kolom; dan F1 *score* sebesar 0,916 untuk rumus *inline*.

Kata Kunci: *Machine Learning*, *Artificial Neural Network*, *Convolutional Neural Network, Image*

ABSTRACT

Mathematical formulae in academic papers or scientific journals are an important part of said documents. However, mathematical formulae are oftentimes not properly recognized by Optical Character Recognition (OCR) processes. One of the causes of this failure is the difference between mathematical formulae and ordinary text. Therefore, mathematical formula detection in those document pages might help with this problem.

The formula detection is done by converting digital document pages into images, then performing text line segmentation and word segmentation and classifying those results with a Convolutional Neural Network. The aim is to help OCR processes by recognizing which parts of the document pages contain formulae and which parts do not. The CNN architectures used to perform classification comes with 64 kernels in each convolutional layer.

For displayed formulae (formulae that doesn't share its space with regular text), the model uses 10 groups of Convolutional-ReLU-Max Pooling layers. For inline formulae (formulae that shares its text line with regular text), 12 groups of Convolutional-ReLU-Max Pooling layers are used. Results of the CNN architectures mentioned above are an F1 score of 0,980 for displayed formulae classification in 1-column documents, 0,940 for 2-column documents, and 0,916 for inline formulae.

Keywords: *Machine Learning, Artificial Neural Network, Convolutional Neural Network, Image*

1. PENDAHULUAN

Menurut riset dari Pew Research Center yang diadakan tahun 2017-2018, median jumlah pengguna internet di 39 negara dunia (termasuk Indonesia) adalah 75% dari populasi negara tersebut [8]. Hal ini mengindikasikan jangkauan internet di dunia yang semakin meluas dibandingkan tahun-tahun sebelumnya. Dengan meluasnya jangkauan internet di dunia, semakin banyak sumber daya ilmu pengetahuan yang dapat diakses masyarakat umum, di antaranya *academic paper* dan jurnal ilmiah. *Academic paper* dan jurnal ilmiah adalah sumber referensi yang berharga bagi peneliti dan pelajar yang ingin membuat penelitian baru atau mengembangkan penelitian sebelumnya yang sudah ada. Mayoritas *academic paper* dan jurnal ilmiah tersedia secara daring dalam bentuk dokumen PDF yang menjamin penampilan yang konsisten di semua *device*.

Pada *academic paper* atau jurnal ilmiah, seringkali rumus matematika yang ada di dalamnya merupakan salah satu bagian terpenting dari isi dokumen. Tetapi, sebagai komponen yang begitu penting dari dokumen, rumus matematika seringkali tidak bisa dikenali dengan baik oleh proses *Optical Character Recognition* atau OCR [8]. Padahal, proses OCR pada rumus matematika bermanfaat untuk mengubah rumus matematika dalam dokumen menjadi *format markup language*, misalnya TeX dan MathML atau Unicode, sehingga dapat digunakan ulang atau dapat digunakan untuk pencarian dalam *search engine* [4]. Sulitnya mengenali rumus matematika dalam proses OCR ini disebabkan perbedaan-perbedaan yang dimiliki rumus matematika dibandingkan dengan baris teks biasa. Misalnya, sebuah rumus dapat mencakup 2 baris atau lebih tetapi tetap menjadi satu kesatuan. Oleh karena itu, rumus matematika perlu dideteksi terlebih dahulu sebelum dilakukan proses OCR, supaya dapat dipisahkan dari elemen-elemen lain pada dokumen, misalnya teks, tabel, dan gambar.

Terkait masalah ini, sudah ada beberapa penelitian yang mencoba deteksi rumus matematika pada dokumen dengan *Support Vector Machine* (SVM) [4] serta gabungan *rule-based method* dan SVM [7]. Tetapi, menurut penelitian mengenai *image classification* yang membandingkan metode SVM dengan *Convolutional Neural Network* (CNN), akurasi metode CNN lebih baik daripada SVM [11]. Penelitian sejenis mengenai segmentasi dokumen kuno yang membandingkan CNN dengan *neural network* lain yang tidak dirancang khusus untuk menangani gambar seperti *Multilayer Perceptron* (MLP) juga mengindikasikan bahwa CNN menghasilkan akurasi yang lebih baik daripada MLP [3]. Sejauh

ini, belum ditemui penelitian mengenai deteksi rumus matematika pada dokumen dengan menggunakan CNN. Padahal, CNN adalah *artificial neural network* yang sangat cocok untuk menangani gambar [1] dan terbukti pada beberapa penelitian sebelumnya bahwa CNN menghasilkan akurasi lebih baik daripada metode SVM.

2. TINJAUAN PUSTAKA

2.1 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah *artificial neural network* yang digunakan dalam bidang *computer vision* untuk klasifikasi gambar dan pengenalan objek. Arsitektur CNN biasanya terdiri dari *convolutional layer, Rectifier Linear Unit* (ReLU) *layer, pooling layer*, dan *fully connected layer* [1]. *Input* untuk *convolutional layer* adalah gambar berukuran $m \times m \times r$, di mana m adalah panjang serta lebar gambar dan r adalah jumlah *channel* gambar (misalnya gambar RGB memiliki 3 *channel*).

Convolutional layer memiliki *kernel* atau *filter* sejumlah k. *Filter* ini berukuran $n \times n \times q$, dengan n lebih kecil daripada dimensi gambar yang menjadi *input* dan q lebih kecil atau sama dengan r yaitu banyaknya *channel* pada gambar *input* tersebut [1]. Dari *filter* tersebut, dilakukan konvolusi (*convolution*) untuk menghasilkan *feature map* sebanyak k dengan ukuran m-n+1. *Feature map* ini berisi fitur-fitur menonjol dari gambar yang menjadi *input* dari *convolutional layer*.

Hasil konvolusi ini kemudian akan dinormalisasi dengan *Rectifier Linear Unit* (ReLU) untuk menghilangkan nilai-nilai negatif dan mengubahnya menjadi 0. Pada *pooling layer*, setiap *feature map* di-*subsample* dengan mengambil nilai terbesar dalam *window* berukuran $p \times p$ (*max pooling*) atau mengambil nilai *mean* (*mean pooling*). *Pooling* dapat dilakukan dengan berbagai *window size* dan *stride* sesuai ukuran gambar [9]. Untuk setiap *feature map* yang didapat dari tahapan *convolution*, dilakukan langkah-langkah *pooling* yaitu menentukan *window size* (biasanya 2 atau 3), menentukan *stride* (jarak gerak) *window* (biasanya 2), mengeser *window* melewati gambar yang sudah di-*filter,* dan terakhir mengambil nilai terbesar atau *mean* dari setiap *window* [5].

Pada layer terakhir, yaitu *fully connected layer*, *input* yang diterima adalah output dari layer sebelumnya (*pooling layer*). Pada *fully connected layer*, dilakukan *learning* dengan menggunakan *feature* berupa hasil konvolusi yang didapat dari *pooling layer*. *Fully connected layer* menghasilkan *output* berupa vektor berdimensi N, di mana N adalah jumlah *class* yang sesuai dengan kategori apa saja yang ada pada *dataset* yang digunakan (misalnya mobil, burung, pesawat).

2.2 VGG-16

VGG *network* adalah arsitektur *Convolutional Neural Network* yang dirancang oleh Karen Simonyan dan Andrew Zisserman dari Visual Geometry Group, Department of Engineering Science, University of Oxford. Arsitektur ini dibuat untuk mengikuti kompetisi ImageNet Challenge 2014 dan berhasil meraih peringkat teratas untuk *localization* dan *classification*. *Input* yang digunakan berupa RGB *image* berukuran 224 × 224 *pixels*. *Convolutional layer* yang digunakan dalam arsitektur ini ada 2 jenis, yaitu *convolutional layer* dengan ukuran *filter* 3x3 (conv3) dan ukuran *filter* 1x1 (conv1). Ukuran *convolutional layer* yang digunakan bermacam-macam, yaitu 64x64, 128x128, 256x256, dan 512x512 [10].

2.3 *Image Dilation*

Morphological image processing berkaitan dengan mengubah struktur geometris yang ada dalam sebuah *image*. Biasanya, *morphological operations* dilakukan pada *binary image*, yaitu gambar yang hanya memiliki warna hitam dan putih saja. *Dilation* adalah salah satu bentuk *morphological transformation. Dilation* menggunakan konvolusi dengan kernel berbentuk kotak atau lingkaran.

Dalam proses *dilation*, kernel yang digunakan untuk *dilation* memiliki titik pusat (*anchor point*). Ketika kernel dikonvolusi dengan gambar, dihitung nilai *pixel* maksimal yang mengalami *overlap* dengan kernel. *Pixel* pada posisi *anchor point* (titik pusat) di gambar yang asli akan digantikan dengan nilai *pixel* maksimum yang telah didapat tadi [6]. Hasil dari prosedur ini adalah area yang berwarna terang akan semakin luas dan area yang berwarna gelap semakin sempit.

2.4 *Text Line Segmentation*

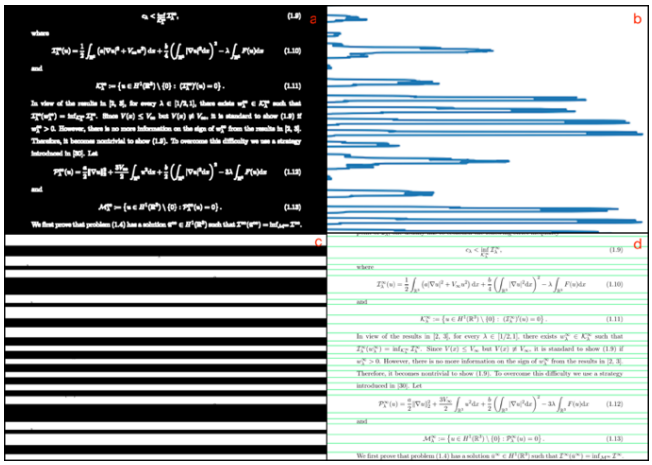
Segmentasi atau ekstraksi baris teks (*text line segmentation/extraction*) merupakan pemisahan baris-baris teks yang terdapat dalam sebuah halaman dokumen. *Text line segmentation* biasanya digunakan sebagai langkah *preprocessing* dalam pengenalan tulisan tangan dan *digital document analysis* [2].

Text line segmentation yang digunakan dalam skripsi ini dibuat berdasarkan metode *text localization* yang diusulkan dalam penelitian berjudul "*A Low Complexity Sign Detection and Text Localization Method for Mobile Applications*" oleh Katherine L. Bouman, Golnaz Abdollahian, Mireille Boutin, dan Edward J. Delp. Algoritma *text localization* tersebut diadopsi untuk melakukan *text line segmentation* pada dokumen oleh Wei-Ta Chu dan Fan Liu dalam penelitian "*Mathematical Formula Detection in Heterogeneous Document Images*". Langkah-langkah dalam algoritma *text line segmentation* tersebut adalah sebagai berikut:

1. Bagi halaman dokumen yang sudah berupa *image* menjadi blok sebesar k x k *pixels*.
2. Buat 3 buah *weighting matrix* berukuran k x k, masing-masing 1 untuk arah vertikal, horizontal, dan diagonal. Setiap matriks diisi angka +1 untuk warna putih dan -1 untuk warna hitam.
3. Hitung derajat homogenitas *pixel* setiap blok dengan melakukan konvolusi setiap blok dengan 3 *weighting matrix* yang didapat pada langkah sebelumnya. Sebuah blok dianggap homogen isi *pixel*-nya apabila panjang vektor blok tersebut tersebut kurang dari sebuah *threshold* dan setidaknya 1 dari 4 blok di sekitarnya juga memenuhi kriteria ini.
4. Warnai blok yang dianggap homogen *pixel*-nya dengan warna hitam dan blok yang tidak homogen *pixel*-nya dengan warna putih. (Dalam skripsi ini, langkah 1-4 digantikan dengan meng-*invert image* dan melakukan *dilation* dan dibandingkan dengan metode aslinya)
5. Hitung *horizontal projection profile* dari semua nilai *pixel* yang ada pada gambar dokumen yang bloknya telah diwarnai hitam putih.

Baris yang *horizontal projection profile*-nya lebih besar dari sebuah *threshold* dianggap sebagai baris teks. *Threshold* yang digunakan dalam algoritma ini adalah 10% dari nilai rata-rata *horizontal projection profile* pada seluruh halaman [4]. *Horizontal projection profile* merupakan *running sum* dari semua *pixel* yang ada pada arah (*direction*) horizontal. Karena *block* yang dianggap homogen diwarnai putih (nilai *pixel* 255), maka semakin besar nilai *horizontal profile projection* pada suatu arah, semakin banyak area pada arah tersebut yang bukan merupakan gambar kosong. Contoh hasil *horizontal profile projection* pada gambar yang telah melalui

proses pewarnaan *block pixel* homogen (langkah 1-4) dapat dilihat pada Gambar 1.

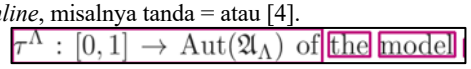


Gambar 1. Langkah-langkah *text line segmentation*

2.5 *Word Segmentation*

Segmentasi atau ekstraksi kata (*word segmentation/extraction*) merupakan pemisahan kata-kata dalam baris-baris teks yang terdapat dalam sebuah halaman dokumen. Dalam skripsi ini, *word segmentation* digunakan sebagai langkah *preprocessing* dalam mengidentifikasi rumus *inline* (rumus yang bercampur dengan teks). Algoritma *word segmentation* untuk membantu mengidentifikasi rumus *inline* yang digunakan mengacu pada penelitian Wei-Ta Chu dan Fan Liu yang berjudul "*Mathematical Formula Detection in Heterogeneous Document Images*". *Word segmentation* ini dilanjutkan dari *text line segmentation*. Langkah-langkah dalam algoritma *word segmentation* tersebut adalah sebagai berikut:

1. Cari *minimum bounding box* dari setiap area yang tidak homogen dari sebuah *text line*.
2. Cek jarak horizontal antara satu *bounding box* dengan yang lain. Jika jarak tersebut kurang dari suatu *threshold*, maka gabungkan kedua *bounding box* karena kemungkinan isi *bounding box* tersebut adalah bagian dari kata atau rumus yang sama.
3. Cek *neighbor* kanan dan kiri dari setiap *bounding box*. Jika tetangga kanan dan kiri sebuah *bounding box* hanya berisi 1 karakter, maka gabungkan ketiga *bounding box* tersebut. Hal ini dilakukan karena jarang kata-kata yang hanya terdiri dari satu huruf. Kemungkinan, satu huruf tersebut merupakan bagian dari rumus *inline*, misalnya tanda = atau [4].

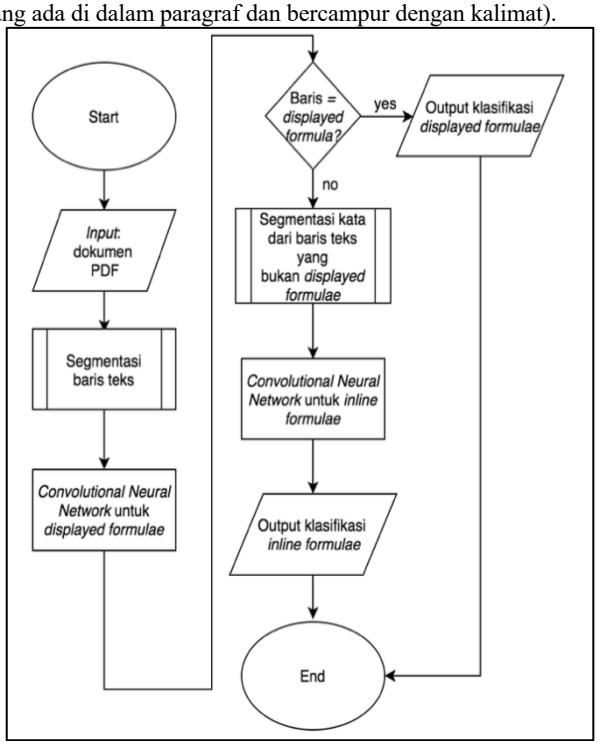


Gambar 2. Contoh hasil *word segmentation*

3. DESAIN SISTEM

3.1 Desain Sistem

Deteksi rumus matematika pada halaman dokumen digital dilakukan dengan mengubah dokumen menjadi *image,* kemudian memisahkan setiap baris teks dan kata-kata yang ada di dalamnya dan mengklasifikasi baris teks dan kata-kata tersebut menggunakan *Convolutional Neural Network.* Pertama, hasil *text line segmentation* diklasifikasi terlebih dahulu, apakah termasuk *displayed formula* (rumus yang tidak bercampur dengan paragraf, 1 baris hanya berisi rumus saja). Jika tidak, maka akan dilakukan *word segmentation* pada baris tersebut dan diklasifikasi apakah setiap kata yang tersegmentasi termasuk *inline formulae* (rumus yang ada di dalam paragraf dan bercampur dengan kalimat).



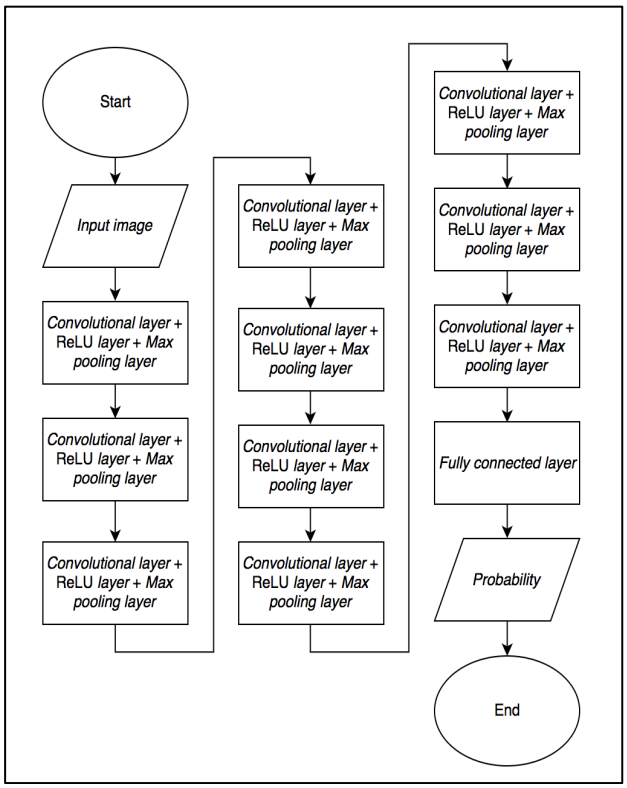
Gambar 3. Desain sistem

3.2 Proses *Training* pada *Convolutional Neural Network*

Training dataset diambil dari *website* www.arxiv.org dengan *search query* 'electron' dan *category* 'all', di mana setiap halaman dokumen yang ada dalam *dataset* diambil baris teks dan kata-katanya untuk dijadikan *training dataset*. Proses *training* dilakukan dengan 6 *epoch* yang berbeda, yaitu 25, 50, 75, 100, 125, dan 150. Terdapat 6 jenis arsitektur *Convolutional Neural Network* yang di-*train*, yaitu:

1. VGG-16 dengan *learning rate* 0.000001
2. 7 jenis arsitektur CNN dengan *convolutional layer* berukuran 64x64, ukuran *convolutional filter* 3x3, dan ukuran *max pooling* 2x2, dan *learning rate* 0.001:
   a. 64x2, yaitu arsitektur dengan 2 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   b. 64x4, yaitu arsitektur dengan 4 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   c. 64x6, yaitu arsitektur dengan 6 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   d. 64x8, yaitu arsitektur dengan 8 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   e. 64x10, yaitu arsitektur dengan 10 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   f. 64x12, yaitu arsitektur dengan 12 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.
   g. 64x14, yaitu arsitektur dengan 14 kumpulan *layer Convolutional*-ReLU-*Max Pooling*.

   Setiap arsitektur ini di-*train* dengan 6 *epoch* yang telah disebutkan di atas untuk ketiga jenis *training dataset*, yaitu *displayed formulae* untuk dokumen dengan 1 kolom, *displayed formulae* untuk dokumen dengan 2 kolom, dan *inline formulae*.



Gambar 4. *Flowchart* ilustrasi arsitektur 64x10

4. PENGUJIAN SISTEM

4.1 Pengujian *Text Line Segmentation*

Pada pengujian *text line segmentation* ini, rincian *true positive, true negative, false positive,* dan *false negative* yang digunakan adalah sebagai berikut:

1. *True positive*/TP: Tepat 1 baris teks di dalam 1 *bounding box*
2. *False positive*/FP: Bukan baris teks tetapi dianggap baris teks. Misalnya, simbol kecil yang biasanya menyertai simbol sigma, titik pada huruf 'i', dan simbol-simbol lain yang berada di atas atau bawah rata-rata ketinggian baris teks yang lainnya.
3. *False negative*/FN: Lebih dari 1 baris teks dalam 1 *bounding box.* Jika ada 2 baris teks di dalam 1 *bounding box*, jumlah *false positive* dianggap 2. Begitu juga jika ada 3 baris teks di dalam 1 *bounding box*, jumlah *false positive* dianggap 3.

Tabel 1. Hasil pengujian *text line segmentation* pada dokumen dengan 1 kolom

Metode	Rata-rata waktu per halaman (detik)	*Precision*	*Recall*	F1 *Score*
Homogenous blocks coloring	10,398	0,896	0,890	0,893
Dilation	1,341	0,963	0,917	0,940

Hasil pengujian *text line segmentation* pada dokumen dengan 1 kolom (Tabel 1) menunjukkan bahwa *precision* dari metode *dilation* hasilnya lebih baik karena lebih sedikit *false positive* (bukan baris teks tetapi dianggap baris teks, misalnya titik pada huruf 'i' atau simbol-simbol lain yang tingginya di atas rata-rata baris teks).

Tabel 2. Hasil pengujian *text line segmentation* pada dokumen dengan 2 kolom

Metode	Rata-rata waktu per halaman (detik)	*Precision*	*Recall*	F1 *Score*
Homogenous blocks coloring	11,938	0,879	0,760	0,815
Dilation	2,739	0,894	0,826	0,859

Pada Tabel 2, dapat dilihat bahwa *precision* untuk metode *homogenous blocks coloring* lebih rendah karena lebih banyaknya simbol yang tersegmentasi sebagai baris teks. Secara keseluruhan F1 *score* untuk *text line segmentation* pada dokumen dengan 2 kolom lebih rendah karena jarak antar baris biasanya lebih berdekatan dibandingkan dokumen dengan 1 kolom, sehingga semakin banyak baris yang seharusnya terpisah tetapi disegmentasi menjadi 1.

4.2 Pengujian *Word Segmentation*

Tabel 3. Hasil Pengujian *Word Segmentation* pada Dokumen dengan 1 Kolom

Kategori	Jumlah	Persentase
1 kata dalam 1 *bounding box*	29485	89,92%
Lebih dari 1 kata dalam 1 *bounding box*	1735	5,29%
Rumus saja dalam 1 *bounding box*	993	3,03%
Rumus dan kata-kata dalam 1 *bounding box*	575	1,75%
TOTAL	32789	100%

Hasil pengujian *word segmentation* pada dokumen dengan 1 kolom dalam Tabel 3 menunjukkan bahwa 89,92% hasil *word segmentation* berupa 1 kata dalam 1 *bounding box*, 5,29% berupa lebih dari 1 kata dalam 1 *bounding box*, 3,03% rumus saja dalam 1 *bounding box*, dan 1,75% rumus dan kata-kata dalam 1 *bounding box*.

Tabel 4. Hasil Pengujian *Word Segmentation* pada Dokumen dengan 2 Kolom

Kategori	Jumlah	Persentase
1 kata dalam 1 *bounding box*	34272	87,70%
Lebih dari 1 kata dalam 1 *bounding box*	3942	10,09%
Rumus saja dalam 1 *bounding box*	357	0,91%
Rumus dan kata-kata dalam 1 *bounding box*	508	1,30%
TOTAL	39079	100%

Pada hasil pengujian *word segmentation* pada dokumen dengan 2 kolom (Tabel 4), dapat dilihat bahwa jumlah hasil yang berupa "lebih dari 1 kata dalam 1 *bounding box*" dan "rumus bercampur dengan kata-kata dalam 1 *bounding box*" lebih banyak daripada

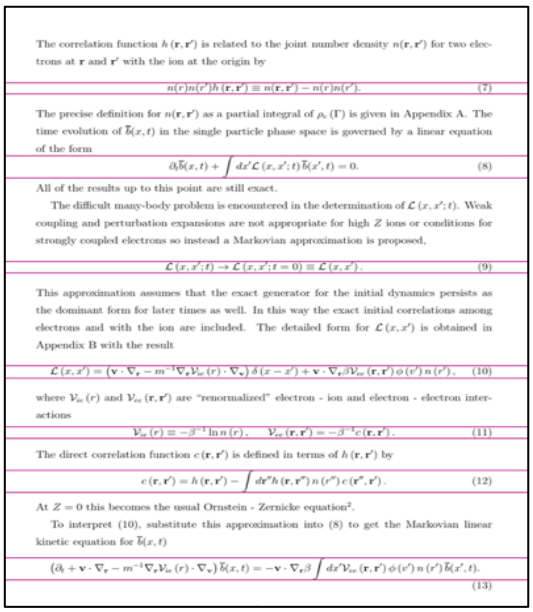
dokumen dengan 1 kolom. Hal ini diakibatkan hasil *text line segmentation* pada dokumen dengan 2 kolom lebih banyak mengandung 2 atau lebih baris yang tersegmentasi menjadi 1 baris.

4.3 Pengujian Model *Convolutional Neural Network* (CNN) untuk Deteksi Rumus *Displayed*

Tabel 5. Hasil Pengujian Model CNN untuk Deteksi Rumus *Displayed* pada Dokumen dengan 1 Kolom

Training Epochs		64x2	64x4	64x6	64x8	64x10	64x12	VGG-16
25	*Precision*	0,619	0,848	0,924	0,945	0,946	0,926	0,500
	Recall	0,970	0,965	0,975	0,985	0,965	0,965	1,000
	F1 *score*	0,756	0,903	0,949	0,965	0,955	0,944	0,667
50	*Precision*	0,582	0,803	0,863	0,954	**0,982**	0,916	0,500
	Recall	0,980	0,940	0,883	0,980	**0,978**	0,933	1,000
	F1 *score*	0,730	0,866	0,873	0,967	**0,980**	0,924	0,667
75	*Precision*	0,584	0,737	0,899	0,903	0,926	0,946	0,500
	Recall	0,968	0,960	0,960	0,975	0,973	0,968	1,000
	F1 *score*	0,728	0,834	0,929	0,938	0,949	0,957	0,667
100	*Precision*	0,586	0,741	0,863	0,945	0,959	0,943	0,500
	Recall	0,950	0,943	0,978	0,983	0,990	0,918	1,000
	F1 *score*	0,725	0,829	0,917	0,963	0,974	0,930	0,667
125	*Precision*	0,574	0,777	0,867	0,940	0,962	0,929	0,500
	Recall	0,950	0,960	0,915	0,975	0,945	0,985	1,000
	F1 *score*	0,716	0,859	0,891	0,957	0,953	0,956	0,667
150	*Precision*	0,645	0,785	0,889	0,963	0,944	0,940	0,500
	Recall	0,970	0,950	0,958	0,973	0,965	0,945	1,000
	F1 *score*	0,774	0,860	0,922	0,968	0,954	0,943	0,667

Hasil pengujian pada Tabel 5 menunjukkan bahwa model CNN yang menghasilkan F1 *score* tertinggi untuk deteksi rumus displayed pada dokumen dengan 1 kolom adalah arsitektur 64x10 yang di-train sebanyak 50 *epochs*. Terlihat bahwa dari arsitektur 64x2 hingga 64x10, F1 *score* yang dihasilkan semakin meningkat dengan bertambahnya kumpulan layer *Convolutional*-ReLU-*Max Pooling*. Tetapi, jika arsitektur yang digunakan lebih kompleks dari 64x10, F1 *score* agak menurun. Pada arsitektur VGG-16 yang lebih rumit, F1 *score* menjadi rendah karena semua input dianggap sebagai rumus. Contoh hasil deteksi rumus *displayed* pada dokumen dengan *layout* 2 kolom dapat dilihat pada Gambar 5.

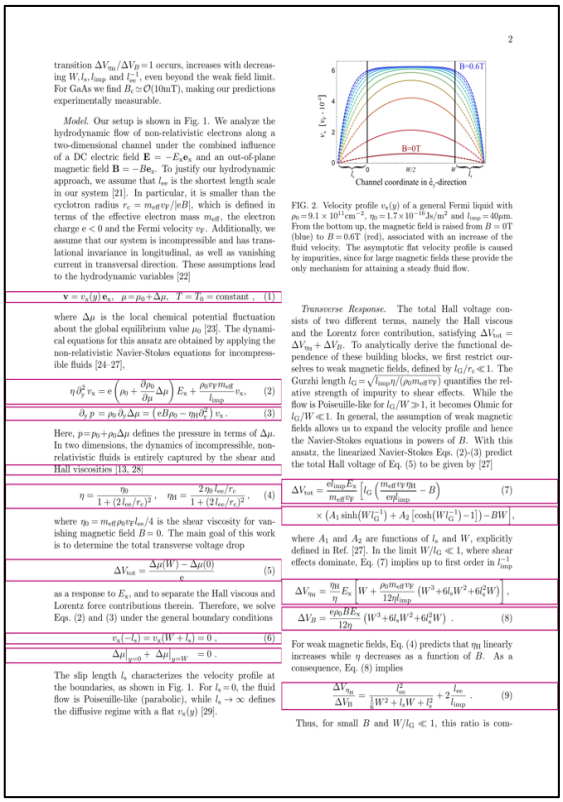


Gambar 5. Contoh hasil deteksi rumus *displayed* pada dokumen dengan 1 kolom

Tabel 6. Hasil Pengujian Model CNN untuk Deteksi Rumus *Displayed* pada Dokumen dengan 2 Kolom

Training Epochs		64x2	64x4	64x6	64x8	64x10	64x12	VGG-16
25	*Precision*	0,648	0,718	0,841	0,886	0,594	0.886	0,500
	Recall	0,993	1,000	0,975	0,995	1,000	0.970	1,000
	F1 *score*	0,785	0,836	0,903	0,938	0,746	0.926	0,667
50	*Precision*	0,757	0,702	0,798	0,881	**0,933**	0.911	0,500
	Recall	0,988	1,000	0,990	0,998	**0,948**	0.933	1,000
	F1 *score*	0,857	0,825	0,884	0,936	**0,940**	0.921	0,667
75	*Precision*	0,721	0,781	0,633	0,885	0,955	0.962	0,500
	Recall	0,980	0,990	0,998	0,998	0,893	0.891	1,000
	F1 *score*	0,831	0,873	0,775	0,938	0,922	0.925	0,667
100	*Precision*	0,770	0,706	0,802	0,875	0,821	0.815	0,500
	Recall	0,980	0,998	0,983	1,000	1,000	0.976	1,000
	F1 *score*	0,862	0,827	0,883	0,933	0,902	0.887	0,667
125	*Precision*	0,754	0,811	0,837	0,797	0,845	0.834	0,500
	Recall	0,973	0,995	0,985	1,000	0,995	0.998	1,000
	F1 *score*	0,849	0,893	0,905	0,887	0,914	0.915	0,667
150	*Precision*	0,707	0,737	0,821	0,833	0,697	0.696	0,500
	Recall	0,973	1,000	0,975	0,995	1,000	0.993	1,000
	F1 *score*	0,819	0,848	0,891	0,907	0,821	0.817	0,667

Hasil pengujian pada Tabel 6 menunjukkan bahwa model CNN yang menghasilkan F1 *score* tertinggi untuk deteksi rumus *displayed* pada dokumen dengan 2 kolom adalah arsitektur 64x10 yang di-*train* sebanyak 50 *epochs*, sama seperti deteksi rumus *displayed* pada dokumen dengan 1 kolom. F1 *score* agak menurun jika arsitektur yang digunakan lebih rumit dari 64x10. Sama seperti hasil pengujian deteksi rumus *displayed* pada dokumen dengan 1 kolom, F1 *score* yang dihasilkan VGG-16 lebih rendah karena hasilnya hanya *true positive* dan *false positive* (semua dianggap rumus). Contoh hasil deteksi rumus *displayed* pada dokumen dengan *layout* 2 kolom dapat dilihat pada Gambar 6.



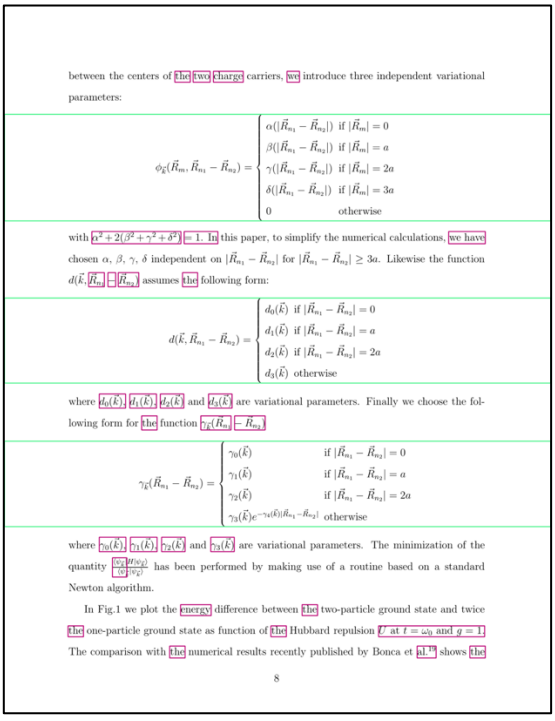
Gambar 6. Contoh hasil deteksi rumus *displayed* pada dokumen dengan layout 2 kolom

4.4 Pengujian Model *Convolutional Neural Network* untuk Deteksi Rumus *Inline*

Tabel 7. Hasil Pengujian Model CNN untuk Deteksi Rumus *Inline*

Training Epochs		64x4	64x6	64x8	64x10	64x12	64x14	VGG-16
25	*Precision*	0,634	0,771	0,786	0,863	0,649	0,860	0,500
	Recall	0,933	0,968	0,980	0,973	0,805	0,505	1,000
	F1 *score*	0,755	0,858	0,872	0,914	0,719	0,636	0,667
50	*Precision*	0,556	0,697	0,689	0,853	**0,866**	0,855	0,500
	Recall	0,938	0,953	0,930	0,945	**0,973**	0,898	1,000
	F1 *score*	0,698	0,805	0,791	0,897	**0,916**	0,876	0,667
75	*Precision*	0,560	0,676	0,790	0,695	0,704	0,762	0,500
	Recall	0,985	0,965	0,990	0,957	0,993	0,913	1,000
	F1 *score*	0,714	0,795	0,879	0,805	0,824	0,830	0,667
100	*Precision*	0,506	0,640	0,754	0,786	0,796	0,707	0,500
	Recall	0,913	0,870	0,995	0,873	0,985	0,988	1,000
	F1 *score*	0,651	0,737	0,858	0,827	0,880	0,825	0,667
125	*Precision*	0,473	0,675	0,744	0,700	0,730	0,702	0,500
	Recall	0,845	0,970	0,957	0,910	0,935	0,998	1,000
	F1 *score*	0,606	0,796	0,837	0,791	0,820	0,824	0,667
150	*Precision*	0,486	0,670	0,729	0,743	0,683	0,859	0,500
	Recall	0,880	0,970	0,998	0,953	0,965	0,973	1,000
	F1 *score*	0,626	0,793	0,843	0,835	0,800	0,912	0,667

Hasil pengujian pada Tabel 7 menunjukkan bahwa model CNN yang menghasilkan F1 *score* tertinggi untuk deteksi rumus *inline* pada dokumen dengan adalah arsitektur 64x12 yang di-train sebanyak 50 *epochs*. Jika arsitektur yang digunakan lebih rumit dari 64x12, F1 *score* mulai menurun. Contoh hasil deteksi rumus *inline* dapat dilihat pada Gambar 7.


between the centers of the two charge carriers, we introduce three independent variational parameters:

$$\phi_E(\vec{R}_m, \vec{R}_{n_1} - \vec{R}_{n_2}) = \begin{cases} \alpha(|\vec{R}_{n_1} - \vec{R}_{n_2}|) & \text{if } |\vec{R}_m| = 0 \\ \beta(|\vec{R}_{n_1} - \vec{R}_{n_2}|) & \text{if } |\vec{R}_m| = a \\ \gamma(|\vec{R}_{n_1} - \vec{R}_{n_2}|) & \text{if } |\vec{R}_m| = 2a \\ \delta(|\vec{R}_{n_1} - \vec{R}_{n_2}|) & \text{if } |\vec{R}_m| = 3a \\ 0 & \text{otherwise} \end{cases}$$

with $\alpha^2 + 2(\beta^2 + \gamma^2 + \delta^2) = 1$. In this paper, to simplify the numerical calculations, we have chosen α, β, γ, δ independent on $|\vec{R}_{n_1} - \vec{R}_{n_2}|$ for $|\vec{R}_{n_1} - \vec{R}_{n_2}| \geq 3a$. Likewise the function $d(\vec{k}, \vec{R}_{n_1} - \vec{R}_{n_2})$ assumes the following form:

$$d(\vec{k}, \vec{R}_{n_1} - \vec{R}_{n_2}) = \begin{cases} d_0(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = 0 \\ d_1(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = a \\ d_2(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = 2a \\ d_3(\vec{k}) & \text{otherwise} \end{cases}$$

where $d_0(\vec{k})$, $d_1(\vec{k})$, $d_2(\vec{k})$ and $d_3(\vec{k})$ are variational parameters. Finally we choose the following form for the function $\gamma_{\vec{k}}(\vec{R}_{n_1} - \vec{R}_{n_2})$:

$$\gamma_{\vec{k}}(\vec{R}_{n_1} - \vec{R}_{n_2}) = \begin{cases} \gamma_0(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = 0 \\ \gamma_1(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = a \\ \gamma_2(\vec{k}) & \text{if } |\vec{R}_{n_1} - \vec{R}_{n_2}| = 2a \\ \gamma_3(\vec{k}) e^{-\gamma_4(\vec{k})|\vec{R}_{n_1} - \vec{R}_{n_2}|} & \text{otherwise} \end{cases}$$

where $\gamma_0(\vec{k})$, $\gamma_1(\vec{k})$, $\gamma_2(\vec{k})$ and $\gamma_3(\vec{k})$ are variational parameters. The minimization of the quantity $\frac{\langle \psi_G | H | \psi_G \rangle}{\langle \psi_G | \psi_G \rangle}$ has been performed by making use of a routine based on a standard Newton algorithm.

In Fig.1 we plot the energy difference between the two-particle ground state and twice the one-particle ground state as function of the Hubbard repulsion U at $t = \omega_0$ and $g = 1$. The comparison with the numerical results recently published by Bonca et al.[9] shows the

8

Gambar 7. Contoh hasil deteksi rumus *inline*

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian aplikasi, dapat diambil beberapa kesimpulan antara lain:

- Arsitektur *Convolutional Neural Network* yang terlalu kompleks seperti VGG-16 tidak cocok untuk mendeteksi rumus matematika dalam bentuk *image* karena *image*-nya sederhana, mayoritas hanya berisi *pixel* hitam dan putih.
- Untuk deteksi rumus *displayed* pada dokumen dengan 1 kolom, arsitektur yang menghasilkan F1 *score* terbaik (sebesar 0,980) yaitu arsitektur dengan 64 *filter* pada setiap convolutional layer, dengan 10 kumpulan *layer Convolutional-ReLU-Max Pooling* (64x10) yang di-*train* sebanyak 50 *epochs.*
- Untuk deteksi rumus *displayed* pada dokumen dengan 2 kolom, arsitektur yang menghasilkan F1 *score* terbaik sebesar 0,940 yaitu arsitektur 64x10 yang di-*train* sebanyak 50 *epochs.*
- Untuk deteksi rumus *inline*, arsitektur yang menghasilkan F1 *score* terbaik sebesar 0,916 yaitu arsitektur dengan 64 *filter* pada setiap convolutional layer, dengan 12 kumpulan *layer Convolutional-ReLU-Max Pooling* (64x12) yang di-*train* sebanyak 50 *epochs.*
- Akurasi deteksi rumus sangat dipengaruhi hasil *text line segmentation* dan *word segmentation*.

5.2 Saran

Saran yang diberikan untuk penyempurnaan dan pengembangan lebih lanjut untuk aplikasi ini adalah sebagai berikut:

- Pembuatan API untuk proses klasifikasi pada aplikasi sehingga pengguna aplikasi tidak perlu meng-*install* berbagai *open source library* yang diperlukan.
- Penambahan *dataset* sehingga dapat mengidentifikasi rumus matematika pada dokumen PDF dengan jenis *font* yang lebih bervariasi.
- Pengujian *Convolutional Neural Network* dengan arsitektur yang lebih bervariasi untuk menemukan akurasi yang lebih baik.

6. DAFTAR REFERENSI

[1] Aggarwal, C. C. 2018. *Neural Networks and Deep Learning: A Textbook.* Cham: Springer Nature.

[2] Amarnath, R., & Nagabhushan, P. 2018. Text line Segmentation in Compressed Representation of Handwritten Document using Tunneling Algorithm. *International Journal of Intelligent Systems and Applications in Engineering*, 251-261.

[3] Chen, K., & Seuret, M. 2017. *Convolutional Neural Networks for Page Segmentation of Historical Document Images.* URI=http://arxiv.org/abs/1704.01474

[4] Chu, W., & Liu, F. 2013. Mathematical Formula Detection in Heterogenous Document Images. *2013 Conference on Technologies and Applications of Artificial Intelligence*, doi:10.1009/taai.2013.38.

[5] Ciaburro, G., & Venkateswaran, B. 2017. *Neural Networks with R: Smart models using CNN, RNN, Deep Learning, and artificial intelligence principles.* Birmingham: Packt Publishing.

[6] Fisher, R., Perkins, S., Walker, A., & Wolfart, E. 2003. *Hypermedia Image Processing Reference*. URI=https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm

[7] Lin, X., Gao, L., Tang, Z., Lin, X., & Hu, X. 2011. Mathematical Formula Identification In PDF Documents. *2011 International Conference on Document Analysis and Recognition*, doi:10.1109/icdar.2011.285.

[8] Poushter, J., Bishop, C., & Chwe, H. 2018. *Social Media Use Continues to Rise In Developing Countries*.

URI=http://www.pewglobal.org/2018/06/19/across-39-countries-three-quarters-say-they-use-the-internet

[9] Shanmugamani, R. 2018. *Deep Learning for Computer Vision.* Birmingham: Packt Publishing.

[10] Simonyan, K., & Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations.* San Diego.

[11] Singh, C. D. 2016. *Image Classification: CIFAR-10 Neural Networks vs Support Vector Machine.* URI=http://chahatdeep.github.io/docs/NNvsSVM.pdf