

**IMPLEMENTASI DEEP LEARNING UNTUK SISTEM
PENCARIAN GAMBAR PRODUK MENGGUNAKAN
ALGORITMA CONVOLUTIONAL NEURAL NETWORK
(CNN)**

LAPORAN TUGAS AKHIR



YOGI DWI ANDRIAN

5180411123

**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA
2022**

**IMPLEMENTASI DEEP LEARNING UNTUK SISTEM
PENCARIAN GAMBAR PRODUK MENGGUNAKAN
ALGORITMA CONVOLUTIONAL NEURAL NETWORK
(CNN)**

Disusun oleh

YOGI DWI ANDRIAN

5180411123

Telah dipertahankan di depan Dewan Penguji
pada tanggal 03 Maret 2022

Nama & Gelar

DEWAN PENGUJI

Jabatan

Tanda tangan

Tanggal

Dr. Enny Itje Sela, S.Si., M.Kom.
NIK 111116086

Ketua Penguji

Saucha Diwandari, S.Kom., M.Eng.
NIK 110717134

Penguji I

Muhammad Fachrie, ST, M.CS.
NIK 110517125

Penguji II

(Dosen Pembimbing)

14/3/2022

14-03-2022

Yogyakarta,
Ketua Program Studi Informatika

Dr. Enny Itje Sela, S.Si., M.Kom.
NIK 111116086

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini, saya

Nama : Yogi Dwi Andrian

NPM : 5180411123

Program Studi: Informatika

Program : Sarjana

Fakultas : Sains & Teknologi

Menyatakan bahwa tugas akhir dengan judul Implementasi *Deep Learning* Untuk Sistem Pencarian Gambar Produk Menggunakan Algoritma *Convolutional Neural Network (CNN)* ini adalah karya ilmiah asli saya dan belum pernah dipublikasikan oleh orang lain, kecuali yang tertulis sebagai acuan dalam naskah ini dan disebutkan dalam daftar pustaka. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa yang diberikan Program Studi Informatika Fakultas Sains & Teknologi Universitas Teknologi Yogyakarta.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada tanggal : 20 Januari 2022

Yang menyatakan



Yogi Dwi Andrian

ABSTRAK

Convolution neural network adalah salah satu metode *Machine Learning* dari pengembangan *Multi Layer Peceptron* (MLP) yang digunakan untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural network* karena dalam tingkat jaringan dan banyak di implementasikan dalam data citra. Semakin berkembangnya teknologi dibidang kecerdasan buatan terutama di bidang *Deep Neural network* meningkatkan perkembangan teknologi pencarian. Pencarian yang mulanya menggunakan teks sekarang sudah mulai mengimplementasikan menggunakan gambar sebagai masukannya dan mengembalikan gambar yang serupa dengan masukan. Cara kerja teknologi ini adalah dengan membandingkan fitur-fitur yang ada pada gambar untuk menghitung kesamaan antar gambar dan mengembalikan gambar-gambar yang menyerupai tingkat kesamaan terhadap gambar masukan. Dalam pembangunan sistem pencarian bisa menggunakan desain arsitektur yang diusulkan atau memanfaatkan model *pretrained* dengan metode *transfer learning*. Terdapat beberapa model *pretrained* yang dapat digunakan untuk melakukan pencarian gambar seperti VGG16 dan Resnet50 akan digunakan sebagai model penelitian dengan metode *transfer learning* ditambah satu model dengan arsitektur rancangan yang diusulkan. Dari tiga model dilakukan *hyperparameter tuning* dan didapatkan model terbaik dari setiap arsitektur yaitu dengan urutan model buatan yang diusulkan, VGG16, dan Resnet50 hasilnya adalah 78.45%, 86.03%, 87.50% untuk akurasi *training* dan 78.70%, 84.43%, 84.46% untuk akurasi validasi dan 79%, 85%, 84% untuk akurasi testing. Arsitektur terbaik pada penelitian ini adalah sistem yang menggunakan metode *transfer learning* dengan model ResNet50 dengan *optimizer* RMSProp dan *learning rate* 0.0001. Model akan digunakan untuk ekstraksi fitur citra gambar masukan dan dataset citra gambar untuk disimpan dalam basis data lalu dihitung *euclidean distance* dan berdasarkan jarak terdekat dijadikan sebuah *output*.

Kata Kunci: *Convolution Neural Network*, *Visual Search*, Citra, *Deep Neural Network*, Sistem Pencarian, *Transfer Learning*, Ekstraksi Fitur.

ABSTRACT

Convolution neural network is one of the Machine Learning methods from the development of Multi Layer Peceptron (MLP) which is used to process two-dimensional data. CNN is included in the type of Deep Neural network because it is at the network level and is widely implemented in image data. The development of technology in the field of artificial intelligence, especially in the field of Deep Neural networks, increases the development of search technology. Searches that originally used text have now started implementing using an image as input and returning an image similar to the input. The way this technology works is by comparing the features in the image to calculate the similarity between images and returning images that resemble the level of similarity to the input image. In building the search system, you can use the proposed architectural design or use a pretrained model with the transfer learning method. There are several pretrained models that can be used to perform image searches such as VGG16 and Resnet50 which will be used as research models with transfer learning methods plus one model with the proposed design architecture. From the three models, hyperparameter tuning was carried out and the best model for each architecture was obtained, with the proposed artificial model sequence, VGG16, and Resnet50 the results were 78.45%, 86.03%, 87.50% for training accuracy and 78.70%, 84.43%, 84.46% for validation accuracy. and 79%, 85%, 84% for testing accuracy. The best architecture in this study is a system that uses the transfer learning method with the ResNet50 model with the RMSProp optimizer and a learning rate of 0.0001. The model will be used for feature extraction of the input image and image image dataset to be stored in the database and then the Euclidean distance is calculated and based on the closest distance is used as an output.

Keywords: *Convolution Neural Network, Visual Search, Image, Deep Neural Network, System Search, Transfer Learning, Feature Extraction.*

KATA PENGANTAR

Puji syukur dipanjatkan atas kehadiran Allah SWT, karena dengan limpahan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dengan judul Implementasi *Deep Learning* Untuk Sistem Pencarian Gambar Produk Menggunakan Algoritma *Convolutional Neural Network (CNN)*.

Penyusunan Tugas Akhir diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana pada Program Studi Informatika Fakultas Sains & Teknologi Universitas Teknologi Yogyakarta.

Tugas Akhir ini dapat diselesaikan tidak lepas dari segala bantuan, bimbingan, dorongan dan doa dari berbagai pihak, yang pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

- a. Kedua orang tua penulis, Bapak Andri Sunarto dan Ibu Eni Kusri yang telah memberikan dukungan, semangat dan doa sehingga penulis dapat menyelesaikan Tugas Akhir ini.
- b. Bapak Dr. Bambang Moertono Setiawan, M.M., C.A., Akt selaku Rektor Universitas Teknologi Yogyakarta.
- c. Dr. Endy Marlina, S.T., M.T. selaku Dekan Fakultas Sains & Teknologi
- d. Ibu Dr. Enny Itje Sela, S.Si., M.Kom selaku Ketua Program Studi Informatika Fakultas Sains & Teknologi.
- e. Muhammad Fachrie, S.T., M.Cs selaku Dosen Pembimbing dan Wali Dosen TI B angkatan 2018 yang telah memberikan bimbingan dan arahan dalam menyelesaikan Laporan Tugas Akhir ini.
- f. Dosen – dosen program studi Informatika yang telah memberikan ilmunya selama penulis menuntut ilmu di Universitas Teknologi Yogyakarta.
- g. Teman – Teman Program Studi Informatika khusus nya Informatika B yang telah memotivasi penulis menyelesaikan laporan ini.
- h. Kepada kakak Mufti Ristio Van Dame yang selalu memberikan semangat dan mengingatkan untuk mengerjakan laporan ini.
- i. Terakhir kepada rekan-rekan magang di VARX - PT Semua Aplikasi

Indonesia.

Akhir kata, penulis menyadari bahwa sepenuhnya akan terbatasnya pengetahuan penyusun, sehingga tidak menutup kemungkinan jika ada kesalahan serta kekurangan dalam penyusunan Tugas Akhir, untuk itu sumbang saran dari pembaca sangat diharapkan sebagai bahan pelajaran berharga dimasa yang akan datang.

Yogyakarta, Januari 2022

Penulis

DAFTAR ISI

ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR.....	vi
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL	xii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Sistematika Penulisan	4
BAB II KAJIAN HASIL PENELITIAN DAN LANDASAN TEORI.....	6
2.1 Kajian Hasil Penelitian.....	6
2.2 Landasan Teori.....	10
2.2.1 Gambar Digital.....	10
2.2.2 Visual Search	11
2.2.3 Feature Extraction	12
2.2.4 Convolutional Neural Network	12
2.2.5 Fungsi Aktivasi	19
2.2.6 ADAM	21
2.2.7 RMSProp.....	23
2.2.8 Transfer Learning.....	24
2.2.9 Confusion Matrix	27
2.2.10 Measure Distance	30
2.2.11 Vektor.....	30
2.2.12 ERD (Entity Relationship Diagram)	31
2.2.13 Python	32
2.2.14 Basis Data MySQL	33
2.2.15 Tensorflow	33
2.2.16 Flask	34
BAB III METODE PENELITIAN.....	35
3.1 Bahan/Data.....	35
3.1.1 Data yang diperoleh	35
3.1.2 Prosedur pengumpulan data	39
3.2 Aturan bisnis (bussiness rule)	41
3.2.1 Analisis Sistem Saat Ini	41
3.2.2 Kelemahan Sistem Saat Ini	42
3.3 Tahapan Penelitian	42
BAB IV ANALISIS DAN PERANCANGAN SISTEM.....	49
4.1 Analisis Sistem Yang Diusulkan.....	49

4.1.1 Analisis Fungsional.....	49
4.1.2 Analisis Non Fungsional.....	50
4.2 Desain Sistem.....	51
4.2.1 Desain Logik	51
4.2.1.1 Diagram Sistem Yang Akan Dibangun.....	51
4.2.1.2 Entity Relationship Diagram (ERD)	52
4.2.1.3 Diagram Use Case.....	53
4.2.2 Desain Fisik.....	54
BAB V IMPLEMENTASI DAN HASIL SERTA PEMBAHASAN	55
5.1. Implementasi	55
5.1.1 Eksperimen Model Pencarian Gambar.....	55
5.1.1.1 Rancangan Dataset	55
5.1.1.2 Konfigurasi Library Python	56
5.1.1.3 Menyiapkan Dataset.....	58
5.1.1.4 Persiapan Data.....	59
5.1.1.5 Pembuatan model	61
5.1.2 Pembangunan Sistem Pencarian Gambar.....	67
5.1.2.1 Perancangan Database.....	67
5.1.2.2 Ekstraksi Fitur	67
5.1.2.3 Menambahkan Data ke Basis Data	68
5.1.2.4 Klasifikasi dan Mengukur Kemiripan.....	70
5.1.2.5 Membuat Tampilan Berbasis Web.....	71
5.1. Hasil	74
5.1.1 Hasil Pelatihan	74
5.1.1.1 Model yang Diusulkan	75
5.1.1.2 Model VGG16.....	76
5.1.1.3 Model ResNet50	77
5.1.1.4 Perbandingan Hasil Pelatihan	79
5.1.2 Hasil Sistem Pencarian Menggunakan Gambar	80
5.2 Pembahasan.....	81
5.2.1 Perbandingan Arsitektur dan Proses Pelatihan	81
5.2.2 Evaluasi dan Analisis Performa Model.....	83
5.2.3 Evaluasi dan Analisis Model Transfer Learning.....	84
5.2.4 Model Sistem Pencarian Gambar.....	84
BAB VI PENUTUP	85
6.1 Simpulan.....	85
6.2 Saran.....	86
DAFTAR PUSTAKA	87

DAFTAR GAMBAR

Gambar 2. 1 Arsitektur <i>Convolutional Neural Network</i>	13
Gambar 2. 2 Proses <i>Convolutional Layer</i>	14
Gambar 2. 3 Proses <i>Convolutional Layer</i> Lengkap	14
Gambar 2. 4 Proses Filter	15
Gambar 2. 5 Contoh <i>Pooling Layer</i>	16
Gambar 2. 6 Operasi <i>Pooling Layers</i>	16
Gambar 2. 7 Proses <i>fully-connected layer</i>	17
Gambar 2. 8 <i>Dropout Layer</i>	18
Gambar 2. 9 Grafik Fungsi Sigmoid	19
Gambar 2. 10 Menggunakan Aktivasi Softmax	21
Gambar 2. 11 Diagram fungsi relu	21
Gambar 2. 12 Proses <i>Transfer Learning</i>	24
Gambar 2. 13 Arsitektur VGG16	25
Gambar 2. 14 Arsitektur ResNet50	26
Gambar 2. 15 Arsitektur yang Diusulkan Memanfaatkan Pre-trained Model	27
Gambar 2. 16 <i>Confusion Matrix</i>	28
Gambar 2. 17 Gambar Digital Menjadi Vektor 1 Dimensi	31
Gambar 3. 1 Data Sub-kategori Bagian 1	35
Gambar 3. 2 Data Sub-kategori Bagian 2	36
Gambar 3. 3 Data Sub-kategori Bagian 3	37
Gambar 3. 4 Data Sub-kategori Bagian 4	38
Gambar 3. 5 Halaman Depan Situs Kaggle	40
Gambar 3. 6 Hasil Pencarian di Kaggle	40
Gambar 3. 7 Hasil Pencarian Data	41
Gambar 3. 8 Dataset yang Berhasil di Unduh	41
Gambar 3. 9 Diagram Sistem Pencarian	42
Gambar 3. 10 Diagram Tahap Penelitian	43
Gambar 3. 11 Arsitektur yang Model Diusulkan	45
Gambar 4. 1 Sistem Yang Akan Dibangun	52
Gambar 4. 2 Diagram ERD	53
Gambar 4. 3 Diagram <i>Use Case</i>	53
Gambar 4. 4 Tampilan <i>User Interface</i>	54
Gambar 5. 1 <i>Source Code Split Folders</i>	56
Gambar 5. 2 <i>Source Code Library</i> Python	58
Gambar 5. 3 <i>Source Code</i> Menyiapkan Dataset	59
Gambar 5. 4 <i>Source Code</i> Memindahkan Folder	59
Gambar 5. 5 <i>Source Code</i> Merubah Ukuran Dataset	60
Gambar 5. 6 <i>Source Code</i> Membagi Dataset	60
Gambar 5. 7 <i>Source Code</i> Augmentasi, Normalisasi, dan <i>Batching</i>	61
Gambar 5. 8 <i>Source Code</i> Model VGG16	64
Gambar 5. 9 <i>Source Code</i> Model ResNet50	65
Gambar 5. 10 <i>Source Code</i> Model yang Diusulkan	66

Gambar 5. 11 <i>Source Code</i> Perancangan <i>Database</i>	67
Gambar 5. 12 <i>Source Code</i> Fitur Ekstraksi.....	68
Gambar 5. 13 <i>Source Code</i> Menambahkan Data ke Basis Data	69
Gambar 5. 14 <i>Source Code</i> Proses Klasifikasi	70
Gambar 5. 15 <i>Source Code</i> Proses Menghitung <i>Euclidean Distance</i>	71
Gambar 5. 16 <i>Source Code</i> Flask.....	72
Gambar 5. 17 <i>Source Code</i> File Html.....	73
Gambar 5. 18 Tampilan <i>User Interface</i> Web.....	74
Gambar 5. 19 Grafik Proses Pelatihan Menggunakan Model yang Diusulkan.....	76
Gambar 5. 20 Grafik Proses Pelatihan Menggunakan VGG16.....	77
Gambar 5. 21 Grafik Proses Pelatihan Menggunakan ResNet50	78
Gambar 5. 22 Pencarian <i>Hats</i>	80
Gambar 5. 23 Hasil Proses CNN.....	81
Gambar 5. 24 Hasil Proses Pencarian <i>Hats</i>	81

DAFTAR TABEL

Tabel 2. 1 Perbandingan Kajian Hasil Penelitian	8
Tabel 2. 2 Notasi-notasi ERD	32
Tabel 3. 1 Banyak Data Setiap Label	38
Tabel 3. 2 Lanjutan Banyak Data Per-label Dalam Dataset.....	39
Tabel 5. 1 Pembagian Dataset.....	56
Tabel 5. 2 Perbandingan Hasil <i>Training</i> Model yang Diusulkan	75
Tabel 5. 3 Perbandingan Hasil <i>Training</i> Model VGG16	76
Tabel 5. 4 Perbandingan Hasil <i>Training</i> Model ResNet50	78
Tabel 5. 5 Tabel Perbandingan Performa Model	79
Tabel 5. 6 Perbandingan Struktur Arsitektur dan Pelatihan.....	82
Tabel 5. 7 Perhitungan Presisi, <i>Recall</i> , <i>F1-Score</i> Pada Model Terbaik	83
Tabel 5. 8 Lanjutan Perhitungan Presisi, <i>Recall</i> , <i>F1-Score</i> Pada Model Terbaik.	84

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam satu dekade terakhir *Machine Learning* menjelma sebagai bidang yang berkembang sangat pesat dan terus dikembangkan para ilmuwan seluruh dunia. Meskipun demikian, penggunaan *Machine Learning* di industri masih terbilang baru dan berbeda dengan di bidang akademik dimana pada bidang industri lebih mementingkan aspek fungsionalitas sedangkan di bidang akademik untuk mencapai hasil tertinggi atau tahapan terbaru. Oleh sebab itu banyak perusahaan berbasis teknologi yang mulai mengimplementasikan teknologi *Machine Learning* terutama perusahaan di bidang *marketplace* seperti Tokopedia, Bukalapak, dan Shopee. Pada perusahaan di bidang tersebut sebuah fitur mesin pencarian produk sangatlah penting, mesin pencari yang ada sekarang ini rata-rata masih menggunakan cara yang tradisional yaitu dengan menggunakan kata kunci berupa teks untuk mencari sebuah produk yang diinginkan. Hal ini perlu dikembangkan lagi mengikuti kebutuhan dan tren di masyarakat yang sudah memiliki gawai untuk menangkap dan mengolah gambar yang ada di sekitarnya. Dengan adanya teknologi *Machine Learning*, sistem pencarian dapat dikembangkan dengan menggunakan gambar sebagai masukannya dan akan menghasilkan produk-produk yang serupa atau mirip. Sistem pencarian menggunakan gambar sangat diperlukan guna untuk mencari sebuah produk yang sesuai dengan apa yang di lihat seperti seorang *influencer* mengenakan sebuah produk dan penggemarnya ingin membeli produk yang mirip namun tidak mengetahui nama produk atau kata kunci untuk mendapatkan produk tersebut, disinilah pencarian menggunakan gambar sangat diperlukan. Teknologi tersebut adalah mesin pencari berdasarkan gambar atau dikenal dengan istilah *visual search*.

Visual search menggunakan gambar dunia nyata (tangkapan layar, gambar daring, atau foto) sebagai *input* untuk melakukan pencarian. Sebelum era *Machine Learning* untuk melakukan *visual search* sulit dilakukan karena informasi yang diperoleh dari gambar sangat sedikit. Informasi tersebut hanya berupa warna,

bentuk, dan semacamnya. Informasi tersebut biasa disebut dengan *local feature* yang artinya fitur dapat dilihat oleh manusia secara langsung. Namun setelah adanya *Machine Learning* terutama pada bidang *Deep Learning* kita tidak hanya dapat mendapatkan local feature dari sebuah gambar namun juga *deep feature* dari sebuah gambar. Fitur inilah yang membuat visual search menjadi lebih akurat dan cepat.

Implementasi dari *Machine Learning* di bidang *Deep Learning* sudah menjadi tren pada saat ini. Ketersediaan berbagai *library* yang mendukung untuk pengembangan model untuk tujuan tertentu sangat mempermudah orang dalam melatih model *Machine Learning* maupun *Deep Learning*. Hal ini membuka berbagai peluang inovasi yang dapat dikembangkan dengan cara tersebut salah satunya adalah *visual search*.

Dalam pembuatan model akan dilakukan beberapa penelitian terhadap beberapa arsitektur yang akan diimplementasikan nantinya dan akan dipilih yang terbaik. Metode yang akan digunakan adalah *Convolutional Neural Network* (CNN). Pemilihan beberapa arsitektur diharapkan dapat menghasilkan ekstraksi fitur dan akurasi yang baik serta dapat dikembangkan lebih lanjut atau *scalable*.

1.2 Rumusan Masalah

Berdasarkan latar belakang maka rumusan masalah adalah sebagai berikut:

1. Bagaimana arsitektur CNN yang terbaik dalam mengembangkan model untuk melakukan ekstraksi fitur?
2. Bagaimana akurasi CNN dapat ditingkatkan menjadi lebih baik untuk melakukan klasifikasi?

1.3 Batasan Masalah

Batasan dalam pengembangan CNN untuk pencarian gambar produk ini mencakup berbagai hal, sebagai berikut:

- a. Model akan dilatih dengan dataset *Real Fashion* yang didapat dari website kaggle.

- b. Perbandingan model yang digunakan untuk mencari performa yang terbaik menggunakan model VGG16, ResNet50, dan model yang diusulkan.
- c. *Library* yang digunakan untuk membuat model yaitu TensorFlow dan model akan disimpan dalam format h5 (keras model).
- d. Sistem pencarian yang dibuat hanya menampilkan produk yang dicari dan produk yang serupa.
- e. Sistem pencarian tidak menyediakan masukkan menggunakan kata kunci berupa teks.
- f. Metode yang digunakan dalam penelitian ini adalah metode *Convolutional Neural Network* untuk melakukan ekstraksi fitur.
- g. Jumlah epoch yang digunakan maksimal adalah 50 saat proses pelatihan.
- h. Data *input* saat melakukan pencarian tidak akan di simpan dalam *database* ataupun direktori program.
- i. Pengguna tidak dapat menambahkan data ekstraksi fitur kedalam database menggunakan GUI.

1.4 Tujuan Penelitian

Penelitian ini dilakukan untuk membuat sistem pencarian berdasarkan gambar dengan menggunakan metode *Convolutional Neural Network* untuk melakukan ekstraksi fitur gambar. Pembuatan model akan dilakukan dengan melakukan 2 buah penelitian, yaitu dengan mencoba membuat model yang diusulkan dan menggunakan *pretrained* model atau model yang sudah pernah dilatih sebelumnya. Beberapa model yang sudah ada yaitu, VGG16, dan Resnet50. Hasil dari penelitian beberapa model akan dibandingkan untuk mendapatkan model yang optimal dan memiliki performa yang baik untuk melakukan ekstraksi fitur.

1.5 Manfaat Penelitian

Mengetahui bahwa metode *Convolutional Neural Network* dapat digunakan untuk melakukan ekstraksi fitur gambar serta dapat menambah pengetahuan tentang pembuatan sebuah sistem pencarian menggunakan gambar.

1.6 Sistematika Penulisan

Adapun sistematika penulisan proposal tugas akhir ini disusun dalam beberapa bab yang dijelaskan sebagai berikut:

BAB I PENDAHULUAN

Bab ini akan berisi mengenai latar belakang yang mendasari melakukan penelitian Pencarian Gambar Produk Menggunakan Algoritma *Convolutional Neural Network*, rumusan masalah yang akan diselesaikan pada bab-bab selanjutnya, batasan-batasan masalah dalam melakukan penelitian, tujuan yang akan dicapai dalam melakukan penelitian, dan manfaat dari hasil dilakukannya penelitian ini.

BAB II KAJIAN HASIL PENELITIAN DAN LANDASAN TEORI

Bab ini akan berisi kajian-kajian penelitian terdahulu yang digunakan oleh penulis sebagai referensi dan tolak ukur dalam pembuatan program mengenai pengembangan *visual search*, *Convolutional Neural Network*, serta modul-modul yang diterapkan untuk membuat sistem pencarian menggunakan gambar. Adapun landasan teori yang berkaitan langsung dengan masalah penelitian dalam menerapkan Algoritma *Convolutional Neural Network* dan pembuatan sistem pencarian menggunakan gambar.

BAB III METODE PENELITIAN

Bagian ini menyajikan dan menjelaskan secara lengkap setiap langkah eksperimen yang dilakukan dalam penelitian seperti menjelaskan tahapan mendapatkan bahan atau data gambar produk, prosedur dalam pengumpulan data, melakukan analisa sistem pencarian saat ini beserta kelemahannya, dan menguraikan tahapan penelitian yang dilakukan untuk membangun sebuah sistem pencarian gambar produk

BAB IV ANALISIS DAN DESAIN SISTEM

Bab ini meliputi langkah - langkah membuat aliran sistem, seperti flowchart sistem pencarian menggunakan gambar dan ERD dari *database* yang digunakan untuk menyimpan data gambar beserta atributnya. Selain itu juga menjelaskan tentang analisis analisis sistem yang akan diusulkan yang terdiri dari analisis fungsional yang terkait dengan fasilitas yang dibutuhkan oleh sistem pencarian menggunakan gambar secara umum dan non fungsional yang meliputi perangkat yang digunakan, desain sistem yang terdiri dari desain logik yang berisi aliran sistem dari penelitian dan desain fisik dari rancangan sistem.

BAB V IMPLEMENTASI DAN HASIL SERTA PEMBAHASAN

Bab V pada bagian implementasi memaparkan *source code* dari pembuatan beberapa model CNN serta sistem pencarian gambar produk, pada bagian hasil memaparkan uraian beberapa model yang dibuat beserta perhitungan kinerja masing-masing model dan hasil dari sistem pencarian menggunakan gambar yang dibuat. Pada bagian pembahasan membahas tentang perbandingan antara model yang akan dipilih untuk digunakan sebagai ekstraksi fitur pada sistem pencarian menggunakan gambar produk.

BAB VI PENUTUP

Bab ini berisi tentang Bagian ini berisi simpulan sementara yang diambil dari isi metode penelitian, dan rancangan. Bab ini juga berisi saran yang diperlukan untuk pengembangan penelitian selanjutnya.

BAB II

KAJIAN HASIL PENELITIAN DAN LANDASAN TEORI

2.1 Kajian Hasil Penelitian

Zhang, Y. dkk., (2018) melakukan penelitian dengan judul *Visual Search at Alibaba*. Pada penelitian ini digunakan salah satu metode *Deep Learning* yaitu *Convolutional Neural Network*. Dalam implementasinya dilakukan *Deep Metrik Embedding* untuk mengukur kedekatan dari dua gambar yang sedang dibandingkan. *Weakly supervised object localization* juga diimplementasikan untuk melakukan identifikasi objek yang terdapat pada gambar yang diberikan, fitur ini digunakan untuk mendapatkan kategori dari gambar yang sudah diberikan. Kategori ini nantinya akan mengurangi dari proses perbandingan gambar dengan data gambar yang sudah ada. *Joint detection and Feature learning* diaplikasikan untuk mendeteksi objek dan mendapatkan fitur-fitur dari objek dan setelah itu akan dilakukan *index reranking* berdasarkan dari kedekatan data gambar ke gambar yang diterima. Model hasil dari penelitian ini mendapatkan nilai *recall@1* sebesar 0.415 yang lebih bagus dari model aslinya yaitu GoogLeNet V1 yang hanya sebesar 0.067 dan model lainnya.

Jing, Y. dkk., (2015) melakukan penelitian dengan judul *Visual Search at Pinterest*. Pada penelitian ini digunakan salah satu metode *Deep Learning* yaitu *Convolutional Neural Network* menggunakan Caffe framework. Pada penelitian ini dilakukan lokalisasi terlebih dahulu sebelum melakukan pencarian gambar, jadi gambar akan ditentukan kelasnya terlebih dahulu, misal dalam hal ini ditentukan kategorinya sehingga dalam proses pencarian tidak perlu dilakukan terhadap semua gambar tapi hanya pada kategori yang sudah ditentukan atau dihasilkan. Hal ini akan mempercepat proses pencarian dan dapat meningkatkan akurasi dari pencarian. Sistem pencarian gambar mengimplementasikan sistem terdistribusi sehingga dapat melakukan beberapa proses secara paralel. Hasil dari penelitian ini adalah pengguna dapat melakukan sebuah pencarian gambar dimana dia dapat memilih objek dalam gambar yang akan dijadikan *input* dalam model.

Fengzi, L. dkk., (2020) melakukan penelitian dengan judul *Neural Networks for Fashion Image Classification and Visual Search*. Dalam penelitian ini penulis mengejar dua tujuan yaitu *Image Classification* dan *Image Search*. Pada *Image Classification* penulis mengimplementasikan transfer learning dengan menggunakan model yang sudah pernah dilatih yaitu VGG19 dan InceptionV3 untuk melakukan klasifikasi gambar. Sedangkan untuk *Image Search* menggunakan *cnn based autoencoder* dan *resnet based autoencoder*, autoencoder merupakan jaringan neural yang dirancang untuk mempelajari hubungan non-linear yang kompleks antara titik data. Dalam penelitian distribusi data yang digunakan tidak sama dalam setiap kategori maka dilakukan teknik SMOTE yaitu teknik untuk menghasilkan data sintetis dari kelas yang memiliki data minoritas.

Simran, A. dkk., (2021) melakukan penelitian dengan judul *Content Based Image Retrieval Using Deep Learning Convolutional Neural Network*. Pada penelitian ini digunakan *Convolutional Neural Network* untuk melakukan ekstraksi fitur pada gambar. Dalam penelitian ini untuk mengukur kemiripan antara gambar yaitu menggunakan Euclidean *distance* perhitungan jarak dari 2 buah titik dalam Euclidean *space*. Penulis melakukan eksperimen dengan beberapa metode untuk mendapatkan karakteristik representasi dari gambar, metode yang digunakan pertama dengan *color histogram* dengan hasil *mean Average Precision* (mAP) sebesar 68.86 dan *mean Average Recall* (mAR) sebesar 70.85, metode selanjutnya menggunakan DWT dengan hasil mAP sebesar 78.85 dan mAR sebesar 79.56, dan yang terakhir menggunakan metode DconvNet-PCA dengan hasil mAP sebesar 85.23 dan mAR sebesar 88.53. jadi metode menggunakan DconvNet-PCA adalah yang terbaik dan metode tersebut diusulkan oleh penulis.

Park, S. dkk., (2019) melakukan penelitian dengan judul *Study on Fashion Image Retrieval Methods for Efficient Fashion Visual Search*. Pada penelitian ini digunakan beberapa model *baseline* untuk dilakukan eksperimen yaitu DenseNet121, ResNet50, SEResNet50, dan SEResNeXt50 yang dimodifikasi pada lapisan *fully connected* (FC) terakhir pada masing-masing model. Dalam melakukan ekstraksi fitur digunakan model CNN lalu hasil semua vektor fitur ekstraksi dinormalisasi L2, kemudian nilai kesamaan dihitung menggunakan *inner*

product. Hasil dari eksperimen yang dilakukan menunjukkan bahwa pelatihan yang menggunakan strategi pelatihan, *module relaxtation* dan kombinasi *loss function* mengarah pada konsistensi dan peningkatan yang signifikan dalam akurasi model, baik dalam hal klasifikasi dan *retrieval performance*.

Tabel 2. 1 Perbandingan Kajian Hasil Penelitian

No	Judul	Penulis	Modul	Hasil/ Kesimpulan
1	<i>Visual Search at Alibaba</i>	Zhang, Y. dkk.	<i>Deep Metrik Embedding, Weakly supervised object localization, Joint detection, dan Feature learning.</i>	Peneliti menerapkan model yang efektif untuk melakukan prediksi kategori. Model CNN di desain untuk joint detection dan feature learning dengan cara menambang perilaku pengguna. Eksperimen ekstensif pada High Recall Set menggambarkan kinerja yang menjanjikan dari modul.
2	<i>Visual Search at Pinterest</i>	Jing, Y. dkk.	<i>Localization, Object Detection, dan Click Prediction</i>	Pengguna dapat melakukan sebuah pencarian gambar dimana dia dapat memilih objek dalam gambar yang akan dijadikan input dalam model, eksperimen menunjukkan bahwa fitur pencarian visual dapat meningkatkan keterlibatan pengguna.

Tabel 2. 1 Lanjutan Perbandingan Kajian Hasil Penelitian

No	Judul	Penulis	Modul	Hasil/ Kesimpulan
3	<i>Neural Networks for Fashion Image Classification and Visual Search</i>	Fengzi, L. dkk.	<i>Data Filtration, Data Augmentation, dan Weight Balancing</i>	Pencarian visual dapat meningkatkan pengalaman pelanggan dengan memungkinkan pengguna untuk mengunggah produk untuk mencari produk serupa. Model <i>Machine Learning</i> dapat memberi label pada foto produk yang diunggah penjual dan dapat meminimalisir kesalahan dalam memberi label.
4	<i>Content Based Image Retrieval Using Deep Learning Convolutional Neural Network</i>	Simran, A. dkk.	PCA	Hasil simulasi menunjukkan bahwa metode DconvNet-PCA yang diusulkan mencapai keunggulan efisiensi melalui akuisisi gambar yang lebih tepat. Selanjutnya, dengan menggunakan mAP dan mAR mendapatkan nilai yang tinggi daripada metode lainnya yaitu mAP sebesar 85.23 dan mAR sebesar 88.53
5	<i>Study on Fashion Image Retrieval Methods for Efficient Fashion Visual Search</i>	Park, S. dkk.	<i>L2 norm, kombinasi loss function, module relaxation</i>	Hasil dari eksperimen yang dilakukan menunjukkan bahwa pelatihan yang menggunakan strategi pelatihan, <i>module relaxation</i> dan kombinasi <i>loss function</i> mengarah pada konsistensi dan peningkatan yang signifikan dalam akurasi model, baik dalam hal klasifikasi dan <i>retrieval performance</i>

Seperti terlihat pada tabel 2.1. perbedaan dari kelima referensi dengan judul yang diangkat oleh penulis terletak pada modul yang digunakan, yaitu dengan penggunaan modul *Deep Metrik Embedding*, *Weakly supervised object localization*, *Joint detection*, *Feature learning*, *Localization*, *Object Detection*, *Click Prediction*, *Data Filtration*, *Data Augmentation*, *Weight Balancing*, *PCA*, *L2 norm*, kombinasi *loss function*, dan *module relaxation* untuk mendukung proses pencarian gambar produk dan keakuratannya, sehingga hasil pencarian menggunakan gambar produk dapat dipertanggung jawabkan.

2.2 Landasan Teori

2.2.1 Gambar Digital

Gambar adalah suatu perpaduan antara titik, garis, bidang, serta warna yang dikomposisikan dengan tujuan untuk mencitrakan sesuatu (objek gambar). Gambar digital merupakan gambar yang dihasilkan dari olah gambar di komputer, pemotretan kamera digital, atau media lain yang disimpan dalam format data tertentu. Beberapa format data gambar adalah JPEG, JPG, PNG, RAW, dll. Beberapa informasi yang disimpan pada format gambar diatas salah satunya ukuran dari gambar, lebar dan tinggi.

Dalam *computer vision* biasanya sebuah gambar memiliki fitur. Fitur adalah informasi yang diekstraksi dari gambar dalam bentuk nilai numerik yang sulit dipahami dan dikorelasikan oleh manusia. Misalkan kita menganggap gambar sebagai data, informasi yang diekstraksi dari data dikenal sebagai fitur. Umumnya, fitur yang diekstraksi dari sebuah gambar memiliki dimensi yang jauh lebih rendah daripada gambar aslinya. Pengurangan dimensi mengurangi biaya pemrosesan sekumpulan gambar. Pada dasarnya ada dua jenis fitur yang diekstrak dari gambar berdasarkan aplikasi, mereka adalah fitur lokal dan global. Fitur global menggambarkan gambar secara keseluruhan untuk menggeneralisasi seluruh objek sedangkan fitur gambar yang masih bisa dikenali oleh manusia, contohnya adalah warna, bentuk, tepi, dan lain lain. Fitur global termasuk representasi kontur, deskriptor bentuk, dan fitur tekstur sedangkan fitur lokal mewakili tekstur dalam *patch* gambar. Matriks Bentuk, Momen Invarian, Gradien Berorientasi Histogram

(HOG) dan Co-HOG adalah beberapa contoh deskriptor global. SIFT, SURF, LBP, BRISK, MSER dan FREAK adalah beberapa contoh deskriptor lokal.

2.2.2 Visual Search

Visual search adalah sebuah bidang di *computer vision* yang perkembangannya cukup pesat saat ini seiring dengan perkembangan teknologi digital. Fitur *visual search* saat ini menjadi salah satu fitur yang vital pada banyak perusahaan berbasis digital. Hal ini disebabkan karena perkembangan dan mudahnya pengguna mendapatkan gambar digital dari internet maupun kamera sendiri. Hal yang mendukung lainnya adalah semakin banyak penggunaan fitur *searching*, fitur ini tergolong sangat vital bagi berbagai macam situs di internet, misalnya Internet Explorer seperti Google dan situs *marketplace* seperti Tokopedia. Penggabungan fitur *searching* dengan berkembangnya pengolahan dan penggunaan gambar ini menciptakan sebuah bidang baru yaitu *visual search*.

Visual search menggunakan gambar dunia nyata (tangkapan layar, gambar Internet, atau foto) sebagai *input* untuk pencarian *online*. *Visual search* dikembangkan dengan membuat sebuah model yang dilatih dengan dataset yang sudah ditentukan. Mengimplementasikan *visual search* membutuhkan tiga tahap yaitu:

1. Ekstraksi fitur (seperti warna, tekstur, bentuk, merek, dll) dan kategorisasi gambar membantu mengkarakterisasi gambar dan secara signifikan mempersempit bidang pencarian.
2. *Hashing* gambar (alias *encoding*) menghasilkan kode hash numerik dan menyimpannya dalam sebuah vektor untuk mengidentifikasi setiap gambar. Hal ini pada gilirannya memungkinkan mekanisme yang sangat efisien untuk menyimpan dan menghitung representasi gambar.
3. Pencarian gambar menggunakan kode hash dan ukuran kesamaan.

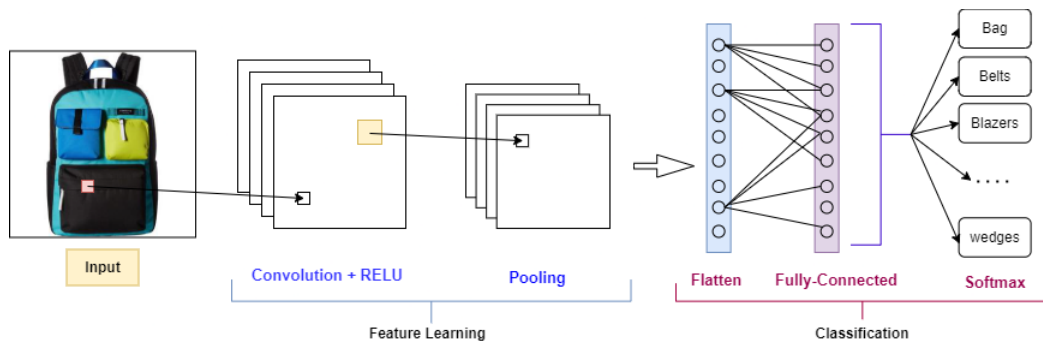
Model *visual search* biasanya menggunakan metode *Convolutional Neural Networks* (CNN) salah satu cabang dari *Deep Learning*. Dalam arsitekturnya terdiri dari beberapa modul penting yang biasanya digunakan yaitu seperti langkah pertama diatas *feature extraction*, dan yang lainnya adalah *classification*.

2.2.3 Feature Extraction

Feature extraction merupakan teknik merepresentasikan gambar dengan mengekstrak beberapa fitur pada gambar. Fitur adalah bagian kecil yang menarik, deskriptif, atau informatif. Ia bisa berupa sudut, tepi, skema warna, tekstur gambar, dan lain-lain. Salah satu kemampuan utama CNN adalah melakukan *feature extraction* pada lapisan *convolution* dan lapisan *pooling*. Lapisan *convolution* bekerja dengan prinsip *sliding window* dan *weight sharing* (mengurangi kompleksitas perhitungan). Lapisan *Pooling* digunakan untuk merangkum informasi yang dihasilkan oleh suatu *convolution* (mengurangi dimensi). Sedangkan *vector* hasil dari beberapa operasi *convolution* dan *pooling* pada *multilayer perceptron* dikenal sebagai *fully connected layer* yang digunakan untuk melakukan suatu pekerjaan (misal klasifikasi).

2.2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu jenis dari *neural network*. CNN sering digunakan untuk memproses sebuah gambar, baik untuk klasifikasi gambar, deteksi objek dalam gambar, pengenalan wajah, dan lain-lain. CNN dalam memproses sebuah gambar memerlukan masukan sebuah *array* dari gambar tersebut. Ukuran *array* tersebut tergantung dari resolusi gambar tersebut. Biasanya *array* akan berukuran $H \times W \times D$, dimana H adalah *Height* atau tinggi gambar, W adalah *Width* atau lebar gambar, dan D adalah *Dimension* atau dimensi gambar. Nilai D mengacu pada jenis gambar, gambar yang berwarna akan memiliki nilai $D = 3$ yang merepresentasikan ke nilai RGB dari gambar tersebut, sedangkan untuk $D = 1$ untuk gambar grayscale atau biner. Ilustrasi dari arsitektur *Convolutional Neural Network* dapat dilihat pada Gambar 2.1 berikut.



Gambar 2. 1 Arsitektur *Convolutional Neural Network*

Berdasarkan gambar 2.1, tahap pertama pada arsitektur CNN adalah tahap konvolusi. Tahap ini dilakukan dengan menggunakan sebuah kernel dengan ukuran tertentu. Perhitungan jumlah kernel yang dipakai tergantung dari jumlah fitur yang dihasilkan. Kemudian dilanjutkan menuju fungsi aktivasi, biasanya menggunakan fungsi aktivasi ReLU (*Rectifier Linear Unit*). Selanjutnya setelah keluar dari proses fungsi aktivasi kemudian melalui proses *pooling*. Proses ini diulang beberapa kali sampai didapatkan peta fitur yang cukup untuk dilanjutkan ke *fully connected neural network*, dan dari *fully connected network* adalah *output class*. Berikut adalah beberapa proses yang harus dilewati saat menggunakan *Convolution neural network* diantaranya :

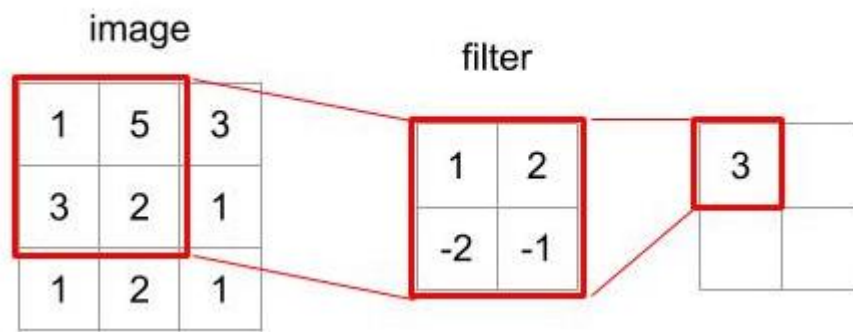
1. Convolution Layer

Convolutional layer berfungsi untuk mengenali atribut-atribut unik pada sebuah objek. Atribut-atribut yang lebih rendah membentuk atribut lebih tinggi contohnya atribut wajah dibentuk dari atribut mata, telinga, dan hidung. Atribut mata dibentuk dari garis, lengkungan dan bintik hitam.

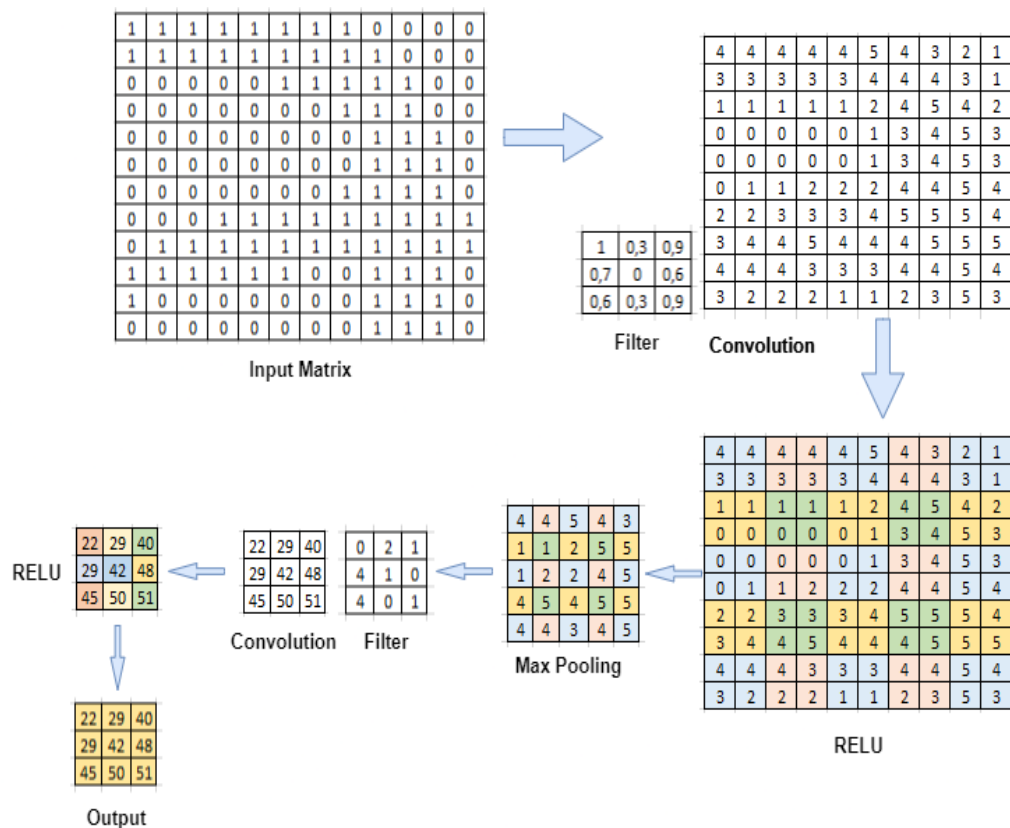
Convolutional layer menerapkan operasi *convolution* ke *input*, meneruskan hasilnya ke lapisan berikutnya. *Convolution* mengubah semua piksel dalam bidang reseptifnya menjadi satu nilai. Misalnya, jika Anda akan menerapkan *convolution* ke gambar, Anda akan mengurangi ukuran gambar serta menyatukan semua informasi di bidang menjadi satu piksel. Hasil akhir dari *convolutional layer* adalah sebuah vektor.

Proses *convolution* adalah proses yang mengaplikasikan filter pada gambar. Pada proses *convolution* ada perkalian matriks terhadap filter dan area pada gambar. Cara kerja lapisan konvolusi Dapat dilihat pada Gambar 2.2 dan Gambar 2.3.

$$(1*1) + (5*2) + (3*-2) + (2*-1) = 3$$

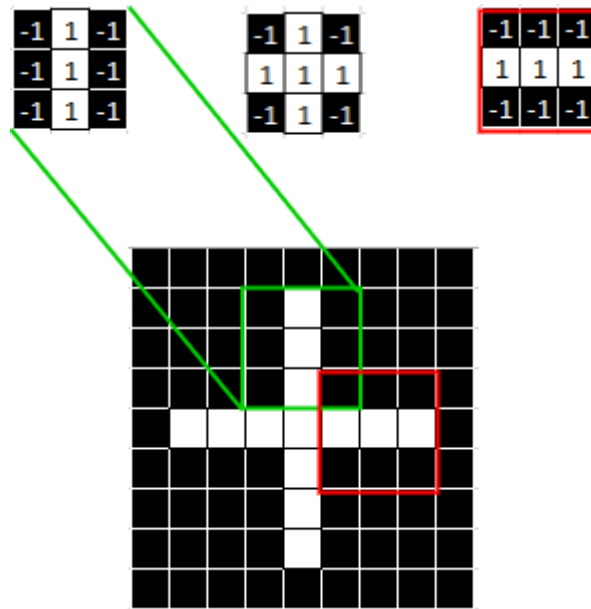


Gambar 2. 2 Proses *Convolutional Layer*



Gambar 2. 3 Proses *Convolutional Layer* Lengkap

Filter hanyalah sebuah matriks yang berisi angka-angka, seperti Gambar 2.4 dibawah ini.



Gambar 2. 4 Proses Filter

Pada gambar 2.4, terdapat 3 buah filter yang merupakan matriks 3x3 dan sebuah objek gambar berupa huruf X. Filter yang berada di sebelah kiri digambarkan dapat mengenali garis yang terdapat pada kotak hijau. Setiap filter berbeda dapat mengenali atribut yang berbeda seperti, filter di kanan dapat mengenali atribut objek x yang berada di kotak merah.

Kita dapat membedakan seekor burung dan kucing berdasarkan bentuknya, dengan filter seperti pada gambar yang paling kanan, kita dapat mendeteksi garis-garis yang bisa menunjukkan apakah itu merupakan seekor burung atau kucing berdasarkan bentuk garisnya.

2. Pooling Layer

Pooling layer adalah layer baru yang biasanya ditambahkan setelah lapisan *convolution*. *Pooling layer* digunakan untuk mengurangi dimensi *array* dengan tetap mempertahankan informasi pada gambar. Dengan demikian, ini mengurangi jumlah parameter untuk dipelajari dan jumlah komputasi yang dilakukan dalam jaringan. Seperti pada Gambar 2.5 di mana ketika resolusi dikurangi sampai batas tertentu kita masih bisa mendapatkan informasi mengenai objek pada gambar.



Gambar 2. 5 Contoh *Pooling Layer*

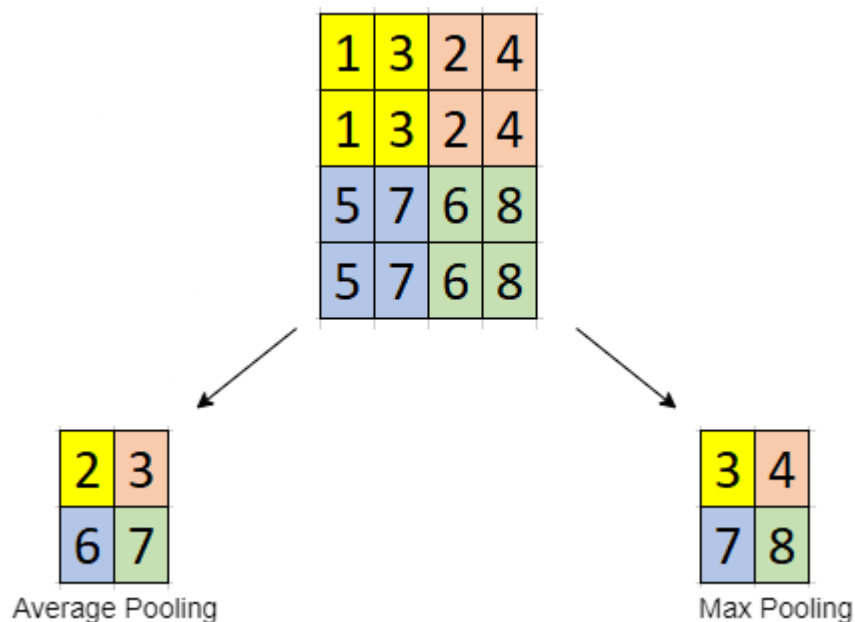
Dalam pooling layer terdapat dua macam pooling yang biasa digunakan.

- *Max Pooling*

Operasi *pooling* yang mengambil nilai maksimum dari *array* gambar

- *Average Pooling*

Operasi *pooling* yang mengambil nilai rata-rata dari *array* gambar.



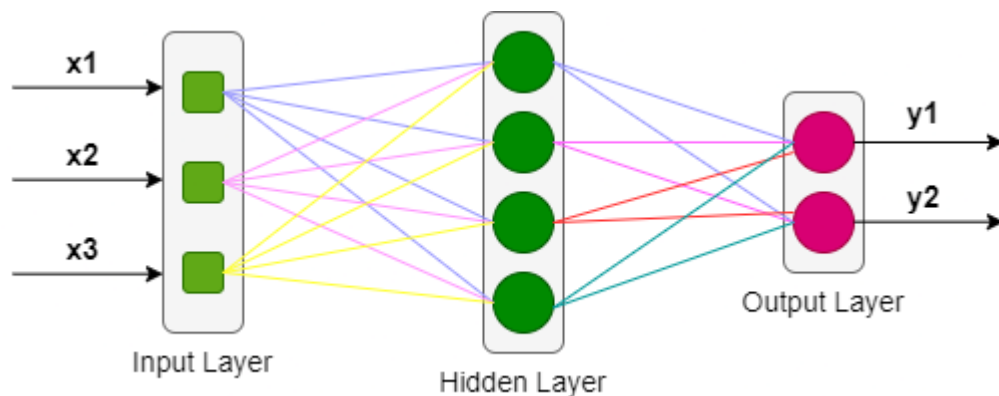
Gambar 2. 6 Operasi *Pooling Layers*

Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan

menggunakan *filter* dengan ukuran 2×2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari *inputnya*. Proses *pooling* dipakai karena pada praktiknya, jumlah filter yang digunakan pada proses *convolution* berjumlah banyak. Jika menggunakan 128 filter pada konvolusi maka akan menghasilkan 128 gambar baru. *Pooling* membantu mengurangi ukuran dari setiap gambar dari proses *convolution*.

3. Fully-Connected Layer

Activation map yang dihasilkan dari lapisan ekstraksi fitur masih berbentuk multidimensional *array*, sehingga mau tidak mau harus dilakukan *reshape activation map* menjadi sebuah vektor agar bisa digunakan sebagai *input* dari *fully-connected layer*. Ketika memasuki lapisan ini *array* gambar akan di ratakan atau flattened, yaitu *array* gambar akan dijadikan 1 dimensi saja. *Array* yang berupa vektor fitur dimasukkan kedalam lapisan ini. Vektor tersebut $\langle x_1, x_2, x_3, x_n \rangle$ akan dimasukkan ke dalam lapisan ini untuk menghasilkan sebuah output $\langle y_1, y_2 \rangle$. Fungsi aktivasi akan digunakan untuk mendapatkan nilai output. Dari *array* output inilah kita dapat gunakan untuk tujuan CNN. Dapat dilihat pada Gambar 2.7 dibawah ini.

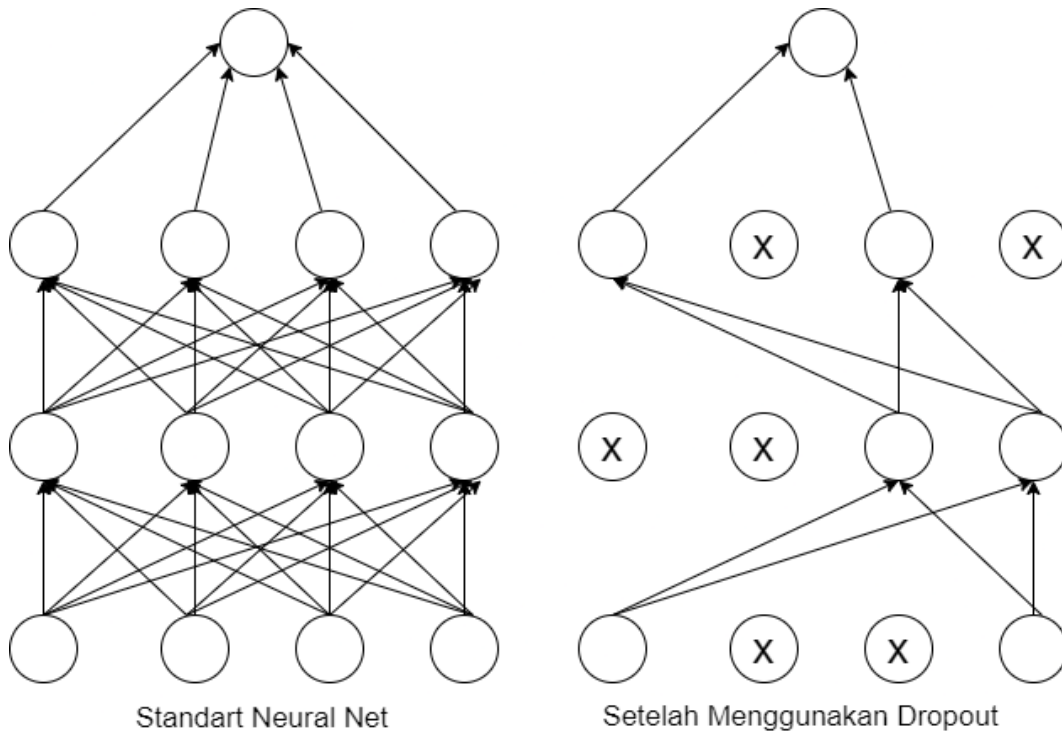


Gambar 2. 7 Proses *fully-connected layer*

4. Dropout Layer

Dropout bekerja dengan mengurangi kompleksitas model jst tanpa merubah arsitektur model tersebut. Nama *dropout* mengacu pada unit/perseptron yang di-*dropout* (dibuang) secara temporer pada sebuah layer. Hal ini berarti bahwa kontribusi neuron yang dibuang akan diberhentikan sementara jaringan dan bobot

baru juga tidak diterapkan pada neuron pada saat melakukan *backpropagation*. Berikut adalah gambar proses *dropout* dapat dilihat pada Gambar 2.8.



Gambar 2. 8 *Dropout Layer*

5. Batch Normalization

Batch normalization adalah teknik untuk melatih jaringan saraf yang sangat dalam yang menormalkan kontribusi ke lapisan untuk setiap mini-batch. Ini berdampak pada penyelesaian proses pembelajaran dan secara drastis mengurangi jumlah periode pelatihan yang diperlukan untuk melatih. *Batch normalization* melakukan penskalaan output lapisan, secara eksplisit dengan menormalkan aktivasi setiap variabel input per mini-batch, misalnya, berlakunya node dari lapisan terakhir. Tinjau bahwa normalisasi mengacu pada penskalaan ulang data agar memiliki rata - rata nol dan standar deviasi satu.

6. Loss Layer

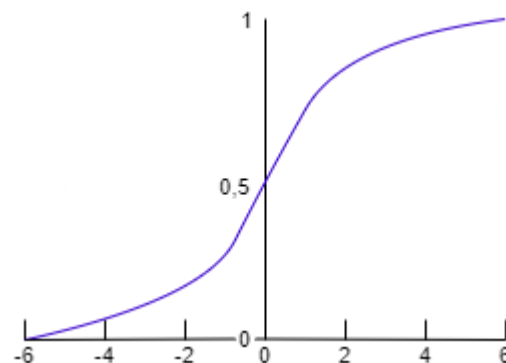
Seluruh JST selalu diakhiri dengan sebuah *loss layer*, yaitu *layer* yang bertanggung jawab menghitung error dari jaringan dalam proses pelatihan sekaligus melakukan fungsi khusus jaringan. Untuk kasus klasifikasi yang digunakan pada umumnya adalah softmax atau sigmoid. Keduanya digunakan biasanya tergantung

dari jumlah kelas *output*, semisal jika hanya terdapat dua atau binary maka digunakan sigmoid dan jika lebih dari itu biasanya digunakan softmax.

2.2.5 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi yang ditambahkan ke jaringan saraf tiruan untuk membantu jaringan mempelajari pola kompleks dalam data. Ketika membandingkan dengan model berbasis neuron yang ada di otak kita, fungsi aktivasi pada akhirnya memutuskan apa yang akan dikeluarkan ke neuron berikutnya. Itulah tepatnya yang dilakukan fungsi aktivasi dalam JST juga. Ini mengambil sinyal output dari sel sebelumnya dan mengubahnya menjadi beberapa bentuk yang dapat diambil sebagai input ke sel berikutnya. Berikut beberapa fungsi aktivasi yang sering digunakan dalam CNN.

1. Sigmoid



Gambar 2. 9 Grafik Fungsi Sigmoid

Fungsi sigmoid memiliki rumus sebagai berikut:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.1)$$

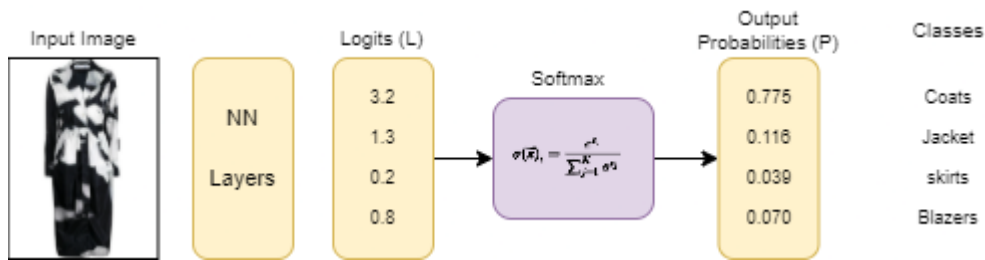
Grafik fungsi sigmoid dapat dilihat pada Gambar 2.8. Fungsi sigmoid adalah salah satu fungsi aktivasi yang paling banyak digunakan saat ini. Jika diperhatikan, antara nilai $x = -2$ hingga 2 , nilai y sangat curam. Artinya, setiap perubahan kecil pada nilai x di wilayah tersebut akan menyebabkan nilai y berubah secara signifikan itu berarti fungsi ini memiliki kecenderungan untuk membawa nilai y ke kedua ujung kurva. Hal ini cenderung membawa aktivasi ke kedua sisi kurva (di atas $x = 2$ dan di bawah $x = -2$ misalnya). Membuat perbedaan yang jelas

pada prediksi. Keuntungan lain dari fungsi aktivasi ini adalah *output* dari fungsi aktivasi akan selalu berada dalam rentang (0, 1).

Fungsi sigmoid memiliki masalah yaitu jika diperhatikan di salah satu ujung fungsi sigmoid, nilai Y cenderung kurang merespons perubahan X yang artinya gradien di wilayah itu akan menjadi kecil. Ini menimbulkan masalah "*vanishing gradients*". Jadi yang terjadi ketika aktivasi mencapai dekat bagian hampir horizontal dari kurva di kedua sisi maka menyebabkan efek *saturating gradients* dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu ekstrim sehingga memiliki gradient yang mendekati nol. Jika hal tersebut terjadi, maka neuron tersebut tidak akan dapat mengalami update yang signifikan dan akan nonaktif.

2. Softmax

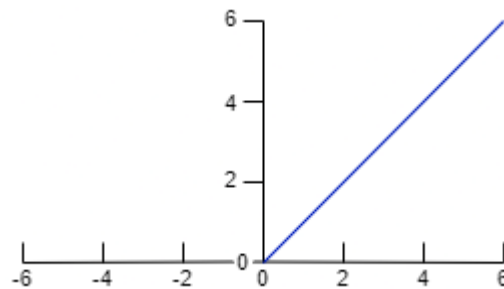
Softmax adalah sebuah fungsi aktivasi yang di gunakan untuk permasalahan klasifikasi, biasanya fungsi aktivasi ini digunakan pada output layer. Pada dasarnya fungsi ini adalah probabilitas eksponensial yang dinormalisasi dari pengamatan kelas yang di wakili sebagai aktivasi neuron. Fungsi eksponensial akan meningkatkan probabilitas nilai maksimum lapisan sebelumnya dibandingkan dengan nilai lainnya. Softmax *function* adalah perhitungan kemungkinan dari masing-masing kelas target atas semua kelas target yang memungkinkan dan membentuk untuk menentukan kelas target untuk input yang diberikan. Nilai *input* bisa positif, negatif, nol, atau lebih besar dari satu, tetapi softmax mengubahnya menjadi nilai antara 0 dan 1, sehingga dapat diinterpretasikan sebagai probabilitas. Jika salah satu inputnya kecil atau negatif, softmax mengubahnya menjadi probabilitas kecil, dan jika inputnya besar, maka itu mengubahnya menjadi probabilitas besar, tetapi akan selalu tetap antara 0 dan 1. Ketika softmax digunakan untuk model klasifikasi multi, maka akan mengembalikan peluang dari masing-masing kelas dan kelas target akan memiliki probabilitas lebih tinggi dari kelas yang yang lain. Berikut contoh perhitungan menggunakan softmax pada Gambar 2.10.



Gambar 2. 10 Menggunakan Aktivasi Softmax

3. ReLU

Relu adalah fungsi aktivasi dalam jaringan saraf tiruan yang banyak digunakan. Pada dasarnya, fungsi *Rectified Linear Unit* (ReLU) adalah "*threshold*" dari 0 hingga tak terhingga. Diagram fungsi relu dapat dilihat pada Gambar 2.11.



Gambar 2. 11 Diagram fungsi relu

Relu memiliki rumus berikut:

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

Dimana x adalah neuron input, yang juga disebut fungsi ramp, mirip dengan penyearah setengah gelombang dalam teknik elektro. Unit linier yang dikoreksi memiliki output 0 jika input kurang dari 0, dan output mentah dinyatakan. Artinya, jika input lebih besar dari 0, outputnya sama dengan inputnya. ReLu berfungsi lebih seperti neuron manusia. Aktivasi ReLu pada dasarnya adalah fungsi aktivasi nonlinier yang paling sederhana. Jika mendapatkan input positif, turunannya hanya 1, dengan kata lain nilai aktivasi hanya memiliki ambang batas 0. Penelitian menunjukkan bahwa ReLu dapat mempercepat penelitian jaringan.

2.2.6 ADAM

ADAM adalah algoritma optimisasi yang dapat digunakan sebagai ganti dari prosedur classical stochastic gradient descent untuk memperbarui bobot secara

iteratif yang didasarkan pada data training. ADAM berbeda dengan algoritma Stochastic Gradient Descent yang memiliki *learning rate* yang tidak berubah setiap pembaruan bobot. Algoritma ADAM menghitung rata – rata bergerak dari gradien dan gradien kuadrat. Parameter beta1 dan beta2 mengendalikan laju pengurangan (*decay rate*) dari rata – rata bergerak. (Brownlee, 2017). Berikut adalah langkah-langkah perhitungan menggunakan ADAM, tapi sebelumnya ada beberapa nilai yang harus diinisialisasikan yaitu sebagai contoh:

- $m = 0$
- $v = 0$
- $\epsilon = 10^{-8}$
- $t = 0$
- $\alpha = 0.001$
- $\beta_1 = 0.9$
- $\beta_2 = 0.999$

Selanjutnya tahap-tahap yang dilakukan yaitu:

1. Tambah t setiap iterasi.

$$t = t + 1 \quad (2.3)$$

2. Menghitung gradient.

$$g_t \leftarrow \nabla \theta f_t(\theta_t - 1) \quad (2.4)$$

3. Menghitung bias first moment.

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.5)$$

4. Menghitung bias second moment.

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t \cdot g_t \quad (2.6)$$

5. Menghitung bias first moment.

$$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t} \quad (2.7)$$

6. Menghitung bias second moment.

$$\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t} \quad (2.8)$$

7. Memperbaiki parameter.

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (2.9)$$

Keterangan:

t = iterasi

g = gradient

m = *first moment*

v = *second moment*

β_1, β_2 = *Exponential decay rates*

α = *Step size* atau *learning rate*

θ = parameter yang akan diperbaiki (pada kasus ini adalah bobot)

Dari ketujuh tahap tersebut diulang sebanyak jumlah dataset yang diambil secara acak hingga semua epoch telah selesai. Perbedaan antara Adam dengan RMSProp yaitu ada pada perubahan step size pada awal perubahan parameter dikarenakan adam melakukan bias correction pada perhitungannya.

2.2.7 RMSProp

RMSprop adalah teknik optimasi berbasis gradien yang digunakan dalam pelatihan jaringan saraf. RMSprop menggunakan learning rate adaptif dan ini berarti bahwa kecepatan belajar berubah dari waktu ke waktu mempertahankan rata-rata dari kuadrat gradient untuk setiap bobot. Berikut adalah rumus dari RMSprop yaitu:

$$\text{MeanSquare}(w, t) = \rho * \text{MeanSquare}(w, t - 1) + 0.1 \left(\frac{\partial E}{\partial w}(t) \right)^2 \quad (2.10)$$

Keterangan:

w = bobot

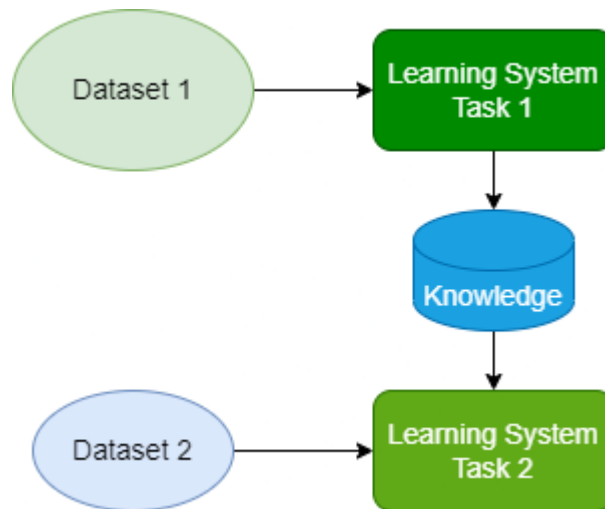
t = *timestamp*

$\rho = 0.9$

$\frac{\partial E}{\partial w}$ = gradient

2.2.8 Transfer Learning

Transfer learning adalah metode pembelajaran mesin di mana model yang dikembangkan untuk suatu tugas digunakan kembali sebagai titik awal untuk model pada tugas kedua. Diagram *transfer learning* dapat dilihat pada Gambar 2.12 dibawah ini.



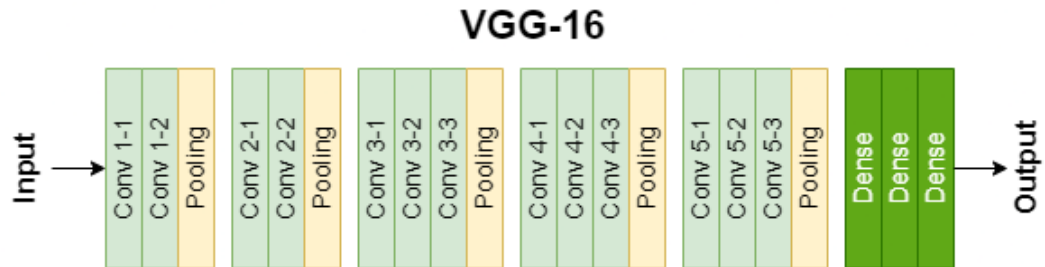
Gambar 2. 12 Proses *Transfer Learning*

Terdapat beberapa model yang sudah sangat umum digunakan dalam proses transfer learning untuk melakukan visual search, model-model tersebut adalah:

1. VGG16

VGG adalah model yang dirilis oleh Karen Simonyan pada tahun 2014. Andrew Zisserman dari Oxford Large Visual Robotics Institute Recognition Challenge 2014 (ILSVRC2014) telah mendapat skor 92.7% Akurasi top-5 di ImageNet, ImageNet sendiri adalah salah satu kumpulan data terbesar Ini dalam lingkup gambar, ada sekitar 14 juta data. VGG adalah Model yang mengalahkan AlexNet saat pertama kali dirilis. VGG sendiri lagi Mengedepankan aspek penting dari CNN, yaitu aspek kedalaman. VGG memiliki 138 juta parameter. VGG16 adalah model CNN yang menggunakan Lapisan konvolusi dengan spesifikasi filter konvolusi kecil (3×3), dengan ukuran tersebut, kedalaman jaringan saraf dapat ditambah dengan lebih banyak lagi lapisan konvolusi. Hasilnya model menjadi lebih akurat daripada model-model yang ada sebelumnya. Model VGG16 memiliki

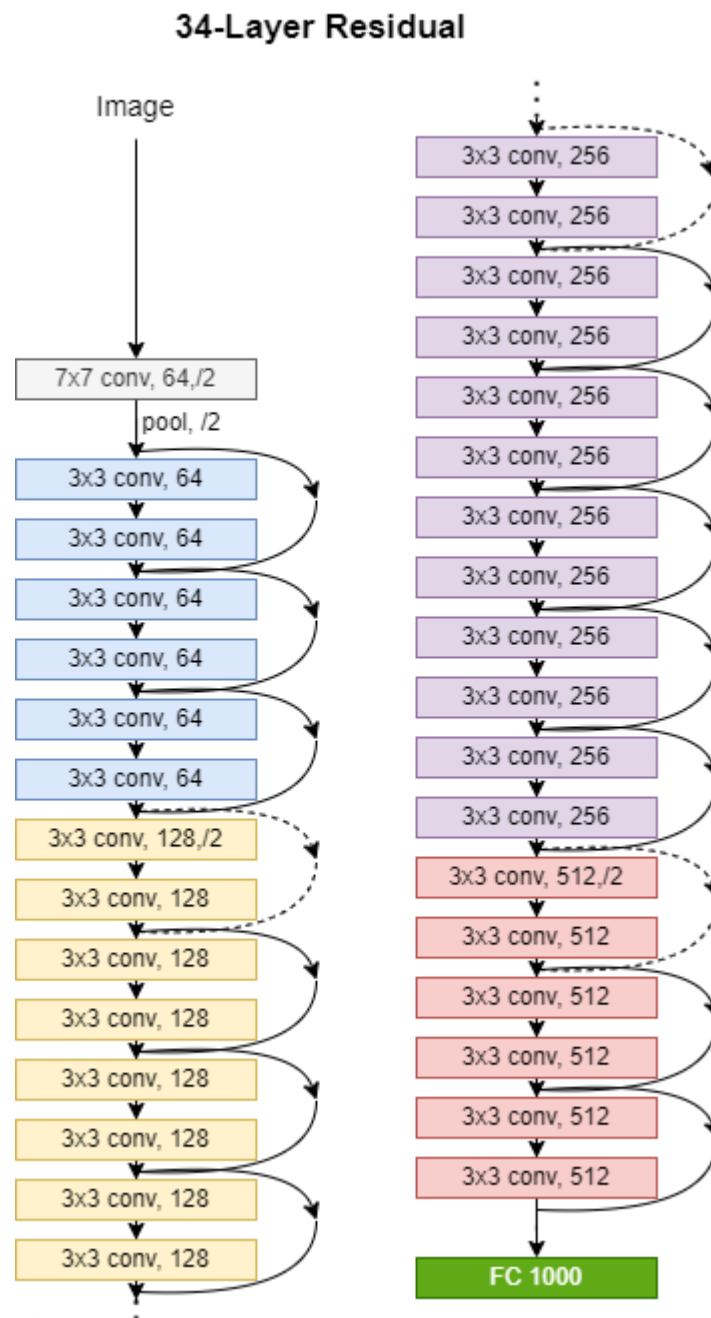
19 *layer*/lapisan yang terdiri dari 16 lapisan konvolusi dan 3 *fully-connected layer*. Berikut arsitektur dari model VGG16 bisa dilihat pada Gambar 2.13.



Gambar 2. 13 Arsitektur VGG16

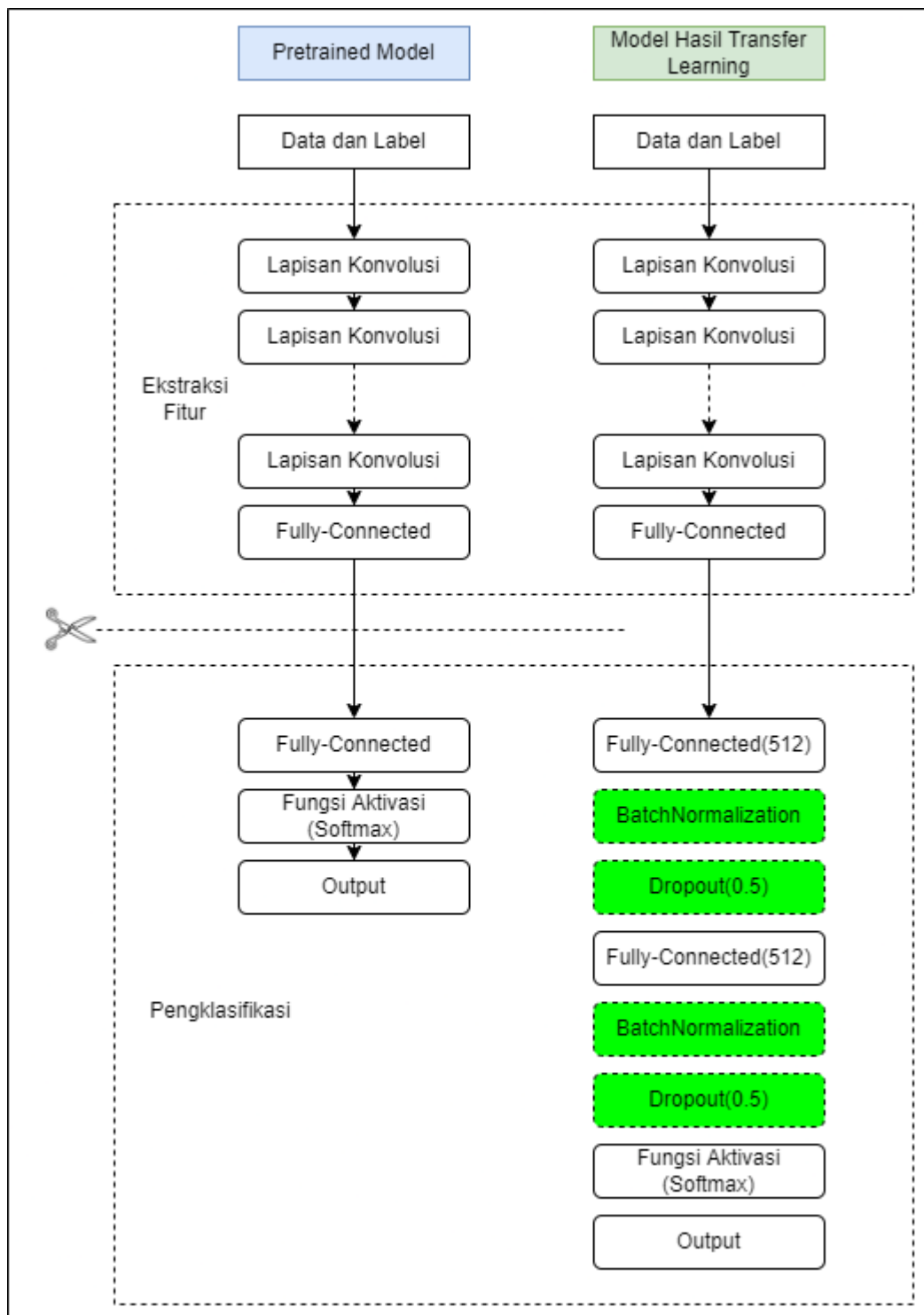
2. ResNet50

Resnet adalah model CNN yang dirancang untuk memecahkan masalah *vanishing gradient* ketika terlalu banyak *layer* yang bertumpuk sehingga mengurangi efektivitas lapisan jaringan yang lebih dalam itu sendiri. Permasalahan ini jika model semakin dalam (lebih banyak *convolutional layer* dalam model) akan menyebabkan penurunan kinerja karena nilai gradien yang semakin kecil. Resnet dapat memiliki ratusan hingga ribuan *convolutional layer* tanpa mengurangi efektivitas *layer* tersebut. Arsitektur ResNet Ini mirip dengan VGG, bedanya adalah jumlah *convolutional layer* yang lebih banyak dan residual network yang menerapkan solusi *identity shortcut connection*. ResNet50 memiliki kedalaman 50 *layer*, di bawah ini pada Gambar 2.14 adalah arsitektur ResNet50 dengan residual 34 *layer*.



Gambar 2. 14 Arsitektur ResNet50

Pada model yang menggunakan *transfer learning* dilakukan sebuah perubahan pada tahap klasifikasinya namun pada tahap ekstraksi fitur atau proses konvolusi digunakan susunan model aslinya seperti VGG dan ResNet. Berikut adalah arsitektur dari model yang menggunakan *transfer learning* dapat dilihat pada Gambar 2.15.



Gambar 2. 15 Arsitektur yang Diusulkan Memanfaatkan Pre-trained Model

2.2.9 Confusion Matrix

Confusion matrix adalah pengukuran kinerja untuk masalah klasifikasi *Machine Learning* yang *output*-nya bisa dua kelas atau lebih. *Confusion matrix*

adalah tabel dengan 4 kombinasi nilai prediksi dan nilai aktual yang berbeda. Ada empat item dalam *confusion matrix* yang mewakili hasil proses klasifikasi, yaitu *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Berikut adalah tabel *confusion matrix* pada Gambar 2.16.

		Nilai Aktual	
		Positive	Negative
Nilai Prediksi	Positive	TP	FP
	Negative	FN	TN

Gambar 2. 16 *Confusion Matrix*

Berikut analogi dan penjelasannya:

- *True Positive* (TP): Pada prediksi bernilai positif dan nilai sebenarnya juga positif.

Analogi: Memprediksi tim sepak bola favorit menang, dan memang ternyata menang.

- *False Positive* (FP): Pada prediksi bernilai negatif dan nilai sebenarnya juga negatif.

Analogi: Memprediksi tim sepak bola favorit kalah, namun ternyata menang.

- *False Negative* (FN): Pada prediksi bernilai negatif tapi nilai sebenarnya adalah positif.

Analogi: Memprediksi tim sepak bola favorit kalah, namun ternyata menang.

- *True Negative* (TN): Pada prediksi bernilai negatif dan nilai sebenarnya juga negatif.

Analogi: Memprediksi tim sepak bola favorit kalah, dan memang ternyata kalah.

Evaluasi dengan confusion matrix menghasilkan nilai akurasi, presisi, *recall*, dan *f1-score*. Berikut merupakan penjelasan hal tersebut adalah:

- Akurasi

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.11)$$

Akurasi adalah ukuran keakuratan model saat menggunakan data aktual untuk memprediksi data. Akurasi dapat dihitung dengan rumus di atas. Kelebihan dari metrik ini adalah sering digunakan untuk membuat model klasifikasi, baik itu klasifikasi dua kelas maupun kategori. Kerugian dari indikator ini adalah dapat "menyesatkan" data yang tidak seimbang.

- Presisi

$$\text{Presisi} = \frac{TP}{TP + FP} \quad (2.12)$$

Presisi adalah metrik dalam kasus klasifikasi, yang digunakan untuk menghitung efek model dalam memprediksi label positif terhadap semua label positif model. Kelebihan dari metrik ini adalah fokusnya pada performa model (prediksi) untuk label data positif, sedangkan kelemahan metrik ini adalah tidak mempertimbangkan label negatif.

- *Recall*

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.13)$$

Recall adalah metrik dalam kasus klasifikasi, yang digunakan untuk menghitung efek model dalam memprediksi label positif untuk semua label data positif. Keuntungan metrik ini adalah menghitung bagian negatif dari prediksi label positif (tidak sama dengan akurasi). Tetapi kelemahannya adalah ketika semua prediksi = 1, *recall* akan memiliki nilai 1 (prediksi negatif tidak dipertimbangkan).

- *F1-Score*

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{presisi}) / (\text{Recall} + \text{Presisi}) \quad (2.14)$$

F1-Score merupakan metrik dalam kasus klasifikasi yang digunakan untuk menghitung seberapa baik hasil prediksi model (precision) dan seberapa lengkap hasil prediksinya (recall). Kelebihan dari metrik ini menutup semua kekurangan

yang ada pada precision dan recall. Namun kekurangannya adalah f1-score tidak memperhitungkan hasil prediksi benar pada label negatif.

2.2.10 Measure Distance

Measure distance digunakan untuk menentukan kemiripan sebuah gambar dengan gambar lainnya. Dalam menentukan kemiripan diperlukan mengukur kedekatan gambar tersebut dengan membandingkan fitur-fitur yang ada di gambar. Kedekatan suatu gambar dapat ditentukan dengan menghitung kedekatan setiap objek yang ada pada gambar tersebut. Salah satu cara untuk menghitung *measure distance* adalah menggunakan metode *Eucidean Distance*. *Eucidean distance* merupakan salah satu parameter yang digunakan untuk merepresentasikan tingkat kemiripan antara dua buah citra. *Eucidean distance* adalah akar dari jumlah kuadrat perbedaan nilai untuk tiap variabel. Perhitungan citra menggunakan rumus *Euclidean* dapat direpresentasikan pada formula dibawah ini.

$$\text{Citra 1 } (v_k) = (v_{k1}, v_{k2}, v_{k3}, \dots v_{kn})$$

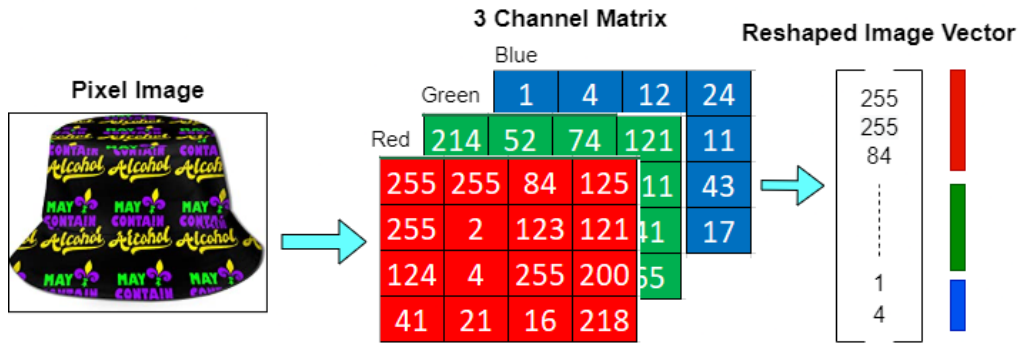
$$\text{Citra 2 } (v_l) = (v_{l1}, v_{l2}, v_{l3}, \dots v_{ln})$$

$$j(v_k, v_l) = \sqrt{\sum_{k,l=1}^N (v_k - v_l)^2} \quad (2.15)$$

Dalam hal ini, v_k dan v_l adalah dua vektor yang jaraknya akan dihitung dengan N menyatakan dimensi vektor.

2.2.11 Vektor

Vektor dalam ilmu fisika dan matematika didefinisikan sebagai suatu obyek geometri yang memiliki besar dan arah dalam ruang. Dalam pemrograman komputer, pengertian vektor adalah gambar digital hasil kombinasi titik dan garis melalui proses rumus matematika sehingga membentuk poligon yang menghasilkan objek gambar tertentu, vektor berbentuk penunjuk (*pointer*) atau larik (*array*) dengan hanya satu dimensi. Berikut pada Gambar 2.17 terdapat ilustrasi gambar digital dirubah menjadi vektor *array* 1 dimensi.



Gambar 2. 17 Gambar Digital Menjadi Vektor 1 Dimensi

Panjang sebuah vektor dalam ruang euklidian dapat dihitung dengan rumus 2.16 dibawah ini.

$$||a|| = \sqrt{a_1^2 + a_2^2 + a_3^2} \quad (2.16)$$





Berdasarkan ruang yang ditempati, vektor membaginya berdasarkan besar dan arahnya. Setidaknya ada 4 jenis vektor antara lain yaitu vektor posisi, vektor berlawanan, vektor nol, dan vektor posisi. Pada penelitian ini akan digunakan sebuah vektor berjenis satuan yang berarti suatu vektor dengan panjang ‘satu satuan’ biasanya vektor satuan hanya digunakan untuk menunjukkan arah. Untuk mendapatkan vektor satuan dapat dilakukan dengan melakukan normalisasi vektor dengan cara suatu vektor dengan panjang sembarang dibagi oleh panjangnya. Berikut formula untuk menormalisasi sebuah vektor.

$$\hat{a} = \frac{a}{||a||} \quad (2.17)$$

2.2.12 ERD (Entity Relationship Diagram)

Entity relationship diagram atau diagram hubungan entitas merupakan data berupa notasi grafis dalam pemodelan data konseptual yang menggambarkan hubungan antara penyimpanan. Model data merupakan sekumpulan cara untuk menggambarkan data-data yang memiliki hubungan satu sama lain, semantiknya, serta batasan konsistensi. Model data terdiri dari model hubungan entitas dan model hubungan. Berikut adalah notasi-notasi dan keterangannya yang terdapat pada ERD dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Notasi-notasi ERD

Notasi	Keterangan
	Entitas merupakan suatu objek yang dapat diidentifikasi dalam lingkungan pemakai
	Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja untuk menjelaskan suatu proses yang terjadi.
	Atribut berfungsi mendeskripsikan karakter entitas(atribut yang berfungsi sebagai key diberi garis bawah)
	Garis sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

2.2.13 Python

Menurut pengertian dari Python Software Foundation (2016), Python adalah bahasa pemrograman interpretatif, berorientasi objek dan semantik yang dinamis. Python memiliki high-level struktur data, dynamic typing dan dynamic binding. Python memiliki sintaks sederhana dan mudah dipelajari untuk penekanan pada kemudahan membaca dan mengurangi biaya perbaikan program. Python mendukung modul dan paket untuk mendorong k modularan program dan code reuse. Interpreter Python dan standard library-nya tersedia secara gratis untuk semua platform dan dapat secara bebas disebar.

Bahasa pemrograman ini muncul pertama kali pada tahun 1991 oleh Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa pemrograman Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Python juga memiliki dukungan komunitas yang aktif sehingga para developer bisa dengan mudah bertanya tentang permasalahan yang dialami, selain itu Python juga menyediakan banyak framework, dan library yang bisa diunduh secara gratis.

2.2.14 Basis Data MySQL

Basis data atau *database* suatu kumpulan data-data yang disusun sedemikian rupa sehingga membentuk informasi yang berguna. Basis data terdiri dari sekumpulan data dengan jenis/karakter yang sama contohnya adalah data-data berupa nama-nama, kelas-kelas, alamat-alamat. Semua data ini dikumpulkan dalam kelompok data baru yang disebut data mahasiswa, demikian pula kumpulan data mahasiswa, data dosen, data keuangan, dan data lainnya dapat dikumpulkan kembali ke dalam kelompok besar. Bahkan dalam proses pengembangannya, data tersebut dapat berupa berbagai bentuk data, seperti program, formulir entri data, laporan, dan lain-lain. Semua ini dapat dikumpulkan dalam basis data.

Salah satu basis data yang populer digunakan saat ini adalah basis data MySQL. MySQL adalah suatu perangkat lunak basis data relasi (*Relational Database Management System* atau RDBMS) seperti halnya MongoDB, Postgresql, Oracle dan sebagainya. Dalam mengoperasikan MySQL digunakanlah sebuah bahasa bernama SQL. SQL atau singkatan dari *Structured Query Language* ialah suatu sintaks perintah-perintah tertentu atau bahasa pemrograman yang digunakan untuk mengelola suatu basis data. Jadi, MySQL dan SQL itu tidaklah sama. Singkatnya, MySQL ialah perangkat lunaknya dan SQL adalah bahasa perintahnya.

2.2.15 Tensorflow

TensorFlow adalah pustaka perangkat lunak *open source* dan gratis untuk pembelajaran mesin. TensorFlow dapat digunakan dalam berbagai tugas tetapi memiliki fokus khusus pada pelatihan dan inferensi jaringan *deep neural*. Tensorflow adalah pustaka matematika simbolis berdasarkan dataflow dan pemrograman. Saat ini, TensorFlow merupakan pustaka pembelajaran mesin paling terkenal di dunia.

Diciptakan oleh tim Google Brain, produk Google satu ini, menggunakan pembelajaran mesin di semua produknya untuk meningkatkan mesin telusur, terjemahan, pemberian keterangan gambar, atau rekomendasi. Ia menggabungkan banyak model dan algoritma *Machine Learning* termasuk *Deep Learning* (neural

network). Framework di susun menggunakan Python front-end API untuk membuat suatu aplikasi penggunaannya, dan menggunakan C++ yang memiliki kinerja terbaik dalam hal eksekusi.

Tensorflow dapat melatih dan menjalankan neural network untuk keperluan mengklasifikasikan tulisan tangan, pengenalan gambar/object, serta menggabungkan suatu kata. Selanjutnya adalah re-current neural network, yang merupakan model sequential, dapat digunakan untuk Natural Language Processing (NLP), PDE (Partial Differential Equation) berdasarkan simulasi. Dan yang paling utama adalah bahwa Tensorflow dapat digunakan pada skala yang besar untuk produksi dengan menggunakan model yang sama pada ketika proses training data.

2.2.16 Flask

Flask adalah kerangka kerja web atau *framework* yang ditulis dengan Python dan diklasifikasikan sebagai *microframework*. Flask bertindak sebagai kerangka aplikasi dan tampilan dari suatu web. Dengan menggunakan Flask dan Python, pengembang dapat membuat jaringan terstruktur dan mengelola behaviour jaringan dengan lebih mudah. Flask termasuk dalam jenis *microframework* karena tidak memerlukan tools atau library tertentu dalam penggunaannya. Fungsi dan komponen yang paling umum seperti validasi formulir dan database tidak diinstal secara default di Flask. Hal ini karena fungsi dan komponen tersebut telah disediakan oleh pihak ketiga, dan Flask dapat menggunakan ekstensi untuk membuat fungsi dan komponen tersebut terlihat seperti diimplementasikan oleh Flask itu sendiri. Selain itu, meskipun Flask disebut sebagai *microframework*, bukan berarti tidak memiliki fungsionalitas. *Microframework* disini berarti bahwa Flask bermaksud untuk membuat sebuah core dari aplikasi dengan sesederhana mungkin tapi tetap dapat dengan mudah ditambahkan. Dengan begitu, fleksibilitas serta skalabilitas dari Flask dapat dikatakan cukup tinggi dibandingkan dengan framework lainnya.

BAB III

METODE PENELITIAN

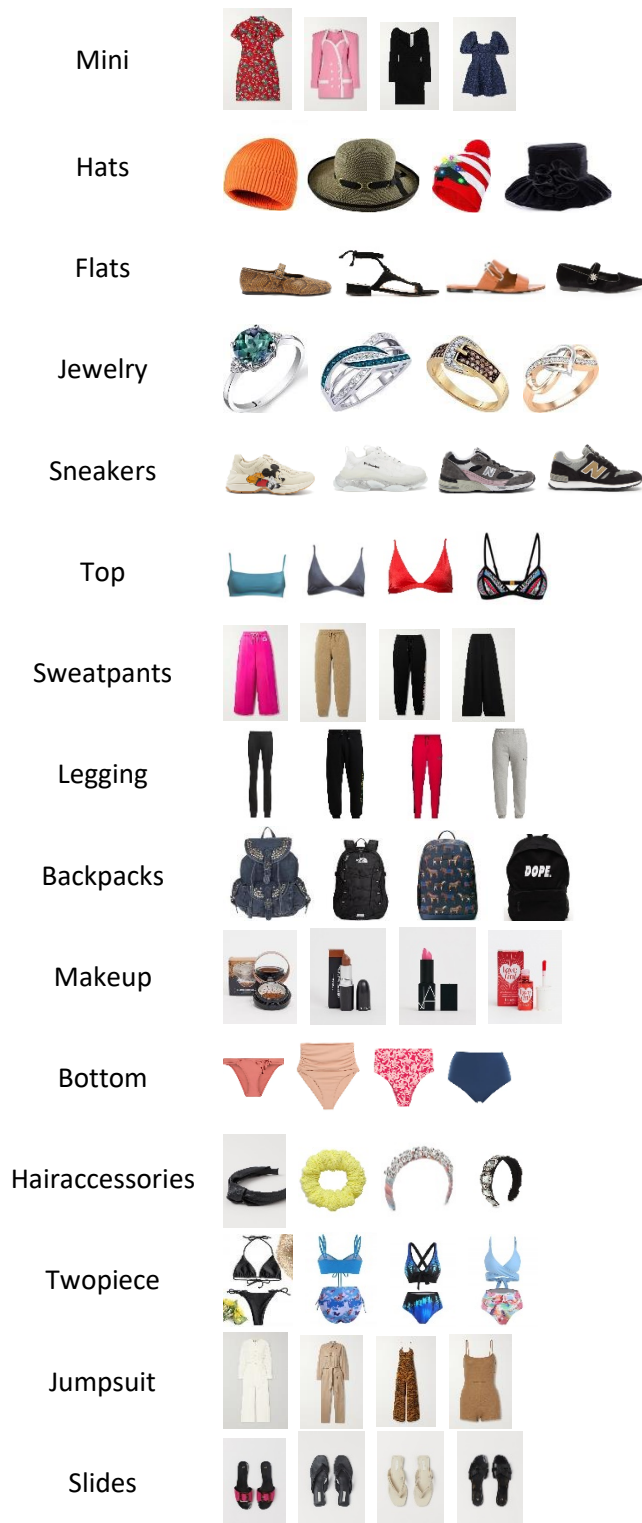
3.1 Bahan/Data

3.1.1 Data yang diperoleh

Pada pembangunan sistem *visual search* ini, dataset yang digunakan adalah dataset *Real Fashion* yang berasal dari website kaggle (<https://www.kaggle.com/hammaadali/real-fashion>). Dataset *Real Fashion* berisi 30.276 data gambar dan terdiri dari 7 kategori dengan total memiliki 42 sub-kategori didalamnya. 42 sub kategori dalam dataset terdiri dari objek yang biasa ditemui di marketplace pada kategori *fashion*. Karena gambar diperoleh dari produk yang ada di marketplace maka rata-rata gambar memiliki background yang bersih dan berfokus pada produk. Berikut sampel data dari semua sub-kategori berisikan nama kategori dan empat sampel gambar dapat dilihat pada Gambar 3.1, Gambar 3.2, Gambar 3.3, dan Gambar 3.4.



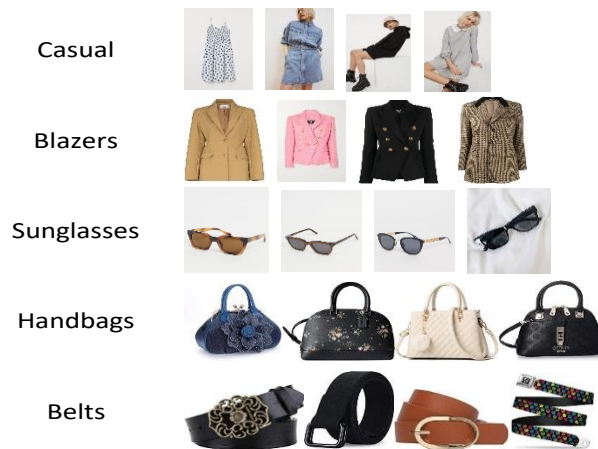
Gambar 3. 1 Data Sub-kategori Bagian 1



Gambar 3. 2 Data Sub-kategori Bagian 2



Gambar 3. 3 Data Sub-kategori Bagian 3



Gambar 3. 4 Data Sub-kategori Bagian 4

Berikut adalah daftar label pada sub-kategori dataset beserta jumlah gambar pada setiap label tersebut dapat dilihat pada tabel 3.1.

Tabel 3. 1 Banyak Data Setiap Label

No	Label	Jumlah Gambar
1	Shorts	411
2	Caps	800
3	Satchel	248
4	Coats	737
5	Wedges	695
6	Scarves	800
7	Jeans	800
8	Loafers	891
9	Mini	634
10	Hats	800
11	Flats	700
12	Jewelry	800
13	Sneakers	904
14	Top	800
15	Sweatpants	816
16	Legging	502
17	Backpacks	800
18	Makeup	800
19	Bottom	800
20	Hairaccessories	755
21	Twopiece	800
22	Jumpsuit	494
23	Slides	757

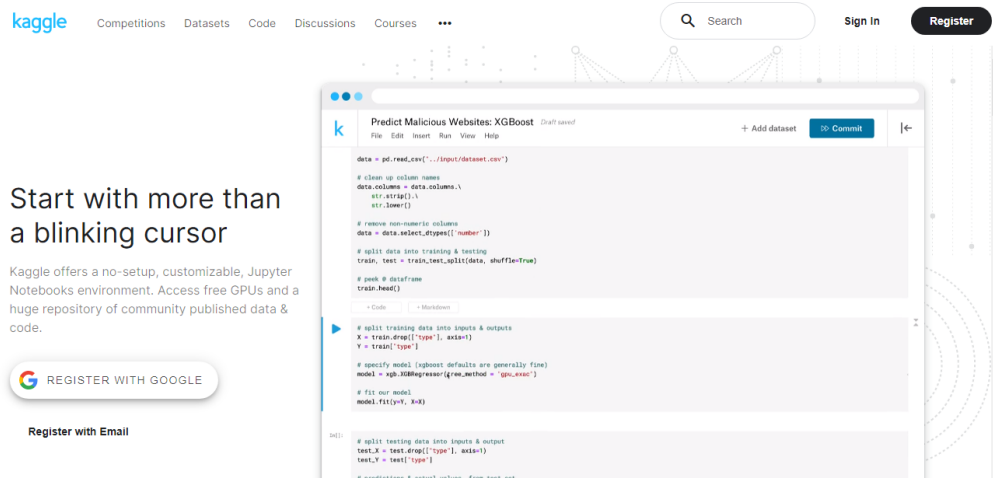
Tabel 3. 2 Lanjutan Banyak Data Per-label Dalam Dataset

No	Label	Jumlah Gambar
24	Midi	612
25	Gloves	800
26	Skirts	800
27	Clutch	800
28	Jackets	870
29	Pants	800
30	Watches	800
31	Maxi	469
32	Boots	700
33	Onepiece	800
34	Ponchos	166
35	Heels	1101
36	Cocktail	800
37	Sandals	965
38	Casual	496
39	Blazers	803
40	Sunglasses	800
41	Handbags	800
42	Belts	801

3.1.2 Prosedur pengumpulan data

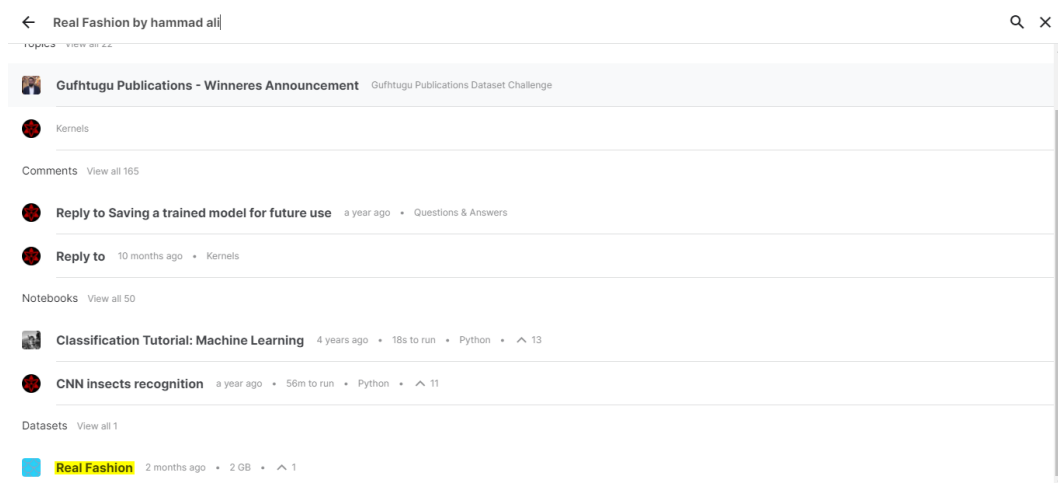
Prosedur pengumpulan data adalah proses yang dilakukan untuk mengumpulkan data yang akan digunakan dalam penelitian, disini data di peroleh dengan mengunduh melalui website penyedia dataset yaitu kaggle. Berikut langkah-langkah pengumpulan data yang akan digunakan untuk penelitian:

1. Buka situs kaggle dengan mengetik alamat (<https://www.kaggle.com/>) di mesin pencarian, lalu tekan enter atau klik cari. Setelah loading membuka alamat selesai maka akan muncul tampilan seperti pada Gambar 3.5 dibawah ini.



Gambar 3. 5 Halaman Depan Situs Kaggle

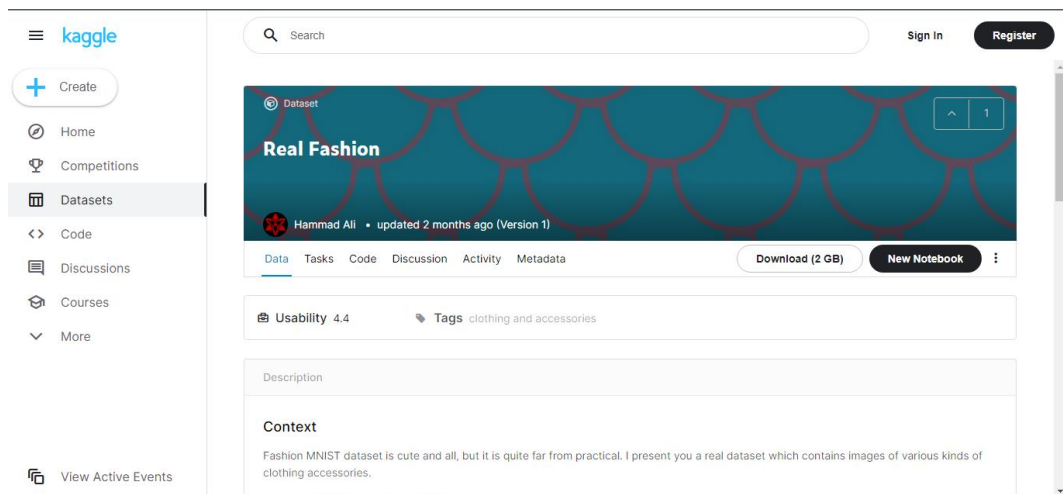
2. Kemudian klik menu *search* lalu tuliskan kata kunci “Real Fashion by hammad ali” dan setelah itu akan muncul beberapa pilihan lalu pilih yang berada pada blok dataset yaitu *Real Fashion* seperti yang ada pada Gambar 3.6.



Gambar 3. 6 Hasil Pencarian di Kaggle

Karena dataset tergolong baru yaitu rilis pada tanggal 11 Oktober 2021 maka terkadang tidak muncul dalam pencarian. Maka alternatifnya yaitu dapat mengetik alamat datasetnya langsung yaitu <https://www.kaggle.com/hammaadali/real-fashion>.

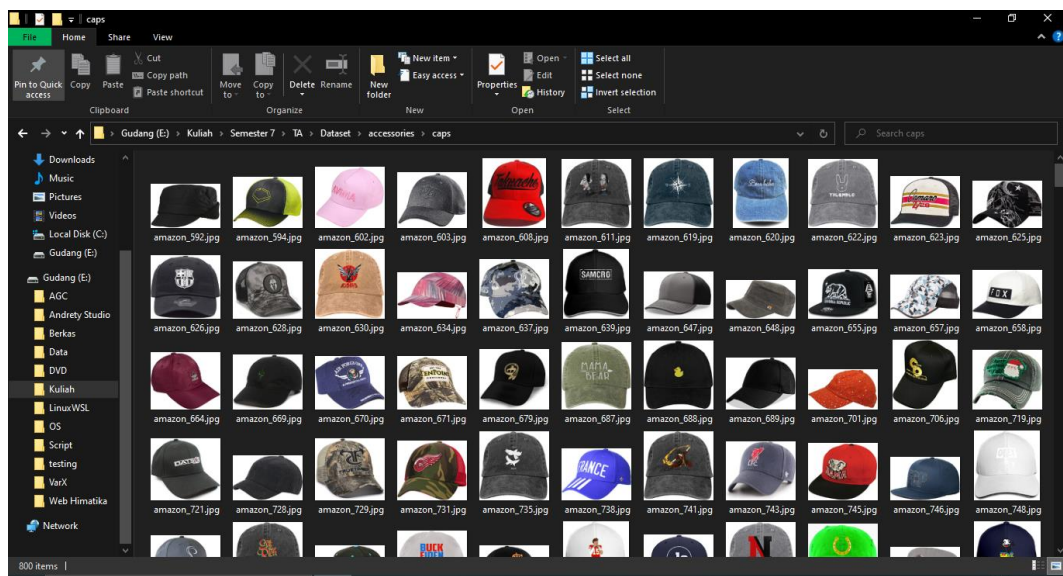
3. Setelah muncul data yang di cari maka selanjutnya unduh data tersebut dengan menekan tombol *download* yang terdapat di halaman tersebut seperti pada Gambar 3.7 dibawah ini.



Gambar 3. 7 Hasil Pencarian Data

Saat akan mengunduh diharuskan *sign-in* terlebih dahulu, jika tidak memiliki akun maka dapat menggunakan email yang dipunya untuk *sign-in*.

4. Setelah dataset selesai di unduh selanjutnya ekstrak file zip dari data yang berhasil di unduh maka akan mendapatkan data gambar produk seperti pada Gambar 3.8 dibawah ini.



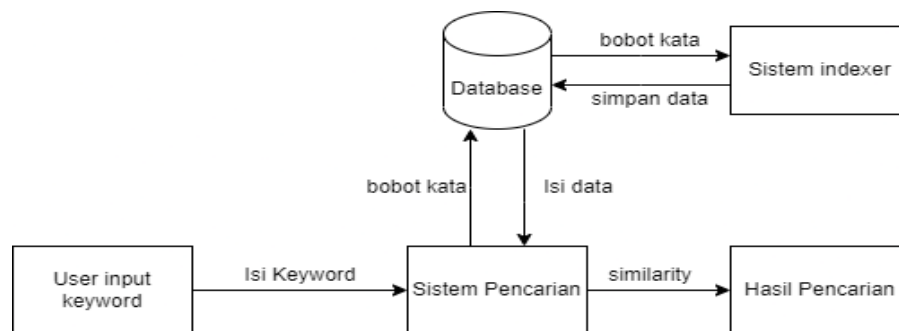
Gambar 3. 8 Dataset yang Berhasil di Unduh

3.2 Aturan bisnis (bussiness rule)

3.2.1 Analisis Sistem Saat Ini

Sistem yang berjalan saat ini umumnya adalah pengguna jika ingin mencari suatu produk di *marketplace* harus mengetikkan nama produk di fitur pencarian

yang ingin dicari. Cara ini memiliki kekurangan apabila pengguna menginginkan suatu barang tetapi tidak tahu nama produk tersebut. Semisal pengguna menginginkan sebuah gaun yang model atau coraknya mirip dengan publik figur tetapi pengguna tidak tahu merk atau nama gaun tersebut, maka pengguna akan kesulitan mencari sebuah produk yang mirip dengan publik figur tersebut. Berikut gambar diagram sistem pencarian menggunakan *keyword* dapat dilihat pada Gambar 3.9.



Gambar 3. 9 Diagram Sistem Pencarian

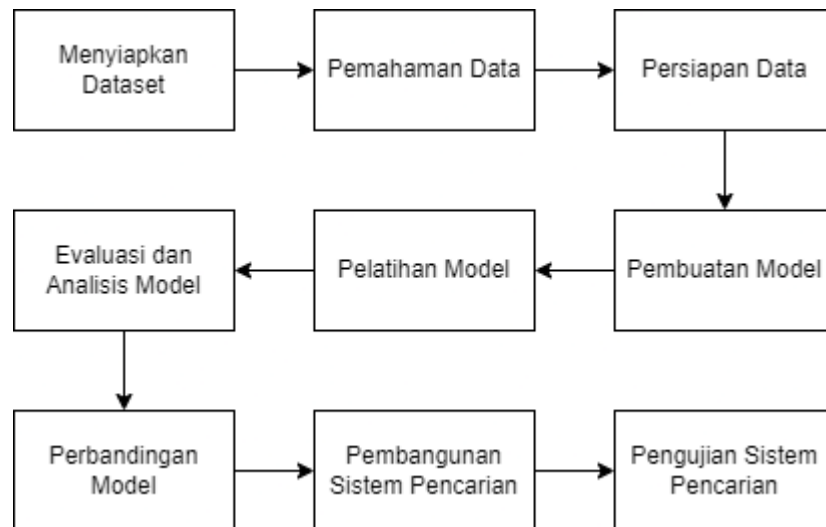
3.2.2 Kelemahan Sistem Saat Ini

Keterbatasan pencarian menggunakan teks adalah pengguna harus mencari sebuah produk dengan jargon jenis produk yang sama persis dengan yang digunakan situs atau aplikasi. Situs atau aplikasi tidak mendukung kueri penelusuran untuk variasi warna (meskipun produk yang dicari tersedia dalam berbagai warna).

Keterbatasan tersebut tidak hanya dialami pengguna atau konsumen tetapi juga penjual. Hal ini sangat tergantung pada penggunaan kombinasi kata yang tepat. Ketika konsumen tidak mendapatkan kata kunci yang benar, atau lebih buruk lagi, tidak tahu bagaimana menjelaskan dengan kata-kata apa yang mereka cari. Penjual kemudian kehilangan peluang untuk terhubung langsung dengan konsumen serta kehilangan peluang meningkatkan pendapatan.

3.3 Tahapan Penelitian

Tahapan penelitian yang dilakukan penulis dalam melakukan penelitian untuk membangun sistem dapat dilihat pada gambar 3.10.



Gambar 3. 10 Diagram Tahap Penelitian

Berikut penjelasan dari diagram tahapan penelitian Tugas Akhir:

a. Menyiapkan Dataset

Dalam proses penyiapan dataset yang dilakukan adalah mengunduh dataset dari situs kaggle ke penyimpanan Google Colab. Mengunduh dataset dari kaggle dapat dilakukan dengan menggunakan API (*Application Programming Interface*) kredensial yang disediakan kaggle berupa file JSON. Untuk menggunakannya ialah dengan mengunduh API kredensial dari akun kaggle lalu di unggah ke Google Colab dan setelah itu jalankan API *command* untuk mengunduh dari dataset tersebut, setelah terunduh maka dataset akan berupa file zip dan selanjutnya adalah melakukan ekstraksi dataset.

b. Pemahaman Data

Setelah dataset dilakukan proses ekstraksi lalu dilakukan sebuah proses pemahaman data (*data understanding*) untuk mengetahui struktur dari dataset tersebut. Dari dataset yang sudah diekstrak didapatkan sebuah struktur yaitu terdapat 7 kategori utama, pada setiap dari kategori utama didalamnya terdapat sub-kategori dengan jumlah yang bervariasi serta berkaitan dengan kategori utama dan pada masing-masing sub-kategori terdapat gambar-gambar dari dataset sesuai kategori utama dan sub-kategori. Dikarenakan gambar-gambar yang dibutuhkan berada pada folder kategori yang berlapis maka struktur dataset akan disederhanakan dengan cara memindah semua sub-kategori yang berada pada

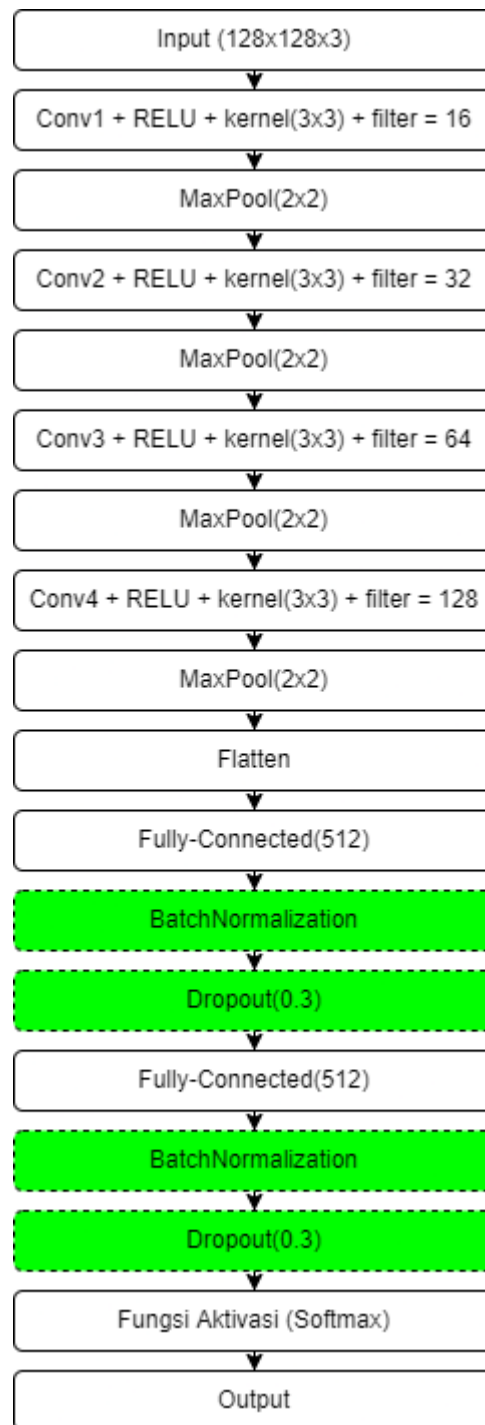
setiap kategori utama menjadi satu direktori saja yang berisi semua sub-kategori yang ada.

c. Persiapan Data

Dataset yang sudah disiapkan akan dilakukan proses *preprocessing* data terlebih dahulu agar meningkatkan performa dari proses pembelajaran model. Pertama data gambar di ubah resolusinya menjadi 128x128 lalu data di dibagi menjadi data *train* dan data *validation* dengan rasio sebesar 9:1 (90% data *train* dan 10% data *validation*). Data akan dilakukan standarisasi dimensi dan dilakukan augmentasi data agar mendapatkan banyak sampel data dengan bentuk yang berbeda-beda, data akan menjadi 3-4 kali lipat dari data sebelumnya.

d. Pembuatan Model

Dalam pembuatan model akan dilakukan dengan dua cara yaitu, cara pertama merancang arsitektur model yang diusulkan tidak menggunakan *pre-trained* model dan cara kedua yaitu memanfaatkan beberapa model *pre-trained* untuk melakukan *transfer learning* dalam perancangan arsitektur model diantara lain model arsitektur yang digunakan adalah VGG16 dan ResNet50. Berikut arsitektur model yang diusulkan tidak menggunakan *pre-trained* model dapat dilihat pada Gambar 3.11.



Gambar 3. 11 Arsitektur Model yang Diusulkan

Berikut merupakan penjelasan mengenai arsitektur yang diusulkan atau tidak memanfaatkan *pre-trained* model atau *transfer learning* yang terdapat pada Gambar 3.11:

1. Gambar input yang digunakan yaitu 128x128 piksel dengan jenis gambar

RGB atau dengan 3 *channel* warna. Data gambar akan di konvolusi untuk mendapatkan nilai *feature* sebagai *inputan* untuk *neural network*.

2. Terdapat 4 *layer* konvolusi dengan diantaranya memiliki ukuran kernel sama dan banyak *filter* yang digunakan untuk konvolusi pertama sebanyak 16 *filter*, konvolusi kedua menggunakan 32 *filter*, konvolusi ketiga menggunakan 64 *filter*, dan konvolusi keempat menggunakan 128 *filter*.
3. *Convolution* pada *layer* pertama memiliki ukuran kernel 3x3 dengan *filter* sebanyak 16 dengan aktivasi fungsi relu.
4. Gambar akan di cari nilai maksimal pada setiap pixel yang dilalui oleh kernel pooling layer dengan ukuran 2 x 2.
5. Pada *layer* kedua gambar akan dikonvolusi dengan ukuran kernel 3 x 3 dan *filter* sebanyak 32 dengan aktivasi fungsi relu.
6. Citra pada tahap ini akan diambil nilai maksimalnya menggunakan layer *pooling* dengan ukuran 2 x 2.
7. Pada *layer* ketiga gambar akan dikonvolusi dengan ukuran kernel 3 x 3 dan *filter* sebanyak 64 dengan aktivasi fungsi relu.
8. Citra pada tahap ini akan diambil nilai maksimalnya menggunakan layer *pooling* dengan ukuran 2 x 2.
9. Pada *layer* keempat gambar akan dikonvolusi dengan ukuran kernel 3 x 3 dan *filter* sebanyak 128 dengan aktivasi fungsi relu.
10. Citra pada tahap ini akan diambil nilai maksimalnya menggunakan layer *pooling* dengan ukuran 2 x 2.
11. Nilai *feature* seluruh gambar yang sudah dikonvolusi akan dijadikan sebuah *vector* untuk inputan pada *neural network* menggunakan *flatten*.
12. Pada neural network terdapat 2 *fully connected layer* yang memiliki 512 jumlah neuron dan pada setiap *fully connected layer* terdapat *layer batchnormalization* dan menggunakan *dropout layer* sebesar 0.3 atau 30%.
13. Lalu yang terakhir ada fungsi aktivasi softmax untuk mendapatkan hasil klasifikasi dari setiap kelas yang ada.

Cara kedua yang digunakan dalam pembuatan model yang diusulkan yaitu memanfaatkan beberapa model *pre-trained* untuk melakukan *transfer learning*. Beberapa tahap klasifikasi yang dirubah dan disesuaikan dengan *output* data yang digunakan:

1. Pada proses *transfer learning* citra akan dilakukan serangkaian proses konvolusi dan *pooling* sesuai arsitektur *transfer learning* yang digunakan.
2. Setelah didapatkan *feature map* dari proses konvolusi yang terdapat pada *base* arsitektur model *transfer learning* selanjutnya nilai *feature* seluruh gambar yang sudah dikonvolusi akan dijadikan sebuah *vector* untuk inputan pada *neural network* menggunakan *flatten*.
3. Pada *neural network* terdapat 2 *fully connected layer* yang memiliki 512 jumlah neuron dan pada setiap *fully connected layer* terdapat *layer batchnormalization* dan menggunakan *dropout layer* sebesar 0.5 atau 50%.
4. Lalu yang terakhir ada fungsi aktivasi softmax untuk mendapatkan hasil klasifikasi dari setiap kelas yang ada sesuai data yang dilatih nantinya.

e. Pelatihan Model

Dalam pelatihan model antara model yang diusulkan tanpa menggunakan *transfer learning* dengan yang memanfaatkan *transfer learning* akan menggunakan parameter-parameter yang sama. Diantara lain parameter yang digunakan ialah *learning rate* sebesar 0.01, 0.001, 0.0001, dan jenis *optimizer* yaitu ADAM dan RMSProp

f. Evaluasi dan Analisis Model

Setelah model akhir didapatkan akan dilakukan evaluasi dengan data uji untuk mengetahui nilai akurasi, presisi, *recall*, dan *f1-score* setiap model. Setelah dilakukan evaluasi selanjutnya dilakukan analisis dimana dari setiap arsitektur memiliki enam model dan dipilih yang memiliki performa terbaik dari nilai akurasi, presisi, *recall*, dan *f1-score* untuk dibandingkan nantinya terhadap arsitektur yang ada.

g. Perbandingan Model

Setelah masing-masing arsitektur memiliki model yang terbaik dari hasil evaluasi dan analisis kemudian dibandingkan masing-masing model arsitektur

diantaranya yaitu model arsitektur yang diusulkan, model arsitektur dengan *transfer learning* VGG16, dan model arsitektur dengan *transfer learning* ResNet50.

h. Pembangunan Sistem Pencarian

Setelah model yang memiliki performa terbaik didapatkan maka selanjutnya adalah membuat sebuah sistem pencarian dengan menggunakan Flask sebagai *framework* dalam pembangunan website. Berikut proses pembangunan sistem pencarian menggunakan gambar:

1. Model akan di *load* pada Flask menggunakan bahasa pemrograman python untuk melakukan ekstraksi fitur.
2. Membuat proses ekstraksi fitur gambar dari dataset untuk mendapatkan vektor citra lalu dimasukan ke dalam *database*.
3. Membuat proses klasifikasi dari input dari model yang sudah di *load* untuk mendapatkan kategori atau kelas dari citra yang di inputkan. Proses klasifikasi dalam pencarian gambar berguna untuk mempersempit lingkup perhitungan dari vektor citra *input* masukan dengan vektor citra dalam *database*.
4. Setelah mendapatkan nama kategori atau kelas dari citra selanjutnya dilakukan proses perhitungan kemiripan dengan *euclidean distance* antara vektor citra *input* dan vektor citra yang terdapat dalam *database* sesuai kelas atau kategori citra. Untuk menghitung kemiripan dilakukan perhitungan.
5. Setelah dilakukan perhitungan dari setiap citra yang ada lalu dilakukan *indexing* untuk mengurutkan citra yang paling mirip dengan citra *input* dan ditampilkan pada halaman web sesuai urutan.

i. Pengujian Sistem Pencarian

Setelah sistem pencarian sudah selesai maka dilakukan uji coba dengan memasukkan gambar ke dalam sistem untuk mendapatkan hasil atau gambar keluaran yang sesuai apa yang di cari atau mirip yang dicari.

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Sistem Yang Diusulkan.

Sistem pencarian menggunakan teks di *marketplace* saat ini memiliki banyak keterbatasan atau kelemahan seperti pengguna harus mencari sebuah produk dengan jargon jenis produk yang sama persis dengan yang digunakan situs atau aplikasi. Situs atau aplikasi tidak mendukung kueri penelusuran untuk variasi warna (meskipun produk yang dicari tersedia dalam berbagai warna).

Sistem yang diusulkan berupa pencarian produk menggunakan gambar atau *visual search* dimana pengguna dapat menangkap gambar atau mengunggah gambar ke sistem pencarian, maka sistem akan mengolah gambar tersebut dan mencari produk yang mirip dengan gambar yang diunggah atau hasil tangkapan kamera.

4.1.1 Analisis Fungsional

Analisis Fungsional merupakan analisis fungsional yang terkait dengan fasilitas yang dibutuhkan oleh sistem secara umum. Kebutuhan fungsional dari sistem yang akan dibangun meliputi:

a. Analisis Kebutuhan *Input*

1. *Input Data Latih / Dataset: input* berupa data gambar yang akan digunakan untuk melatih model.
2. *Input Data Database: input* berupa data gambar yang akan diekstraksi fitur yang ada pada gambar tersebut dan akan diproses sampai disimpan dalam *database*.
3. *Input Data Pencarian: input* berupa data gambar ke mesin pencarian yang nanti disediakan untuk mendapatkan hasil pencarian produk yang serupa.

b. Analisis Kebutuhan Proses

1. Proses *preprocessing*, yaitu proses untuk mengolah data gambar.
2. Proses augmentasi, yaitu proses memperbanyak jumlah sampel data untuk menghindari *under-sampling*.

3. Proses pembuatan model, yaitu proses melatih model dengan *dataset* yang sudah disiapkan dan melakukan evaluasi terhadap hasil pelatihan model.
 4. Proses ekstraksi fitur, yaitu proses mengekstraksi setiap fitur yang ada pada gambar. Hasil proses ekstraksi fitur biasanya akan disimpan dalam bentuk vektor.
 5. Proses tes klasifikasi kategori, yaitu proses mengklasifikasikan kelas atau kategori dari masukkan data gambar untuk menguji keakuratan dari model yang sudah dibuat.
 6. Proses penambahan data ke *database*, yaitu proses menambahkan dataset yang sudah di ekstraksi fiturnya dalam bentuk vektor ke dalam *database* MySQL.
 7. Proses klasifikasi kategori, yaitu proses mengklasifikasikan kelas atau kategori dari masukkan data gambar untuk untuk mendapatkan id kategori untuk mempersempit proses pencarian gambar.
 8. Proses perbandingan kesamaan, yaitu proses menghitung kedekatan sebuah vektor gambar dari *input* dengan vektor gambar yang ada dalam *database* dengan menggunakan rumus *euclidean distance*.
 9. Proses *reindexing* citra sesuai kedekatan dengan *input*, yaitu proses indeksing dari jarak paling kecil. Hasil data yang sudah di *indexing* akan dikembalikan sebagai bentuk *output* sistem.
 10. Proses pencarian, yaitu proses menghitung kedekatan antar gambar masukkan dengan gambar yang ada didalam database, dan menampilkan hasil dari proses tersebut.
- c. Analisis Kebutuhan Output
1. Hasil pencarian gambar yang serupa dengan masukkan di pencarian.

4.1.2 Analisis Non Fungsional.

A. Perangkat Keras (Hardware) yang Digunakan

Perangkat keras yang digunakan untuk mengoperasikan sistem yang akan dibuat dapat adalah:

1. Laptop Dell Latitude E6420.

2. Processor Intel Core i5 2520M.
3. Mouse Logitech.
4. RAM 6GB.
5. Monitor Philips 14 inch.
6. Hardisk 240 GB, SSD 240GB.

B. Perangkat Lunak (Software) yang Digunakan

Perangkat lunak yang digunakan untuk membangun sistem yang akan dibuat adalah :

1. Sistem Operasi Windows 10.
2. Jupyter Notebook.
3. Google Colab
4. Draw.io

4.2 Desain Sistem

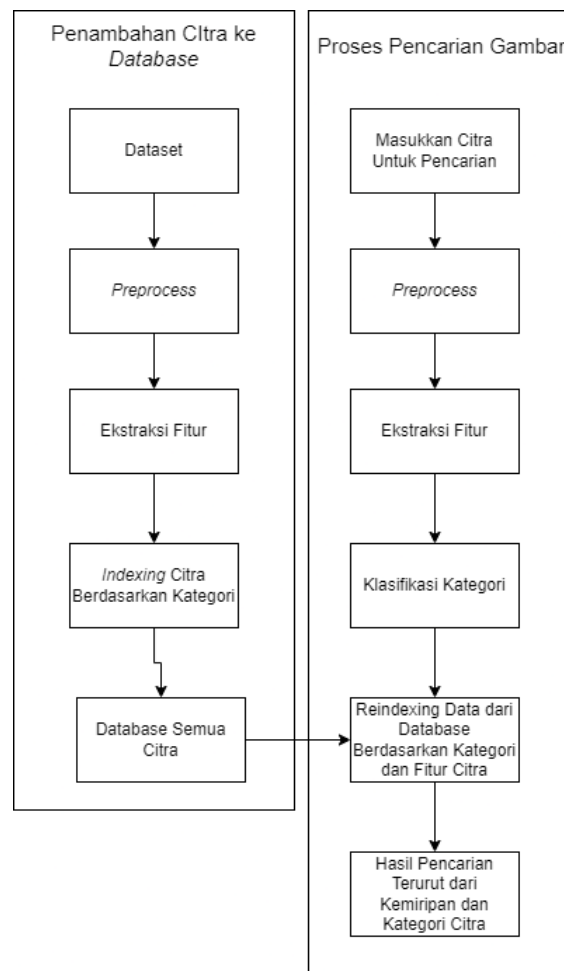
4.2.1 Desain Logik

Desain logik digunakan untuk menggambarkan sistem yang akan di bangun. Rancangan sistem pada penelitian ini berfokus pada alur algoritma yang digambarkan melalui diagram atau flowchart.

4.2.1.1 Diagram Sistem Yang Akan Dibangun

Sistem akan terbagi menjadi 2 bagian, yaitu bagian untuk menambahkan gambar kedalam database dan bagian pencarian gambar. Pada bagian penambahan gambar, langkah pertama yaitu mengumpulkan semua gambar yang akan ditambahkan kedalam database, setelah itu setiap gambar akan dilakukan *preprocessing* dan menggunakan model yang terbaik performanya untuk melakukan ekstraksi fitur, setelah itu vektor hasil ekstraksi fitur disimpan ke dalam *database* sesuai kategori dimana kategori dari gambar ditandai oleh nama folder yang menampung gambar tersebut. Pada bagian lainnya yaitu pencarian gambar, langkah awal mirip dengan proses penambahan gambar ke dalam database, yaitu lokalisasi objek, menggunakan model yang terbaik performanya untuk ekstraksi fitur dan klasifikasi kategori. Setelah melalui proses tersebut hasil fitur dan

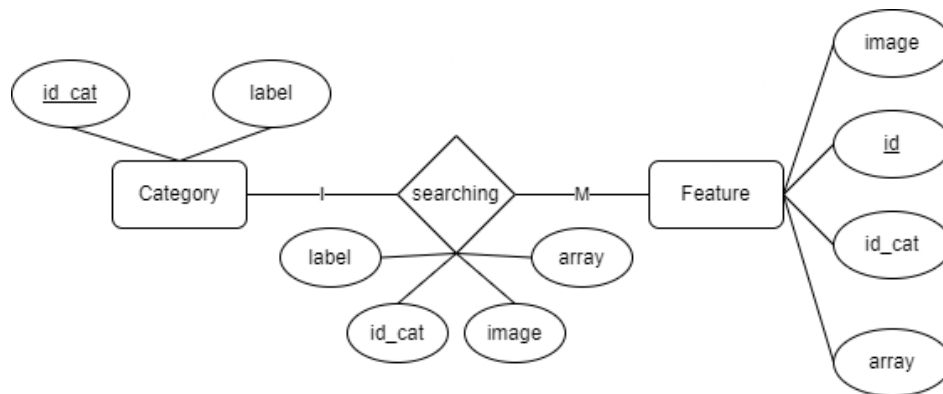
klasifikasi yang diperoleh dari gambar akan dibandingkan dengan fitur-fitur yang ada dalam database sesuai kategori hasil klasifikasi. Lalu dilakukan *Re-Indexing* terhadap data dari database tersebut sesuai dengan kedekatan fitur. Hasil urutan inilah yang akan dikembalikan. Diagram dapat dilihat pada gambar 4.1.



Gambar 4. 1 Sistem Yang Akan Dibangun

4.2.1.2 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan proses yang menunjukkan hubungan antar entitas dan relasinya pada basis data. Rancangan ERD pada sistem pencarian menggunakan gambar ini dapat dilihat pada Gambar 4.2.

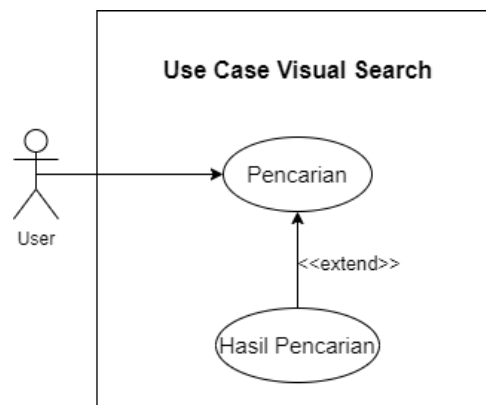


Gambar 4. 2 Diagram ERD

Dapat dilihat pada diagram sistem pencarian terdapat dua entitas di dalamnya. Kedua entitas tersebut adalah *category* dan *feature*. Setiap entitas memiliki atributnya masing-masing yaitu entitas *category* memiliki atribut *id_cat* dan *label*, sedangkan entitas *feature* memiliki atribut *image*, *id*, *id_cat*, dan *array*. Pada setiap relasi dari ER diagram mengandung beberapa atribut yang digunakan dalam prosesnya yaitu *label*, *id_cat*, *image*, *array*. Proses yang dilakukan ialah proses pencarian dimana proses ini nantinya digunakan untuk mengkalkulasi kemiripan data *input* dengan data pada atribut *array* sesuai *label* gambar masukkan dan akan mengembalikan sebuah data binary dari gambar untuk diproses dan ditampilkan pada tampilan website.

4.2.1.3 Diagram Use Case

Gambar diagram pada Gambar 4.3 merupakan *use case* sistem pencarian gambar, adapun penjelasan dan ilustrasinya sebagai berikut ini:



Gambar 4. 3 Diagram Use Case

- *User*: Orang yang menggunakan sistem untuk melakukan pencarian menggunakan gambar
- *Pencarian*: Merupakan langkah pertama untuk melakukan pencarian menggunakan gambar, *user* dapat mengunggah sebuah gambar untuk dilakukan pencarian dan akan mendapatkan hasil pencarian yang berupa beberapa gambar yang menurut sistem memiliki kemiripan dengan gambar yang diunggah sebelumnya.

4.2.2 Desain Fisik

Berikut pada Gambar 4.4 adalah rancangan antar muka untuk melakukan pencarian berdasarkan gambar atau *visual search*.

Tugas Akhir

<https://yogidwiandrian.masuk.id/>

IMPLEMENTASI DEEP LEARNING UNTUK SISTEM PENCARIAN GAMBAR PRODUK MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN)

Choose File No File Chosen

Submit

Query:

Img Input

Urutan Hasil Klasifikasi: Class 1, Class 2, Class 3

Result:

Vektor dari gambar input:

vektor

Vektor dari database :

vektor

Hasil Perhitungan Euclidean antara Vektor input dan vektor pada DB : 0.0

Menampilkan gambar asli dari vektor pada DB :

Img Input

Kemiripan 50%

Vektor dari gambar input:

vektor

Vektor dari database :

vektor

Hasil Perhitungan Euclidean antara Vektor input dan vektor pada DB : 0.0

Menampilkan gambar asli dari vektor pada DB :

Img Input

Kemiripan 50%

Gambar 4. 4 Tampilan *User Interface*

BAB V

IMPLEMENTASI DAN HASIL SERTA PEMBAHASAN

5.1. Implementasi

Penelitian ini dilakukan dengan menerapkan dua implementasi, yaitu melakukan eksperimen pembangunan model untuk pencarian gambar dan pembangunan sistem pencarian gambar. Hal ini dilakukan karena peneliti ingin membuat model yang optimal untuk diterapkan pada sistem pencarian.

5.1.1 Eksperimen Model Pencarian Gambar

Eksperimen ini dilakukan untuk untuk membuat sistem pencarian dengan gambar menggunakan algoritma *Convolutional Neural Network* (CNN). Pembangunan model dilakukan dengan melakukan 2 buah eksperimen, yaitu dengan mencoba membuat model dengan rancangan arsitektur yang diusulkan dan menggunakan *pretrained* model atau model yang telah dilatih sebelumnya. Untuk perancangan model yang diusulkan dilakukan dengan menggunakan arsitektur sekuensial atau residual. Sedangkan pembangunan model dengan memanfaatkan *pretrained* model akan mencoba beberapa model yang sudah ada dan banyak digunakan yaitu ada VGG16 dan ResNet50. Selain itu saat pelatihan akan dilakukan juga eksperimen dengan melakukan *hyperparameter tuning* manual untuk mendapatkan hasil yang lebih optimal dari masing-masing arsitektur. Hasil dari eksperimen beberapa model ini akan dilakukan perbandingan untuk mendapatkan model yang terbaik yang nantinya akan digunakan untuk membangun sistem pencarian menggunakan gambar.

5.1.1.1 Rancangan Dataset

Dataset yang digunakan pada pembuatan model CNN ini adalah data gambar produk *fashion*. Model dapat berjalan dengan baik jika menggunakan data *train* yang banyak dan berkualitas untuk mengenali pola gambar, semakin banyak data yang digunakan dan berkualitas maka semakin banyak belajar mengenali

sebuah pola gambar. Dataset yang digunakan memiliki 7 jenis kategori *fashion* dan dari masing-masing kategori memiliki sub-kategori yang jumlahnya bervariasi jika ditotal semua terdapat 42 sub-kategori, nama dari sub-kategori inilah dan data gambar di dalamnya yang akan digunakan untuk proses *training*.

Dataset dibagi menjadi 2 yaitu data *training* dan data *validation*, data *training* sendiri berjumlah 27642 data gambar dan data *validation* berjumlah 3084 data gambar atau bisa dikatakan pembagian dataset dengan perbandingan 9:1. Berikut *source code* proses pembagian dataset menggunakan *library splitfolders* pada Gambar 5.1.

```
import splitfolders
path = os.path.join('/content/dataset')
splitfolders.ratio(path,
output="/content/split_data", ratio=(.9, .1))
```

Gambar 5. 1 *Source Code Split Folders*

Pembagian dataset dapat dilihat pada Tabel 5.1 dibawah ini.

Tabel 5. 1 Pembagian Dataset

Data Training	Data Validation
90%	10%
27642	3084

5.1.1.2 Konfigurasi Library Python

Berikut merupakan beberapa *library* python yang digunakan peneliti untuk mendukung program ini, diantaranya sebagai berikut:

1. *Packages* os, shutil, glob: berfungsi sebagai komunikasi antara sistem operasi dengan program pada Python.
2. *Packages* Tensorflow: sebuah *framework* komputasional untuk membuat model *Machine Learning*.
3. *Packages* Keras: merupakan *framework* jaringan syaraf tiruan tingkat tinggi yang ditulis dengan bahasa python dan mampu berjalan di atas TensorFlow, CNTK, atau Theano
4. *Packages* split-folders: digunakan untuk membagi dataset menjadi folder data *train*, *validation*, dan *test*.
5. *Packages* Numpy: digunakan untuk menghitung operasi matematika pada

array.

6. *Packages* Matplotlib: digunakan untuk melakukan visualisasi data multiplatform yang dibangun di atas *array* NumPy
7. *Packages* PIL: PIL atau Pillow digunakan untuk mengolah data gambar seperti visualisasi sampai melakukan *image processing*.
8. *Packages* Sklearn: Sklearn atau *Scikit-learn* digunakan untuk mendapatkan fungsi Confusion Matrix dan *classification report* untuk melakukan evaluasi metrik.
9. *Packages* itertools: library yang mengandung beberapa fungsi yang berguna dalam pemrograman fungsional dan salah satu jenis fungsi yang dihasilkan adalah pengulangan yang tak terbatas.

Dibawah ini merupakan *source code* konfigurasi yang digunakan dalam sistem dapat di lihat pada Gambar 5.2 dibawah ini.

```

# I/O python
import os, shutil, glob, time

# Preproses data
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
import splitfolders
import numpy as np

# Untuk pembuatan model
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.layers import Dropout, Flatten, Dense,
BatchNormalization, Conv2D, MaxPooling2D
from keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow import keras
from keras.models import Sequential, Model

# Transfer learning
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.mobilenet_v2 import
MobileNetV2

# Visualisasi
from keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import plotly.express as px
from PIL import Image

# Evaluasi
from sklearn.metrics import confusion_matrix,
classification_report
import itertools

```

Gambar 5. 2 *Source Code Library Python*

5.1.1.3 Menyiapkan Dataset

Hal pertama yang dilakukan untuk menyiapkan dataset ialah mengunduh dataset dari situs kaggle ke penyimpanan Google Colab. Mengunduh dataset dari kaggle dapat dilakukan dengan menggunakan API (*Application Programming Interface*) kredensial yang disediakan kaggle berupa file JSON. Untuk menggunakannya ialah dengan mengunduh API kredensial dari akun kaggle lalu di unggah ke Google Colab dan setelah itu jalankan API *command* untuk mengunduh dari dataset tersebut, setelah terunduh maka dataset akan berupa file zip dan

selanjutnya adalah melakukan ekstraksi dataset. Dibawah ini merupakan source code untuk menyiapkan dataset yang dapat di lihat pada Gambar 5.3 dibawah ini.

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d hammaadali/real-fashion
!unzip real-fashion.zip
```

Gambar 5. 3 *Source Code* Menyiapkan Dataset

5.1.1.4 Persiapan Data

Setelah data produk *fashion* didapatkan kemudian dilakukan persiapan data untuk nantinya dapat dimasukan dalam model yang dibuat. Dari hasil ekstraksi dataset yang ada pada persiapan dataset terdapat 7 folder kategori utama dan didalamnya masing-masing terdapat beberapa sub-folder kategori, pada proses pelatihan nanti hanya digunakan sub-folder kategori beserta data gambarnya maka sub-folder kategori yang terdapat dari masing-masing folder kategori utama dipindahkan ke dalam satu folder yang akan berisi semua sub-folder yang ada. Dibawah ini merupakan source code untuk memindahkan sub-folder yang berbeda ke satu folder yang dapat di lihat pada Gambar 5.4.

```
# Membuat folder dataset
os.mkdir('/content/dataset')

# Inisiasi data yang akan dipindah
sources = ['dress', 'accessories', 'bag', 'bottom', 'shoes',
'outerwear', 'swimwear']
source = "/content/"
destination = "/content/dataset"

# Melakukan pemindahan sub-kategori dataset ke satu folder
for category in sources:
    cur_cat = os.path.join(source, category)
    for sub in os.listdir(category):
        cur_sub = os.path.join(cur_cat, sub)
        shutil.move(cur_sub, destination, copy_function =
shutil.copytree)
    os.rmdir(cur_cat)
```

Gambar 5. 4 *Source Code* Memindahkan Folder

Setelah folder sub-kategori berada di satu tempat maka dilanjutkan dengan melakukan tahap *preprocessing*. Semua data tersebut akan diubah ukurannya atau di *resize* agar ukurannya sama dan menghemat komputasi saat pelatihan nantinya.

Semua gambar akan di ubah ukurannya menjadi 128x128 piksel. Berikut adalah *source code* untuk merubah ukuran dataset dapat dilihat pada Gambar 5.5.

```
# Merubah ukuran gambar menjadi 128x128
path = "/content/dataset"
for cat in os.listdir(path):
    cur_cat = os.path.join(path, cat)
    for item in os.listdir(cur_cat):
        cur_item = os.path.join(cur_cat, item)
        if os.path.isfile(cur_item):
            im = Image.open(cur_item)
            rgb_im = im.convert('RGB')
            imResize = rgb_im.resize((128, 128),
Image.ANTIALIAS)
            imResize.save(cur_item, 'JPEG', quality=90)
            print(f'Kategori {cat} berhasil di resize')
```

Gambar 5. 5 *Source Code* Merubah Ukuran Dataset

Setelah dilakukan penyeragaman ukuran data gambar maka selanjutnya adalah langkah untuk membagi dataset menjadi data *train* dan data *validation* sesuai rancangan dataset yaitu sebesar 9 banding 1, 90% untuk data *train* dan 10% data *validation*. Berikut adalah *source code* untuk melakukan pembagian dataset dengan *library* *splitfolders* dapat dilihat pada Gambar 5.6.

```
# Melakukan split dataset
path = os.path.join('/content/dataset')
splitfolders.ratio(path
,output="/content/split_data",ratio=(.9,.1))
```

Gambar 5. 6 *Source Code* Membagi Dataset

Setelah data berhasil dibagi menjadi data *train* dan data *validation* maka selanjutnya adalah melakukan proses augmentasi gambar dan normalisasi data. Proses augmentasi berfungsi untuk memperbanyak data dengan cara menyalin data asli lalu dirubah menjadi gambar baru dengan cara seperti *merotasinya*, melakukan *zooming*, memotong, membalikan secara horizontal atau vertikal, dsb. Proses normalisasi menggunakan fungsi *preprocess_input* yang disediakan dari masing-masing model arsitektur, *preprocess_input* akan menyesuaikan gambar dengan format yang dibutuhkan model. Beberapa model menggunakan gambar dengan nilai mulai dari 0 hingga 1, atau dari -1 hingga +1, dan atau menggunakan gaya "caffe", yang tidak dinormalisasi, tetapi terpusat. Untuk melakukan kedua proses augmentasi dan normalisasi peneliti menggunakan *ImageDataGenerator* yang

berfungsi untuk *pipelining* data gambar untuk *Deep Learning*. *ImageDataGenerator* dalam penelitian ini juga digunakan untuk melakukan *batch sizing* yang berfungsi jumlah sampel data yang akan disebar ke neural network dalam satu kali epoch. *Batch size* yang digunakan sebesar 268 maka saat proses pelatihan akan diambil sebanyak 268 data secara acak dari sampel dataset yang ada untuk tiap epoch sampai semua epoch memenuhi batas sampelnya. Berikut adalah *source code* untuk melakukan augmentasi, normalisasi dan *batching* menggunakan *ImageDataGenerator* dapat dilihat pada Gambar 5.7.

```
# Melakukan split dataset
train_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input,
                   shear_range=0.2,
                   zoom_range=0.2,
                   rotation_range=10,
                   vertical_flip=True,
                   horizontal_flip=True)

val_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input,)
WIDTH = 128
HEIGHT = 128
BATCH_SIZE = 268

train_generator = train_datagen.flow_from_directory(
    '/content/split_data/train',
    target_size=(WIDTH, HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    '/content/split_data/val',
    target_size=(WIDTH, HEIGHT),
    batch_size=BATCH_SIZE,
    shuffle = False,
    class_mode='categorical')
```

Gambar 5. 7 *Source Code Augmentasi, Normalisasi, dan Batching*

5.1.1.5 Pembuatan model

Pada tahap ini akan dibuat sebuah *function* atau fungsi yang didalamnya berisi untuk membangun arsitektur model, optimasi model dan *callback*, dan inisialisasi proses pelatihan. Fungsi ini memiliki sebuah parameter yaitu *optimizer* dan *learning rate* yang bertujuan untuk melakukan *hyperparameter tuning* manual

dengan cara mengatur kedua parameter tersebut secara berbeda-beda setiap dijalankan. Berikut penjelasan tahap-tahapan dalam pembuatan model.

1. Membangun arsitektur model

Pada tahap ini akan dibangun 2 jenis arsitektur yang berbeda yaitu dengan menggunakan *pretrained* model (VGG16 dan ResNet50) dan tidak menggunakan *pretrained* model (model yang diusulkan) untuk rancangan kedua arsitekturnya dapat dilihat pada Gambar 3.8 dan Gambar 3.9. Struktur model yang dibangun menggunakan model Sequential, dimana setumpuk lapisan disusun secara linear atau berurutan dan masing masing lapisan akan menyaring secara bertahap data masukan untuk mendapatkan data keluaran yang di inginkan. Umumnya dalam arsitektur CNN mempunyai 2 tahap yaitu *feature learning* dan *classification*. Input gambar pada model CNN memiliki ukuran 128x128x3, dimana 3 menandakan gambar memiliki 3 channel yaitu *Red*, *Green*, dan *Blue* (RGB). Pada model yang memanfaatkan transfer learning saat pelatihan akan mengunci atau *freeze layer* agar tidak dapat dilatih lagi parameternya karena nilai *weight* dari model aslinya dapat berubah.

2. Optimasi model dan callback

Sebelum dilakukan proses pelatihan maka model CNN yang telah di susun harus dikonfigurasi dengan method `model.compile()`. Dalam pengoptimalan untuk memperbaharui bobot menggunakan optimasi ADAM dan RMSProp. Penggunaan loss function *categorical_crossentropy* karena data pelatihan berbentuk kategori, sehingga pengevaluasian loss dilakukan berdasarkan kategori data, sedangkan metric yang digunakan dalam pelatihan ini adalah akurasi.

Callback ialah salah satu instrumen tambahan yang begitu penting saat proses pelatihan data, *callback* berguna untuk menginferensi proses jalannya pelatihan. Callback yang digunakan pada penelitian ini ialah model checkpoint dan early stopping. Model *checkpoint* berguna untuk menyimpan model keras atau bobot model terbaik dengan memperhatikan *validation loss* selama pelatihan, sehingga model atau bobot dapat dimuat atau digunakan kembali nanti untuk melanjutkan pelatihan dari keadaan tersimpan. *Early stopping* berguna untuk

meghentikan sebuah pelatihan jika tidak ada perkembangan akurasi sesuai jumlah *epoch* yang ditunggu.

3. Proses pelatihan

Ketika proses pelatihan dari kumpulan data terjadi beberapa pengulangan/*epoch* untuk mendapatkan kinerja model yang maksimal, nilai akurasi setiap pengulangan dapat meningkat atau menurun, sehingga untuk mendapatkan nilai terbaik diperlukan fungsi *checkpoint*. Checkpoint digunakan untuk menyimpan model terbaik dengan nilai *loss error* paling kecil dan nilai *loss function* yang rendah sebanding lurus dengan akurasi model. Tahap terakhir dari proses pelatihan *neural network* dalam sebuah *epoch* adalah *backpropagation*, yaitu proses menggunakan *stochastic gradient descent* untuk memperbarui bobot dan bias untuk mengurangi kerugian pelatihan secara keseluruhan, di mana peneliti menggunakan ADAM. Semakin kecil nilai *loss*, semakin baik model dan semakin baik tingkat akurasinya, sebaliknya semakin besar nilai *loss*, semakin buruk tingkat akurasinya. Setelah semua model berhasil dilatih maka tahap selanjutnya adalah membandingkan semua model untuk mencari model yang memiliki performa terbaik untuk selanjutnya digunakan dalam pembangunan sistem pencarian menggunakan gambar. Berikut adalah *source code function* masing-masing model yang sudah terdapat arsitektur model, optimasi model, *callback*, dan proses pelatihan dapat dilihat pada Gambar 5.8, Gambar 5.9, dan Gambar 5.10.

```

def vgg_model(opt, lr):
    # Membangun arsitektur model
    base_vgg = VGG16(include_top=False,
                      weights='imagenet',
                      input_shape=(WIDTH, HEIGHT, 3))

    base_vgg.trainable = False

    model = Sequential([
        base_vgg,
        Flatten(),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(N_CLASS, activation='softmax')
    ])

    # Optimasi Model
    model.compile(optimizer=opt(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    # Callback Implementation
    save_model =
    f'/content/gdrive/MyDrive/TA/models/vgg_{str(opt)}_{lr}.h5'
    earlyStopping = EarlyStopping(monitor='val_loss', patience=5,
    verbose=0, mode='min')
    modelCheckpoint = ModelCheckpoint(save_model,
    save_best_only=True, monitor='val_accuracy', mode='max',
    verbose=1)

    # proses pelatihan
    start = time.time()
    history = model.fit(train_generator,

    steps_per_epoch=len(train_generator.filesnames) // BATCH_SIZE,
                        epochs=50,
                        validation_data=validation_generator,

    validation_steps=len(validation_generator.filesnames) //
    BATCH_SIZE,
                        callbacks=[earlyStopping,
    modelCheckpoint])
    elapsed = time.time() - start
    print('Computation time = ' + str(round(elapsed,2)) + 's')

    return history, model

```

Gambar 5. 8 Source Code Model VGG16

```

def resnet_model(opt, lr):
    # Membangun arsitektur model
    base_resnet = ResNet50(include_top=False,
                           weights='imagenet',
                           input_shape=(WIDTH, HEIGHT, 3))

    base_resnet.trainable = False

    model = Sequential([
        base_resnet,
        Flatten(),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(N_CLASS, activation='softmax')
    ])

    # Optimasi Model
    model.compile(optimizer=opt(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    # Callback Implementation
    save_model =
f'/content/gdrive/MyDrive/TA/models/resnet_{str(opt)}_{lr}.h5'
    earlyStopping = EarlyStopping(monitor='val_loss', patience=5,
verbose=0, mode='min')
    modelCheckpoint = ModelCheckpoint(save_model,
save_best_only=True, monitor='val_accuracy', mode='max',
verbose=1)

    # proses pelatihan
    start = time.time()
    history = model.fit(train_generator,

steps_per_epoch=len(train_generator.fileNames) // BATCH_SIZE,
                      epochs=50,
                      validation_data=validation_generator,

validation_steps=len(validation_generator.fileNames) //
BATCH_SIZE,
                      callbacks=[earlyStopping,
modelCheckpoint])
    elapsed = time.time() - start
    print('Computation time = ' + str(round(elapsed,2)) + 's')

    return history, model

```

Gambar 5. 9 Source Code Model ResNet50

```

def resnet_model(opt, lr):
    # Membangun arsitektur model
    base_resnet = ResNet50(include_top=False,
                           weights='imagenet',
                           input_shape=(WIDTH, HEIGHT, 3))

    base_resnet.trainable = False

    model = Sequential([
        base_resnet,
        Flatten(),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(512, activation = "relu"),
        BatchNormalization(),
        Dropout(0.5),
        Dense(N_CLASS, activation='softmax')
    ])

    # Optimasi Model
    model.compile(optimizer=opt(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    # Callback Implementation
    save_model =
f'/content/gdrive/MyDrive/TA/models/resnet_{str(opt)}_{lr}.h5'
    earlyStopping = EarlyStopping(monitor='val_loss', patience=5,
verbose=0, mode='min')
    modelCheckpoint = ModelCheckpoint(save_model,
save_best_only=True, monitor='val_accuracy', mode='max',
verbose=1)

    # proses pelatihan
    start = time.time()
    history = model.fit(train_generator,

steps_per_epoch=len(train_generator.fileNames) // BATCH_SIZE,
                      epochs=50,
                      validation_data=validation_generator,

validation_steps=len(validation_generator.fileNames) //
BATCH_SIZE,
                      callbacks=[earlyStopping,
modelCheckpoint])
    elapsed = time.time() - start
    print('Computation time = ' + str(round(elapsed,2)) + 's')

    return history, model

```

Gambar 5. 10 *Source Code Model yang Diusulkan*

5.1.2 Pembangunan Sistem Pencarian Gambar

Alur dalam pembangunan sistem pencarian gambar yaitu perancangan *database*, ekstraksi fitur, menambahkan data ke *database*, klasifikasi serta mengukur kemiripan, dan membuat tampilan berbasis web.

5.1.2.1 Perancangan Database

Dalam perancangan database terdapat 2 tabel yaitu *category* dan *feature*, dalam tabel *category* terdapat 2 kolom yaitu berisi id kategori dan nama label/kategori, untuk tabel *feature* terdapat 4 kolom yaitu id, id kategori yang berasal dari tabel *category*, *array* yang berisi data vektor gambar, dan *path* yaitu berisi nama gambar asli yang diekstraksi. Berikut adalah *source code* perancangan database pada Gambar 5.11.

```
CREATE DATABASE ta;
CREATE TABLE `ta`.`category` (
  `id_cat` INT(3) NOT NULL,
  `label` VARCHAR(24),
  PRIMARY KEY (`id_cat`)
);
CREATE TABLE `ta`.`feature` (
  `id` INT(12) NOT NULL AUTO_INCREMENT,
  `id_cat` INT(3),
  `array` BLOB NOT NULL,
  `image` LONGBLOB NOT NULL,
  PRIMARY KEY (`id`)
);
ALTER TABLE `ta`.`feature`
  ADD FOREIGN KEY (`id_cat`) REFERENCES
  `test`.`category` (`id_cat`);
```

Gambar 5. 11 *Source Code* Perancangan Database

5.1.2.2 Ekstraksi Fitur

Ekstraksi fitur disini ialah proses mengubah data gambar menjadi sebuah data vektor *array* dari hasil proses *convolution* dan *pooling* menggunakan model CNN yang sebelumnya sudah dibuat. Hasil dari fitur ekstraksi inilah yang nantinya akan berguna untuk mengukur kedekatan antar data vektor gambar. Dalam proses ekstraksi fitur, arsitektur model CNN akan di *load* dan akan digunakan dari proses *input* sampai *layer fully-connected* sebagai *output*-nya setelah data gambar di

ratakan menjadi 1 dimensi (*flatenned*) dan akan menghasilkan sebuah data vektor gambar yang akan disimpan atau diukur dengan data vektor lainnya, setelah berhasil di *load* maka selanjutnya yaitu mengubah ukuran gambar sesuai ukuran *input* pada model yaitu sebesar 128x128 piksel, selanjutnya gambar akan dirubah dalam bentuk *numpy array*, setelah itu dimensi *array* ditambah dari yang sebelumnya per piksel menjadi per *channel* (1 piksel terdiri dari 3 *channel*), setelah itu data *array* dilakukan preprocess dengan mengubah skala gambar dan proses terakhirnya yaitu data gambar yang telah di *preprocess* dimasukkan dalam model untuk menghasilkan data vektor dari gambar. Berikut adalah *source code* ekstraksi fitur pada Gambar 5.12.

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import
preprocess_input
from tensorflow.keras.models import Model, load_model
import numpy as np

class FeatureExtractor:
    # Membuat object dari class
    def __init__(self):
        # load model dari input sampai layer dense3
        base_model = load_model('static/model/model_resnet.h5')
        self.model = Model(inputs=base_model.input,
outputs=base_model.get_layer("dense_3").output)
        pass

    def extract(self, img):
        img = img.resize((128, 128)).convert("RGB")
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        feature = self.model.predict(x)[0]
        return feature / np.linalg.norm(feature)
```

Gambar 5. 12 *Source Code* Fitur Ekstraksi

5.1.2.3 Menambahkan Data ke Basis Data

Setelah fungsi ekstraksi fitur dibuat langkah selanjutnya yaitu mengubah semua dataset menjadi data vektor gambar dan dimasukkan dalam basis data. Informasi yang dibutuhkan untuk menambahkan data ke basis data yaitu nama kategori, *array* vektor gambar, dan binary object gambar. Saat penambahan data gambar asli kedalam *database*, *database* hanya dapat menampung gambar yang

sudah dalam bentuk *Binary Large Object* (BLOB) maka dari itu gambar asli harus dikonversi kedalam bentuk binary. Dalam konversi gambar ke bentuk binary object dilakukan sebuah proses yaitu melakukan pengkodean atau *encoding* menggunakan Base64. Pengkodean Base64 digunakan untuk mengonversi byte yang memiliki data biner atau teks menjadi karakter ASCII. Setelah melakukan pengkodean maka tipe data yang dibutuhkan *database* dengan data gambar sudah sama sehingga data gambar sudah dapat ditambahkan. Berikut adalah *source code* menambahkan data ke basis data pada Gambar 5.15.

```
import os
import base64
from PIL import Image
from pathlib import Path
from feature_extractor import FeatureExtractor
from database import insert, check
from connection import connect

if __name__ == "__main__":
    fe = FeatureExtractor()
    conn = connect()

    path = "static/dataset/"

    for cat in os.listdir(path):
        id = check(cat, conn)
        if id == None:
            pass
        else:
            print(f"Proses ekstraksi fitur pada kategori
{cat}")
            for img_path in
sorted(Path(f"./static/dataset/{cat}").glob("*jpg")):
                feature = fe.extract(img=Image.open(img_path))

                file = open(img_path, 'rb').read()
                file = base64.b64encode(file)
                insert(id[0], feature, file, conn)

    print("Proses ekstraksi fitur selesai")

    conn.close()
```

Gambar 5. 13 *Source Code* Menambahkan Data ke Basis Data

5.1.2.4 Klasifikasi dan Mengukur Kemiripan

Terdapat 2 tahap terakhir yaitu melakukan proses klasifikasi dan mengukur kemiripan, proses klasifikasi berguna untuk memprediksi kategori atau label dari gambar masukan, sedangkan proses mengukur kemiripan ialah proses mengukur *euclidean distance* antara vektor gambar masukan dengan vektor gambar yang sudah ada dalam basis data. Kedua proses tersebut saling berkaitan dimana proses klasifikasi akan menghasilkan id atau nama kategori, hasil inilah yang nantinya digunakan untuk memfilter gambar yang ada dalam basis data, jadi dalam proses mengukur kemiripan hanya menggunakan antara data masukan dengan data yang sudah di filter sesuai kategori atau label gambar masukan. Hal ini berguna untuk mengurangi komputasi karena sistem tidak akan menghitung *euclidean distance* dari vektor masukan dengan seluruh vektor yang ada dalam database. Bisa dibayangkan jika basis data memiliki ribuan sampai jutaan vektor gambar maka proses yang dibutuhkan mencari vektor yang memiliki kedekatan sangatlah lama dan memiliki biaya yang tinggi. Berikut adalah source code dalam melakukan klasifikasi pada Gambar 5.14 dan pada Gambar 5.15 ialah proses menghitung jarak euclidean distance antara vektor masukan dengan vektor yang berada dalam basis data.

```
model = keras.models.load_model("static/model/model_resnet.h5")
def predict(x):
    predictions = model(x)
    pred = predictions[0]
    label = np.argmax(pred)
    return label
```

Gambar 5. 14 Source Code Proses Klasifikasi

```

# Prediction class
tensor = transform_image(rgb_img)
# return id category and probability classes
category, prob = predict(tensor)
features, img_binary = select(conn, category)

# Run search
query = fe.extract(img)
# Menhitung jarak euclidean
dists = np.linalg.norm(features-query, axis=1)
# indexing 30 hasil teratas
ids = np.argsort(dists)[:30]
scores = [(convert_dist(dists[id]), img_binary[id]) for id in
ids]

```

Gambar 5. 15 *Source Code* Proses Menghitung *Euclidean Distance*

5.1.2.5 Membuat Tampilan Berbasis Web

Dalam pembuatan web untuk melakukan pencarian gambar yaitu digunakan sebuah kerangka aplikasi dan tampilan web menggunakan Flask. Digunakannya Flask dalam pembuatan sistem berbasis web karena mudah dalam penggunaannya dan mudah saat integrasi ke fungsi-fungsi lain. Untuk pembuatan tampilan agar dapat melakukan fungsi sistem pencarian gambar dan menampilkan informasi yang didapatkan, Flask diintegrasikan dengan file html untuk mengatur tata letak dan isi tampilan web. Berikut adalah source code dari Flask dapat dilihat pada Gambar 5.16.

```

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        uploaded_file = request.files["query_img"]
        filename = secure_filename(uploaded_file.filename)
        if filename != '':
            file_ext = os.path.splitext(filename)[1].lower()
            if file_ext not in app.config['UPLOAD_EXTENSIONS']:
                return render_template("index.html",
ext='Uploaded file is not a valid image. Only JPG, JPEG and PNG
files are allowed')
            else:
                # open image PIL
                img = Image.open(uploaded_file.stream)
                rgb_pred = img.convert('RGB')
                rgb_blob = img.convert('RGB')
                # image to blob
                buffered = BytesIO()
                rgb_blob.save(buffered, format="JPEG")
                img_encode =
base64.b64encode(buffered.getvalue())
                # decode image base64 to load on HTML
                img_decode = img_encode.decode("utf-8")
                # Prediction class
                tensor = transform_image(rgb_pred)
                # return id category and probability classes
                category, prob = predict(tensor)
                features, img_binary = select(conn, category)
                # Run search
                query = fe.extract(img)
                # Menghitung jarak euclidean
                dists = np.linalg.norm(features-query, axis=1)
                # indexing 30 hasil teratas
                ids = np.argsort(dists)[:30]
                scores = [(dists[id], img_binary[id]) for id in
ids]
                return render_template("index.html",
query_path=img_decode, scores=scores, probability=prob,
vinput=query, vdb=features[ids[0]], hasil=dists[ids[0]],
hasil_gb=img_binary[ids[0]])
            else:
                return render_template("index.html", ext = "Please
select image file to upload")
        else:
            return render_template("index.html")

if __name__ == "__main__":
    app.run()

```

Gambar 5. 16 Source Code Flask

Berikut adalah *source code* dari file html yang dihubungkan dengan flask pada Gambar 5.17.

```
<body>
  <div class="container">
    <h2 style="text-align:center;">IMPLEMENTASI DEEP
    LEARNING UNTUK SISTEM PENCAIRAN GAMBAR PRODUK MENGGUNAKAN
    ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN)</h2>
    <form method="POST" enctype="multipart/form-data">
      <input type="file" class="form-control"
name="query_img" accept="image/png, image/jpeg, image/jpg"><br>
      <input type="submit" class="btn btn-primary">
</form>{% if ext %}
<div class="alert alert-danger" role="alert">ext}}div>{% endif
%}<h2>Query:</h2>if query_path %}
  
  {% endif %}
  <h4>Urutan Hasil Klasifikasi: {{ probability
}}</h4>
  <h2>Proses Perhitungan:</h2>
  <h4>Berikut ialah contoh perhitungan gambar input
dengan 1 vektor yang ada dalam database (terdapat ribuan vektor
dalam DB)</h4>
  <label for="1">Vektor gambar Input hasil ekstraksi
fitur oleh model: </label>
  <textarea id="1" name="1" rows="1" cols="50">
    {{ vinput }}</textarea>
  <label for="2">1 contoh Vektor gambar yang ada
dalam database: </label>
  <textarea id="2" name="2" rows="1" cols="50">{{ vdb }}
  </textarea><br><h4>Antara vektor gambar input
terhadap semua vektor gambar yang ada dalam database sesuai
kategori hasil Klasifikasi dihitung kedekatannya dengan
euclidean distance</h4>
  <h4>Hasil perhitungan dengan rumus euclidean distance ialah: {{
hasil }}</h4><h4>Setelah dilakukan semua perhitungan lalu
dilakukan sorting dimana hasil yang mendekati 0 atau sama
dengan 0ialah hasil teratas dan hanya diambil 30 teratas karena
ingin menampilkan 30 gambar yang mirip.
  Dalam database satu baris terdapat kolom id, id_cat, vektor,
dan gambar asli. Jadi tinggal menampilkan gambar asli sesuai
nomor id vektor teratas. </h4><h4>Berikut gambar hasil setelah
sorting 1 teratas: </h4><figure style="margin-right: 15px;
margin-bottom: 15px;">
  </figure>
<h2>Results:</h2>
{% for score in scores %}
<figure style="float: left; margin-right: 20px; margin-bottom:
20px;"> 
<figcaption>Kemiripan {{ score[0] }}%</figcaption>
</figure>{% endfor %}div></body>
```

Gambar 5. 17 Source Code File Html

Dari *hasil source code* Gambar 5.16 dan Gambar 5.17 menghasilkan sebuah tampilan *user interface* web yang terlihat pada Gambar 5.18.

IMPLEMENTASI DEEP LEARNING UNTUK SISTEM PENCARIAN GAMBAR PRODUK MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN)

Choose File | No file chosen

Submit

Query:

Proses CNN:

Array gambar untuk diinputkan ke model :

Hasil setelah dimasukkan model CNN (probabilitas masing-masing kelas):

Di urutkan dari probabilitas yang terbesar sesuai index hasil CNN :

Berikut index dan nama kelas :

Setelah diurutkan kelas yang memiliki probabilitas tertinggi pada index
Index sesuai nama kelas yaitu
Dari hasil diatas digunakan untuk mendapatkan vektor gambar pada DB sesuai nama kelas yaitu

Results:

Gambar 5. 18 Tampilan *User Interface* Web

5.1. Hasil

5.1.1 Hasil Pelatihan

Peneliti melakukan klasifikasi terhadap citra *produk fashion* dengan 42 kelas atau kategori. Proses pertama yang dilakukan adalah dengan meload data yang akan digunakan saat proses pelatihan kemudian saat proses pelatihan akan menghasilkan model dengan performa terbaik model tersebut itulah yang nantinya akan digunakan saat proses pengujian. Parameter yang dapat digunakan untuk mengukur tingkat keberhasilan adalah nilai akurasi. Dalam proses pelatihan digunakan packages keras pada python dengan *backend* tensorflow. Pada proses pelatihan terdapat 3 arsitektur berbeda yaitu arsitektur rancangan yang diusulkan dan arsitektur yang memanfaatkan model yang sudah jadi antara lain VGG16 dan ResNet50. Dari ketiga arsitektur terdapat enam variasi model yang dibuat dengan *optimizer* dan *learning rate* yang membedakannya. Semua hasil pelatihan dicatat

dan dibandingkan saat model disimpan dari *accuracy* validasi terbaik menggunakan *callback* model *checkpoint*, namun grafik dari pelatihan tetap akan mencatat sampai pelatihan berhenti.

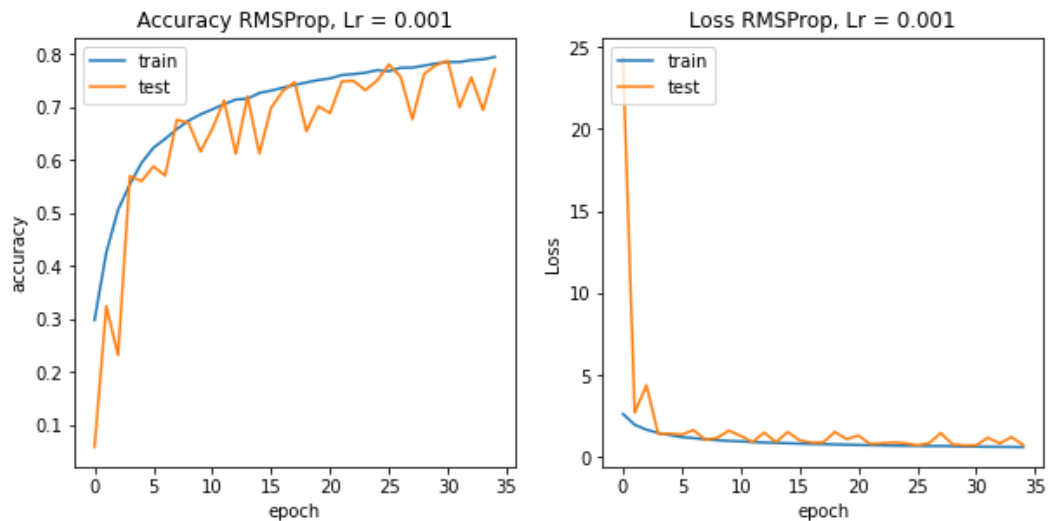
5.1.1.1 Model yang Diusulkan

Saat membuat model ini, arsitektur yang digunakan adalah jaringan residual. Berikut merupakan hasil *training* dengan menggunakan 6 variasi model yang dapat dilihat pada Tabel 5.2.

Tabel 5. 2 Perbandingan Hasil *Training* Model yang Diusulkan

No Model	Jenis Optimizers	Stop Epoch	Learning Rate	Accuracy Training	Accuracy Validasi	Accuracy Testing
1	ADAM	11	0.01	70.03	73.34	74
2	ADAM	27	0.001	77.31	78.80	79
3	ADAM	19	0.0001	74.39	75.51	76
4	RMSProp	15	0.01	71.58	73.20	74
5	RMSProp	31	0.001	78.45	78.70	79
6	RMSProp	13	0.0001	69.78	71.47	71

Dari hasil perbandingan yang disajikan pada Tabel 5.2 model yang menggunakan *optimizer* ADAM dengan *learning rate* sebesar 0.001 atau model 2 dan model yang menggunakan RMSProp dengan *learning rate* sebesar 0.001 atau model 5 juga memiliki hasil yang tidak jauh berbeda. Model 2 memiliki keunggulan pada akurasi validasi yang mendapatkan skor tertinggi yaitu 78.80% serta tidak jauh berbeda dengan model 5 yang mendapatkan skor 78.70% dan keduanya hanya memiliki selisih 0.10% saja, namun pada akurasi training model 5 memiliki keunggulan yaitu mendapatkan skor sebesar 78.45% dibanding model 2 yang mendapatkan skor 77.31% dan berarti keduanya memiliki selisih sebesar 1.14% jauh lebih besar dari selisih pada akurasi validasi. Sedangkan akurasi testing mendapatkan nilai yang sama antara kedua model tersebut yaitu model 2 dan 5, dikarenakan model 5 unggul pada jumlah selisih masing-masing akurasi maka model 5 menjadi model yang terbaik pada arsitektur model rancangan yang diusulkan dengan *optimizer* yaitu RMSProp dan *learning rate* sebesar 0.001. Berikut adalah tampilan grafik *training* model 5 dapat dilihat pada Gambar 5.19.



Gambar 5. 19 Grafik Proses Pelatihan Menggunakan Model yang Diusulkan

Berdasarkan grafik pada Gambar 5.19 dapat dilihat bahwa *accuracy train* dan *accuracy test* tidak jauh berbeda sehingga dapat disimpulkan bahwa model tersebut tidak terjadi *overfitting*.

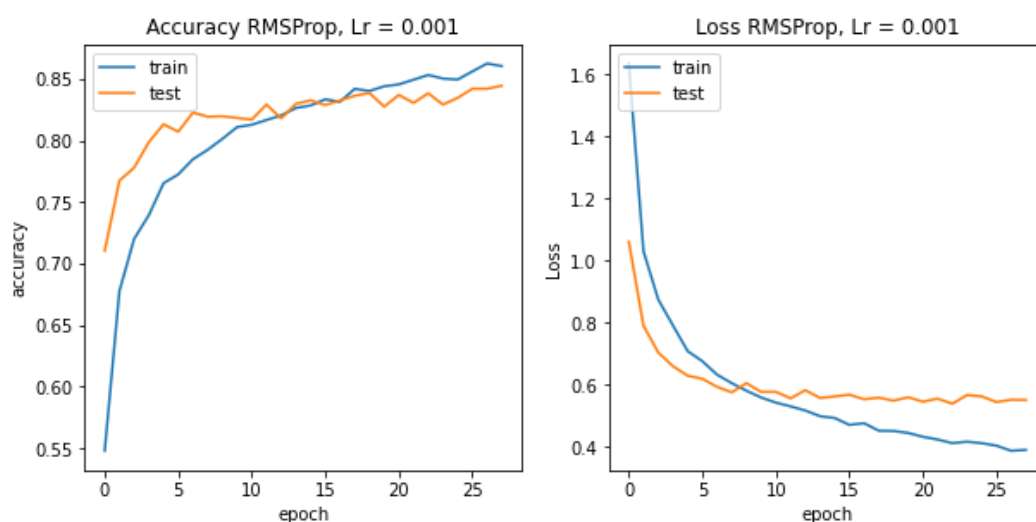
5.1.1.2 Model VGG16

VGG16 adalah sebuah model yang mudah dimengerti dan simpel serta menjadi salah satu model yang sering kali digunakan untuk belajar *Machine Learning*, hal inilah yang menjadi salah satu kelebihan dari model ini. Kelemahan VGG16 itu sendiri, yaitu mempunyai terlalu banyak parameter, ukuran model yang sangat besar (parameter sekitar 134.7 juta). Berikut merupakan hasil *training* dengan menggunakan 6 variasi model yang dapat dilihat pada Tabel 5.3.

Tabel 5. 3 Perbandingan Hasil *Training* Model VGG16

No Model	Jenis Optimizers	Stop Epoch	Learning Rate	Accuracy Training	Accuracy Validasi	Accuracy Testing
1	ADAM	17	0.01	81.11	82.84	83
2	ADAM	13	0.001	81.58	82.73	83
3	ADAM	46	0.0001	83.76	84.19	84
4	RMSProp	10	0.01	79.59	81.92	82
5	RMSProp	28	0.001	86.03	84.43	85
6	RMSProp	46	0.0001	84.48	84.26	84

Dari hasil perbandingan yang disajikan pada Tabel 5.3 didapatkan model dengan performa terbaik dibandingkan 5 varian lainnya yaitu model dengan jenis *optimizer* RMSProp dan *learning rate* 0.001. Model tersebut memiliki skor akurasi *training* sebesar 86.03%, akurasi validasi 84.43%, dan akurasi *testing* 85% pada epoch ke 28. Berikut merupakan visualisasi akurasi dan loss dapat dilihat pada Gambar 5.20.



Gambar 5. 20 Grafik Proses Pelatihan Menggunakan VGG16

Dilihat dari grafik *loss* terjadi plebaran sedikit antara train loss dengan test loss pada epoch sekitar 15. Disinilah peran *callback* *earlystopping* sangat berguna dimana ketika model sudah tidak mengalami perubahan di beberapa epoch maka proses pelatihan akan otomatis dihentikan untuk menghindari model mengalami penurunan performa. Berdasarkan grafik akurasi *train* dan akurasi *test* keduanya memiliki skor yang tidak jauh al ini juga menandakan bahwa proses pelatihan terjadi *goodfit*.

5.1.1.3 Model ResNet50

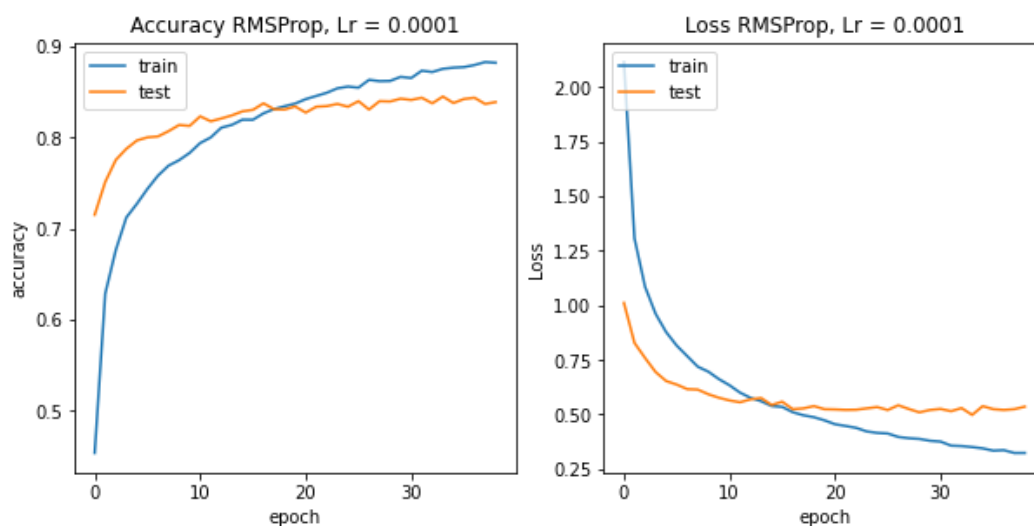
Model ini berfokus pada akurasi perhitungan dan juga merupakan model dengan lapisan terbanyak. Hal ini dimungkinkan karena resnet menggunakan jaringan residual sehingga tidak perlu mengkhawatirkan permasalahan penurunan *gradient* yang membuat performa model menurun jika model terlalu banyak lapisan. Dibandingkan dengan model lain, keunggulan model ini tentu saja

akurasi yang tinggi. Kelemahan model ini sendiri adalah arsitektur model terlalu memperhatikan faktor kedalaman model dan tidak mengembangkan faktor lebar. Berikut merupakan hasil *training* dengan menggunakan 6 variasi model yang dapat dilihat pada Tabel 5.4.

Tabel 5. 4 Perbandingan Hasil *Training* Model ResNet50

No Model	Jenis Optimizers	Stop Epoch	Learning Rate	Accuracy Training	Accuracy Validasi	Accuracy Testing
1	ADAM	17	0.01	83.47	83.62	84
2	ADAM	15	0.001	84.39	83.38	84
3	ADAM	31	0.0001	86.27	83.99	84
4	RMSProp	9	0.01	80.92	82.63	82
5	RMSProp	12	0.001	83.62	83.38	83
6	RMSProp	34	0.0001	87.50	84.46	84

Dari hasil perbandingan yang disajikan pada Tabel 5.3 didapatkan model dengan performa terbaik dibandingkan 5 varian lainnya yaitu model dengan jenis *optimizer* RMSProp dan *learning rate* 0.0001. Model ini memiliki keunggulan pada akurasi *training* yaitu sebesar 87.50% dan akurasi validasi 84.46% sedangkan untuk akurasi testing yang mendapatkan skor 84% sama dengan model lainnya pada jenis *optimizer* yang menggunakan ADAM. Berikut merupakan visualisasi akurasi dan loss dapat dilihat pada Gambar 5.21.



Gambar 5. 21 Grafik Proses Pelatihan Menggunakan ResNet50

Dari Gambar 5.21 dapat dilihat jika akurasi *training* model mendapatkan hasil yang sangat memuaskan. Grafik akurasi dan *loss* keduanya hampir memiliki pola yang sama yaitu pada epoch sekitar 15 terjadi sebuah *cross* akurasi maupun *loss* pada data *train* dan test. Berdasarkan grafik akurasi dan *loss* keduanya memiliki skor dengan selisih yang tidak cukup jauh sehingga dapat disimpulkan bahwa model tersebut tidak terjadi *overfitting*.

5.1.1.4 Perbandingan Hasil Pelatihan

Perbandingan hasil pelatihan dilakukan dengan membandingkan hasil penelitian dari setiap model yang terbaik pada arsitekturnya masing-masing. Setelah itu akan dipilih model terbaik untuk membangun sistem pencarian gambar. Berikut adalah perbandingan dari model terbaik pada setiap arsitekturnya yang ada pada Tabel 5.5.

Tabel 5. 5 Tabel Perbandingan Performa Model

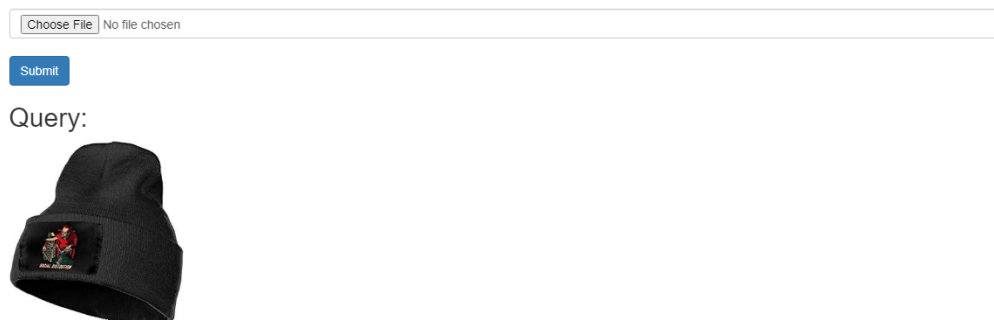
Arsitektur	Jenis Optimizers	Learning Rate	Accuracy Training	Accuracy Validasi	Accuracy Testing
yang diusulkan	RMSProp	0.001	78.45	78.70	79
VGG16	RMSProp	0.001	86.03	84.43	85
ResNet50	RMSProp	0.0001	87.50	84.46	84
Hasil			ResNet50	ResNet50	VGG16

Berdasarkan Tabel 5.5 yaitu pada akurasi *training* ResNet50 memiliki keunggulan sebesar 87.50% dibanding model lainnya yang sebesar 86.03% pada model VGG16 dan 78.45% pada model yang diusulkan, kemudian pada nilai akurasi validasi ResNet50 memiliki keunggulan yang sangat sedikit yaitu sebesar 84.46% selisih 0.03% dibanding model ResNet50 yang sebesar sebesar 84.43% sedangkan model yang diusulkan mendapatkan 78.70%, kemudian pada nilai akurasi testing model VGG16 memiliki keunggulan yaitu sebesar 85% dibanding model lainnya yang sebesar 84% pada model ResNet50 dan 79% pada model yang diusulkan. Maka dari hasil yang telah diuraikan model ResNet50 menjadi model dengan performa terbaik dari hasil keunggulan pada nilai akurasi *training*, akurasi validasi, dan akurasi testing meskipun selisih sangat sedikit pada akurasi validasi dari ResNet50.

5.1.2 Hasil Sistem Pencarian Menggunakan Gambar

Pengembangan sistem pencarian citra dilakukan dengan memilih model terbaik dari hasil penelitian, dari hasil pelatihan didapatkan model terbaik yaitu model ResNet50 dengan parameter *learning rate* 0.0001 dan *optimizer* menggunakan RMSProp selanjutnya melakukan ekspor model dengan bantuan *library* keras, yang akan menyimpan bobot model pelatihan. Model ini dibutuhkan untuk melakukan ekstraksi fitur dan klasifikasi dalam sistem pencarian gambar. Setelah model disimpan, dibuat sistem berbasis web menggunakan bahasa Python dengan bantuan aplikasi Flask. *Interface* sistem berisi *input* dan sampai 30 hasil pencarian gambar. Tampilan *interface* dapat dilihat pada Gambar 5.18 dan untuk hasil pencarian dapat dilihat pada Gambar 5.22 dan Gambar 5.23.

IMPLEMENTASI DEEP LEARNING UNTUK SISTEM PENCARIAN GAMBAR
PRODUK MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK
(CNN)



Gambar 5. 22 Pencarian *Hats*

sebagian kecil dari semua data dalam kategori tersebut. Kelemahan pada model ini adalah lapisan konvolusi yang digunakan terlalu sedikit sehingga model tidak bisa mendapatkan fitur yang lebih spesifik dari gambar, meskipun secara arsitektur model ini bisa dibuat sangat dalam tapi jika memaksakan kedalam arsitektur tanpa diikuti dengan banyaknya data training akan membuat model overfit terhadap data tersebut dan fokus terhadap noise dari gambar dibanding dengan fitur dari gambar sendiri. Kelebihan model ini adalah meskipun memiliki jumlah lapisan sedikit namun bisa mendapatkan nilai akurasi yang cukup bagus.

b. Arsitektur Model VGG16

Kelemahan pada model VGG16 itu sendiri, yaitu mempunyai terlalu banyak parameter, ukuran model yang sangat besar (parameter sekitar 138 juta) dan juga waktu pelatihan yang lama. Kelebihan VGG16 ini model yang sangat simpel, mudah untuk dimengerti dan menjadi salah satu model yang sering digunakan untuk belajar machine learning.

c. Arsitektur Model ResNet50

Model ini berfokus pada *computational accuracy* dan juga merupakan model dengan *layer* paling banyak. Hal ini memungkinkan karena ResNet50 menggunakan *residual network* sehingga tidak perlu mengkhawatirkan permasalahan *diminishing gradient* yang membuat performa model menurun jika model terlalu banyak lapisan. Kelebihan model ini tentunya adalah akurasi yang tinggi dibanding dengan model lainnya. Kelemahan dari model ini sendiri adalah arsitektur model terlalu fokus pada faktor kedalaman model dan tidak mengembangkan faktor lebar.

Berikut pada Tabel 5.5 adalah perbandingan dari masing-masing struktur arsitektur yang digunakan dan waktu proses pelatihan setiap epoch.

Tabel 5. 6 Perbandingan Struktur Arsitektur dan Pelatihan

Model	Input	Jumlah Convolutional	Jumlah filter	Ukuran Kernel	Fungsi Aktivasi	Waktu per epoch
Sendiri	128x128	4	240	3x3	ReLU dan Softmax	140s
VGG16	128x128	13	4.224	3x3	ReLU dan Softmax	191s

Tabel 5. 7 Lanjutan Perbandingan Struktur Arsitektur dan Pelatihan

Model	Input	Jumlah Convolutional	Jumlah filter	Ukuran Kernel	Fungsi Aktivasi	Waktu per epoch
ResNet50	128x128	52	26.560	(4x4), (8x8), (16x16), (32x32), (64x64)	ReLU dan Softmax	162s

5.2.2 Evaluasi dan Analisis Performa Model

Pada bab perbandingan hasil pelatihan didapatkan model yang terbaik yang sudah ditentukan yaitu model dengan arsitektur ResNet50 menggunakan *optimizers* RMSProp dan *learning rate* sebesar 0.0001. Berikut adalah perhitungan presisi, *recall*, dan *f1-score* pada setiap kategori atau kelas yang ada dan dapat dilihat pada Tabel 5.7.

Tabel 5. 8 Perhitungan Presisi, *Recall*, *F1-Score* Pada Model Terbaik

Kategori	Presisi	Recall	F1-Score
<i>backpacks</i>	0.97	0.97	0.97
<i>belts</i>	0.93	0.94	0.93
<i>blazers</i>	0.62	0.78	0.69
<i>boots</i>	0.86	0.96	0.91
<i>bottom</i>	1.00	0.99	0.99
<i>caps</i>	1.00	0.93	0.96
<i>casual</i>	0.96	0.86	0.91
<i>clutch</i>	0.86	0.90	0.88
<i>coats</i>	0.85	0.86	0.86
<i>cocktail</i>	0.88	0.86	0.87
<i>flats</i>	0.75	0.64	0.69
<i>gloves</i>	0.96	0.97	0.97
<i>hairaccessories</i>	0.93	0.74	0.82
<i>handbags</i>	0.89	0.90	0.89
<i>hats</i>	0.81	0.96	0.88
<i>heels</i>	0.83	0.69	0.75
<i>jackets</i>	0.64	0.51	0.56
<i>jeans</i>	0.92	0.86	0.89
<i>jewelry</i>	0.89	0.97	0.93
<i>jumpsuit</i>	0.87	0.80	0.83
<i>legging</i>	0.80	0.80	0.80
<i>loafers</i>	0.82	0.84	0.83
<i>makeup</i>	0.99	0.97	0.98

Tabel 5. 9 Lanjutan Perhitungan Presisi, *Recall*, *F1-Score* Pada Model Terbaik

Kategori	Presisi	Recall	F1-Score
<i>maxi</i>	0.69	0.23	0.35
<i>midi</i>	0.51	0.77	0.62
<i>mini</i>	0.72	0.81	0.76
<i>onepiece</i>	0.94	0.96	0.95
<i>pants</i>	0.64	0.70	0.67
<i>ponchos</i>	0.55	0.65	0.59
<i>sandals</i>	0.59	0.68	0.63
<i>satchel</i>	0.88	0.84	0.86
<i>scarves</i>	0.91	0.93	0.92
<i>shorts</i>	0.94	0.76	0.84
<i>skirts</i>	0.82	0.93	0.87
<i>slides</i>	0.77	0.75	0.76
<i>sneakers</i>	0.91	0.92	0.92
<i>sunglasses</i>	0.98	1.00	0.99
<i>sweatpants</i>	0.81	0.77	0.79
<i>top</i>	0.96	0.96	0.96

5.2.3 Evaluasi dan Analisis Model Transfer Learning

Model yang mengimplementasikan metode ini akan memiliki nilai akurasi yang lebih tinggi dibandingkan dengan model yang dibuat sendiri. Faktor-faktor yang mempengaruhi hal ini adalah faktor ukuran dataset yang tidak terlalu banyak beserta faktor dari pengetahuan model sebelumnya. Ukuran dataset yang tidak terlalu banyak akan lebih cocok menggunakan metode *transfer learning* dikarenakan model sudah memiliki pengetahuan dan bobot dari dataset sebelumnya, setelah itu model hanya perlu beradaptasi terhadap dataset baru atau yang ditambahkan.

5.2.4 Model Sistem Pencarian Gambar

Pada subbab 5.1.5 diperoleh model yang memiliki performa terbaik yaitu ResNet50 dengan *optimizer* RMSProp dan *learning rate* 0.0001, maka dalam pembuatan sistem pencarian menggunakan gambar untuk melakukan klasifikasi dan ekstraksi fitur digunakan model ResNet50 dengan *optimizer* RMSProp dan *learning rate* 0.0001.

BAB VI PENUTUP

6.1 Simpulan

Berdasarkan hasil pengujian model *Deep Learning* dengan menggunakan metode *Convolutional Neural Network* dan pembangunan sistem pencarian menggunakan gambar pada penelitian ini, maka dapat ditarik kesimpulan sebagai berikut:

1. Implementasi pembangunan sistem pencarian berdasarkan gambar menggunakan CNN dilakukan beberapa tahap yaitu pembuatan model untuk melakukan klasifikasi dan ekstraksi fitur dari gambar, melakukan *load* model ke dalam sistem, melakukan ekstraksi fitur gambar untuk dimasukkan dalam basis data, membuat sebuah web untuk menerima *input* gambar dari *user*, setelah itu dilakukan klasifikasi gambar dan dilakukan perhitungan kemiripan gambar lalu ditampilkan hasilnya pada web yang berupa *output* gambar yang mirip beserta nilai kedekatannya.
2. Arsitektur terbaik pada penelitian pembangunan sistem ini adalah sistem yang menggunakan metode *transfer learning* dengan model ResNet50. Pada arsitektur ResNet50 ini menggunakan jenis optimizer RMSProp dan menggunakan *learning rate* sebesar 0.0001. Model ini mendapatkan nilai akurasi *training* yaitu sebesar 87.50% dan akurasi validasi 84.46% sedangkan untuk akurasi testing yang mendapatkan skor 84%.
3. Akurasi CNN dapat ditingkatkan dengan cara melakukan augmentasi data yang digunakan dalam proses pelatihan dan melakukan *hyperparameter tuning*.
4. Sistem dapat menampilkan *output* gambar yang serupa dengan yang sudah di masukkan ke sistem yaitu dengan cara hasil ekstraksi fitur gambar dari *input* yang berupa *array* vektor dan dataset yang sudah di ekstraksi fitur berupa kumpulan *array* vektor yang berada dalam basis data dihitung jarak eculideannya, setelah selesai dihitung semua antara

array vektor *input* dengan kumpulan *array* vektor *output* lalu diurutkan berdasarkan hasil jarak terdekat dan ditampilkan pada web.

6.2 Saran

1. Untuk meningkatkan performa model dapat dilakukan *hyperparameter tuning* dengan variasi yang lebih banyak lagi agar akurasi model dapat meningkat menjadi lebih baik.
2. Menggunakan lebih banyak model *transfer learning* lainnya seperti mobilenet, inception, dan sebagainya untuk penelitian model agar mendapatkan referensi atau pembandingan yang lebih banyak.
3. Dapat menggunakan *object detection* dalam mencari fitur dalam gambar karena *object detection* mampu mengenali lebih dari satu kelas dalam gambar dan akan meningkatkan ketepatan dalam pencarian gambar.

DAFTAR PUSTAKA

- Fengzi, L., Kant, S., Araki, S., Bangera, S. dan Shukla, S.S. (2020), *Neural Networks For Fashion Image Classification And Visual Search*, *SSRN Electronic Journal*
- Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J. dan Tavel, S. (2015), *Visual Search At Pinterest*, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015-Augus, 1889–1898
- Park, S., Shin, M., Ham, S., Choe, S. dan Kang, Y. (2019), *Study On Fashion Image Retrieval Methods For Efficient Fashion Visual Search*, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019-June, 316–319
- Simran, A., Shijin Kumar, P.. dan Bachu, S. (2021), *Content Based Image Retrieval Using Deep Learning Convolutional Neural Network*, *IOP Conference Series: Materials Science and Engineering*, 1084(1), 012026
- Zhang, Y., Pan, P., Zheng, Y., Zhao, K., Zhang, Y., Ren, X. dan Jin, R. (2018), *Visual Search At Alibaba*, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 993–1001