Sistem Manajemen Inventori Dengan Pengenalan Barang Secara Otomatis Menggunakan Metode Convolutional Neural Network

Ferbian Loekman [1], Lina [2*]

[1] Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Tarumanagara, DKI Jakarta
[2] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara, DKI Jakarta
Email: [1] ferbian.825190005@stu.untar.ac.id, [2*] lina@fti.untar.ac.id

(Naskah masuk: 16 Jan 2023, direvisi: 20 Feb 2023, diterima: 23 Feb 2023)

Abstrak

Saat ini teknologi *barcode* masih luas penggunaannya untuk mendata stok barang. Namun pada faktanya *barcode* juga memiliki kelemahan. Misal, *barcode* rentan mengalami kerusakan sehingga data di dalamnya sulit terbaca oleh *scanner*. Selain itu, *barcode* juga hanya bisa di-*scan* pada jarak tertentu, serta letaknya yang berbeda-beda di setiap produk juga membuat *user* harus mencari letak *barcode* terlebih dahulu sebelum di-*scan*. Teknologi RFID yang ditawarkan untuk menjadi jalan keluar dari permasalahan pada teknologi *barcode* juga memiliki beberapa hambatan untuk penerapannya, salah satu contohnya adalah biaya yang mahal. Pada penelitian ini dibuat sebuah sistem manajemen inventori berbasis *website* menggunakan HTML, CSS, dan PHP. Hasil *black box testing* fungsionalitas *web* menunjukkan hasil yang sangat baik, tingkat keberhasilannya mencapai 93,94%. Teknologi *computer vision* khususnya *object recognition* yang menggunakan arsitektur ResNet dalam CNN juga diterapkan untuk mengenali barang melalui *input*-an citra objek secara otomatis. Setelah melakukan *training* data terhadap sepuluh kelas yang sudah ditentukan, didapatkan sebuah model dengan *validation loss* sebesar 1.0834e-04 dan *validation accuracy* mencapai 100%. Berdasarkan *testing* yang dilakukan, model sudah mampu mengenali satu objek dalam satu *frame* foto dengan tingkat akurasi mencapai 90%. Namun akurasinya menurun untuk skenario *testing* dengan dua dan lima objek dalam satu foto, sehingga mendapatkan hasil tingkat akurasi 56% dan 54%.

Kata Kunci: Sistem Manajemen Inventori, *Website*, *Object Recognition*, CNN, *ResNet Architecture*.

Inventory Management System with Automatic Recognition of Goods Using the Convolutional Neural Network Method

Abstract

Currently, barcode technology is still widely used to record stock items. But in fact, the barcode also has weaknesses. For example, a barcode is prone to damage so the data in it is difficult to read by a scanner. Apart from that, the barcode can only be scanned at a certain distance, and its location varies for each product, which also makes the user have to find the location of the barcode first before being scanned. The RFID technology offered to be a way out of problems in barcode technology also has several obstacles to its application, one example is the high cost. In this study, a website-based inventory management system was created using HTML, CSS, and PHP. The results of black box testing of web functionality show very good results, the success rate reaches 93.94%. In this study, computer vision technology especially objects recognition that uses the ResNet architecture in CNN is also applied to recognize goods through automatic object image input. After training the data for the ten classes that have been determined, a model with validation loss of 1.0834e-04 is obtained and the validation accuracy reaches 100%. Based on the testing carried out, the model can recognize one object in one photo frame with an accuracy rate of up to 90%. However, the accuracy decreases for testing scenarios with two and five objects in one photo, resulting in an accuracy rate of 56% and 54%.

Keywords: *Inventory Management System, Website, Object Recognition, CNN, ResNet Architecture*

I. PENDAHULUAN

Penggunaan teknologi dalam kehidupan sehari – hari sudah menjadi keniscayaan, situasi pandemi Covid-19 yang masih sangat dinamis sampai saat ini juga ikut mendorong transformasi digital dengan sangat cepat dalam segala sektor. Salah satu sektor yang ikut terdampak adalah sektor bisnis, baik bisnis berskala besar, maupun bisnis yang masih dijalankan oleh individu atau perorangan [1].

Gudang merupakan hal yang sangat penting dan tidak bisa dipisahkan dari dunia bisnis, terutama bisnis yang menjual barang-barang industri [2]. Oleh sebab itu, diperlukan sistem yang mengatur proses dalam pergudangan yang baik bagi para pelaku bisnis, agar dapat membantu proses bisnis bisa berjalan lebih baik. Pergudangan atau *warehousing* merupakan salah satu kunci utama dari berjalannya rantai pasok sebuah bisnis, operasional logistik dapat dijalankan dengan efisien dan biayanya dapat ditekan seminimal mungkin apabila sistem manajemen gudang sudah diterapkan dengan metode yang tepat [3]. *Warehouse Management System* (WMS) terdiri dari banyak kegiatan, jika dijabarkan satu persatu WMS meliputi seluruh kegiatan pergudangan mulai dari penerimaan stok, penyimpanan stok, mengatur barang keluar (sesuai pesanan), pengambilan barang, *packaging* barang, dan pengiriman barang. Selain kegiatan yang sudah disebutkan, masih ada satu kegiatan yang juga termasuk ke dalam *warehouse management* yaitu manajemen persediaan atau *inventory management* yang lebih berfokus pada mengontrol ketersediaan barang. Beberapa contoh kegiatan dalam *inventory management* misalnya, mengatur dan memperhitungkan berapa banyak barang yang harus distok (disesuaikan dengan kebutuhan pembeli, tidak boleh kekurangan ataupun berlebih), memperhatikan posisi letak beserta cara penyimpanan barang yang harus disesuaikan dengan karakteristik masing-masing barang agar tetap terjaga kualitasnya, dan memperhitungkan berapa banyak barang yang harus dipesan [4].

Berbagai macam perancangan *Inventory Management System* (IMS) sudah dilakukan oleh para peneliti untuk mencari metode yang paling tepat. Hasilnya pun beragam, misalnya inovasi yang menerapkan teknologi *barcode* atau *QR-Code*. Contohnya seperti pada penelitian yang telah dilakukan oleh Syam, M. L. pada tahun 2022 dengan judul Sistem Informasi Stok Barang Menggunakan *QR-Code* Berbasis Android [5]. Sistem yang dikembangkan telah berhasil untuk membantu admin gudang dalam menjalankan tugasnya karena pengecekan stok dapat dilakukan lebih efektif dan efisien.

Di Indonesia pengaplikasian teknologi *barcode* atau *QR-Code* sering kita jumpai pada saat berbelanja, misalnya di *supermarket*. Teknologi ini memiliki banyak kelebihan pada saat proses *checkout* barang di meja kasir jika dibandingkan dengan pencatatan manual. Namun, terdapat beberapa kekurangan yang ditemukan pada teknologi *barcode* seperti kapasitasnya yang terbatas untuk menyimpan data [6]. Permasalahan *barcode* yang sulit terbaca pada produk yang basah, letaknya yang sulit ditemukan dan adanya kemungkinan *barcode* mengalami kerusakan atau hilang sehingga tidak bisa dibaca *scanner* juga membuat antrean pada meja kasir menjadi tambah panjang, dan pembeli harus menunggu lebih lama [7].

Untuk menjawab kekurangan dari teknologi *barcode*, digunakanlah teknologi *radio-frequency identification* (RFID) untuk melakukan pencatatan barang masuk, barang keluar, dan memonitor sisa stok barang dalam gudang [8]. Penerapan teknologi RFID dalam WMS dinilai lebih efektif karena RFID mampu membaca data dengan jarak sekitar 10 meter, jarak yang sangat jauh jika dibandingkan *barcode*. Seiring penggunaan teknologi RFID, ditemukan juga beberapa kelemahan. Kelemahan utama dari RFID adalah mahalnya biaya yang dibutuhkan untuk pembelian komponen RFID dan penerapannya yang sulit [9].

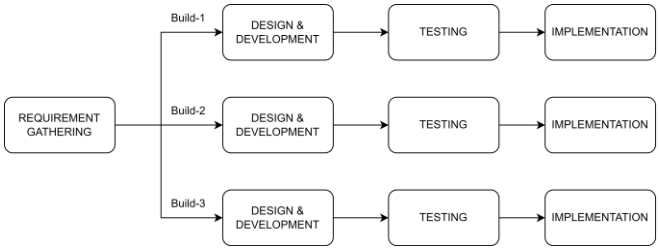
Inovasi terus dilakukan sampai pada akhirnya mulai diterapkannya teknologi *computer vision* seperti yang sudah dilakukan oleh Ardiansyah, M. N. *et al*. pada tahun 2021 [10]. Metode *Convolutional Neural Network* (CNN) sendiri telah digunakan untuk berbagai aplikasi pengenalan, diantaranya untuk pengenalan terhadap data medis [11][12][13], pengenalan wajah dan emosi manusia [14], serta aplikasi umum lainnya [15][16]. Pada penelitian ini dilakukan pengembangan sebuah *Inventory Management System* (IMS) dengan menggunakan metode CNN untuk proses pengenalan barangnya. Sistem yang dirancang akan berfokus pada aktivitas *receiving items* yang terdiri dari penambahan barang, *input* jumlah barang yang ditambahkan, pemeriksaan jumlah stok dan menampilkan tanggal kedaluwarsa barang. Diharapkan dari data jumlah stok dan tanggal kedaluwarsa yang ditampilkan dapat membantu *user* untuk mengambil keputusan. Fitur yang membedakan antara sistem yang dirancang dengan sistem yang sudah ada adalah sistem dapat mengidentifikasi nama barang yang dimasukkan secara otomatis menggunakan foto produk. Sehingga waktu yang dibutuhkan pada saat proses *receive items* di gudang diharapkan dapat berjalan lebih cepat karena produk tidak perlu lagi dipasang *barcode* dan di-*scan* satu persatu. Sehingga pengelolaan barang di gudang dapat dilakukan lebih efektif, efisien, *cost-effective*, dan tentunya juga akan meminimalkan *human error* di gudang.

Sistem yang dirancang bertujuan untuk melakukan *stock taking* (menghitung stok barang) dengan menggunakan teknologi *object recognition*. Penelitian yang dilakukan menggunakan *framework* YOLOv3, *dataset* yang digunakan dalam penelitian berasal dari gambar yang diambil sendiri menggunakan kamera ditambah dengan gambar yang diambil secara acak dari internet. Proses *testing* dilakukan dalam beberapa skenario yaitu, deteksi objek dengan mengubah sudut pandang, deteksi objek tunggal pada gambar objek jamak, deteksi multi objek, deteksi jumlah objek tunggal, dan deteksi jumlah objek jamak.

Sistematika penulisan makalah ini terdiri atas pendahuluan pada bab 1, metodologi penelitian pada bab 2, hasil dan pembahasan pada bab 3, serta kesimpulan dan saran pada bab 4.

II. METODOLOGI PENELITIAN

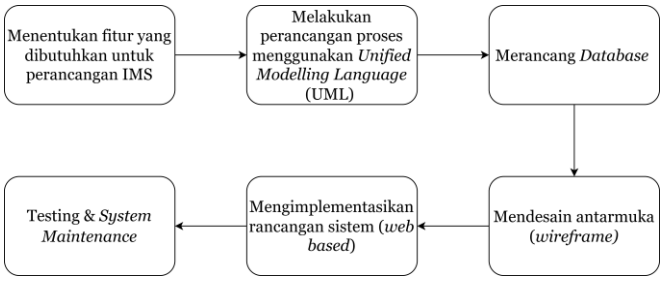
Metode yang digunakan dalam pembuatan sistem IMS dalam penelitian ini terdiri dari dua bagian, *Software Development Life Cycle* (SDLC) khususnya *incremental model* untuk pengembangan aplikasi berbasis *website*-nya. Sedangkan, metode CNN akan digunakan untuk *object recognition*-nya. Gambar 1 di bawah menampilkan ilustrasi tahapan membangun *software* dengan metode *incremental* dalam SDLC.



Gambar 1. *Incremental Model in* SDLC
Sumber: Singh & Kaur (2019)

A. Metodologi Pembuatan *Website*

Agar proses pembuatan *website* dapat berjalan secara sistematis, jelas, terarah dan terukur maka diperlukan alur tata laksana yang tetap berpedoman pada metode yang sudah ditentukan. Setidaknya terdapat enam tahapan yang sudah ditentukan, mulai dari penentuan fitur apa saja yang akan dibuat (*requirement*), perancangan sistem menggunakan *unified modelling language* (UML) dan perancangan *database*, *design* antarmuka dalam bentuk *wireframe*, implementasi rancangan (pembuatan program), *testing* dan *system maintenance* menggunakan *black box testing*. Tahapan dapat dilihat pada Gambar 2 di bawah.



Gambar 2. Tahapan Pembuatan *Website*

Berikut ini adalah penjabaran singkat dari masing–masing tahapan dalam pembuatan *website*:

a. Menentukan Fitur yang Dibutuhkan

Tujuan dasar sebuah sistem dirancang atau dibuat adalah untuk membantu menyelesaikan permasalahan *real*, sehingga sistem yang dibangun tidak menjadi sia-sia. Pada tahap ini peneliti menentukan fitur berdasarkan pengetahuan mengenai kendala yang sedang terjadi yang diperoleh dari observasi langsung dan dengan mencari serta membaca kekurangan dari penelitian terkini.

b. Perancangan Proses Menggunakan UML

Pada tahap ini akan dibuat gambaran visual alur dari sistem yang akan dibuat, selain mempermudah *developer* pada saat tahap pengembangan, UML juga dapat membantu *user* memahami sistem yang dibuat. Model desain yang akan digunakan adalah *use case diagram* dilengkapi dengan *use case scenario*, *activity diagram*, *sequence diagram* dan *class diagram*.

c. Perancangan *Database*

Perancangan *database* akan dilakukan bertahap mulai dari penentuan *entity*, menentukan relasi dan *multiplicity* antar *entity*, dan juga *attributes* yang ada di masing-masing *entity* menggunakan rancangan basis data konseptual. Sampai dengan digambarkannya ERD (*Entity Relationship Diagram*), dalam prosesnya ERD dapat mengalami perubahan disesuaikan dengan kebutuhan *user*.

d. Perancangan Desain Antarmuka (*Wireframe*)

Agar lebih mudah untuk membayang tampilan *website* yang akan dibuat, maka diperlukan ilustrasi kasar (*low fidelity) user interface* berupa *wireframe* menggunakan aplikasi *draw.io*. *Wireframe* juga akan membantu memberi informasi mengenai alur *website* secara sederhana.

e. Pengimplementasian Rancangan Sistem

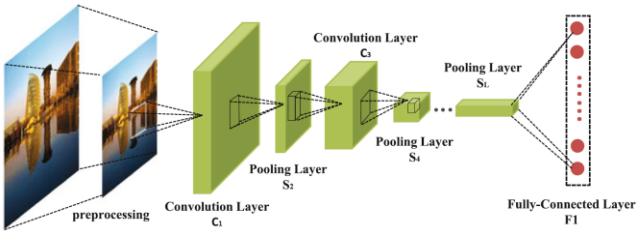
Semua perancangan yang sudah dilakukan akan digunakan pada tahap pengimplementasian rancangan sistem. Rancangan sistem akan dibuat berbasis *web* dengan menggunakan HTML, CSS, dan PHP. *Framework* Bootstrap juga digunakan agar *web* yang dibuat *responsive* serta dapat diselesaikan lebih cepat. Sedangkan untuk *database*-nya akan menggunakan MySQL melalui XAMPP sebagai *web server*.

f. *Testing* dan *System Maintenance*

Tahap terakhir setelah semua program selesai dibuat adalah melakukan *testing*. Metode *testing* yang akan digunakan adalah *black box testing*, lalu jika ditemukan *bugs* selama proses *testing* maka artinya sistem harus diperbaiki (*maintenance*). Skenario *testing* dibuat berdasarkan *requirement* awal dan alur serta semua fungsionalitas tombol yang ada juga diperiksa.

B. Metodologi *Object Recognition*

CNN merupakan salah satu algoritma dalam *deep learning* yang banyak digunakan untuk menyelesaikan permasalahan yang berkaitan dengan *computer vision*. Sejauh ini CNN telah menunjukkan performa dan hasil yang baik dalam pengolahan data berupa *image* atau citra pada penelitian, sehingga banyak peneliti yang menggunakan CNN sebagai metode dasar saat ingin melakukan pengelolaan citra [17]. Arsitektur dasar yang dimiliki CNN dapat dilihat pada Gambar 3.



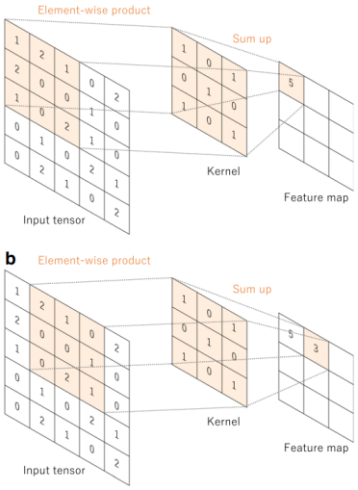
Gambar 3. Arsitektur Dasar CNN
Sumber: Xia, Z. (2019)

Algoritma CNN tersusun dari beberapa *building blocks* atau lapisan (*layer)* yang terdiri dari *convolutional layers*, *pooling layers*, dan *fully connected layers*. Pada tahapan awal, gambar yang di-*input* akan mengalami *feature extraction* pada dua lapisan pertama yakni *convolutional* dan *pooling layers.* Selanjutnya, pada lapisan ketiga (*fully connected layers)*, hasil dari *feature* yang sudah diekstraksi akan dipetakan (*maps)* menjadi *final output.* Perlu diketahui juga bahwa piksel gambar digital pada komputer, disimpan dalam bentuk himpunan angka 2 dimensi (biasa disebut *tensor)* [17].

Setiap *layers* memiliki fungsinya masing-masing, berikut ini penjabaran dari masing-masing *layers* yang ada dalam CNN:

1. *Convolutional Layers*

Sesuai dengan namanya, "*convolutional", layer* ini dapat dikatakan sebagai komponen paling mendasar pada CNN. Pada lapisan ini akan dilakukan *feature extraction*, dalam prosesnya terdapat filter yang terdiri dari *array of numbers* (biasa disebut dengan istilah *kernel)* yang nantinya akan diaplikasikan dengan cara dihitung, secara menyeluruh pada *tensor* (gambar yang di-*input*). Hasil dari perhitungannya disebut *features maps* atau *activation maps.* Ilustrasi perhitungan pada *convolutional layers* dapat dilihat pada Gambar 4.



Gambar 4. Contoh Perhitungan Pada *Convolutional Layers* Dengan *Kernel* 3×3

2. *Pooling Layers*

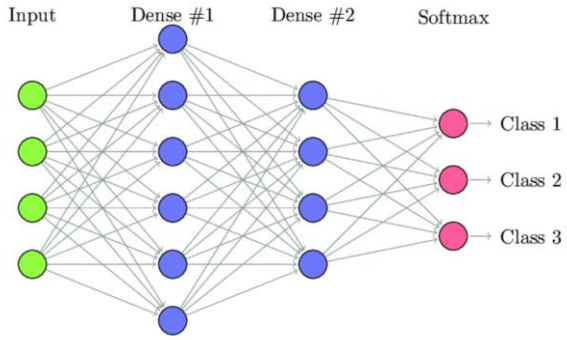
Fungsi pada lapisan ini adalah menurunkan dimensi dari *feature map* dengan tetap menyimpan data yang penting, oleh karena itu nama lain *pooling layers* adalah *downsampling*. Terdapat dua metode dalam *pooling layers* yaitu *max pooling* dan *global average pooling*, dari kedua metode tersebut *max pooling* lebih populer dan umum digunakan [16]. Contoh ilustrasi *pooling layers* dapat dilihat pada Gambar 5.

Gambar 5. Contoh Perhitungan Pada *Pooling Layers*

3. *Fully Connected Layers*

Lapisan ini membutuhkan *input* berupa vektor satu dimensi yang berasal dari *feature maps convolution* atau *pooling layer* yang ditransformasi. Setiap vektor akan terhubung dengan satu atau lebih *fully connected layer* (*dense*). Pada lapisan *fully connected layer* ini komputer akan belajar mengenai *input* yang dimasukkan dengan cara memberi bobot atau *weight*, yang nantinya akan digunakan sebagai acuan untuk menghitung probabilitas dengan bantuan *softmax function*, sehingga dapat menentukan *class* yang sesuai. Ilustrasi *fully connected layer* dapat dilihat pada Gambar 6, dan rumus *softmax function* dapat dilihat pada formula (1).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (1)$$



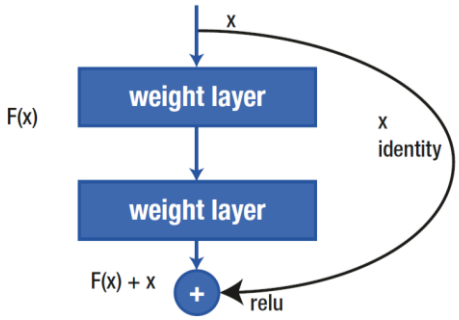
Gambar 6. Ilustrasi *Fully Connected Layers*

Hasil dari perhitungan *Softmax* akan berupa probabilitas antara 0 sampai dengan 1, yang berguna untuk menentukan klasifikasi kelasnya. Kelas akan ditentukan berdasarkan *output* yang memiliki angka probabilitas tertinggi (mendekati satu).

Dalam proses *training* data menggunakan metode CNN terkadang akan ditemukan masalah yang disebabkan oleh lapisan *neural network* yang terlalu dalam (menyebabkan

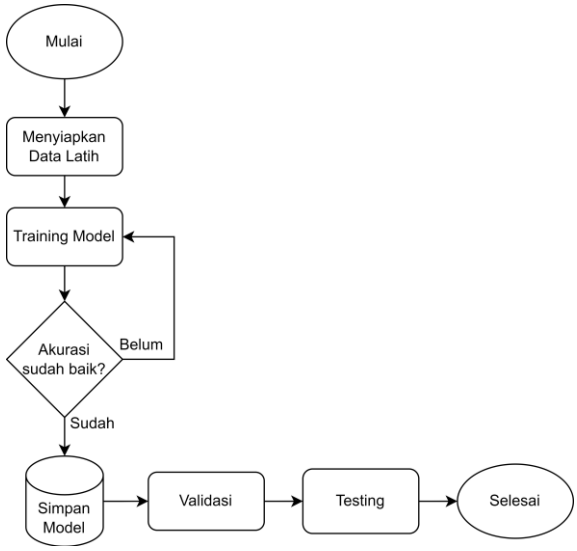
training error dan *validation error* yang tinggi), sehingga kinerja akan memburuk seiring dengan pertambahan *layer*. Hal ini dikenal dengan istilah '*Vanishing Gradients'* [16]. Terdapat berbagai arsitektur yang dapat digunakan dalam algoritma CNN. Salah satu arsitektur yang paling stabil untuk mengatasi permasalahan *vanishing gradients* adalah Resnet [17].

Hal utama yang membedakan arsitektur ResNet dibandingkan arsitektur lainnya adalah konsep *skip connections* dan *residual block* yang digunakan bersamaan untuk mengatasi permasalahan *vanishing gradient*. Ilustrasinya dapat dilihat di atas pada Gambar 7. Pada Gambar 7 di bawah, dapat dilihat terdapat lompatan *output* hasil dari *layer* sebelumnya ke *layer* berikutnya, melompati beberapa *layer* sekaligus yang berada diantaranya (digambarkan dengan panah melengkung). Dengan lompatan-lompatan yang dilakukan, memungkinkan untuk dilakukan *training* pada lapisan yang lebih dalam, tanpa mengurangi performa dari *network* [18].



Gambar 7. Ilustrasi Lompatan Dalam Arsitektur ResNet
Sumber: Verdhan, V. (2021)

Alur pembuatan program untuk *object recognition* dapat dilihat pada Gambar 8 di bawah.



Gambar 8. *Flowchart* Pelatihan Model

Penjelasan singkat mengenai hal yang dilakukan pada setiap tahapan pembuatan program *object recognition* adalah sebagai berikut:

a. Menyiapkan Data Latih

Sebelum komputer dapat melakukan pengenalan objek secara otomatis tentu komputer harus diajari terlebih dahulu di-*training* agar dapat mengklasifikasikan objek dengan benar dan tepat. Oleh sebab itu, tahap pertama yang harus dilakukan adalah menentukan jumlah dan nama *class* apa yang akan dikenali, serta mempersiapkan data latihnya. Dalam kasus ini data yang akan dilatih berupa citra (*image*). Pada penelitian ini terdapat sepuluh kelas yang akan dilatih, kesepuluh kelas tersebut dapat dilihat pada Tabel 1.

Tabel 1. Sepuluh Kelas Yang di Training

No.	Nama Barang
1.	Chacha
2.	Cup Noodles
3.	Fanta
4.	Onion Rings
5.	Pretz Stik
6.	Snickers
7.	Susu Ultra
8.	Totole Kaldu Ayam
9.	Waffer Nabati
10.	Wang Lao Ji Kaleng

Masing–masing kelas akan memiliki jumlah gambar dengan total 100. Gambar yang digunakan dalam penelitian ini merupakan gambar yang berasal dari internet [20] dan gambar yang diambil secara mandiri. Gambar data latih yang digunakan untuk *training* dapat dilihat pada Gambar 9.

Gambar 9. Data Latih (Kelas: Pretz Stik)

Jumlah gambar dengan total 100 untuk setiap *class* masih tergolong sedikit, *deep learning* pada umumnya biasanya memiliki data latih untuk setiap kelasnya bisa menyentuh ribuan bahkan jutaan data. Untuk mengatasi hal tersebut digunakanlah salah satu *library* dari Keras yaitu *ImageDataGenerator*.

b. *Training Model*

Pada tahap ini akan digunakan *library* misalnya *numpy*, *matplotlib*, dan *TensorFlow*. *Library* ini akan membantu proses pelatihan model agar tidak perlu membuat kode manual lagi dari dasar. Dengan *TensorFlow*, akan memudahkan untuk langsung membagi data menjadi data *train* dan *validation*. Selain itu, data tersebut juga langsung diberi label sesuai

dengan nama *folder* tempat data disimpan, secara otomatis menjadi nama kelas (*class names*).

Teknik *transfer learning* dari *TensorFlow Hub* (TFHub) juga akan diterapkan. Arsitektur yang digunakan dalam pelatihan model menggunakan ResNet-50 yang diambil dari *repository* TFHub. Dalam tahap pengembangannya, *feature vector* yang diambil dari *repository* TFHub bisa disesuaikan lagi dengan kebutuhan (*tuning*). Rangkuman dari arsitektur yang digunakan untuk *training* setelah di *tuning* dapat dilihat pada Gambar 10.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 2048)              23564800

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 24,619,018
Trainable params: 1,054,218
Non-trainable params: 23,564,800
_________________________________________________________________
```

Gambar 10. Model *Summary* untuk *Training*

c. Menyimpan Model

Model yang dilatih harus disimpan agar dapat digunakan kembali tanpa harus berulang-ulang *train* lagi, sebab *training* model membutuhkan waktu yang tidak sebentar. Model yang disimpan juga diharapkan merupakan model dengan akurasi terbaik yang bisa didapatkan selama *training*. Oleh sebab itu, ada beberapa cara yang dapat digunakan untuk menyimpan model salah satunya adalah *callbacks*.

Pada penelitian ini yang menjadi *monitor* apakah model tersebut sudah baik atau belum adalah nilai *validation* loss-nya (*val_loss*). Semakin kecil *val_loss* diharapkan model dapat mengklasifikasikan objek dengan lebih baik. Secara otomatis komputer akan membandingkan *val_loss* yang dihasilkan dari setiap *epoch* yang dilakukan. Setiap ada nilai *val_loss* terkecil yang dihasilkan maka model pada *epoch* tersebut akan langsung disimpan ke dalam komputer.

d. Validasi

Perhitungan hasil validasi sudah dilakukan secara otomatis saat *training* berlangsung karena data gambar yang dimiliki langsung di-*split* 20% untuk *validation*.

e. *Testing*

Pengujian model dilakukan dengan cara meng-*input* foto yang belum pernah dilatih sebelumnya. Terdapat tiga skenario tes yang sudah dilakukan, ketiga skenario tersebut yakni: satu objek dalam satu *frame* foto, dua objek dalam satu *frame* foto dan lima objek dalam satu *frame* foto. Jumlah gambar yang di-*input* untuk *testing* berjumlah 50 untuk setiap skenarionya. Khusus untuk yang skenario satu objek dalam satu *frame* foto, masing-masing kelas akan terdiri dari 5 foto. Foto diambil menggunakan kamera secara mandiri.

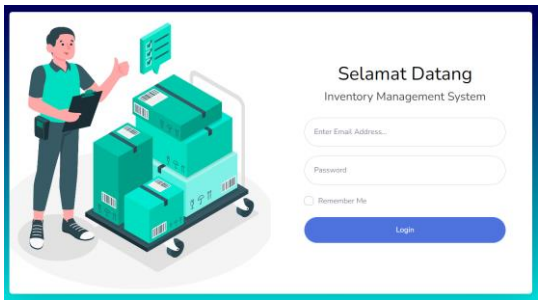
III. HASIL DAN PEMBAHASAN

A. Hasil Pembuatan *Website*

Website yang dibuat terdiri dari halaman *login* (Gambar 11), halaman *dashboard* (Gambar 12), halaman data produk (Gambar 13), halaman pencatatan barang masuk (Gambar 14), halaman validasi *item recognition* (Gambar 15), halaman validasi produk (Gambar 16), dan halaman admin (Gambar 17). Setiap halaman memiliki fungsinya masing-masing berikut ini adalah beberapa hasil tampilan *website* yang sudah dibuat beserta penjelasan singkatnya.

1. Halaman *Login*

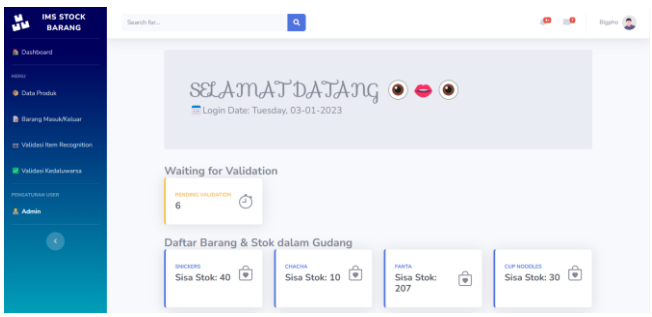
Sama seperti pada *web* atau aplikasi pada umumnya, halaman *login* akan muncul pada saat pertama kali *user* ingin mengakses IMS yang dirancang. Pada halaman ini *user* harus memasukkan email dan *password* yang sudah didaftarkan dalam *database*. Terdapat dua *role* dalam sistem yang telah dibuat, *staff* biasa dan admin. Admin adalah *role* yang berhak untuk mendaftarkan *user* baru ke dalam sistem.



Gambar 11. Tampilan Halaman *Login*

2. Halaman *Dashboard*

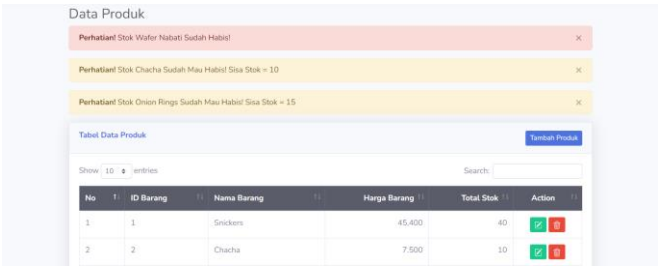
Saat *user* sudah berhasil masuk ke dalam sistem inventori maka akan tampil halaman *dashboard* seperti pada Gambar 12 di bawah. Pada *dashboard* ditampilkan jumlah *row* data yang harus di validasi, dan total stok dari masing-masing barang yang ada di gudang.



Gambar 12. Tampilan Halaman *Dashboard*

3. Halaman Data Produk

Halaman data produk akan menampilkan daftar produk yang ada di gudang beserta harga produk dan total stoknya. Selain itu pada halaman bagian atas akan muncul *alert* yang memberi informasi jika ada produk yang stoknya habis (warna merah), dan produk yang sisa stoknya tinggal sedikit (warna kuning). Pada halaman ini *staff* dapat menambah produk jika ada produk baru, dan juga bisa mengubah nama barang atau harga menggunakan tombol *edit*. Tombol *delete* untuk menghapus data juga sudah tersedia.



Gambar 13. Tampilan Halaman Data Produk

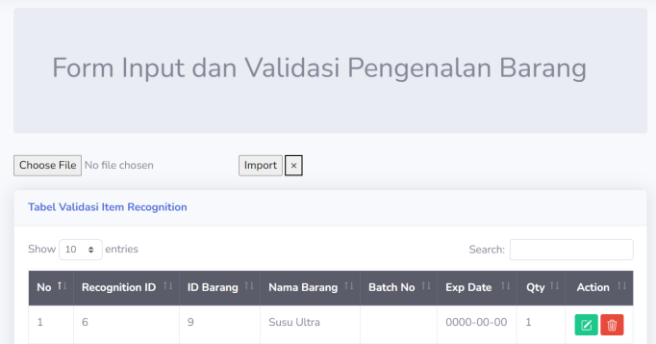
4. Halaman Pencatatan Barang Masuk

Setiap barang yang masuk ke dalam gudang harus dicatat dalam *database.* Halaman ini akan menampilkan catatan detail setiap barang yang masuk ke dalam *database. Staff* yang ingin mencatat barang masuk secara manual juga dapat menekan tombol "Tambah Barang Masuk". Tombol *edit* juga disediakan pada halaman ini apabila ada detail barang masuk yang salah di-*input*. Pada bagian atas akan terlihat *alert* berwarna merah apabila ada barang yang sudah kedaluwarsa, bagi *staff* yang melihat *alert* ini dapat menekan tombol *report* agar barang bisa segera masuk ke dalam *list validasi* produk.

Gambar 14. Tampilan Halaman Pencatatan Barang Masuk

5. Halaman Validasi *Item Recognition*

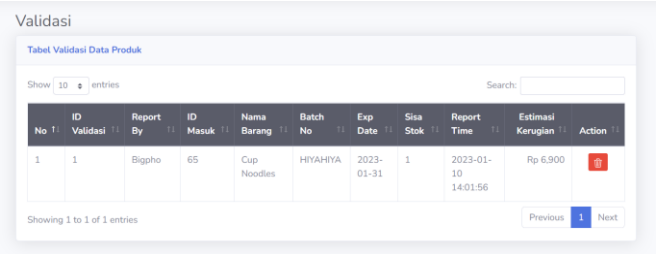
Halaman ini berfungsi untuk *user* meng-*input file extension* .csv hasil *generate* dari program *object recognition*. *File* yang di-*import* nantinya akan masuk ke *database* dan harus divalidasi kembali oleh *staff*, mengingat salah satu kekurangan dari *object recognition* yaitu tidak selalu benar dalam mengenali objek yang difoto.



Gambar 15. Tampilan Halaman Validasi Pengenalan Barang

6. Halaman Validasi Produk

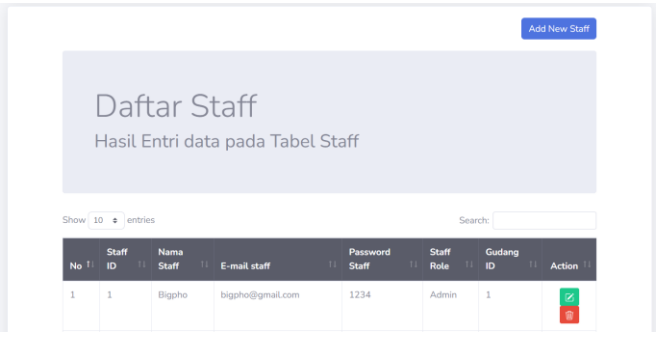
Data dalam tabel validasi yang ditampilkan pada halaman ini merupakan data yang berasal dari tabel stok barang yang di-*report* oleh *staff*. Hanya *staff* yang memiliki *role* admin yang bisa mengakses menu ini karena setelah divalidasi barang yang ada dalam daftar akan di-*delete*.



Gambar 16. Tampilan Halaman Validasi Produk

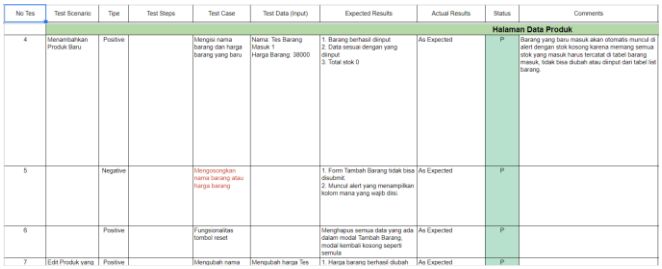
7. Halaman Admin

Halaman ini berfungsi untuk *user management*, sama seperti halaman validasi produk, hanya *staff* yang memiliki *role* admin yang dapat mengakses menu ini. Pada halaman ini admin dapat mengontrol siapa saja yang dapat mengakses sistem. Tersedia tombol *Add New Staff*, *edit*, dan *delete* yang dapat berguna bagi admin untuk mengontrol *user*.



Gambar 17. Tampilan Halaman Admin

B. Hasil *Black Box Testing Web*



No Tes	Test Scenario	Tipe	Test Steps	Test Case	Test Data (Input)	Expected Results	Actual Results	Status	Comments
					Halaman Data Produk				
4	Menambahkan Produk Baru	Positive		Mengisi nama barang dan harga barang yang baru	Nama: Tes Barang Masuk 1 Harga Barang: 38000	1. Barang berhasil diinput 2. Data sesuai dengan yang diinput 3. Total stok 0	As Expected	P	Barang yang baru masuk akan otomatis muncul di alert dengan stok kosong karena memang semua stok yang masuk harus tercatat di tabel barang masuk, tidak bisa diubah atau diinput dari tabel list barang.
5		Negative		Mengosongkan nama barang atau harga barang		1. Form Tambah Barang tidak bisa disubmit 2. Muncul alert yang menampilkan kolom mana yang wajib diisi.	As Expected	P	
6		Positive		Fungsionalitas tombol reset		Menghapus semua data yang ada dalam modal Tambah Barang, modal kembali kosong seperti semula	As Expected	P	
7	Edit Produk yang	Positive		Merubah nama	Merubah harga Tes	1. Harga barang berhasil diubah	As Expected	P	

Gambar 18. Tampilan Hasil *Black Box Testing*

Berdasarkan hasil *black box testing web* yang sudah dilakukan, fungsionalitas dari *website* yang dikembangkan sudah menunjukkan hasil yang baik. *Testing* yang telah dilakukan berjumlah 33 skenario tes, yang terdiri dari 23 skenario positif dan 10 skenario negatif. Skenario dibuat berdasarkan *requirement* awal yang sudah ditentukan. Tampilan tes yang dilakukan dapat dilihat pada Gambar 18, untuk rangkuman hasil tes dapat dilihat pada Tabel 2.

Tabel 2. Rangkuman Hasil *Black Box Testing*

Test Results							
	Skenario	**P**	**PBN**	**N/A**	**F**	**Total Status**	**%**
Positive	23	30	1	0	2	33	93,94%
Negative	10						
TOTAL	33	30	1	0	2		

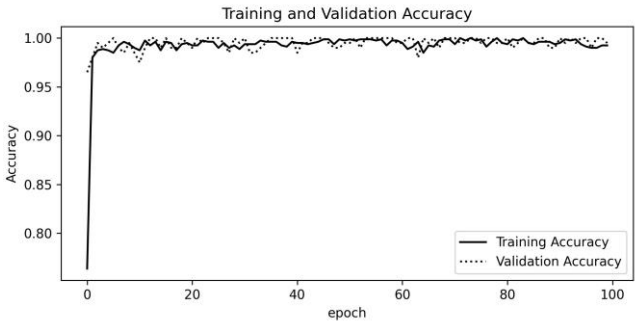
C. Hasil Model *Object Recognition*

Train citra yang dilakukan menggunakan arsitektur ResNet menunjukkan hasil yang cukup baik. Jumlah *epoch* yang dilakukan adalah sebanyak 100 kali dengan menggunakan *optimizer Adam*, dan *Softmax activation function* untuk mengklasifikasikan *class*-nya. Dari 100 *epoch* yang telah dilakukan, diambil satu model yang paling baik. *Epoch* ke-40 akan digunakan karena memiliki *validation loss* yang terendah. Selain *validation loss* akurasi yang dimiliki juga cukup tinggi. *Log output* hasil *train epoch* ke-40 dapat dilihat pada Tabel 3.

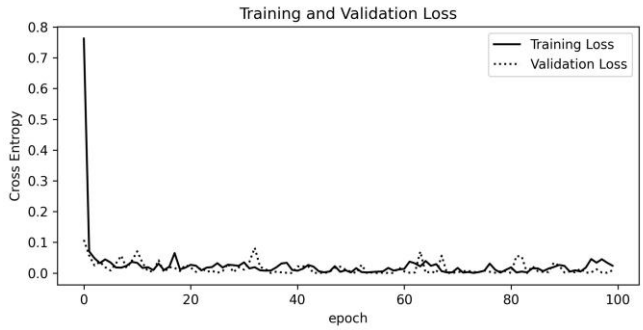
Tabel 3. *Log Output* Hasil *Train Epoch* ke-40

Epoch	***Loss***	***Acc***	***Val Loss***	***Val Acc***
40	0,0104	0,9962	1,0834e-04	1,0000

Dari 100 *epoch* yang telah dilakukan didapatkan dua buah grafik yakni grafik *training* terhadap *validation accuracy* yang dapat dilihat pada Gambar 19 dan *training* terhadap *validation loss* yang dapat dilihat pada Gambar 20.

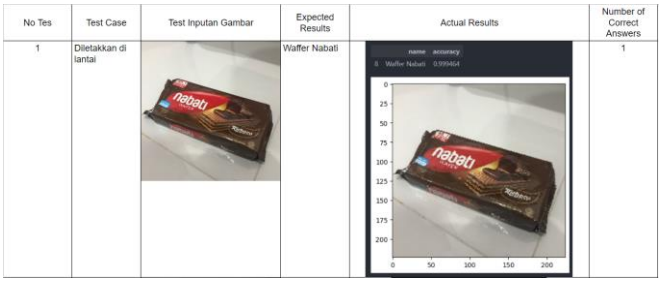


Gambar 19. Grafik *Training and Validation Accuracy*



Gambar 20. Grafik *Training and Validation Loss*

D. Hasil *Testing* Model



Gambar 21. Tampilan Dokumentasi Hasil *Testing Model*

Gambar 22. Tampilan *Output* Hasil Prediksi Pada Program

Gambar 21 di atas merupakan contoh dokumentasi hasil *testing* yang dilakukan terhadap skenario *testing* satu objek dalam satu *frame* foto. Kolom *Number of Correct Answers* merupakan kolom yang menunjukkan berapa jumlah barang yang berhasil dikenali oleh model. Kolom *Actual Result* diisi dengan foto hasil prediksi dari program, untuk lebih jelasnya dapat melihat Gambar 22 sebagai contoh *output*.

Skenario tes pertama (satu objek dalam satu *frame*) yang dilakukan, dari 50 objek yang di-*input*, program berhasil mengenali objek sebanyak 45 kali dan salah mengenali 5 kali, sehingga didapatkan persentase akurasinya sebesar 90%. Untuk skenario tes kedua (dua objek dalam satu *frame*), dari 100 objek yang di-*input*, program berhasil mengenali 56 objek, sehingga akurasinya 56%. Dan yang terakhir, untuk lima objek dalam satu *frame* dengan total 250 objek yang di-*input* program berhasil mengenali 136 objek sehingga menghasilkan akurasi sebesar 54%. Rangkuman dari hasil *testing* model ketiga skenario yang telah dilakukan dapat dilihat pada Tabel 4.

Tabel 4. Rangkuman Hasil Uji Model *Object Recognition*

No.	Skenario Tes	Jumlah Objek	Jumlah Benar	Akurasi
1.	Satu objek dalam satu *frame*	50	45	90%
2.	Dua objek dalam satu *frame*	100	56	56%
3.	Lima objek dalam satu *frame*	250	136	54%

IV. KESIMPULAN DAN SARAN

Pada penelitian ini dihasilkan sebuah sistem manajemen inventori yang dapat digunakan untuk mencatat barang masuk berbasis *web*. Sistem ini juga dilengkapi dengan *alert* yang dapat membantu mengingatkan *user* jika ada barang yang sisa stoknya sedikit atau sudah habis, dan juga jika ada barang yang sudah kedaluwarsa. Berdasarkan hasil pengujian, fitur *web* yang dibuat sudah berhasil dijalankan sampai mencapai 93,94%. Selain itu, *website* juga dilengkapi dengan fitur *object recognition* yang dikembangkan menggunakan arsitektur ResNet dalam CNN. Kesepuluh kelas yang dilatih sudah berhasil dikenali dengan tingkat akurasi sangat baik saat mengenal satu objek dalam satu *frame*, karena sudah mencapai 90% pada saat *testing*. Namun untuk dua objek dan lima objek dalam satu *frame* masih kurang baik, karena masih ada di angka 56% dan 54%.

Saran bagi peneliti yang ingin mengembangkan aplikasi ini adalah coba kembangkan aplikasi berbasis *mobile* karena proses pengambilan foto akan lebih mudah daripada dengan *webcam* laptop. Lalu, tambahkan arsitektur lain selain ResNet agar program dapat mengenali banyak objek dalam satu *frame* foto dengan tingkat akurasi yang lebih baik. Fitur dalam *web* juga dapat dilengkapi lagi, misalnya fitur pencatatan barang keluar. Dan yang terakhir adalah menggunakan bahasa pemrograman yang lebih mudah untuk mengintegrasikan Python (program untuk *object recognition*-nya) dengan aplikasi utama (*web* atau *mobile*).

REFERENSI

[1] R. Komalasari, B. Harto, and R. Setiawan, "UMKM Go-Digital sebagai Adaptasi dan Inovasi Pemasaran Arkha Minoritas pada Pandemi COVID-19," *IKRA-ITH ABDIMAS*, vol. 4, no. 1, pp. 1–7, 2021.

[2] H. W. W. Pitoy, A. B. H. Jan, and J. S. B. Sumarauw, "Analisis Manajemen Pergudangan Pada Gudang Paris Superstore Kotamobagu," *Jurnal EMBA: Jurnal Riset Ekonomi, Manajemen, Bisnis dan Akuntansi*, vol. 8, no. 3, 2020, doi: https://doi.org/10.35794/emba.v8i3.29929.

[3] A. Kamali, "Smart warehouse vs. traditional warehouse," *CiiT International Journal of Automation and Autonomous System*, vol. 11, no. 1, pp. 9–16, 2019.

[4] A. Yanuar, ST., M.MGT. and M. Rahmatulah, ST., MT., "Analisa dan Perancangan Warehouse Management System (WMS) Pada UKM Online," *Jurnal Logistik Bisnis*, vol. 9, no. 02, pp. 81–89, Nov. 2019, doi: 10.46369/LOGISTIK.V9I02.569.

[5] M. L. Syam and Erdisna, "Sistem Informasi Stok Barang Menggunakan *QR-Code* Berbasis Android," *Jurnal Informatika Ekonomi Bisnis*, vol. 4, no. 1, pp. 17–22, Feb. 2022, doi: 10.37034/infeb.v4i1.108.

[6] A. L. Tungadi and E. A. Lisangan, "Simulasi Penerapan Active RFID pada Fungsi Bisnis Penjualan sebagai Komponen ERP pada Peruahaan Ritel," in *Seminar Nasional Komunikasi dan Informatika*, 2021.

[7] Martinus, M. S. Wahab, Yudi, and H. Ham, "Data Transmission Using RFID System on Smart Shopping Carts for Checkout Process Efficiency in Supermarket at Indonesia," *Procedia Comput Sci*, vol. 179, pp. 902–912, Jan. 2021, doi: 10.1016/J.PROCS.2021.01.080.

[8] B. Saputra, R. E. Indrajit, and E. Dazki, "Perancangan Warehouse Management System Berbasis IOT Pada PT. Agility Internasional," *SMARTICS Journal*, vol. 7, no. 2, pp. 72–77, 2022.

[9] J. Zhao, F. Xue, and D. A. Li, "Intelligent Management of Chemical Warehouses with RFID Systems," *Sensors 2020, Vol. 20, Page 123*, vol. 20, no. 1, p. 123, Dec. 2019, doi: 10.3390/S20010123.

[10] M. N. Ardiansyah, P. S. Muttaqin, M. D. Prasetio, and N. Novitasari, "Identifikasi Objek/Produk untuk Proses Stock Taking Barang menggunakan Konsep Object Recognition," *Jurnal Rekayasa Sistem & Industri (JRSI)*, vol. 8, no. 01, pp. 28–34, Jun. 2021, doi: 10.25124/jrsi.v8i1.455.

[11] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an

overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.

[12] L. Lina, M. Augustine, O. Oktaviana, and A. Chris, "Pendeteksian Leukosit Secara Otomatis Melalui Citra Preparat Berbasis Region Proposal Network," *Jurnal Teknika*, vol. 11, no. 03, pp. 190-196, Nov. 2022, doi: 10.34148/teknika.v11i3.541.

[13] H. Yoo, S. Han, and K. Chung, "Diagnosis Support Model of Cardiomegaly Based on CNN Using ResNet and Explainable Feature Map," *IEEE Access*, vol. 9, pp. 55802–55813, Apr. 2021, doi: 10.1109/ACCESS.2021.3068597

[14] L. Lina, A.A. Marunduh, W.Wasino, and D. Ajienegoro, "Identifikasi Emosi Wajah Pengguna Konferensi Video Menggunakan Convolutional Neural Network dengan Arsitektur VGG-16", *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 9, no. 05, pp. 1047-1054, Okt. 2022, doi: 10.25126/jtiik.202295269.

[15] Z. Fouad, M. Alfonse, M. Roushdy, and A.-B. M. Salem, "Hyper-parameter optimization of convolutional neural network based on particle swarm optimization algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 6, pp. 3377–3384, Dec. 2021, doi: 10.11591/eei.v10i6.3257.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[17] V. Verdhan, *Computer Vision Using Deep Learning*. Berkeley, CA: Apress, 2021. doi: 10.1007/978-1-4842-6616-8.

[18] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu, "RPC: A Large-Scale Retail Product Checkout Dataset," Jan. 2019, doi: https://doi.org/10.48550/arXiv.1901.07249.