Form Recognition dan *Character Mapping* Menggunakan *Image Segmentation* dan *Optical Character Recognition*

http://dx.doi.org/10.28932/jutisi.v7i1.3340
Riwayat Artikel
Received: 27 Januari 2021 | *Final Revision*: 20 Maret 2021 | *Accepted*: 22 Maret 2021

Christian Wibisono[#1], Setia Budi✉[#2]

[#] *Jurusan Magister Ilmu Komputer, Universitas Kristen Maranatha*
Jl. Surya Sumantri no. 65, Sukawarna
[1]1979002@maranatha.ac.id
[2]Setia.Budi@it.maranatha.edu

Abstract — **Industry 4.0 revolves around the way of thinking in the manufacturing factory business. Speed and accuracy become the main focus to survive and grow in an uncertain situation. This study aims to build a system blueprint that will increase both speed and accuracy to input from data. This research uses several computer vision technologies like Convolutional Neural Network (CNN) that is used to do form classification and image segmentation, there is also Optical Character Recognition (OCR) which extracts specific information from a document that has been classified with CNN and then transforms it into a JSON format which has more generic format and can be used in most common platform.**

Keywords— **CNN; Form; OCR; Recognition**

I. PENDAHULUAN

Seiring dengan perkembangan teknologi yang semakin maju, maka setiap perusahaan *textile* dituntut untuk ikut berkembang agar bisa bertahan. Khususnya di era industri 4.0 di mana sudah dituntut untuk menerapkan *smart factory*. Hal ini menjadi perhatian khusus untuk melakukan studi yang bisa mendukung kelangsungan sebuah perusahaan *textile.*

Permasalahan yang umum dihadapi perusahaan *textile* biasanya berkaitan dengan performa kecepatan mengelola suatu task ataupun akurasi yang dihasilkan dari sebuah action. Kecepatan yang rendah berpotensi untuk meningkatkan kerugian dikarenakan efisiensi yang tidak sepadan dengan *labour cost* yang ada, meskipun pekerjaan yang dikerjakan benar hal ini tidak menjamin *profit* jika tidak diselaraskan dengan kecepatan kerja yang tinggi [9]. Di sisi lain akurasi memiliki peranan penting yang dapat menentukan keberlangsungan suatu perusahaan *textile*. Keakuratan data menjadi sebuah keharusan untuk perusahaan *textile* bisa mencapai *profit* yang baik, Data merupakan landasan pokok yang membangun sebuah *sistem* [9]. Tidak peduli sebaik apapun sebuah *sistem* yang diimplementasikan, jika data yang ada tidak akurat maka *output* yang dihasilkan juga tidak akan akurat. Kondisi ini bisa menyebabkan sebuah perusahaan mengalami kerugian bahkan kebangkrutan jika sampai salah mengambil keputusan yang diakibatkan kondisi data yang tidak akurat. Pada studi yang dilakukan pada salah satu perusahaan *textile*, kecepatan yang rendah mengakibatkan tingkat delay pengiriman barang cukup tinggi yang mana hal ini bisa mengakibatkan citra perusahaan terkait menjadi buruk di depan pelanggan. Sedangkan Tingkat akurasi yang rendah mengakibatkan kesalahan perhitungan *costing* yang mengakibatkan perusahaan merugi saat menentukan harga jual.

Untuk menjawab kebutuhan perusahaan *textile* terkait dalam meningkatkan kecepatan dan akurasi, Maka perlu dilakukan perubahan cara kerja dalam memperlakukan penginputan data ke dalam sebuah sistem. Data yang umum dilakukan input pada perusahaan ini pada umumnya berupa *form supplier*, di mana perusahaan ini memiliki banyak *supplier* yang memiliki form transaksi beragam sehingga tingkat kerumitan yang ada cukup kompleks. *Form* sendiri dapat diartikan sebagai informasi yang biasanya dipindahkan dari sebuah media cetak ke dalam sistem yang digunakan. Proses input ini memakan waktu yang cukup banyak dikarenakan kecepatan dari setiap orang yang mengerjakan berbeda, akurasi yang dihasilkan juga akan rentan terhadap kesalahan input yang berpotensi menyebabkan kerugian terhadap perusahaan.

Untuk mempercepat proses *input form*. Maka diperlukan sebuah sistem yang mampu meningkatkan kecepatan melakukan *input form*, namun juga harus meningkatkan akurasi data secara keseluruhan. Untuk memenuhi

kebutuhan ini maka studi ini dilaksanakan untuk membuat *blueprint* sistem yang nantinya bisa digunakan oleh perusahaan untuk mengembangkan sistem untuk menyesuaikan dengan proses bisnis yang ada secara internal. Di harapkan hasil dari studi ini mampu meningkatkan kecepatan input sehingga *image* perusahaan tetap baik di hadapan pelanggan, di satu sisi akurasi yang meningkat diharapkan bisa membantu perusahaan dalam proses *costing* agar penentuan harga jual bisa menghasilkan profit.

Sistem yang dibuat memiliki fungsi untuk melakukan *pengelompokan form* dan *scanning form*. Untuk pengelompokan form akan menggunakan arsitektur *CNN*, arsitekrur ini merupakan salah satu komponen dalam *computer vision* yang memiliki fungsi untuk menghasilkan *model* yang mampu mengelompokan sebuah *image* berdasarkan konten visual yang dimilikinya [2]. Sedangkan untuk *scanning form* akan digunakan arsitektur *OCR*, *OCR* sendiri merupakan salah satu arsitektur yang ada di dalam *computer vision* yang memiliki fungsi untuk menerjemahkan setiap informasi *alphabetic* yang ada di dalam sebuah *image* ke dalam format *ASCII* [1]. Nantinya hasil format *ASCII* ini akan dilakukan transformasi ke dalam bentuk *JSON*, agar bisa digunakan pada *platform* aplikasi secara *global*.

Dengan adanya sistem ini, pengguna tidak perlu lagi memindahkan informasi yang ada ke dalam aplikasi. Namun hanya perlu melakukan *review* dari hasil pengelompokan dan *scanning* yang dikerjakan oleh sistem. Dengan demikian kecepatan akan meningkat dikarenakan campur tangan *end user* berkurang drastis dalam melakukan proses input dan akurasi akan meningkat dikarenakan pengecekan yang dilakukan oleh sistem ini memiliki tingkat *error* yang lebih kecil dibandingkan pengecekan yang dilakukan oleh *user*.

Dengan Kebutuhan yang harus dipenuhi oleh perusahaan terkait maka didapat rumusan masalah yang menjadi fokus untuk meningkatkan efektifitas penggunaan waktu dan akurasi data. Apakah sistem yang mampu meningkatkan kecepatan dan akurasi dalam melakukan penginputan *form supplier*, sehingga delay pengiriman barang bisa dikurangi dan akurasi *costing* produk lebih akurat.

Pada penelitian ini ada beberapa batasan yang diterapkan di mana batasan ini dimaksudkan agar penelitian terkait bisa lebih fokus pada permasalahan pokok. Adapun batasan yang dimaksud berkaitan dengan *form* yang discan harus diisi secara digital tidak ditulis tangan. Dan dataset yang digunakan pada studi ini bersifat public dan tidak mengandung data rahasia ataupun informasi sensitif yang bisa merugikan pihak lain.

Tujuan dari studi ini adalah untuk menciptakan sebuah sistem, yang mampu menunjang proses input data form yang sebelumnya dilakukan sepenuhnya oleh *staff* administrasi. sistem akan otomatis mengelompokan *form* sesuai kategorinya lalu melakukan *scanning* dokumen untuk mendapat *value* yang akan dipindahkan. Dengan sistem ini *staff* administrasi yang sebelumnya melakukan input data akan beralih fungsi menjadi pihak yang melakukan verifikasi kesesuaian data yang dihasilkan secara automatis oleh sistem. Sistem yang dibuat pada penelitian ini bukanlah produk *final* yang bisa langsung digunakan oleh perusahaan. Sehingga perlu dilakukan *tuning* untuk menyesuaikan kebutuhan proses bisnis yang ada sebelum bisa digunakan pada sistem utama.

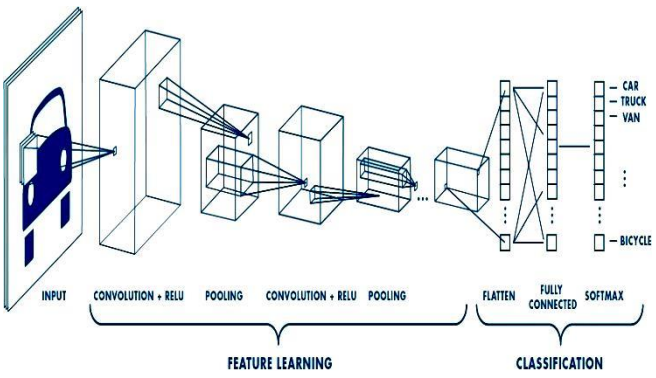
II. LANDASAN TEORI

Pada bagian ini akan dibahas teori – teori yang berhubungan dengan *Convolutional Neural Network* (CNN) dan *Optical Character Recognition* (OCR).

A. Convolutional Neural Network (CNN)

CNN merupakan salah satu arsitektur dalam ilmu *deep learning*, *CNN* sendiri merupakan sebuah metode yang memiliki fungsi untuk mengenali objek berupa data digital. Dengan menerapkan *CNN*, hal ini memungkinkan sebuah *device* komputer untuk mengenali objek tertentu yang tersimpan dalam *image*. Contoh saat *image* yang berisi gambar manusia dibaca oleh komputer, komputer akan melihat *model* yang sudah ada lalu mengidentifikasi gambar tersebut dan menghasilkan output bahwa gambar tersebut adalah manusia. Jika digambarkan secara sederhana, cara kerja ini sama seperti mata manusia yang bisa mengenali objek yang berada dalam suatu gambar atau lukisan.

CNN sendiri merupakan arsitektur yang dikembangkan dari *Multilayer Perception* (MP) yang memiliki kegunaan untuk mengolah data gambar dua dimensi. Secara garis besar arsitektur *CNN* memiliki 3 *layer* dan 2 bagian utama. Untuk *layer* yaitu *convolutional layer*, *Pooling layer*, dan *fully-connected layer*. Sedangkan untuk bagian utama terdiri dari *Feature Learning* dan *classification* yang bisa dilihat pada Gambar 1. Metode ini yang nanti akan digunakan untuk mengenali suatu *form* berdasarkan *label* yang sudah diberikan. Berikutnya akan dibahas mengenai kegunaan dari masing - masing *layer*.



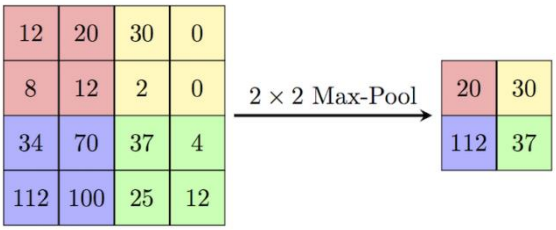
Gambar 1 *Flow CNN.*

Dari gambaran umum mengenai *CNN* yang sudah dibahas selanjutnya akan dibahas mengenai *convolutional*

p-ISSN : 2443-2210
e-ISSN : 2443-2229

layer. *Convolutional layer* Merupakan sebuah *layer* yang digunakan untuk memindai gambar digital, untuk *convolutional layer* sendiri memiliki *neuron* yang berupa matriks tiga dimensi yang terdiri dari panjang, lebar, dan tebal [3]. Untuk penentuan panjang, lebar dan tebal tergantung dari *image* yang dipindah semakin besar ukuran pixel gambar maka akan semakin besar ukuran matriks 3 dimensi dari *neuron* yang ada. Tujuan dari *neuron* ini adalah untuk mengambil sebagian *value* dari *image* lalu *neuron* yang sudah menangkap nilai citra ini dilakukan operasi *dot matrix* dengan *filter* yang digunakan pada *convolutional layer*. Untuk hasilnya setiap hasil perkalian *dot matrix* tersebut akan menghasilkan *output* yang memiliki istilah sebagai *activation map*.

Hasil perhitungan dari *convolutional layer* ini nantinya akan diproses oleh *Pooling layer*. *Pooling layer* memiliki fungsi untuk mereduksi dimensi *matrix* dari *output* yang dihasilkan oleh *Convolutional layer*. Untuk operasi reduksi teknik yang digunakan adalah *downsampling* adapun metode yang digunakan adalah *max pooling*. Di mana teknik ini akan mengambil nilai terbesar dari dari masing - masing *level layer* yang ada seperti Gambar 2 [3]. Adapun untuk hasil output dari *pooling layer* ini akan digunakan pada *Convolutional layer* di tahap selanjutnya, hal ini akan terus dilakukan sampai semua informasi didapat dengan nilai *error* yang kecil.

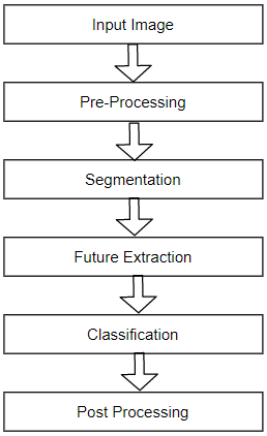


Gambar 2 *Max Pooling.*

Hasil dari *pooling layer* ini nantinya akan diproses pada *fully-connected*. *Fully-connected* merupakan *layer* terakhir dari tahap *CNN* di mana tugas dari *layer* ini adalah membentuk *data* yang sebelumnya sudah dipecah-pecah pada *layer* sebelumnya [6].

A. Optical Character Recognition

OCR merupakan sebuah fungsi yang berguna untuk menerjemahkan *text* yang ada di dalam *image* menjadi hasil *output* dalam bentuk *ASCII text* [2]. Keunikan dari *OCR* ini sendiri adalah fungsi ini mampu membedakan antara *icon* ,*image*, dan *text* di dalam citra digital. Sehingga hasil *output* hanya menghasilkan *text* saja. Adapun struktur pemrosesan dari *image* menjadi *text* dalam *OCR* memiliki beberapa tahap yang terdiri dari *Preprocessing, segmentation, extraction, classification* dan *post processing* yang dapat dilihat pada Gambar 3.



Gambar 3 Pemodelan *OCR.*

Adapun pada tahap *Preprocessing* yang dilakukan adalah merubah semua *image* di dalam *dataset* yang sebelumnya memiliki warna menjadi *gray image* lalu sesudah itu akan di *convert* menjadi *binary* data [8]. Jika sudah maka tahapan selanjutnya adalah melakukan reduksi *noise* yang ada misalnya *gaussian noise*.

Tahapan berikutnya adalah melakukan segmentasi, di mana setiap data *binary* yang sudah dibersihkan akan dilakukan proses pemecahan data. Di mana setiap huruf akan dibuat ke dalam bentuk biner, dalam melakukan proses ini *OCR* akan mengetahui mana data yang merupakan *character* dan mana data yang bukan merupakan *character* [8]. Sesudah tahapan ini maka langkah selanjutnya yang dilakukan adalah *feature extraction*, di mana jika ditemukan *intersection* diantara *binary* maka *OCR* akan menggabungkan huruf-huruf tersebut menjadi kata. Hal ini akan berulang kali dilakukan hingga menjadi 1 paragraf dan menyelesaikan seluruh *image* atau *document* yang di scan.

Langkah terakhir adalah *classification*, di mana dari setiap paragraf yang terbentuk *OCR* akan memberikan *label*, sehingga saat proses rekonstruksi *OCR* tidak akan keliru mengambil *value* selain character [8]. Jika sudah tahapan selanjutnya adalah *post processing* di sini terkait dengan mengolah hasil *output* dari *OCR*, bisa juga mengolah kembali *dataset* agar proses pembacaan *OCR* menjadi lebih cepat.

Adapun juga digunakan beberapa *library* yang merupakan pendukung agar arsitektur *OCR* dapat digunakan dengan baik, *library* ini adalah *pytesseract* dan *OpenCV*. *Pytesseract* merupakan *library* yang menerapkan algoritma dari *OCR*, *pytesseract* sendiri digunakan untuk mendeteksi *object* berupa *character* dari suatu gambar, dan menerjemahkannya ke dalam bentuk *string* [7]. Sedangkan *OpenCV* merupakan salah satu *library* python yang berhubungan dengan pengelolaan *object detection*, modul yang akan digunakan dari library *OpenCV* adalah object detection di mana *OpenCV* mampu mengenali setiap *object* yang ada di dalam gambar [7].

B. Penelitian terkait

Penerapan *CNN* dan *OCR* untuk melakukan klasifikasi dan recognition sudah pernah dilakukan oleh penelitian - penelitian yang ada. Setiap penelitian yang dilakukan memiliki metode tersendiri yang sudah disesuaikan dengan *dataset* yang ada. Pada penerapan CNN setiap penelitian memiliki setup parameter global yang berbeda dan penerapan beberapa teknik turunan yang ditambahkan, Untuk OCR sendiri penelitian yang ada berfokus pada cara melakukan segmentasi karakter.

Diego R Faria dan kawan - kawan menggunakan dataset CIFAR untuk melakukan klasifikasi *image* dengan format RGB yang tidak memiliki sub direktori dengan menggunakan arsitektur *Inception V3* dan menggunakan pendekatan *transfer learning* untuk proses pembuatan modelnya [3]. Pada penelitian ini tidak dilakukan proses preprocessing untuk menyesuaikan dengan model yang digunakan pada *transfer learning*, sebelum melakukan proses training Diego R Faria melakukan pengelompokan *image* yang didasarkan pada area label yang menunjukan identitas unik sebuah *image.* Dari proses ini didapat tiga kategori utama yaitu *image human*, *car*, dan *animal*, pada kategori human dilakukan percobaan menggunakan 500 epoch dengan 4000 epoch untuk mengetahui pengaruh epoch terhadap sebuah image yang memiliki kompleksitas tinggi tanpa adanya preprocessing yang hasilnya bisa dilihat pada Gambar 4**.** Untuk kategori *car* dan *animal* Diego R Faria hanya menggunakan 500 *epoch* dikarenakan percobaan pada kategori *human* sudah memberikan hasil bahwa *epoch* mempengaruhi tingkat akurasi suatu *model* [3].

Accuracy for 500 epochs	Accuracy for 4000 epochs
88%	95%

Gambar 4 Pengaruh epoch percobaan Diego R Faria.

Liu Liu dan kawan - kawan menggunakan *dataset* yang dibangun sendiri dan diberikan label dengan menerapkan *bounding box*, *dataset* yang dibangun oleh Liu Liu terdiri dari 1500 gambar dengan format *grayscale* yang dibagi menjadi 2 kategori berbeda dan tidak dilakukan *preprocessing* apapun sebelum dilakukan *training model* [13]. Proses pembentukan *model* dibangun menggunakan arsitektur *CNN-AGG* di mana arsitektur ini menjalankan clustering *CNN* yang hasilnya akan disambungkan secara seri. Adapun model variabel yang ditentukan pada penelitian ini menggunakan 20 *epochs*. *Model* yang dihasilkan memiliki tingkat akurasi sebesar 83% dengan data *loss* sebanyak 0.42%.

Manohar Karki dan kawan - kawan menggunakan *dataset MINST* untuk melakukan proses klasifikasi *character* bangla dan proses perbaikan *kernel* sebelum dilakukan *document scanning*, sebelum dilakukan proses *training dataset* yang ada dilakukan proses *preprocessing* seperti *gausian noise* untuk menghilangkan *noise* yang ada di dalam *image* [12]. Pada proses klasifikasi Manohar Karki menggunakan *transfer learning* dengan arsitektur C*NN ImageNet*, *model* yang dihasilkan memiliki fungsi untuk membedakan dokumen yang memiliki karakter bangla dengan akurasi sebesar 95% dan data *loss* sebesar 0.30%. Selanjutnya dilakukan proses *document scanning* dan perbaikan *kernel* dengan menerapkan *CRN*. *CRN* akan melakukan rekonstruksi setiap pixel character hingga didapat nilai *motion* yang stabil. Akurasi pembacaan sebelum dilakukan rekonstruksi adalah 80% sedangkan akurasi pembacaan setelah dilakukan rekonstruksi adalah 87%.

Sukhvinder Singh dan kawan - kawan menggunakan dataset yang dibuat sendiri untuk melakukan proses *dokumen scanning* [14]. *Dataset* yang ada dibagi ke dalam 3 kategori di mana dipisahkan berdasarkan ukuran dari *text* yang ada di dalam gambar. Kategori pertama di isi gambar dengan ukuran *text* kecil, kategori kedua di isi dengan *text* ukuran sedang dan kategori ketiga di isi dengan *text* ukuran besar. Dari setiap kategori Sukhvinder Singh melakukan morphologi matematis untuk mengambil setiap karakter *alphabet* yang ada dan melakukan *preprocessing* untuk menghilangkan *noise* berupa garis tepi. Setelah itu dilakukan proses penggabungan semua kategori ke dalam satu direktori yang sama lalu dilakukan *document scanning*. Dari penelitian ini didapat informasi bahwa perbesaran *pixel* dan penghilangan garis tepi bisa meningkatkan akurasi pembacaan *OCR*. Akurasi sebelum dilakukan proses morphologi untuk kategori gabungan sebesar 79.31% sedangkan setelah dilakukan proses morphologi akurasi yang dihasilkan sebesar 82.15% [14].

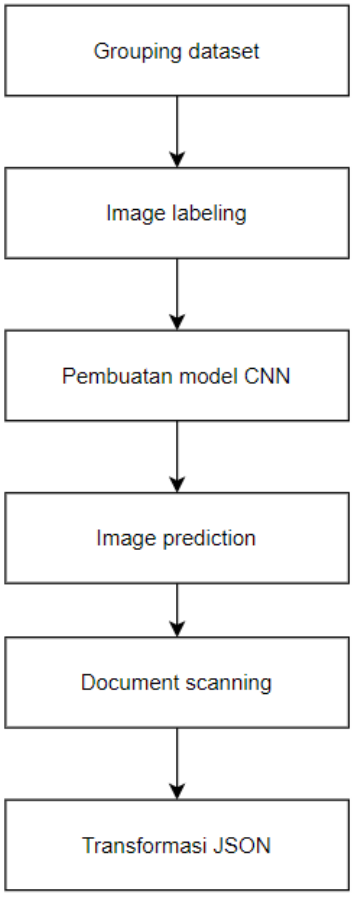
Rifiana Arief dan kawan - kawan menggunakan *dataset* yang dibuat sendiri untuk melakukan *document scanning*. Dataset ini terdiri dari sekumpulan gambar yang memiliki resolusi dan pixel yang besar, penelitian ini dijalankan pada *environment hadoop* sehingga memungkinkan clustering untuk mempercepat proses *scanning document* [15]. Dataset yang ada tidak dilakukan proses *preprocessing* dan penerapan *OCR* pada penelitian ini menggunakan 3 tools yang berbeda. Tujuan dari penelitian ini adalah untuk mendapatkan akurasi dan speed terbaik dari setiap tools yang digunakan, dengan menggunakan *OCR.net* didapat akurasi sebesar 90% dengan speed 8 detik untuk setiap dokumen dengan menggunakan *OCR.net* versi desktop yang menggunakan satu GPU *RTX 960* dan memory sebesar 2GB. Dan didapat akurasi sebesar 95% dengan speed 5 detik untuk setiap dokumennya untuk penggunaan *tools Google Vision*, dengan menggunakan satu GPU *NVIDIA TESLA K80* dan *memory* sebesar 4GB.

III. PERANCANGAN

Pada bagian ini akan dibahas mengenai perancangan yang disusun untuk membuat sistem *scanning*. Yang mana akan dibahas mulai dari *dataset* yang ada, *grouping dataset*

p-ISSN : 2443-2210
e-ISSN : 2443-2229

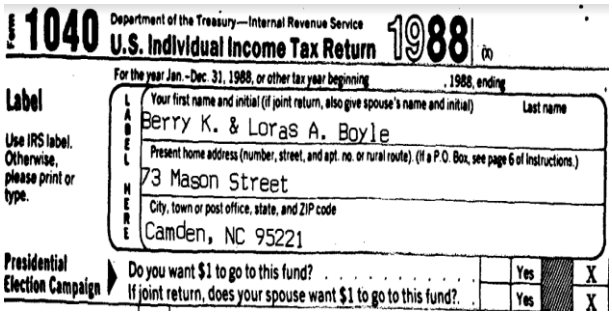
untuk mengelompokan dataset yang berbeda, *image labeling* untuk memberikan identitas unik pada gambar, pembuatan model *CNN* di mana akan dibuat dua model dengan arsitektur yang berbeda, *image prediction* untuk memberikan prediksi menggunakan *model* yang dipilih dan *document scanning* untuk mengambil informasi *character alphanumeric* dari gambar lalu akan dilakukan proses transformasi menjadi format JSON. Bagan penelitian bisa dilihat pada Gambar 5.



Gambar 5 Perancangan system.

A. *Basic Data*

Pada penelitian ini digunakan dua buah *dataset form* yang berbeda yaitu *dataset* NIST dan *dataset* TOBACO. Penggunaan *dataset* menggunakan lebih dari satu sumber dikarenakan perlu adanya keberagaman jenis *form* yang diharapkan bisa menjadi landasan yang baik dari *model* yang akan dibuat. Kedua *dataset* ini memiliki struktur yang berbeda di mana pada *dataset* NIST *form* yang ada akan tersebar secara *random* pada *folder-folder* yang ada, sedangkan *dataset* TOBACO sudah tersusun dalam direktori yang mewakilinya. Pada tahapan selanjutnya perlu dilakukan generalisasi agar kedua *dataset* memiliki *value* yang sama. Untuk contoh *form* bisa dilihat pada Gambar 6.


Form 1040 Department of the Treasury—Internal Revenue Service U.S. Individual Income Tax Return 1988
For the year Jan.-Dec. 31, 1988, or other tax year beginning , 1988, ending
Label
Use IRS label. Otherwise, please print or type.
Your first name and initial (if joint return, also give spouse's name and initial) Last name
Berry K. & Loras A. Boyle
Present home address (number, street, and apt. no. or rural route). (If a P.O. Box, see page 6 of Instructions.)
73 Mason Street
City, town or post office, state, and ZIP code
Camden, NC 95221
Presidential Election Campaign
Do you want $1 to go to this fund? Yes X
If joint return, does your spouse want $1 to go to this fund? Yes X

Gambar 6 Contoh *Form.*

B. Pengelompokan dan pelabelan

Langkah awal yang perlu dilakukan pada bagian ini adalah melakukan *grouping* dari kedua *dataset* yang ada. Adapun perancangan yang diinginkan adalah dengan mengambil 2 jenis *form* dari *dataset* NIST yang nantinya akan digabungkan ke dalam *dataset* TOBACO. Sesudah *grouping* maka tahapan selanjutnya adalah melakukan proses *labeling*.

Proses *labeling* disini dimaksudkan untuk memberikan identitas unik kepada setiap *image* yang ada, teknik yang akan digunakan untuk memberikan *label* adalah teknik *hot encode* di mana setiap *image* akan diberikan satu buah *label* yang mewakili seluruh isi konten yang ada di dalam sebuah *image*.

Teknik *hot encode* ini memiliki kelebihan dibandingkan pemberian *label* yang biasa menggunakan *bounding box*, di mana proses yang dihasilkan akan jauh lebih cepat namun di sisi lain teknik ini memiliki kekurangan. Di mana proses pelabelan menggunakan nama *folder* sebagai *key* bisa mempengaruhi performa training pada tahapan pembuatan model berbasiskan *CNN*, dan juga teknik ini hanya bisa digunakan untuk keperluan yang tidak memerlukan multi *label* dalam sebuah *image* yang sama.

C. *Pembuatan Model CNN*

Tahapan selanjutnya setelah proses *Hot Encode* selesai adalah melakukan pemilahan data untuk dipisahkan menjadi tiga bagian yang berbeda yaitu *data training*, *data validation* dan *data test*. *Data training* memiliki fungsi sebagai data yang digunakan oleh algoritma *CNN* untuk membangun pemodelan yang diinginkan, *data validation* digunakan untuk melakukan cross *check* terhadap *model* yang dibangun untuk mengetahui tingkat akurasi *model* saat dilakukan *cross check* dengan *data validation*. Dan *data test* digunakan untuk menguji akurasi *model* yang sudah selesai *training* untuk melakukan prediksi terhadap *data* yang benar – benar tidak digunakan sama sekali pada saat pembuatan *model*. Adapun pembagian data untuk per masing – masing kategori adalah *data training* 70%, *data validation* 20% dan *data test* 10%.

Sesudah proses pembagian data selesai dijalankan maka tahapan selanjutnya yang dilakukan adalah melakukan

proses *training model*. Proses *training* dijalankan menggunakan Algoritma *CNN* dengan dua jenis arsitektur yang berbeda yaitu *VGG-16* dan *Exception* Adapun ada beberapa *variable global* yang sudah diset sebelumnya agar saat melakukan perbandingan, kedua arsitektur memiliki bobot yang seimbang.

Setelah *model* selesai dibuat maka tahapan selanjutnya adalah melakukan prediksi *model*. Prediksi *model* disini bertujuan untuk mengetahui apakah pemodelan yang dibentuk mampu memberikan prediksi secara tepat terhadap *data test*. Pengecekan ketepatan cukup sederhana di mana akan dibandingkan hasil prediksi dari *model* yang ada dengan label awal dari *image*, jika sesuai maka akan menghasilkan nilai positif dan jika berbeda maka akan menghasilkan nilai negatif.

D. Document Scanning dan transformasi

Tahapan selanjutnya setelah *model* yang dihasilkan bisa memberikan prediksi adalah melakukan *document scanning*, pada tahap ini setiap prediksi yang memiliki nilai positif akan dilakukan proses *scanning* menggunakan *OCR*. Di mana setiap karakter yang ada di dalam gambar akan ditransformasi menjadi bentuk *ASCII*. Namun sebelum dilakukan proses *scanning*, dokumen terkait akan dilakukan transformasi menjadi *grayscale* dan *threshold*. Tujuan dari transformasi ini untuk meningkatkan akurasi pembacaan dari algoritma *OCR* itu sendiri.

Sesudah *image* diubah ke dalam bentuk *threshold* maka langkah selanjutnya adalah menerapkan algoritma *OCR* untuk melakukan *scanning* dokumen yang dimaksud. Hasil *scan* yang dimaksud lalu akan dikonversi menjadi format JSON.

IV. HASIL PENELITIAN

Dalam studi ini akan dilakukan pengerjaan sesuai dengan alur yang sudah dijabarkan pada bagian perancangan. Adapun akan digunakan teknik - teknik seperti *hot encode*, *OpenCV*, *CNN*, dan *OCR* untuk mendapatkan hasil dari studi ini.

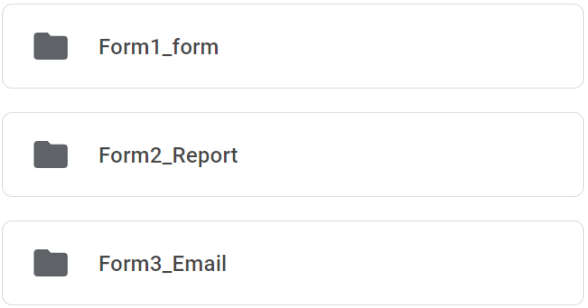
Tahap awal adalah melakukan penggabungan *dataset* NIST dan *dataset* TOBACO, adapun alasan penggabungan *dataset* adalah untuk mendapatkan keberagaman *form* yang lebih unik. Di mana pertama dilakukan analisa terhadap *dataset* NIST yang memiliki *image* yang tersebar ke dalam direktori - direktori. Image yang ada di dalam direktori ini tidak menunjukan keseragaman sehingga perlu dilakukan pengelompokan lebih lanjut agar satu direktori mewakili keseragaman image.

Dataset NIST ini memiliki sebuah pasangan *file* dengan format fmt yang memiliki nama sama dengan *image*, file fmt ini memiliki informasi ID unik yang bisa menjadi kunci untuk mengelompokan *image*, *content file* fmt ini bisa dilihat pada Gambar 7. dari informasi *file* fmt ini akan dilakukan pengelompokan *image* yang didasarkan kepada informasi *key* yang ada di dalam *file* fmt dan hasil *path image* akan di simpan pada *array* untuk proses selanjutnya.

```
1040_1
1040_1_L_H1_V1
1040_1_L_H2_V1
1040_1_L_H3_V1
```

Gambar 7 *Key content* file fmt.

Selanjutnya adalah menggabungkan alamat yang sudah disimpan ke dalam *array*, ke dalam *dataset* TOBACO dengan cara melakukan *looping array* yang setiap *loop* nya akan melakukan *cut image* ke dalam direktori yang sudah ada pada *dataset* TOBACO. *Dataset* TOBACO sendiri sudah memiliki direktori - direktori yang sudah mewakili konten *image* yang ada di dalamnya. Adapun direktori yang sudah mewakili konten *image* bisa dilihat pada Gambar 8.



Gambar 8 Direktori hasil pengelompokan.

Setelah seluruh *image* sudah berada di dalam direktori yang mewakili konten yang ada, tahapan selanjutnya adalah melakukan proses *labeling* menggunakan teknik *hot encode*. Teknik ini akan merubah semua *image* ke dalam bentuk *numpy array* lalu akan di sisipkan id yang merupakan nama *parent* direktori sebagai penanda unik suatu *image*.

Prinsip yang diterapkan adalah melakukan *looping* semua *image* yang ada di dalam direktori. Lalu melakukan identifikasi dengan cara membandingkan direktori *image* dengan kondisi penentuan label. Simulasi pemberian kondisi bisa dilihat pada Tabel I.

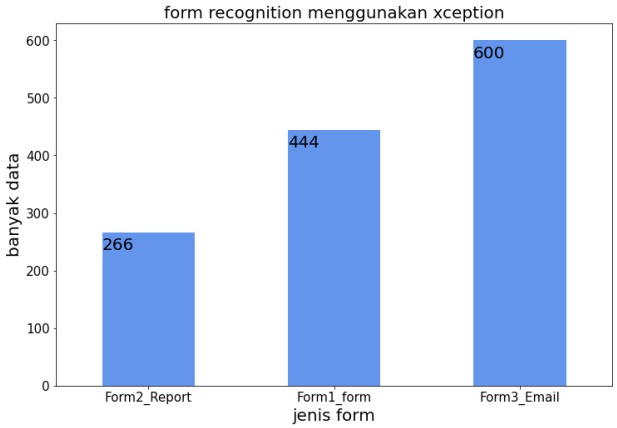
TABEL I
SIMULASI PEMBERIAN LABEL.

Gambar	Form1	Form2	Form 3
gambar 1	1	0	0
gambar 2	0	1	0
gambar 3	0	0	1
gambar 4	0	1	0
gambar 5	1	0	0

p-ISSN : 2443-2210
e-ISSN : 2443-2229

Nilai 1 menunjukan bahwa *image* yang diproses memiliki nama direktori yang sesuai dengan kondisi penentuan label. Saat didapat nilai 1 maka sistem akan memodifikasi *numpy array image* dengan menambahkan informasi label sebelum dikonversi kembali menjadi *image* semula. Jika didapat nilai 0 menunjukan direktori *image* tidak sesuai dengan kondisi penentuan label dan sistem tidak melakukan *action* apapun.

Setelah proses pemberian label selesai dilakukan, maka tahapan selanjutnya melakukan analisa data yang ada. Gabungan kedua *dataset* ini memiliki jumlah *image* sebanyak 1400 *image*. Adapun seluruh image ini tersebar pada tiga buah direktori, di mana form1 berisi 444 *image*, form2 berisi 266 *image* dan form3 berisi 600 *image*. Yang jumlah *image* setiap direktorinya bisa dilihat pada Gambar 9.



Gambar 9 Jumlah *image dataset* gabungan.

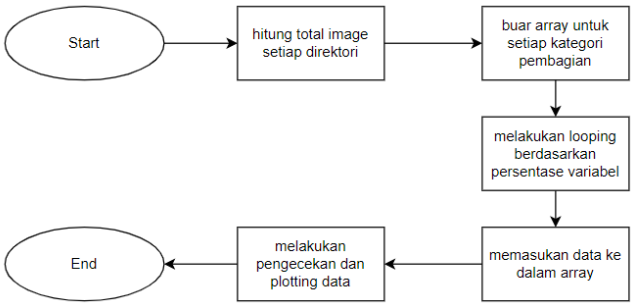
Sebelum melakukan proses *training* pemodelan *CNN*, perlu dilakukan *setting parameter global*. Hal ini dikarenakan akan dibuat dua buah *model CNN* yang menggunakan arsitektur yang berbeda, sehingga perlu parameter yang membuat kedua *model* bisa dibandingkan dengan kondisi yang sama. Informasi *parameter* ini bisa dilihat pada Tabel II.

TABEL II
PARAMETER GLOBAL.

parameter	value
img_size	125
batch_size	32
epochs	15
train_size	0.7
val_size	0.2
test_size	0.1
seed	4321

Berdasarkan Tabel II nilai *epoch* yang di ambil adalah 15 dikarenakan pola training *CNN* tidak difokuskan pada konten detail yang ada di dalam *image*, tetapi lebih berfokus pada pola dari konten *image*. Sehingga tidak diperlukan jumlah *epoch* yang tinggi untuk mencapai nilai akurasi yang stabil. Sedangkan nilai *seed* dimaksudkan agar kondisi *random* pada layer *CNN* arsitektur yang dijalankan memiliki nilai yang sama sehingga tidak perlu dilakukan proses *cross validation* untuk mendapatkan nilai akurasi yang dijalankan dengan kondisi random jika tujuannya hanya untuk menentukan pemilihan *model*. Nilai dari seed 4321 hanya menunjukan id *booking* untuk kondisi *training* pada arsitektur yang dijalankan.

Langkah berikutnya adalah melakukan pemisahan data sesuai *parameter* pada Tabel II. Di mana *data training* sebanyak 70%, *data validation* sebanyak 20% dan *data test* sebanyak 10%, flow proses pemisahan data ini bisa dilihat pada Gambar 10. Proses pembagian *image* ini didasarkan pada jumlah keseluruhan setiap kategori *form* sehingga setiap *segment* akan memiliki perbandingan yang sama dengan kondisi perbandingan awal *image* seperti gambar Gambar 9 Setelah pemisahan data dilakukan maka dilakukan *preprocessing convert* dan *resize*.



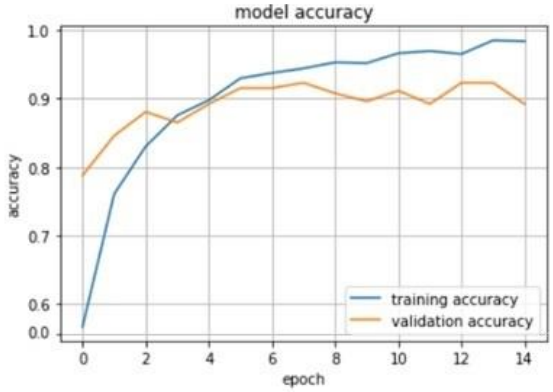
Gambar 10 Flow pembagian *dataset.*

Convert dibutuhkan untuk merubah format gambar untuk disesuaikan, proses *convert* dimaksudkan untuk merubah format gambar yang sebelumnya memiliki tiga layer *RGB* menjadi tiga layer *grayscale*, tujuan proses ini untuk mempercepat proses *training model* yang nanti akan dilakukan. Sedangkan *Rezise* dilakukan untuk memperkecil ukuran gambar selain untuk mempercepat, training *CNN* tidak perlu mengetahui seluruh detail gambar cukup pola gambar yang bisa diidentifikasi.

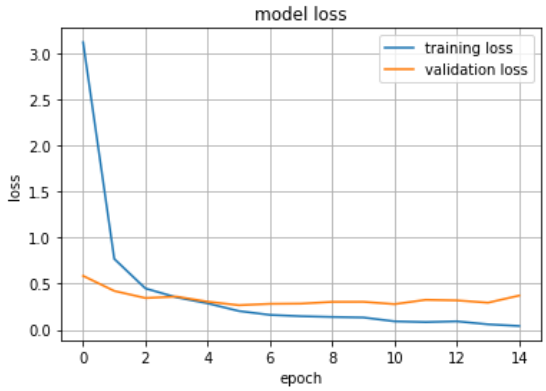
Tahapan selanjutnya yang dilakukan adalah melakukan *training model* menggunakan *CNN*, ada dua arsitektur yang digunakan pada tahapan ini yaitu *VGG-16* dan *Exception*. Dilakukan *training model VGG-16* seperti Kode program 1. *Model VGG-16* ini memiliki tingkat akurasi dan *loss* yang baik untuk 15 epoch yang dijalankan. Akurasi model *VGG-16* bisa dilihat pada Gambar 11, sedangkan tingkat *loss model VGG-16* bisa dilihat pada Gambar 12.

```
model.compile(optimizer=optimizers.Adam(lr=learnin
g_rate), loss='categorical_crossentropy', metrics=
['accuracy'])

train_model = model.fit(x_train, y_train,
                        batch_size=batch_size,
                        epochs=epochs,
                        verbose=1,
                        validation_data=(x_val, y_
                        val))
```

Kode program 1 Pembuatan *model VGG-16*.



Gambar 11 Akurasi *model VGG-16*.

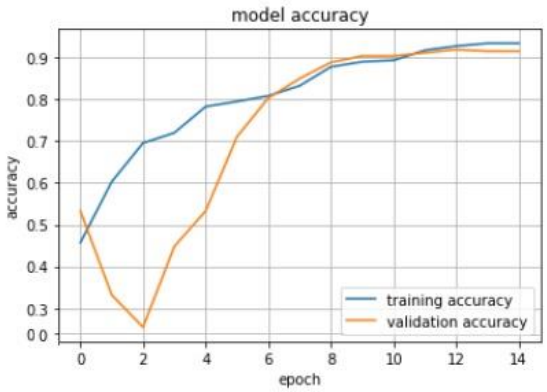


Gambar 12 Loss *model VGG-16*.

Sesudah Pemodelan *VGG-16* selesai dibuat, maka yang dikerjakan berikutnya adalah membuat pemodelan *Exception*. Pembuatan pemodelan *Exception* bisa dilihat pada Kode program 2, model ini menghasilkan akurasi seperti yang sedikit lebih kecil dibandingkan dengan akurasi pemodelan *VGG-16* seperti Gambar 13.

```
model = Model(base_model.input, predictions)


model.compile(optimizer=optimizers.Adam(lr=learnin
g_rate), loss='categorical_crossentropy', metrics=
['accuracy'])

train_model = model.fit(x_train, y_train,
                        batch_size=batch_size,
                        epochs=epochs,
                        verbose=1,
                        validation_data=(x_val, y_v
                        al))
```

Kode program 2 Pembuatan model *Exception*.

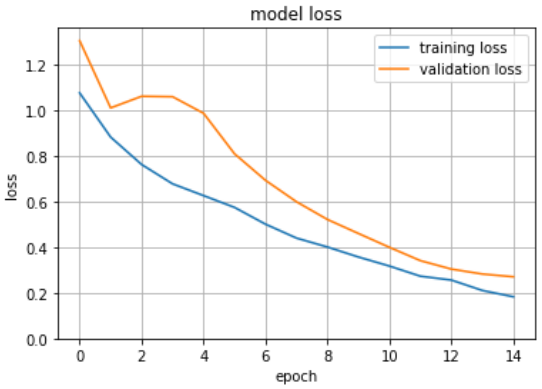


Gambar 13 Akurasi *model Exception*.

Ada hal yang unik pada akurasi pemodelan *Exception*, di mana akurasi validasi menurun di awal *epoch*. Hal ini dikarenakan pemodelan *Exception* memiliki dua buah *convolutional layer*, kedua *layer* ini akan melakukan kalkulasi secara independen saat melakukan ekstrasi fitur dalam sebuah gambar. Hal ini menyebabkan turunnya nilai *loss validation* yang cukup signifikan di awal *epoch*, namun akan meningkat saat kalkulasi kedua *layer* mulai menunjukan nilai yang identik. Prinsip kerja ini mirip dengan *transfer learning* yang terjadi di antara dua model yang berbeda, dalam kasus ini prinsip yang sama diterapkan dalam satu model yang sama.

Pemodelan *Exception* menghasilkan tingkat kalkulasi loss per *epoch* yang lebih rendah dibandingkan pemodelan *VGG-16* yang bisa dilihat pada Gambar 14. Hal ini dikarenakan perhitungan rumit dari dua buah *convolutional layer* yang bisa meningkatkan kemampuan *model* dalam memaksimalkan pemanfaatan fitur dalam sebuah *image*.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 14 Loss *model Exception.*

Setelah kedua arsitektur *CNN* ini selesai dibangun maka tahapan selanjutnya adalah melakukan perhitungan akurasi dan loss secara keseluruhan untuk pemodelan *Exception* dan *VGG-16*. Pemodelan *VGG-16* memiliki tingkat *loss* sebanyak 0.40% dan akurasi sebesar 93%, sedangkan untuk pemodelan *Exception* memiliki tingkat akurasi sebesar 90% dan tingkat *loss* sebesar 0.33% yang bisa dilihat pada Tabel III.

TABEL III
PERBANDINGAN *ACCURACY* DAN *LOSS.*

Arsitektur	***Accuracy***	***Loss***
VGG-16	93%	0.40%
Exception	90%	0.33%

Setelah mengetahui tingkat akurasi dan *loss* dari setiap *model* yang ada. Dipilih penggunaan *model Exception* untuk melakukan prediksi *Form*. Hal ini dikarenakan tingkat *loss* yang lebih kecil dibandingkan pemodelan *VGG-16 yang* menyebabkan validitas prediksi pemodelan *Exception* bisa lebih dipercaya, meskipun nilai akurasi pemodelan *Exception* lebih kecil namun masih berada dalam tingkat akurasi yang diperbolehkan, di mana akurasi sebesar 89% tidak berada dalam jangkauan *overfitting* ataupun *underfitting* sehingga model ini bisa digunakan untuk data form yang sifatnya *universal*. Langkah selanjutnya adalah melakukan prediksi menggunakan pemodelan *Exception* yang bisa dilihat pada Kode program 3.

```
predictions = model.predict_classes(x_test, verbos
e=1)
predictions_list = predictions.tolist()
predicted_classes = lb.classes_

count_true = 0;
count_false = 0;

for i, prediction in enumerate(predictions_list):
    state = True
    if (predicted_classes[prediction] != test_labe
l[i]) :
        state = False
        count_false += 1
    else :
        count_true += 1
    print("prediksi : ", predicted_classes[predict
ion], "   |   Real class : ", test_label[i],   "
|   Result : ", state)

print("\nberhasil : ", count_true)
print("gagal : ", count_false)
print(" rate : ", count_true/len(test_label))
```

Kode program 3 Prediksi *model.*

Prediksi ini dilakukan oleh model terhadap *data test*, yang mana data ini tidak digunakan sama sekali dalam proses pembuatan *model* dan juga validasi. Di dapat informasi tingkat akurasi prediksi dari *model Exception* bisa memberikan akurasi prediksi tepat sebesar 90% seperti Tabel IV. Kolom prediksi menunjukan hasil prediksi *model* terhadap data yang ada di dalam *dataset*, kolom real class menunjukan label yang diberikan pada *image* pada proses *labeling* yang menunjukan kategori sebuah *image*. Sedangkan kolom *Result* menunjukan hasil prediksi, jika bernilai *True* menunjukan bahwa nilai prediksi sama dengan nilai *real class*. Sedangkan jika bernilai *False* menunjukan nilai prediksi berbeda dengan nilai real class.

TABEL IV
HASIL PREDIKSI TERHADAP *DATA TEST*

Prediksi	**Real Class**	**Result**
Form1_form	Form1_form	TRUE
Form2_report	Form2_report	TRUE
Form1_form	Form1_form	TRUE
Form2_report	Form2_report	TRUE
Email	Email	TRUE
Email	Email	TRUE
Email	Form1_form	FALSE
Email	Email	TRUE
Form2_report	Form2_report	TRUE
Form1_form	Form1_form	TRUE

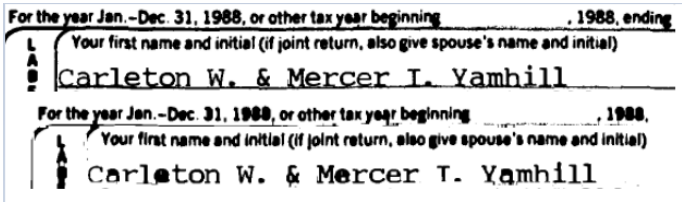
Tahap selanjutnya adalah melakukan *preprocesing* sebelum melakukan *document scanning*. Proses pertama yang dilakukan adalah transformasi gambar *form* menjadi format biner seperti Gambar 15. Format biner yang dimaksud adalah kondisi di mana sebuah image dikonversi menjadi gambar yang memiliki warna hitam dan putih saja, yang mana saat fitur diterjemahkan ke dalam *numpy array* hanya memiliki nilai 1 dan 0. Hal ini dikarenakan *OCR* akan lebih mudah membaca dokumen yang memiliki format biner dibandingkan *RGB* ataupun *Grayscale*.

Gambar 15 Transformasi biner.

Tahap berikutnya adalah melakukan *dilation*. Teknik ini digunakan untuk menghilangkan garis tepi yang ada pada *form* yang nantinya akan dilakukan *scanning* yang bisa dilihat pada Kode program 4. Hal ini diperlukan agar *character* yang ada di dalam gambar bisa terbaca meskipun sebelumnya terkena garis tepi. Contoh perbandingan antara gambar yang masih memiliki garis tepi dan tidak bisa dilihat pada Gambar 16.

```
# menghilangin garis horizontal
horizontal_kernel = cv2.getStructuringElement(cv2.MORPH
_RECT, (25,1))
detected_lines = cv2.morphologyEx(thresh, cv2.MORPH_OPE
N, horizontal_kernel, iterations=2)
cnts = cv2.findContours(detected_lines, cv2.RETR_EXTERN
AL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
for c in cnts:
    cv2.drawContours(image, [c], -1, (255,255,255), 2)

# repair kernel
repair_kernel = cv2.getStructuringElement(cv2.MORPH_REC
T, (1,6))
result = 255 - cv2.morphologyEx(255 -
 image, cv2.MORPH_CLOSE, repair_kernel, iterations=1)

# menghilangin garis vertikal
import numpy as np
vertical = np.copy(thresh)

rows = vertical.shape[0]
verticalsize = rows // 30

verticalStructure = cv2.getStructuringElement(cv2.MORPH
_RECT, (1, verticalsize))

vertical = cv2.erode(vertical, verticalStructure)
vertical = cv2.dilate(vertical, verticalStructure)

cv2_imshow(vertical)

cnts = cv2.findContours(vertical, cv2.RETR_EXTERNAL, cv
2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
for c in cnts:
    cv2.drawContours(image, [c], -1, (255,255,255), 2)
```

Kode program 4 Menghilangkan garis tepi.

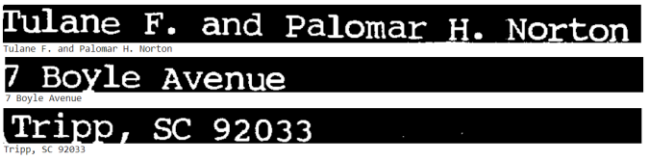


Gambar 16 Perbandingan gambar tanpa garis tepi.

Tahap selanjutnya adalah melakukan *cropping* untuk mengambil kolom id form, yang bisa dilihat pada Gambar 17. Meskipun pada tahapan ini scanning dokumen sudah dapat dilakukan, namun akan membutuhkan waktu yang lama untuk melakukan proses satu dokumennya dikarenakan ukuran pixel yang besar. Dengan dilakukan *cropping* maka akan mengurangi ukuran gambar dan meningkatkan waktu scan.

Gambar 17 *Crop* id *form.*

Sesudah kolom ID form berhasil di *scan* dan didapat value ASCII, maka sistem akan melakukan *scanning* detail kolom yang ingin di ambil informasinya, contoh detail informasi *form* yang di scan bisa dilihat pada Gambar 18.



Gambar 18 Detail informasi *form* yang di ambil.

Langkah akhir yang perlu dilakukan adalah melakukan transformasi seluruh data yang sudah di *scan* menjadi format *JSON* yang bisa dilihat pada Kode program 5, adapun format *JSON* yang dihasilkan bisa dilihat pada Gambar 19.

```
import json

data_set = {"form ID": extractedInformation, "nama": extracted
Information2,
            "alamat": extractedInformation3, "kota": extracted
Information4}

json_dump = json.dumps(data_set)

print(json_dump)
```

Kode program 5 Transformasi *JSON*.

```
{"form ID": "1040\n\f", "nama": "Tulane F. and Palomar H. Norton\n\f",
 "alamat": "7 Boyle Avenue\n\f", "kota": "Tripp, SC 92033\n\f"}
```

Gambar 19 Format JSON yang dihasilkan.

V. SIMPULAN

Sesuai dengan hasil pada Tabel IV mengenai prediksi kategori *form* dan Gambar 19 untuk format *JSON* yang digunakan, dapat disimpulkan jika sistem yang dibuat pada studi ini berhasil untuk melakukan pengkategorian *form* dan *scanning* dokumen. Pengguna tidak perlu lagi melakukan pemindahan dan pengkategorian *form* namun lebih kepada melakukan analisis dan perbaikan jika ada proses prediksi ataupun hasil *scanning* yang tidak sesuai. Sistem ini bukan produk final yang bisa langsung digunakan oleh perusahaan *textile*, perlu dilakukan penyesuaian dan perencanaan yang baik untuk dapat mengaplikasikan *blueprint* ini menjadi sebuah sistem yang utuh untuk suatu perusahaan.

DAFTAR PUSTAKA

[1] Wichian Premchaiswadi, Phaisarn, Nuchree, “The Fast Scheme for Document Page Segmentation in OCR using Window and Optimum Image,” *International Journal of Advanced Research in Computer Science.*, vol. 35, pp. 350-355, Feb. 2006.

[2] Mosalam K, Gao Y, "Deep Transfer Learning for Image-Based," *Journal of Computer-Aided Civil and Infrastructure.*, vol. 22, pp. 233-239, Jan. 2018.

[3] Diego R Faria, Jordan J Bird, "A Study on CNN Transfer Learning for Image Classification," conference:UKCI, 2018, paper 37, p. 125.

[4] Tridib Chakraborty, Suparna Karmakar Chowdhury Md Mizan, "Text Recognition using image processing," *International Journal of Advanced Research in Computer Science*., vol. 8, pp. 765-768, June. 2017.

[5] Apurva A Desai, "Gujarati handwritten numeral optical character reorganization through neural network, Pattern Recognition," *Pattern Recognition*., vol. 43, pp. 2582-2589, Aug. 2014.

[6] Jianxin Wu, Convolutional neural networks manual, 1st ed., Wen Tau Yih. Tianjin, China: Lei Tao, 2020.

[7] Jeff Donahue, Trevor Darrell, Jitendra Malik Ross Girshick, "Region-based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*., vol. 38, pp. 1, Dec. 2015.

[8] Bradski. (2009) OpenCV, Open source Computer Vision library. [Online]. Tersedia: http://opencv.willowgarage.com/wiki/.

[9] Mert Onulrapl Gokalp, Kerem Kayabay, Mehmet Ali Akyol, P.Erhan Eren, Altan Kocyigit, “Big Data For Industry 4.0: A Conceptual Framework,” *International Journal on Computational Science and Computational Intelligence*., vol. 10, pp. 431-434, Dec. 2016.

[10] Patricio Cerda, Gael Varoquaux, Balazs Kegl, “Similarity encoding for learning with dirty categorical variables,” *Journal of Machine Learning Research*., vol. 107, pp. 634-638, Sept. 2018.

[11] Patric Wspanialy, Justin Brooks, Medhat Moussa, “An Image Labeling Tool and Agricultural Dataset for Deep Learning,” *Journal of Machine Learning Research*., vol. 112, pp. 300-304, Apr. 2020.

[12] Manohar Karki, Qun Liu, Robert DiBiano, Saikat Basu, Supratik Mukhopadhyay, “Pixel-level Reconstruction and Classification for Noisy Handwritten Bangla Characters,” *International Journal on Computational Science and Computational Intelligence*., vol. 14, pp. 511-516, Aug. 2018.

[13] Liu Liu, Kaile Liu, Zhengchai Cong, Jiali Zhao, Yefei Ji, Jun He, “Long Length Document Classification by Local Convolutional Feature Aggregation,” *Algoritms.*, vol. 11, pp. 109, July. 2018.

[14] Sukhvinder Singh, Surender Kumar Grewal, “Text Extraction and Character Recognition form Image using Mathematical Morphology and OCR technique,” *International Journal of Science and Research.*, vol. 7, pp. 952-955, June. 2014.

[15] Rifiana Arief, Achmad Benny Mutiara, Tubagus Maulana Kususma, Hustinawaty, “Automated Extraction of Large Scale Scanned Document Images using Google Vision OCR in Apache Hadoop Environment,” *International Journal of Advanced Computer Science and Application.*, vol. 9, pp. 112-116, Jan. 2018.