IDENTIFIKASI KOMPONEN *GUI* PADA *PROTOTIPE* APLIKASI *MOBILE*

Mungki Astiningrum[1], Arie Rachmad Syulistyo[2], Aura Kanza Caesaria[3]

[1,2,3] Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang
[1]mungki.astiningrum@polinema.ac.id, [2]arie.rachmad.s@polinema.ac.id, [3]aurakanzaaa@gmail.com

Abstrak

OS *Android* merupakan OS *Mobile* terpopuler sejumlah 71,6% disusul IOS, dll. Hal ini menjadi peluang besar untuk menjadi *android developer*. Langkah awal yang harus dipersiapkan adalah memiliki pengetahuan dasar *Java* dan XML. Proses yang harus disiapkan pertama kali adalah pembuatan *User Interface* aplikasi. Namun, sebagai pemula sering kali kebingungan dalam menentukan komponen yang harus digunakan dalam pembuatan *interface.* Oleh karena itu diperlukan sebuah sistem yang dapat mengenali komponen yang akan digunakan.

Perkembangan ilmu pengetahuan dan teknologi pengolahan citra digital memungkinkan untuk mengklasifikasi komponen GUI secara otomatis dengan bantuan aplikasi pengolahan citra. Dalam pengenalan komponen ini digunakan metode *Convolutional Neural Network* (CNN). Metode ini dapat mengenali komponen pada sebuah *image*. Dengan adanya aplikasi ini dapat membantu pengguna dalam menentukan komponen yang akan digunakanan untuk membuat tampilan suatu aplikasi. Tingkat keberhasilan klasifikasi komponen GUI yang didapatkan menggunakan metode *Convolutional Neural Network* adalah 85.32%. Sehingga dapat disimpulkan bahwa identifikasi komponen GUI pada prototipe aplikasi *mobile* sudah sesuai**.**

Kata kunci : ***Image Classification, Convolutional Neural Network, GUI***

1. Pendahuluan

Perkembangan *smartphone* saat ini tumbuh sangat pesat, dengan munculnya berbagai inovasi teknologi dan berbagai macam produsen serta didukung oleh para pengembang aplikasi yang dapat meningkatkan kualitas serta kuantitas *smartphone* [1]. Ada berbagai macam sistem operasi yang tercipta hingga saat ini sistem operasi yang dikenal saat ini adalah *Android* dan *IOS.* Berdasarkan *wearesocial*.com saat ini *OS Android* merupakan *OS Mobile* terpopuler di dunia dengan jumlah 71,6% disusul *IOS* dan *OS mobile* lainnya pada tahun 2017. Di Indonesia sendiri pada tahun 2017 pengguna *Smartphone Android* jumlahnya hampir 80%. Tingginya pengguna *Smartphone Android* tentu saja *software* atau aplikasi sangat berperan dalam kegunaan *smartphone.*

Hal ini menjadi sebuah peluang besar untuk menjadi *Android Developer*. Agar dapat membuat sebuah aplikasi *Android* haruslah kita memiliki pengetahuan dasar Java dan XML. Pengetahuan ini sangat penting karena Bahasa Java dianggap sebagai Bahasa pemrograman dasar untuk pengembangan *android.* Sedangkan XML adalah Bahasa *markup* yang banyak digunakan dalam membuat tampilan suatu aplikasi. Dengan mempelajari sintak dasar XML akan membantu dan mempermudah dalam pengembangan aplikasi *android* seperti merancang *layout* tata letak antarmuka pengguna (UI) dan *parsing* data dari internet. Untuk membuat tampilan kita harus memahami aturan dasar dan bagaimana cara menulis XML dan cara membacanya.

Sebagai pemula seringkali kebingungan menentukan langkah awal yang harus dilakukan untuk membuat sebuah tampilan dan bagaimana cara membuatnya. Ingin membuat tampilan sederhana namun tidak tahu komponen yang harus digunakan didalamnya. Oleh karena itu diperlukan sebuah sistem yang dapat membantu proses pembelajaran dalam menentukan komponen yang akan digunakan dengan menggunakan pengolahan citra digital. Sistem dapat mengenali komponen yang terdapat dalam *screenshot* halaman aplikasi. Penulis mencoba mengangkat suatu ide dimana diperlukan metode untuk menganalisa objek dari inputan berupa potongan *screenshot* aplikasi untuk mengenali komponen apa saja yang terdapat didalamnya dan memberikan informasi pembuatan komponen berupa kode program. Untuk mengenali komponen dalam *screenshot* aplikasi, penulis mencoba menggunakan metode *Convolutional Neural Network* (CNN). Metode ini dapat mendeteksi dan mengenali objek pada sebuah *image* lebih baik karena dapat menciptakan suatu pola atau kemampuan belajar (*self organizing*). Dengan melakukan identifikasi komponen GUI menggunakan CNN diharapkan dapat membantu pengguna dalam menentukan

komponen yang akan digunakan untuk membuat tampilan suatu aplikasi.

2. Tinjauan Pustaka

2.1 Pengolahan Citra Digital (*Image Processing*)

Sebuah gambar disebut dengan citra digital apabila gambar yang dihasilkan dari proses sebuah komputer, kamera, *scanner* atau perangkat elektronik lainnya. Pengolahan citra digital diproses oleh komputer dengan menggunakan algoritma. Citra digital direpresentasikan dengan matriks, sehingga pengolahan pada citra digital pada dasarnya memanipulasi elemen-elemen matriks. Operasi yang dapat dilakukan pada sebuah citra ada berbagai macam, diantaranya operasi titik, operasi global, operasi berbasis bingkai, operasi geometri dan operasi bertetangga [2]

2.2 Prototipe

Prototipe dapat di klik dan ketika anda mengklik, anda mendapat respon. *Prototipe* yang dapat di klik mensimulasikan bagaimana pengguna berinteraksi dengan antarmuka UI dengan cara yang nyata, meningkatkan efektivitas komunikasi anda. Ini memungkinkan desainer untuk menguji alur dan menemukan masalah terhadap interaksi pada tahap awal [3]

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN/ConvNet) adalah salah satu algoritma dari *deep learning* yang merupakan pengembangan dari *Multilayer Percepton* (MLP) yang dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya gambar atau suara [4]

- *Convolution Layer*

Convolutional Layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*). Sebagai contoh layer pertama pada *feature extraction layer* biasanya adalah conv. *Layers* dengan ukuran 5x5x3. Panjang 5 *pixels*, Tinggi 5 *pixels* dan tebal 3 buah sesuai dengan *channel* dari *image* tersebut.
Untuk menghitung banyaknya neuron aktivasi dalam sekali *output* adalah sebagai berikut :

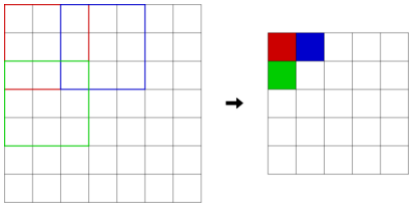
$$output = \frac{W-N+2P}{S} + 1 \quad (1)$$

Keterangan :

W = Panjang/ Tinggi Input
N = Panjang/ Tinggi Filter
P = *zero padding*
S = *stride*

- *Stride*

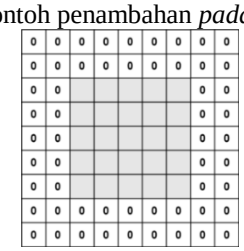
Stride adalah jumlah piksel yang bergeser di atas matriks *input*. Ketika *stride* 1 maka filter akan bergeser sejumlah 1 piksel. Ketika *stride* 2 maka piksel akan bergeser 2 piksel setelah matriks *input* seara *vertical* dan *horizontal* [5]. Dapat dilihat pada Gambar 1 yang merupakan ilustrasi pergeseran *stride*:



Gambar 1 Contoh pergeseran *stride*

- *Padding*

Padding mengacu pada proses menambahkan angka nol secara simetris ke matriks *input*. Cara ini merupakan cara umum yang digunakan untuk mempertahankan ukuran spasial dari volume *input* sehingga *input* dan *output* ukurannya sama. Gambar 2 ini adalah contoh penambahan *padding* 2:



Gambar 2 *zero-pading* 5x5 menjadi 9x9

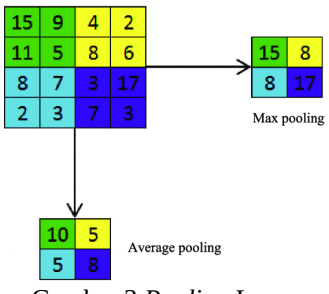
- *ReLU* (*Rectified Linear Units*)

Rectified Linear Unit adalah fungsi aktivasi yang paling umum digunakan dalam *deep learning*. Fungsinya mengembalikan nilai 0 jika menerima *input negative*, tetapi setiap nilai positif nilai x akan dikembalikan ke nilai aktivasi itu sendiri [6]. Jadi dapat ditulis dengan rumus berikut:

$$f(x) = \max(0, x) \quad (2)$$

- *Pooling Layer*

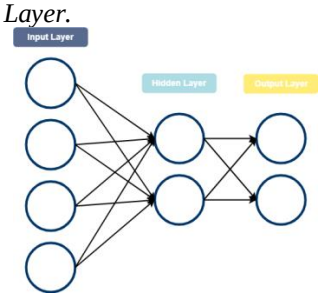
Pooling layer biasanya berada setelah *conv layer*. Pada prinsipnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh jika kita menggunakan *Max Pooling* 2x2 dengan *stride* 2, maka pada setiap pergeseran *filter*, nilai *maximum* pada area 2x2 *pixel* tersebut yang akan dipilih, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Gambar 3 merupakan contoh gambar operasi *max* dan *average pooling*:



Gambar 3 *Pooling* Layer

- *Fully Connected Layer*

Lapisan *fully connected* adalah lapisan di mana semua *neuron* aktivasi dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan. Dapat dilihat Gambar 4 merupakan proses dari *Fully Connected Layer.*



Gambar 4 Proses dari *Fully Connected Layer*

- Fungsi Aktivasi *Softmax*

Fungsi aktivasi *softmax* digunakan untuk mendapatkan hasil klasifikasi. Fungsi aktivasi menghasilkan nilai yang diinterpretasi sebagai probabilitas yang belum dinormalisasi untuk tiap kelas. Nilai kelas dihitung dengan menggunakan fungsi *softmax* [7]

- *Forward Propagation*

Proses *forward propagation* pada jaringan CNN dilakukan untuk meneruskan nilai pada lapisan masukan hingga pada lapisan keluaran. Nilai ini diteruskan melalui lapisan konvolusi, *subsampling* dan lapisan *fully connected* sesuai dengan urutan lapisan tersebut ditempatkan pada jaringan yang digunakan.

- *Backward Propagation*

Proses untuk memperbaharui nilai *filter* dan bobot pada jaringan adalah proes propagasi balik. Perhitungan perubahan nilai bobot dihitung dimulai dari lapisan *fully connected.* Pada lapisan ini perubahan bobot dicari dengan mencari derivatif *loss function* terhadap bobot [8].

- *Confusion* Matriks

Confusion matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion* matriks mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya [9].

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P+N$

Gambar 5 *Confusion* Matriks

Keterangan:

TP : True Positives
TN : True Neggatives
FN : False Negatives
FP : False Positives

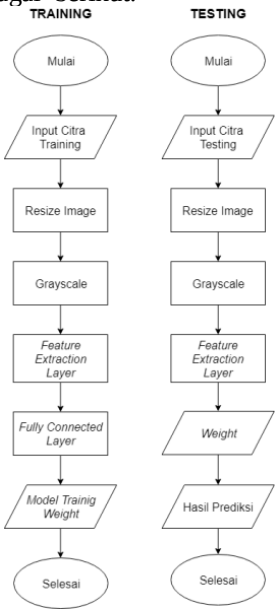
3. Metodologi

3.1 Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini adalah pengumpulan data penunjang yang dilakukan dengan pengambilan data-data dari dataset ReDraw[1]. Dataset ReDraw berisi lebih dari 14.382 dasain aplikasi Android yang mencakup 15 komponen GUI. ReDraww: *Machine Learning* yang berbasis prototyping dari GUI untuk *mobile apps.* Dataset ini dapat diperoleh melalui *website* https://www.android-dev-tools.com/redraw. Dari dataset yang digunakan, data citra ini kemudian dibagi menjadi data *training, validation* dan *testing* untuk keperluan pembuatan model dan pengecekan model yang terbentuk/ validasi model. Pembagian data menjadi data *training, validation* dan *testing* menggunakan *scenario* 75% untuk data *training*, 15% untuk data *validation* dan 10% untuk *testing* seperti penelitian pada umumnya.

3.2 Metode Pengolahan Data

Metode pengolahan data merupakan kerangka penelitian yang telah dikemukakan oleh penulis pada pendahuluan maka penelitian ini menggunakan kerangka sebagai berikut:



Gambar 6 Rancangan Proses *Training* dan *Testing*

Gambar 6 merupakan rancangan proses yang terdapat dalam sistem. Proses *training* dilakukan untuk mendapatkan nilai bobot unggul yang nantinya akan diuji pada proses *testing.* Sedangkan

proses *testing* dilakukan untuk menguji sistem dengan nilai bobot unggul yang paling unggul dari proses *testing.* Dimana data siap di cek akurasi ketepatannya setelah dilakukan *training.*

3.3 Metode Pengujian

Dalam metode pengujian terdapat 2 pengujian yang dilakukan yaitu:

- Pengujian Unit
  Pengujian unit ditujukan untuk memastikan setiap fungsi berjalan sesuai ketentuam
- Pengujian Akurasi
  Pengujian akurasi perhitungan diperlukan untuk menguji tujuan utama dari penelitian ini bahwa dengan menerapkan metode *Convolutional Neural Network* (CNN) sebagai proses identifikasi komponen GUI.

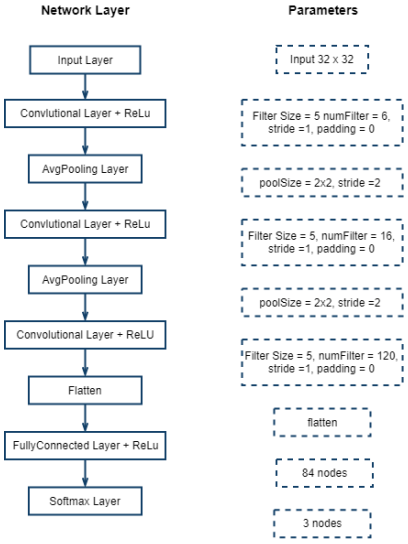
4. Perancangan

4.1 Deskripsi Sistem

Aplikasi identifikasi komponen GUI merupakan sebuah peneitian yang akan dikembangkan menggunakan metode *Convolutional Neural Network.* Terdapat dua proses penting untuk mengidentifikasi komponen GUI yaitu: proses *training* dan proses *testing.* Proses *training* digunakn untuk memperoleh bobot (*weight*) yang nantinya akan dijadikan model pada proses *testing*. Sedangkan proses *testing* digunakan untuk mengindentifikasikan citra berdasarkan model yang telah di training sebelumnya. Pada proses testing user menginputkan citra yang diinginkan untuk di identifikasi.

4.2 Analisis Sistem

Dalam algoritma *Convolutional Neural Network* (CNN) pembentukan arsitektur jaringan dapat mempengaruhi hasil dari akurasi model.



Gambar 7 Arsitektur CNN

Gambar 7 merupakan arsitektur jaringan pada proses *training* untuk menghasilkan model pada penelitian ini. Penelitian ini menggunakan *input* gambar dengan ukuran 32x32x1. Sehingga proses *training* didapatkan model dari arsitektur tersebut. Gambar 8 Model CNN merupakan model yang terbentuk:

no	Nama	Size			Parameter		
0	Input	32*32*1					0
1	Conv2d_1	((32-5+(2*0))/1)+1	=	28*28*6	((5*5)+1)*6	=	156
2	MaxPool_1			14*14*6			0
3	Conv2d_2	((14-5+(2*0))/1)+1	=	10*10*16	((5*5*6)+1)*16	=	2416
4	MaxPool_2			5*5*16			0
5	Conv2d_3	((5-5+(2*0))/1)+1	=	1*1*120	((5*5*16)+1)*120	=	48120
6	Flatten			120			0
7	Dense			84	(120*84)+84	=	10164
8	Output			3	(84+1)*3	=	255
Total							61111

Gambar 8 Model CNN

5. Implementasi dan Pengujian

Melakukan *testing* pada sistem yang sudah dibuat. Tahap ini dibutuhkan sebagai ukuran bahwa sistem dapat berjalan sesuai tujuan atau bekum. Terdapat beberapa bentuk pengujian dalam penelitian ini, yaitu:

5.1 Pengujian Sistem

Pengujian sistem digunakan untuk melihat sistem telah berjalan dengan sesuai atau belum. Pengujian ini dilakukan dengan menguji fitu-fitur dari aplikasi yang dirancang, mulai dari halaman home, pengujian testing input gambar sampai output diperoleh, dan halaman about.

5.2 Pengujian Akurasi

Pengujian akurasi dilakukan untuk menguji seberapa besar sistem dapat identifikasi komponen GUI dengan menggunakan metode *Convolutional Neural Network.* Pada pengujian ini menggunakan sebanyak 136 sampel komponen GUI yang terdiri dari 50 *ImageButton*, 50 *TextView*, dan 36 *SwitchButton.* Hasil *confusion matriks* adalah sebagai berikut:

Tabel 1. *Confusion Matriks*

Matriks		Pred Class		
		Image Button	Text View	Switch Button
Act Class	Image Button	167	31	2
	Text View	17	178	5
	Switch Button	1	8	27

Berdasarkan Tabel 1. Daiatas merupakan hasil prediksi dari model terhadap data *testing* menunjukkan hasil yang cukup baik. Perhitungan akurasi dari keseluruhan matriks diatas adalah sebagai berikut:

$$akurasi = \frac{prediksi\ benar}{jumlah\ dataset} \times 100\%$$

$$akurasi = \frac{371}{436} \times 100\% = 85{,}32\%$$

Jadi akurasi yang dihasilkan oleh model dengan input citra 32x32 pixel, learning rate 0,0001, epoch 1000 didapatkan nilai akurasi sebesar 85,32%

6. Hasil dan Pembahasan

Pada penelitian ini, peneliti melakukan klasifikasi tiga kelas komponen GUI, yaitu *ImageButton, TextView,* dan *SwitchButton* dengan menggunakan algoritma *Convolutional Neural Network* (CNN) terhadap 3242 dataset dengan perbandingan 75% training, 15% validation dan 10% testing didapatkan hasil akurasi sebesar 85,32%. Untuk mengetahui faktor-faktor yang mempengaruhi kinerja dari metode CNN, maka dilakukan pengujian dengan menggunakan parameter yang berbeda-beda. Adapun parameter yang di ubah dalam proses pengujian antara lain: Jumlah layer pada proses *feature extraction layer,* jumlah *epoch, pooling layer* dan *learning rate*.

6.1 Pengujian berdasarkan Jumlah Layer

Jumlah layer juga dapat mempengaruhi hasil dari akurasi proses *training* dan *testing*.

Tabel 2 Hasil Pengujian berdasar Jumlah Layer

No	Layer	Accuracy	LossVal
1.	28x28x6 Konv + AvgPool	45.54%	8.7782
2.	28x28x6 Konv + AvgPool + 10x10x16+ A vgPool	45.54%	8.7551
3.	28x28x6 Konv + AvgPool + 10x10x16+ A vgPool + 1x1x120 Konv	85,07%	0.6906

Berdasarkan Tabel 2 dapat dilihat semakin banyak jumlah layer maka semakin banyak informasi yang didapatkan untuk meningkatkan akurasinya. Namun, semakin banyak *layer* yang digunakan maka membutuhkan waktu yang lebih lama untuk melakukan proses *training* maupun *testing*.

6.2 Pengujian berdasar Jumlah *Epoch*

Epoch adalah ketika seluruh dataset sudah melalui proses training pada *Neural Network* sampai dikembalikan ke awal dalam satu putaran. Untuk mempermudah dan mempercepat proses *training dataset* dibagi per *batch* (*batch size*). Pada penelitian ini menggunakan *batch size* sebesar 100, dimana seluruh jumlah *dataset* akan dibagi dengan ukuran *batch size.* Tabel 3 berikut adalah hasil perbandingan *epoch* dari hasil *training.*

Tabel 3 Akurasi berdasarkan *Epoch*

Epoch	*Accuracy Validation*	*Loss Validation*	*Accuracy Test*	*Loss Test*
100	62.31%	0.8993	57.93%	1.0611
300	82.77%	0.5327	74.71%	0.7353
500	84.77%	0.4539	79.54%	0.6242
700	82.92%	0.4663	80.23%	0.5933
1000	85.08%	0.6906	85.32%	0.4603

6.3 Pengujian *Pooling Layer*

Pooling Layer merupakan proses pengurangan ukuran matriks dari hasil proses konvolusi. Proses ini bertujuan untuuk mengurangi nilai parameter sehingga mengendalikan *overfitting* pada proses *training model*.

Tabel 4 Akurasi berdasarkan *Pooling*

Pooling layer	*Accuracy validation*	*Loss validation*
Max Pooling	46.15%	8.6559
Avg Pooling	85.08%	0.4614

Pada Tabel 4 diatas, merupakan percobaan terhadap *maxPooling* dan *averagePooling*. Hasil *averagePooling* dapat dilihat lebih baik daripada *maxpooling.*

6.4 Pengujian Nilai Learning Rate

Penentuan nilai dari *learning rate* sangat berpengaruh padaa performa hasil. Hasil *learning rate* adalah sebagai berikut:

Tabel 5 *Learning Rate*

Learning rate	Accuracy validation	Loss validation
0.01	45.69%	8.7574
0.001	83.85%	0.6124
0.0001	85.23%	0.4831

Berdasarkan Tabel 5 dapat dilihat *learning rate* 0,0001 menunjukkan hasil yang cukup baik dengan *accuracy validation* sebesar 85.08% dan *accuracy test* sebesar 85.32%. Penentuan nilai *learning rate* yang digunakan pada penelitian ini adalah *trial and error*, sehingga tidak dapat secara langsung menetukan nilai *learning rate* yang paling optimum.

7. Kesimpulan

Berdasarkan hasil analisis yang telah dilakukam, diperoleh beberapa kesimpulan yaitu:

a. Model CNN yang digunakan pada penelitian ini menggunakan citra berukuran 32x32, ukuran *filter* 5x5, dengan menggunakan *hyperparameter learning rate* 0.0001, *batch size* sebesar 100 dan *epoch* sebesar 1000. Pembagian data *training* yang digunakan adalah 3242, *validation* 650 dan *testing* 436. Pengujian menghasilkan tingkat akurasi *testing* dalam melakukan klasifikasi gambar komponen GUI sebesar 85.32%. Tingkat ketepatan untuk mengenali komponen GUI pada prototipe aplikasi *mobile* ini tergantung dari arsitektur dan *hyperparameter* yang digunakan untuk mendapatkan hasil yang baik.

Daftar Pustakan

[1] N. M. Juniver V Mokalu, "Dampak Teknologi Smartphone Terhadap Perilaku Orang Tua di Desa Touure Kecamatan Tompaso," *ActaDiurna,* vol. 5.1, p. web, 2016.

[2] A'ala, "Deteksi Retak Permukaan Jalan Raya Berbasis Pengolahan Citra Menggunakan Metode

Ekstaksi Ciri Wavelet," Program Studi Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Yogyakarta, Yogyakarta, 2016.

[3] MockingBot, "What's difference between Wireframe, Prototype & Mockup?," Medium, 12 September 2016. [Online]. Available: https://medium.com/mockingbot/whats-the-difference-between-wireframe-prototype-mockup-17615f77938f. [Accessed 16 May 2019].

[4] S. Sena, "Pengenalan Deep Learning Part7: Convolutional Neural Network (CNN)," Medium, 13 November 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deeplearning-part-7-convolutionalneural-network-cnn-b003b477dc94. [Accessed 28 November 2018].

[5] Prabhu, "Understanding of Convolutional Neural Network (CNN)—Deep Learning," Medium, 4 March 2018. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed 20 April 2019].

[6] DanB, "Rectified Linear Units (ReLU) in Deep Learning," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. [Accessed 22 Mei 2019].

[7] K. L. Andrea Vedaldi, "MatConvNet: Convolutional Neural Networks for MATLAB.," 2016.

[8] Z. Zhang, "Derivation of Backpropagation in Convolutional Neural Network (CNN)," University of Tennessee, Knoxvill, TN, 2016.

[9] P. E, Data Mining: Konsep dan Aplikasi menggunakan Matlab, 1 ed, Yogyakarta: Andi Offset, 2012.