# Pembuatan Aplikasi Untuk Mengklasifikasikan Citra Digital Wajah *Anime* Menggunakan Metode Prototypical Networks Pada *Anime* Hunter X Hunter

Hafidh Soekma Ardiansyah[1], Andi Iwan Nurhidayat[2]

*Manajemen Informatika, Universitas Negeri Surabaya*
*Jl. Ketintang, Ketintang, Kec. Gayungan, Surabaya, Jawa Timur 60231*
[1]hafidh.19031@mhs.unesa.ac.id
[2]andyl34k5@unesa.ac.id

***Abstrak*— Kecerdasan buatan adalah cabang ilmu komputer yang mereplika kecerdasan manusia, salah satunya dengan menggunakan machine learning. Deep learning, salah satu metode dalam machine learning, membutuhkan banyak data untuk belajar. Untuk mengatasi keterbatasan ini, penulis menggunakan metode prototypical networks. Metode ini membandingkan vector embeddings artificial neural networks dengan vector embeddings rata-rata yang disimpan dalam file JSON sebagai pusat setiap label. Penulis membandingkan metode densenet121, efficientnet_b4, inception_v4, mobilenetv3_large_100, resnet50, vgg16, dan vit_base_patch16_224_dino, dengan pendekatan prototypical networks. Metode vit_base_patch16_224_dino menghasilkan hasil terbaik, dengan accuracy 0.6600, precision 0.6642, recall 0.6600, dan f1-score 0.6584. Penelitian ini menunjukkan bahwa metode vit_base_patch16_224_dino prototypical networks tersebut efektif untuk klasifikasi citra digital wajah anime Hunter X Hunter.***

***Kata kunci*— Kecerdasan Buatan, Machine Learning, Vector Embeddings, Prototypical Networks, Euclidean Distance**

***Abstract*— Artificial intelligence is a branch of computer science that aims to replicate human intelligence, and one of its methods is through the use of machine learning. Deep learning, which is a subfield of machine learning, relies heavily on data for learning. To address this limitation, the author employs the prototypical networks method. This method compares vector embeddings of artificial neural networks with the average vector embeddings stored in a JSON file, which serve as the centers for each label. The author compares the following methods: densenet121, efficientnet_b4, inception_v4, mobilenetv3_large_100, resnet50, vgg16, and vit_base_patch16_224_dino, using the prototypical networks approach. The vit_base_patch16_224_dino method produces the best results, with an accuracy of 0.6600, precision of 0.6642, recall of 0.6600, and an f1-score of 0.6584. This research demonstrates that the prototypical networks method with vit_base_patch16_224_dino is effective for classifying digital images of anime faces from Hunter X Hunter.***

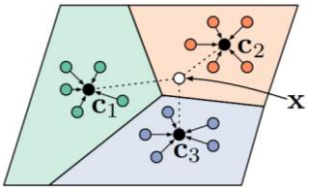
***Keywords*— Artificial Intelligence, Machine Learning, Vector Embeddings, Prototypical Networks, Euclidean Distance**

## I. PENDAHULUAN

*Machine learning* adalah sebuah cabang dari kecerdasan buatan ilmu komputer yang belajar dari sebuah data, untuk menghasilkan sebuah hasil. *Deep learning* salah satu cabang *machine learning* adalah kumpulan dari *artficial neural networks*, yang terdiri dari *input layers*, *hidden layers*, dan *output layers* yang dimana jumlah *hidden layers* memiliki jumlah yang banyak dan menjadi *deep* atau dalam [1]. Kekurangan pada *deep learning* adalah membutuhkan data yang sangat banyak dan data tersebut harus mempunyai label untuk setiap datanya dan juga semakin banyak data yang dibutuhkan, maka komputasi yang dibutuhkan juga berbanding lurus dengan data yang ada [1]. Sebagai contoh, pada data bernama *ImageNet: A Large-Scale Hierarchical Image Database* memiliki total data sejumlah 1.2 juta citra digital data latih, 50.000 citra digital data validasi, dan 150.000 citra digital data tes [2].

Salah satu cara untuk mengatasi masalah pada *deep learning* adalah menggunakan salah satu metode yang dinamakan Prototypical Networks. Prototypical networks adalah metode dalam *deep learning* yang memungkinkan pemahaman objek hanya dengan contoh data terbatas. Perbedaan definisi dengan *deep learning* umum adalah penambahan konsep *support set*, *K-way*, *N-shot*, dan *query*. *Support set* adalah data *training*, *K-way* adalah jumlah label, *N-shot* adalah jumlah data per label, dan *query* adalah data *testing* pada prototypical networks. Cara kerja prototypical networks melibatkan model yang telah dilatih dengan data yang beragam. Data *N-shot* dengan *K-way* dimasukkan ke dalam model untuk menghasilkan *vector embeddings*. Rata-rata *support set* menjadi *centroid* sebagai titik pusat pengukuran untuk setiap *K-way*. Untuk prediksi, *vector embeddings* dari model diukur jaraknya dengan *centroid*. Jarak terdekat menentukan label yang sesuai. *Vector embeddings* dari setiap *K-way* dipetakan ke dalam *embedding space*. *Query* juga dipetakan ke dalam *embedding space*. Jarak antara *query* dan setiap *K-way* diukur menggunakan metode seperti *euclidean distance* atau *cosine similarity*. Hasil diurutkan dari terkecil ke

terbesar untuk setiap *K-way*. Label dari *query* ditentukan oleh jarak terdekat [3].




Gambar. 1 Ilustrasi dari protoypical networks

Prototypical networks bekerja seperti layaknya manusia yang bisa memahami dan mengenali sebuah objek dengan melihat beberapa sampel citra digital saja. Jika ada label baru, manusia hanya perlu melihat beberapa sampel citra digital baru tersebut, setelah itu manusia langsung bisa memahami dan mengenali label baru tersebut. Tidak seperti *deep learning* yang harus membutuhkan banyak data untuk belajar dan juga jika terdapat label baru, *deep learning* harus belajar ulang dari awal lagi. Setelah itu, kebanyakan aplikasi pengenalan wajah berfokus pada pengenalan wajah manusia, bukan wajah selain manusia.

*Anime* adalah animasi asal Jepang yang digambarkan dengan tangan maupun menggunakan teknologi komputer. Wajah pada *anime* sendiri memiliki tampilan yang sangat berbeda dengan wajah manusia, serta wajah *anime* memiliki banyak ragam jenis gaya, seni, dan pola tergantung dari siapa pembuatnya. Maka dari itu, penulis ingin membandingkan beberapa metode dengan pendekatan prototypical networks. Setelah itu, penulis membuat sebuah aplikasi untuk menampilkan hasil dan tempat untuk demo dari hasil penelitian penulis. Disini penulis menggunakan studi kasus *anime* Hunter X Hunter dikarenakan *anime* tersebut adalah salah satu *anime* terbaik yang pernah dibuat, dengan dibuktikan bahwa *anime* tersebut mendapatkan peringkat 11 (Januari 2023), dari semua *anime* yang pernah tayang di muka bumi ini dan memiliki jumlah pengguna yang memfavoritkan *anime* tersebut sejumlah 2.632.930 (Januari 2023) pengguna menurut situs "MyAnimeList". Maka dari itu studi kasus Hunter X Hunter sangat cocok untuk menjadi pengujian pada penelitian penulis ini.

## II. METODE PENELITIAN

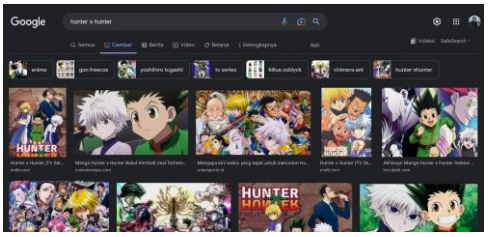
Pada tahap ini, penulis menjelaskan tahapan-tahapan yang penulis lakukan, ketika dalam proses penelitian yang penulis lakukan.

### *A. Studi Literatur*

Penulis memulai penelitian dengan studi literatur untuk menemukan masalah, mencari sumber-sumber terkait, dan sebagai acuan dalam penelitian. Studi literatur penting untuk meminimalisir kesalahan dalam penelitian dengan menggunakan referensi yang mirip. Pencarian dilakukan melalui situs seperti Google Scholar, Research Gate, dan Conference Paper dengan kata kunci seperti *face recognition*, *few shot learning*, *meta learning*, dan *artificial intelligence in anime*. Penulis juga melakukan studi literatur tentang *anime* Hunter X Hunter dan karakter yang digunakan dalam penelitian.

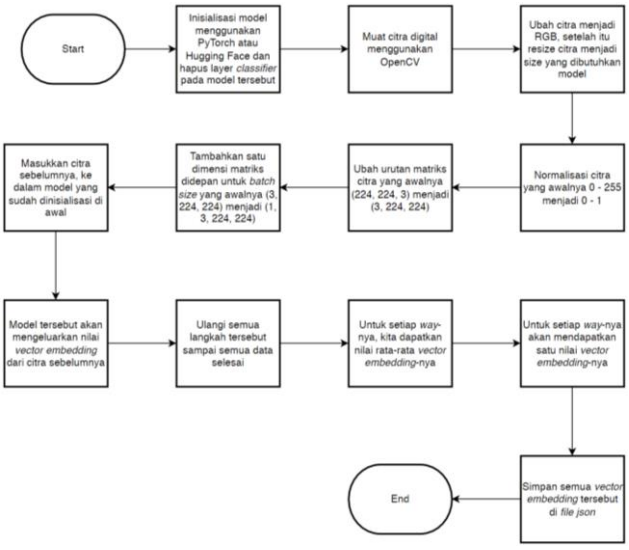
### *B. Pengambilan dan Pemrosesan Data*

Pada tahap ini, penulis melakukan menggunakan aplikasi mesin pencari Google untuk mencari citra digital wajah *anime* Hunter X Hunter, dengan batasan maksimal lima citra digital untuk setiap karakter. Namun, beberapa data yang ditemukan memiliki lisensi yang mengharuskan penulis untuk tidak menyebarkannya ke khalayak umum. Untuk menghindari *overfitting* dan *underfitting*, penulis mencari data secara manual dengan tujuan mendapatkan data dari *domain* yang berbeda. Dengan mencari data secara manual, penulis dapat memperoleh data yang beragam dengan jumlah antara satu hingga lima data.




Gambar. 2 Hasil pencarian Hunter X Hunter di Google

Setelah menemukan lima citra digital wajah *anime* untuk setiap karakter, penulis membuat *database file JSON* yang menyimpan nilai rata-rata *vector embeddings* dari setiap karakter tersebut. Untuk mendapatkan *vector embeddings*, penulis menggunakan algoritma sebagai berikut:




Gambar. 3 Algoritma untuk mendapatkan *vector embeddings database*

### *C. Penentuan Model*

Pada tahap ini, penulis menyiapkan beberapa model yang penulis gunakan. Model yang penulis gunakan adalah:

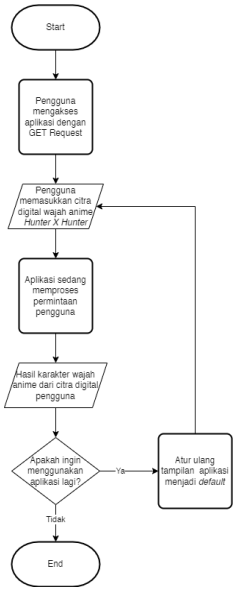
TABEL I
PERBANDINGAN MODEL YANG DIGUNAKAN

| No | Nama Model | Jumlah Parameter | *Input* Matriks | *Output* Matriks |
|---|---|---|---|---|
| 1 | DensetNet (*densetnet121*) [4] | 6.953.856 | (1, 3, 224, 224) | (1, 1024) |
| 2 | EfficientNet (*efficientnet_b4*) [4] | 17.548.616 | (1, 3, 224, 224) | (1, 1792) |
| 3 | InceptionNet (*inception_v4*) [6] | 41.142.816 | (1, 3, 224, 224) | (1, 1536) |
| 4 | MobilNet (*mobilenetv3_large_100*) [7] | 4.202.032 | (1, 3, 224, 224) | (1, 1280) |
| 5 | ResNet (*resnet50*) [8] | 23.508.032 | (1, 3, 224, 224) | (1, 2048) |
| 6 | VGG (*vgg16*) [9] | 134.260.544 | (1, 3, 224, 224) | (1, 4096) |
| 7 | Vision Transformer (*vit_base_patch16_224_dino*) [10] | 85.798.656 | (1, 3, 224, 224) | (1, 768) |

*D. Analisis Sistem*

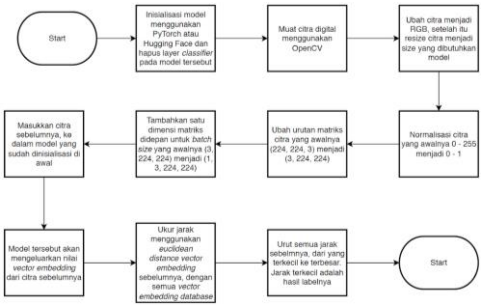
Pada tahap ini, penulis melakukan analisis sistem untuk memahami dan mengevaluasi permasalahan yang dipecahkan dalam penelitian ini. Penulis membuat sebuah aplikasi berbasis *website* karena dianggap fleksibel dan dapat digunakan pada berbagai perangkat, asalkan terdapat *web browser* dan koneksi internet. Aplikasi ini menerima citra digital wajah *anime* Hunter X Hunter yang hanya memiliki satu wajah (*single label*), dan keluarannya berupa nama karakter dari wajah tersebut (*single label*). Analisis sistem yang dilakukan meliputi aspek-aspek yang diperlukan dalam pembuatan aplikasi sebagai berikut:

1) *Spesifikasi komputer minimal*
   - RAM: 8GB
   - CPU: Intel Core i5 Generasi ke 8 atau AMD Ryzen seri 3000
   - Penyimpanan: 45GB
   - Sistem Operasi: Windows atau Linux atau macOS
   - (Opsional) VGA: Nvidia seri GTX
2) *Aplikasi dan pustaka yang harus*
   - Bahasa Pemrograman Python 3.9.13
   - Anaconda dengan Python 3.9.x
   - OpenCV
   - Streamlit
   - PyTorch
   - (Opsional) CUDA Toolkit dan CUDNN
3) *Analisis input dan output*
   - Data *input*: Citra digital wajah *anime* Hunter X Hunter
   - Data *output*: Nama karakter *anime* dari wajah karakter tersebut
4) *Hasil output*
   - Gon Freecs
   - Killua Zoldyck
   - Leorio Paladinknight
   - Kurapika
   - Hisoka Morrow
5) *Alur kerja sistem*



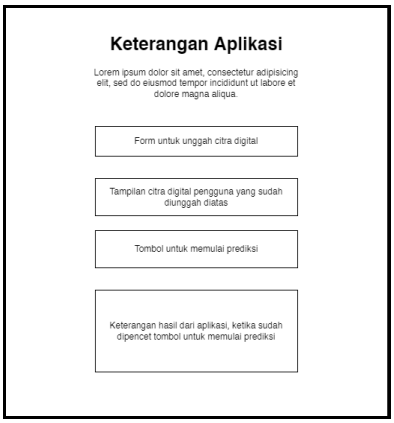

Gambar. 4 Alur kerja sistem

6) *Alur kerja pengenalan wajah anime*




Gambar. 5 Alur kerja pengenalan wajah *anime*

*7) Wireframe tampilan aplikasi*




Gambar. 6 *Wireframe* tampilan aplikasi

## *E. Implementasi dan Pengujian Sistem*

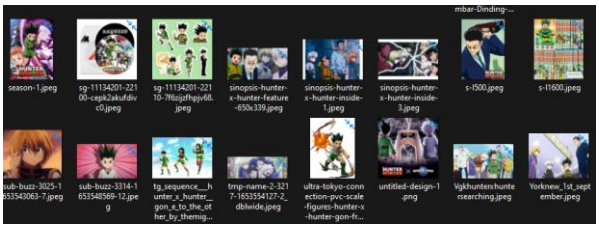
Penulis menggunakan Python 3.9.13 untuk implementasi sistem dan menggunakan pustaka Streamlit untuk membuat aplikasi. Beberapa pustaka yang digunakan adalah PyTorch dan OpenCV. Aplikasi tersebut di-*deploy* di Hugging Face, yang merupakan platform populer untuk *deploy* aplikasi AI. Dalam tahap pengujian, penulis menguji sistem menggunakan beberapa gambar wajah anime Hunter X Hunter dari berbagai episode dan sudut. Penulis juga membuat program khusus untuk menguji sistem secara otomatis dan menghitung matriks kebaikan seperti *accuracy*, *precision*, *recall*, dan *f1-score*.

# III. HASIL PENELITIAN DAN PEMBAHASAN

## *A. Pengambilan Data*

Penulis menggunakan alat *open-source* yang umum digunakan untuk pengambilan data di internet secara otomatis dan cepat. Alat tersebut adalah *ohyicong/Google-Image-Scraper* dan *Bionus/imgbrd-grabber* yang dapat diakses melalui repositori GitHub. Penulis memasukkan kata kunci terkait Hunter X Hunter seperti gon, leorio, kurapika, killua, dan hisoka untuk mengambil data.

Penulis mendapatkan 3522 data citra digital dengan format yang bervariasi seperti JPG, PNG, WEBP, dll. Resolusi citra dan nama citra juga berbeda-beda. Data tersebut perlu diproses agar dapat dianalisis dan digunakan dalam pembuatan model. Penulis menggunakan laptop Asus VivoBook seri A412DA dengan spesifikasi Ryzen 5 3500U, RAM 12GB, dan sistem operasi Windows 11 untuk melakukan proses pengolahan data tersebut.




Gambar. 7 Contoh beberapa hasil pada proses pengambilan data

## *B. Pemrosesan Data*

### *1) Tahap Pertama*

Pada tahap pertama, penulis mengubah nama setiap data menjadi format UUID untuk memastikan keunikan nama dan memudahkan analisis data. Selanjutnya, format citra digital dari semua data diubah menjadi PNG untuk keperluan pemrosesan dan kualitas yang lebih baik dibandingkan format lain seperti JPG.

Penulis menggunakan repositori GitHub *XavierJiezou/anime-face-detection* untuk mendeteksi wajah karakter *anime* dari citra digital tersebut. Hasil deteksi wajah tersebut disimpan sebagai data citra tersendiri dengan resolusi 224x224, format PNG, dan diberi nama UUID. Penulis kemudian mengelompokkan wajah-wajah tersebut secara manual ke dalam direktori-direktori terpisah sesuai dengan nama karakternya, yaitu gon, hisoka, killua, kurapika, dan leorio.


Gambar. 8 Contoh hasil sebelum dan sesudah wahjah krakter dipotong pada bagian wajahnya

### *2) Tahap Kedua*

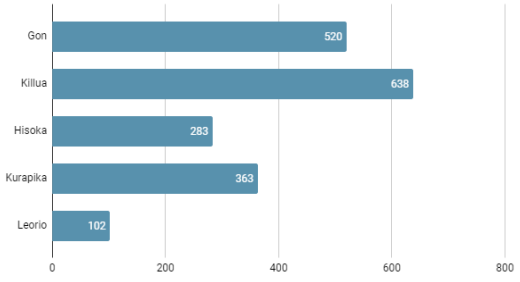
Pada tahap kedua, penulis berfokus pada mempersipkan data menjadi beberapa data yang berbeda, diantaranya adalah:

- *1 Shot – Support Set Dataset* adalah sebuah *dataset* yang memiliki satu contoh data untuk setiap *way*-nya
- *3 Shot – Support Set Dataset* adalah sebuah *dataset* yang memiliki tiga contoh data untuk setiap *way*-nya
- *5 Shot – Support Set Dataset* adalah sebuah *dataset* yang memiliki lima contoh data untuk setiap *way*-nya
- *Image Query Dataset* adalah sebuah *dataset* digunakan untuk menguji dan mengkomparasi model yang sudah penulis siapkan.
- *Image Similarity Dataset* adalah sebuah *dataset* yang membandingkan kedua citra

digital untuk menentukan apakah citra tersebut sama atau tidak. Tujuannya adalah untuk mengidentifikasi data yang tidak sesuai dengan model yang ada. Label yang digunakan adalah 1 untuk citra yang sama dan 0 untuk citra yang berbeda. Dataset ini didasarkan pada *dataset query* citra, dengan penambahan label baru untuk dataset kesamaan citra.

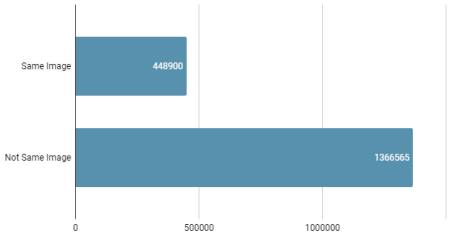
Setiap *dataset* digunakan untuk penanganan yang berbeda. *Dataset* yang mengandung kata *shot* digunakan untuk membandingkan metode yang digunakan penulis terhadap data yang dibutuhkan. Dalam penelitian asli, peneliti menggunakan konfigurasi 1, 3, dan 5 untuk membandingkan setiap model yang digunakan. *Image query* digunakan sebagai pengujian model yang digunakan penulis, sedangkan *image similarity* digunakan untuk menentukan nilai *threshold* yang tepat agar model tidak mengalami kesalahan saat pengguna memasukkan citra digital selain yang disediakan oleh penulis.

## C. *Analisis Data*



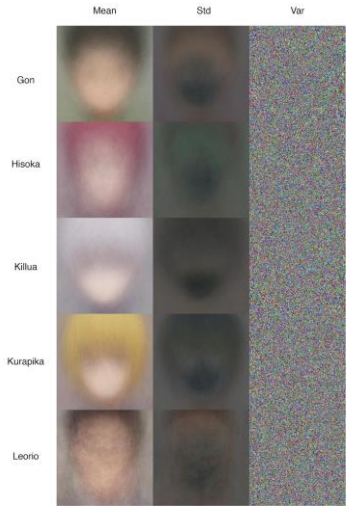

Gambar. 9 Perbandingan jumlah citra digital wajah untuk setiap karakter pada *image query dataset*

Pada gambar. 9, wajah karakter Killua muncul lebih banyak dibandingkan karakter lainnya, menurut situs "MyAnimeList", Killua adalah karakter paling favorit dalam *anime* Hunter X Hunter. Namun, terdapat ketidakseimbangan dalam tipe data citra yang digunakan untuk menguji model penulis. Meskipun demikian, ketidakseimbangan ini tidak mempengaruhi hasil prediksi model dan maka dari itu penulis tidak menggunakan metode *oversampling* atau *undersampling*.



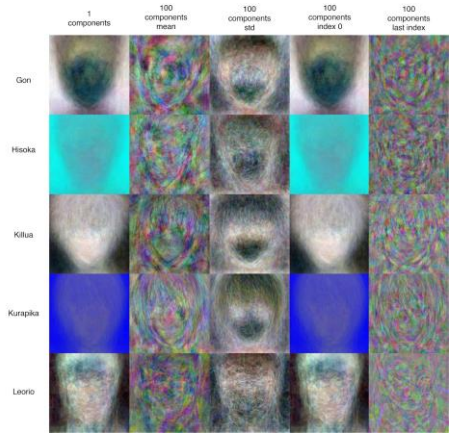

Gambar. 10 Perbandingan jumlah citra digital wajah untuk setiap karakter pada *image similarity dataset*

Pada gambar. 10, label yang telah dibuat oleh penulis dalam *image similarity dataset* didasarkan pada tipe *image query dataset*. Perhatikan bahwa jumlah label *same image* lebih banyak daripada *not same image*. Hal ini disengaja oleh penulis untuk memastikan bahwa aplikasi penulis dapat lebih banyak mendeteksi citra digital yang dimasukkan oleh pengguna yang tidak termasuk dalam label yang disediakan. Tujuannya adalah agar aplikasi lebih efektif dalam menangkap citra digital yang tidak disediakan oleh penulis.



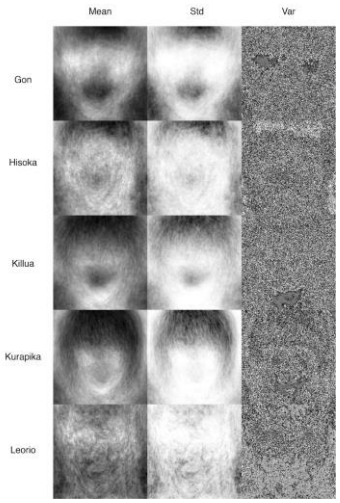

Gambar. 11 Perbandingan hasil visualisasi *mean*, *standard deviation*, dan *variance* untuk citra digital pada *image query dataset*

Pada gambar. 11, penulis menganalisis perbandingan visualisasi *mean*, *standard deviation*, dan *variance* pada karakter-karakter dalam tipe *image query dataset*. Hasil *variance* terlihat abstrak dan tidak jelas, sedangkan hasil *mean* menunjukkan karakteristik inti dari setiap karakter. Sebagai contoh, pada karakter Killua, rata-rata warna rambutnya adalah putih. Namun, karakter Leorio masih sulit dianalisis karena jumlah data untuk karakter tersebut relatif sedikit. Hasil *standard deviation* memberikan efek seperti filter negatif yang membuat warna terlihat lebih gelap, tetapi visualisasi dengan menggunakan *mean* dan *standard deviation* membuat bentuk wajah karakter lebih mudah dikenali.



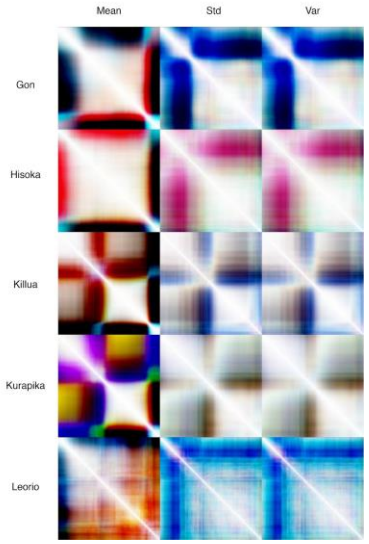

Gambar. 12 Perbandingan hasil visualisasi *image eigen* pada *1 components, 100 components mean, 100 components standard deviation, 100 components index 0, 100 components last index* pada *image query dataset*

Pada gambar. 12, penulis menginginkan representasi fitur yang sederhana melalui penggunaan *image eigen*. Terlihat bahwa penggunaan 1 dan 100 komponen menghasilkan visualisasi nilai yang sama. Namun, terdapat keunikan pada karakter Hisoka dan Kurapika, di mana latar belakang dan objek terlihat serupa, dan wajahnya tidak terlihat dengan jelas. Setiap parameter menghasilkan visualisasi yang beragam dan unik.




Gambar. 13 Perbandingan hasil visualisasi *mean, standard deviation,* dan *variance* menggunakan *salient detection*

Pada gambar. 13, penulis menggunakan metode *salient detection* untuk mengidentifikasi objek yang paling menonjol. Hasilnya menunjukkan bahwa nilai *variance* tidak menghasilkan visualisasi yang jelas, seperti yang sudah ditemukan sebelumnya. Namun, yang menarik adalah hasil *mean* dan *standard deviation* hampir mirip dengan gambar sebelumnya tanpa menggunakan metode *salient detection*. Perbedaannya lebih terlihat pada *salient detection* yang tidak memperhatikan nilai piksel, tetapi fokus pada penemuan objek yang paling penting.




Gambar. 14 Perbandingan hasil visualisasi *mean, standard deviation,* dan *variance* untuk *pixel correlation* pada *image query dataset*

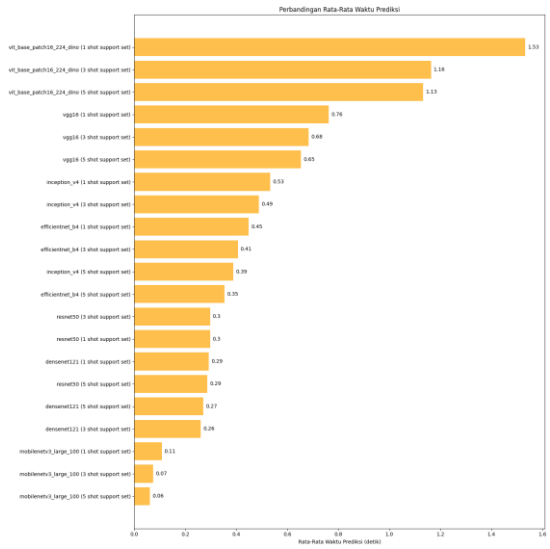
Pada gambar. 14, terlihat hasil visualisasi *pixel correlation mean*, *standard deviation*, dan *variance* pada tipe *image query dataset*. Hasil dari *standard deviation* dan *variance* untuk semua karakter memiliki kesamaan yang tinggi, tetapi rata-rata memiliki pola yang berbeda. Karakter Killua dan Kurapika memiliki bentuk yang hampir sama dengan sedikit perbedaan. Semakin berwarna hasil visualisasi, semakin jelas korelasi antar pikselnya. Karakter Hisoka memiliki korelasi tertinggi antara satu piksel dengan piksel lainnya, hampir memenuhi seluruh bagian gambar.

### *D. Pengujian Model*

Pada tahap pengujian model ini, penulis menguji model yang sebelumnya sudah penulis siapkan dengan beberapa *dataset* yang sudah penulis siapkan juga sebelumnya. Terdapat lima *dataset* yang sudah penulis siapkan dan tujuh model yang penulis siapkan, jadi terdapat 35 model yang penulis uji pada penelitian kali ini.

#### *1) Pengujian Waktu Prediksi Model*

Penulis telah melakukan uji coba terhadap beberapa konfigurasi model yang sebelumnya telah dibuat. Pengujian ini untuk mengetahui waktu prediksi ketika model memproses satu citra digital. Berikut adalah hasil pengujian yang telah dilakukan.

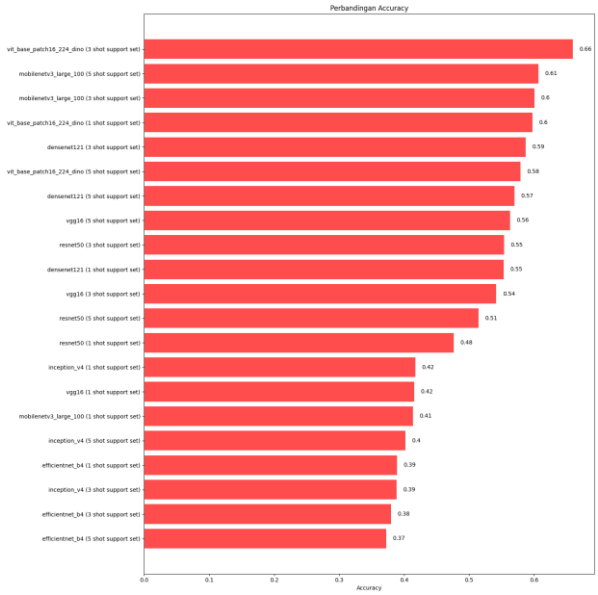



Gambar. 15 Perbandingan hasil waktu prediksi untuk setiap model dan setiap *dataset shot* pada prototypical networks

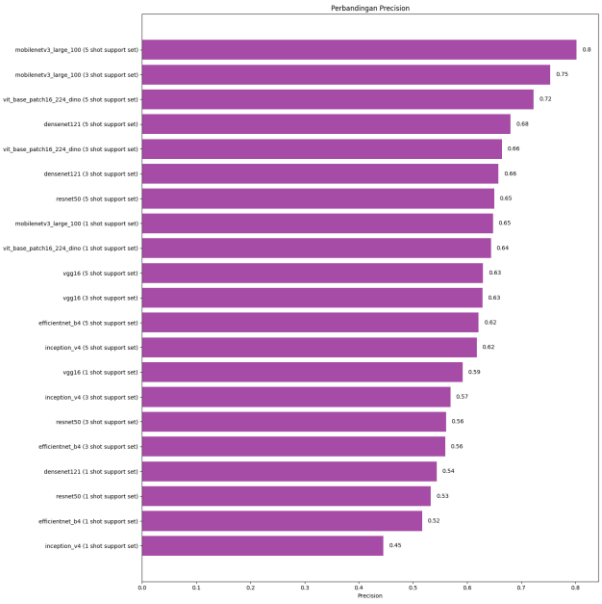
Pada gambar. 15, terlihat bahwa waktu prediksi terlama adalah pada model *vit_base_patch16_224_dino* dengan *1 shot support set*, yaitu 1.53 detik. Sedangkan waktu prediksi tercepat adalah pada model *mobilenetv3_large_100* dengan *5 shot support set*, yaitu 0.06 detik. Perbedaan ini bisa dimengerti karena *vit_base_patch16_224_dino* menggunakan model transformer yang lebih kompleks dibandingkan dengan *mobilenetv3_large_100* yang menggunakan model CNN. Selain itu, *mobilenetv3_large_100* juga dirancang untuk perangkat mini komputer, sehingga wajar jika model ini memiliki waktu prediksi yang lebih cepat dibandingkan dengan model lainnya.

*2) Pengujian Klasifikasi Model*

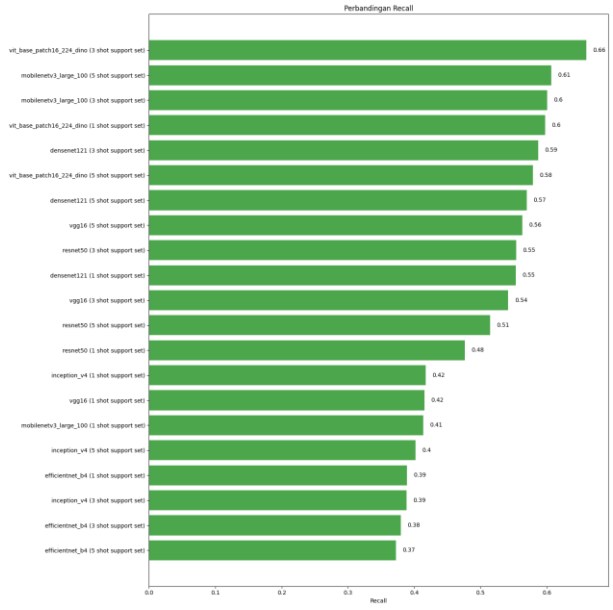
Penulis telah melakukan uji coba terhadap beberapa konfigurasi model yang sebelumnya telah dibuat. Pengujian ini melibatkan beberapa matriks pengukuran seperti akurasi (*accuracy*), presisi (*precision*), *recall*, dan skor f1 (*f1-score*) pada kasus klasifikasi model. Berikut adalah hasil pengujian yang telah dilakukan.



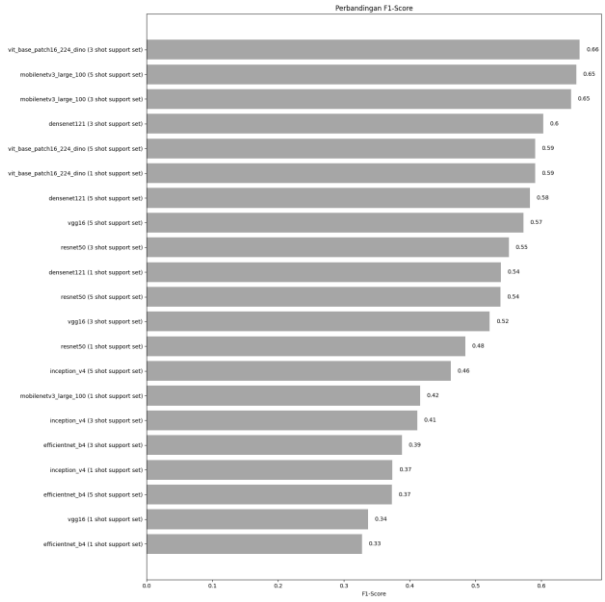

Gambar. 16 Perbandingan hasil *accuracy* pengujian klasifikasi untuk setiap model dan setiap *dataset shot* pada prototypical networks




Gambar. 17 Perbandingan hasil *precision* pengujian klasifikasi untuk setiap model dan setiap *dataset shot* pada prototypical networks




Gambar. 18 Perbandingan hasil *recall* pengujian klasifikasi untuk setiap model dan setiap *dataset shot* pada prototypical networks




Gambar. 19 Perbandingan hasil *f1-score* pengujian klasifikasi untuk setiap model dan setiap *dataset shot* pada prototypical networks

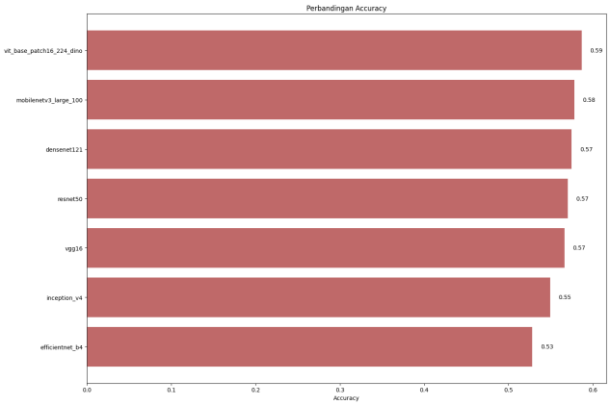
- *Highest Accuracy: vit_base_patch16_224_dino (3 shot support set) = 0.66*
- *Lowest Accuracy: efficientnet_b4 (5 shot support set) = 0.37*
- *Highest Precision: mobilenetv3_large_100 (5 shot support set) = 0.8*
- *Lowest Precision: inception_v4 (3 shot support set) = 0.45*
- *Highest Recall: vit_base_patch16_224_dino (3 shot support set) = 0.66*
- *Lowest Recall: efficientnet_b4 (5 shot support set) = 0.37*
- *Highest F1-Score: vit_base_patch16_224_dino (3 shot support set) = 0.66*
- *Lowest F1-Score: efficientnet_b4 (1 shot support set) = 0.33*
- *Best Overall: vit_base_patch16_224_dino (3 shot support set)*
- *Worst Overall: efficientnet_b4 (5 shot support set)*

Hasil penelitian menunjukkan bahwa terdapat dua model yang diuji, yaitu model terbaik (*vit_base_patch16_224_dino*) dan model terburuk (*efficientnet_b4*). Model terbaik menggunakan konfigurasi *3 shot support set*, sedangkan model terburuk menggunakan konfigurasi *5 shot support set*. Model *vit_base_patch16_224_dino*, yang berbasis transformers, menghasilkan performa yang sangat baik dalam mendapatkan fitur dari citra digital dibandingkan dengan menggunakan model CNN.

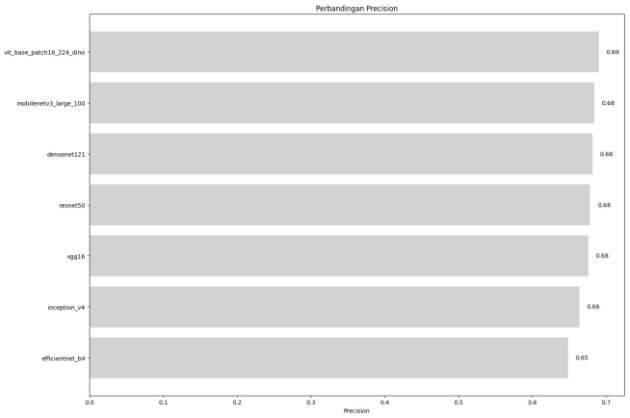
Namun, yang menarik adalah model ini memberikan hasil yang lebih baik dengan 3 shot support set daripada dengan 5 shot support set, yang menunjukkan bahwa terdapat data pada 5 shot support set yang membuat performa model menurun atau terjadi *poisoning data*. Model *efficientnet_b4*, yang didesain untuk waktu prediksi cepat bukan kualitas prediksi yang bagus, menghasilkan performa terburuk. Model ini sulit mendapatkan fitur karena menggunakan basis CNN yang fokus pada *downscale* setiap fitur, yang dapat mempengaruhi kedalaman model. Terlihat bahwa model *efficientnet_b4* cenderung salah memprediksi karakter Killua sebagai Hisoka, mungkin karena kedalaman model yang cukup dalam menyebabkan vanishing gradient dan kesalahan prediksi.

*3) Pengujian Similarity Model*

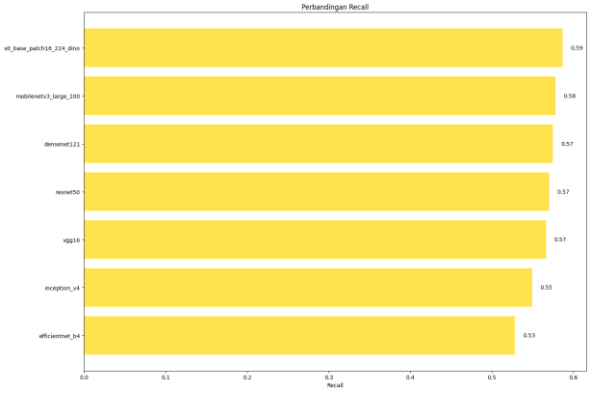
Penulis telah melakukan uji coba terhadap beberapa konfigurasi model yang sebelumnya telah dibuat. Pengujian ini melibatkan beberapa matriks pengukuran seperti akurasi (*accuracy*), presisi (*precision*), *recall*, dan skor f1 (*f1-score*) pada kasus *similarity* model. Berikut adalah hasil pengujian yang telah dilakukan.



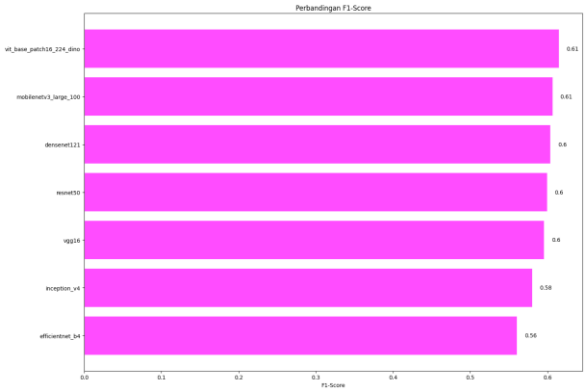

Gambar. 20 Perbandingan hasil *accuracy* pengujian *similarity* untuk setiap model pada prototypical networks




Gambar. 21 Perbandingan hasil *precision* pengujian *similarity* untuk setiap model pada prototypical networks




Gambar. 22 Perbandingan hasil *recall* pengujian *similarity* untuk setiap model pada prototypical networks



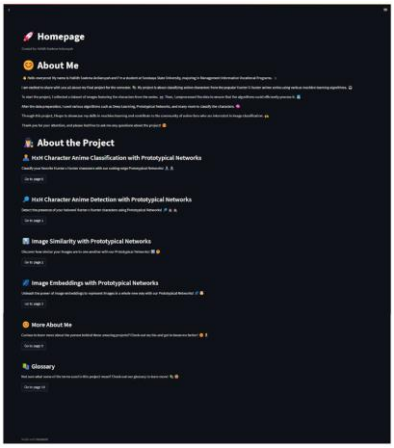

Gambar. 23 Perbandingan hasil *f1-score* pengujian *similarity* untuk setiap model pada prototypical networks

- *Highest Accuracy: vit_base_patch16_224_dino = 0.59*
- *Lowest Accuracy: efficientnet_b4 = 0.53*
- *Highest Precision: vit_base_patch16_224_dino = 0.69*
- *Lowest Precision: efficientnet_b4 = 0.65*
- *Highest Recall: vit_base_patch16_224_dino = 0.59*
- *Lowest Recall: efficientnet_b4 = 0.53*
- *Highest F1-Score: vit_base_patch16_224_dino = 0.61*
- *Lowest F1-Score: efficientnet_b4 = 0.56*
- *Best Overall: vit_base_patch16_224_dino*
- *Worst Overall: efficientnet_b4*

Setelah itu penulis melakukan pengujian supaya model bisa mengetahui, jika pengguna memasukkan citra digital diluar dari label yang penulis sediakan. Hasil model yang didapatkan sama dengan *testing classification* sebelumnya. Alasan model tersebut mendapatkan hasil terbaik dan terburuk juga sama dengan alasan pada *testing classification*.

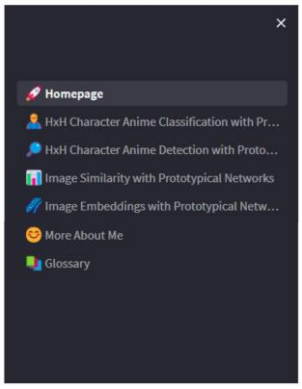
### *E. Pembuatan Aplikasi*

Pada tahap pembuatan aplikasi ini, penulis menggunakan pustaka yang ada pada bahasa pemrograman Python yaitu Streamlit yang memungkinkan penulis untuk membuat aplikasi berbasis *website* yang basis kodenya semua berasal dari bahasa pemrograman Python, karena semua pengerjaan penelitian ini dibuat dengan menggunakan bahasa pemrograman Python.



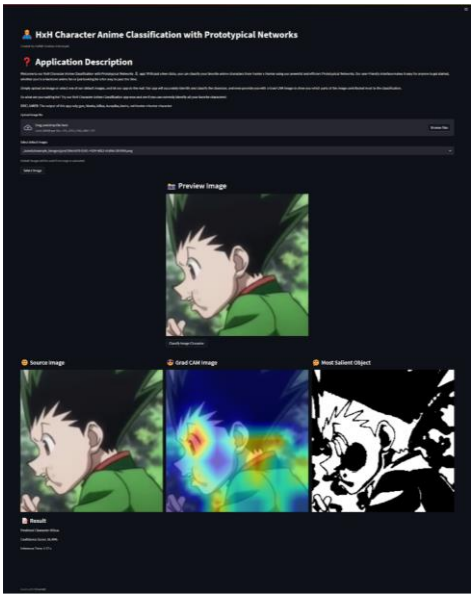

Gambar. 24 Tampilan awal, ketika pengguna membuka website yang penulis buat

Hal pertama yang penulis lakukan adalah dengan membuat tampilan awal terlebih dahulu. Bisa dilihat pada gambar. 24, tampilan awal berisi informasi singkat mengenai apa saja yang ada di aplikasi yang penulis buat, serta beberapa informasi lainnya.



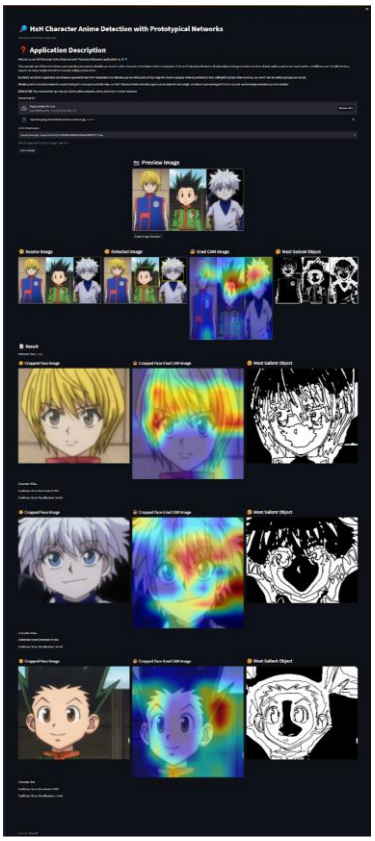

Gambar. 25 Tampilan *sidebar*, yang berisi tombol untuk berpindah ke halaman lain

Pada gambar. 25, terdapat sebuah *sidebar* yang berada di sisi kiri dan bisa dapat memencet tombol di atas kanan pojok pada gambar sebelumnya. Sidebar ini berfungsi untuk berpindah ke setiap halaman lain lebih cepat, tanpa harus menuju ke halaman *homepage* terlebih dahulu.




Gambar. 26 Tampilan untuk halaman “HxH Character Anime Classification with Prototypical Networks”

Pada gambar. 26 berisi tampilan untuk halaman “HxH Character Anime Classification with Prototypical Networks”. Tampilan tersebut digunakan untuk mendemonstrasikan model dengan pendekatan prototypical networks untuk karakter *anime* Hunter X Hunter. Terdapat beberapa tampilan seperti citra asli yang pengguna masukkan, *grad-cam*, dan *salient object*. Setelah itu pada kolom *result* terdapat hasil karakter yang diprediksi, seberapa besar percaya diri model memprediksi karakter tersebut serta waktu yang dibutuhkan model untuk menjalankan satu prediksi tersebut.

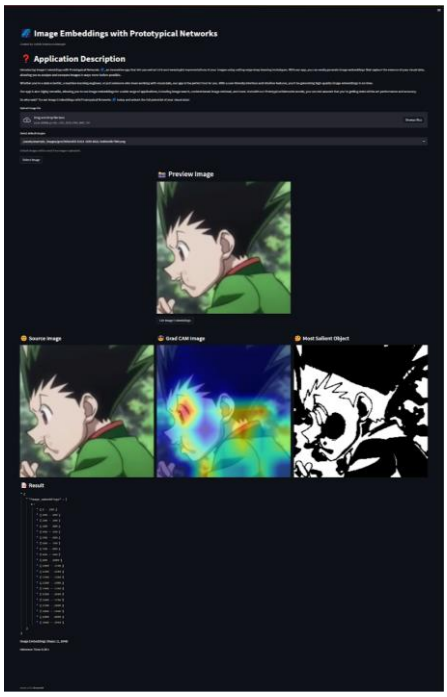



Gambar. 27 Tampilan untuk halaman “HxH Character Anime Detection with Prototypical Networks”

Pada gambar. 27 adalah halaman yang berisi deteksi karakter *anime* yang memungkinkan untuk mendeteksi wajah terlebih dahulu, setelah itu menggunakan prototypical networks untuk mengklasifikasikannya. Terdapat beberapa tampilan seperti citra asli yang pengguna masukkan, *grad-cam*, dan *salient object*. Serta pada kolom result terdapat hasil potongan wajah dari setiap karakter yang dideteksi dan untuk setiap karakter yang dideteksi, penulis menggunakan prototypical networks untuk mengklasifikasikan karakter tersebut. Untuk setiap karakter yang dideteksi, terdapat hasil karakter yang prediksi, seberapa besar percaya diri model mendeteksi karakter *anime*, seberapa besar percaya diri model memprediksi karakter tersebut, serta waktu yang dibutuhkan model untuk menjalankannya.

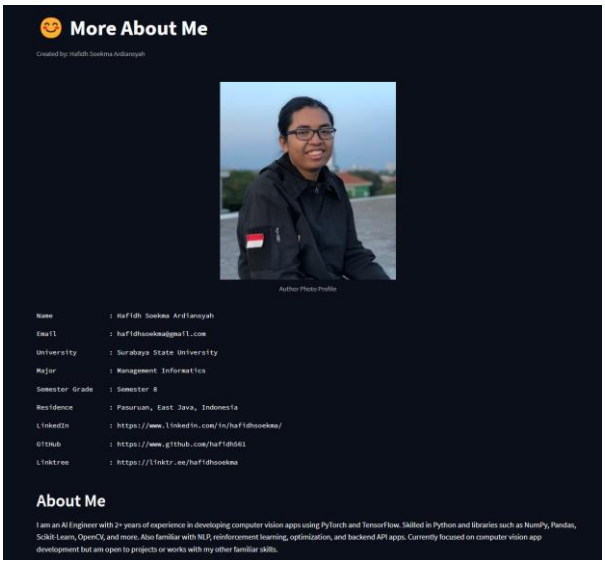

Gambar. 28 Tampilan untuk halaman “Image Similarity with Prototypical Networks”

Pada gambar. 28 adalah gambar yang berisi model yang digunakan untuk mengukur kesamaan kedua citra dengan menggunakan metode pendekatan prototypical networks. Terdapat perbedaan dengan beberapa halaman lain, seperti sekarang pengguna harus memasukkan dua citra, dikarenakan memang membutuhkan dua citra untuk mengetahui apakah kedua citra tersebut sama atau tidak. Untuk hasil yang ditampilkan berisi *grad-cam* dan *salient object*. Serta pada *section result* berisi sebuah nilai *similarity score* yang dimana digunakan untuk mengetahui skor yang didapat dari sebuah kesamaan citra, serta hasil labelnya citra yang sama atau tidak dan waktu yang dibutuhkan untuk menjalankan *image similarity*.



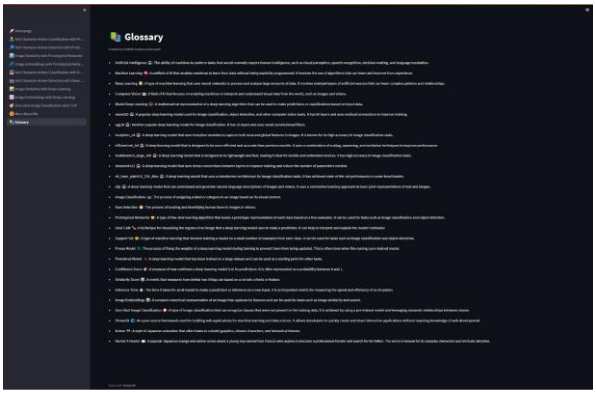

Gambar. 29 Tampilan untuk halaman “Image Embeddings with Prototypical Networks”

Pada gambar. 29 adalah halaman yang digunakan untuk mendapatkan nilai *vector embeddings* dari sebuah model yang menggunakan pendekatan prototypical networks. Untuk tampilan *layout*, kurang lebih sama dengan beberapa halaman lainnya, bedanya disini terdapat *section* yang menampilkan nilai dari *vector embeddings* dari sebuah citra yang dimasukkan.




Gambar. 30 Tampilan untuk halaman “More About Me”

Pada gambar. 30 terdapat halaman yang berisi informasi tentang beberapa istilah informasi penulis dari mulai nama, foto, alamat social media, dll.




Gambar. 31 Tampilan untuk halaman “Glossary”

Pada gambar. 31 terdapat halaman yang berisi informasi tentang beberapa istilah yang ada pada *website* yang sudah penulis buat, dikarenakan tidak semua orang paham mengenai beberapa istilah yang ada dan sangat disarankan pengguna yang masih belum paham membaca pada halaman *Glossary*.

## IV. KESIMPULAN DAN SARAN

### A. KESIMPULAN

Kecerdasan buatan adalah salah satu cabang ilmu pada ilmu komputer yang ada memungkinkan komputer untuk meniru kecerdasan alami yaitu manusia. Salah satu cabang kecerdasan buatan adalah *deep learning* yang menggunakan *artificial neural network* sebagai algoritma untuk mendapatkan hasil yang diinginkan, tetapi *deep learning* membutuhkan banyak sekali data dan penulis menggunakan prototypical network untuk membuat model tanpa melatih banyak data. Kesimpulan yang penulis dapat pada penelitian kali ini adalah:

1) Hasil pembuatan aplikasi untuk mengklasifikasikan citra digital wajah *anime* menggunakan metode prototypical networks berjalan dengan baik, tanpa ada kendala. Tampilan yang digunakan juga berbasis pustaka Streamlit yang memiliki tampilan minimalis dan mudah digunakan untuk kalangan umum.
2) Penulis mendapatkan hasil yang cukup memuaskan pada basis model *vit_base_patch16_224_dino* yang hampir semua bagus dalam semua aspek serta model. Serta mendapatkan hasil yang buruk untuk basis model *efficientnet_b4* di hampir semua aspek.

### B. SARAN

Saran penulis untuk penelitian kedepannya adalah dengan memperbanyak label yang ada, seperti menggunakan semua karakter yang ada pada *anime* Hunter X Hunter atau menggunakan pendekatan prototypical network yang lain dengan cara *fine tuning* model *deep learning* pada citra wajah *anime*, supaya mendapatkan *vector embeddings* yang lebih mudah dipahami oleh model tersebut. Setelah itu, saran penulis

lainnya adalah mengintegrasikannya dengan aplikasi lain seperti website HTML (*Hyper Text Markup Language*), CSS (*Cascading Style Sheets*), JS (*JavaScript*) atau aplikasi piranti cerdas dengan cara menghubungkannya menggunakan metode komunikasi REST API (*Representational State Transfer Application Program Interface*), gRPC *(Google Remote Procedure Call*), SOAP (*Simple Object Access Protocol*), GraphQL (*Graph Query Language*), dll.

## REFERENSI


[1] Tsimenidis, Stefanos. 2020. *Limitations of Deep Neural Networks: a discussion of G. Marcus' critical appraisal of deep learning*. Thessaloniki: arXiv.

[2] Krizhevsky, Alex, et al. 2012. *ImageNet Classification with Deep Convolutional Neural Networks*. Toronto: Curran Associates.

[3] Snell, Jake, et al. 2017. *Prototypical Networks for Few-shot Learning*. Toronto: arXiv.

[4] Gao Huang, et al. 2017. *Densely Connected Convolutional Networks*. Ithaca: IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[5] Mingxing Tan and Quoc V. Le. 2019. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Ithaca: arXiv.

[6] Szegedy, Christian, et al. 2017. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. Pittsburgh: arXiv.

[7] Andrew, Howard, et al. 2019. *Searching for MobileNetV3*. Seoul: IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[8] Kaiming He, et al. 2015. *Deep Residual Learning for Image Recognition*. Las Vegas: IEEE Conference on Computer Vision and Pattern Recognition.

[9] Simonyan, Karen and Zisserman, Andrew. 2014. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. San Diego: International Conference on Learning Representations.

[10] Dosovitskiy, Alexey, et al. 2020. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. Vienna: International Conference on Learning Representations.

[11] Xiaodi Hou and Liqing Zhang. 2007. *Saliency Detection: A Spectral Residual Approach*. Shanghai: IEEE Conference on Computer Vision and Pattern Recognition.

[12] Daoqiang Zhang, et al. 2005. *Representing Image Matrices: Eigenimages Versus Eigenvectors*. Berlin: International Symposium on Neural Networks.

[13] Selvaraju, Ramprasaath R, et al. 2016. *Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization*. Venice: IEEE International Conference on Computer Vision.