Vol. 5, No. 1, April 2024, hal. 159-168

p-ISSN: 2723-567X
e-ISSN: 2723-5661

Jurnal Computer Science and Information Technology (CoSciTech)

http://ejurnal.umri.ac.id/index.php/coscitech/index

Vision Transformer untuk Identifikasi 15 Variasi Citra Ikan Koi

Rayhan Uthama[*1], Yuhandri[2], Billy Hendrik[3]
Email: [1]uthamarayhan1@gmail.com, [2]yuhandri.yunus@gmail.com, [3]billy_hendrik@upiyptk.ac.id

[123]Magister Teknik Informatika, Fakultas Ilmu Komputer, Universitas Putra Indonesia

Diterima: 28 April 2024 | Direvisi: - | Disetujui: 13 Mei 2024

Abstrak

Penelitian ini bertujuan untuk mengklasifikasikan berbagai jenis ikan koi dengan menggunakan *Vision Transformer* (ViT). Terdapat penelitian terdahulu [1] menggunakan *Support Vector Machine* (SVM) sebagai *classifier* untuk mengidentifikasi 15 jenis ikan koi dengan dataset pelatihan dan pengujian masing-masing berjumlah 1200 dan 300 gambar. Penelitian tersebut dilanjutkan oleh penelitian [2] yang mengimplementasikan *Convolutional Neural Network* (CNN) sebagai *classifier* untuk mengidentifikasi 15 jenis ikan koi dengan dataset yang berjumlah sama. Sebagai hasilnya penelitian tersebut mencapai tingkat akurasi 84% klasifikasi. Meskipun akurasi yang diperoleh dari penggunaan CNN cukup tinggi, masih ada ruang untuk peningkatan dalam akurasi klasifikasi. Mengatasi kendala seperti batasan dalam akurasi klasifikasi pada penelitian sebelumnya dan eksplorasi lebih lanjut terhadap penggunaan algoritma dan teknik baru, penelitian ini mengusulkan arsitektur ViT untuk meningkatkan akurasi dalam klasifikasi ikan Koi. ViT merupakan algoritma *deep learning* yang diadopsi dari algoritma *Transformer* yang bekerja dengan mengandalkan tugas mekanisme perhatian diri (s*elf attention*). Karena kekuatan representasi data yang lebih baik dibanding algoritma *deep learning* lainnya termasuk CNN, peneliti telah menerapkan tugas *Transformer* ini pada bidang *computer vision*, yang salah satu hasil dari penerapan tersebut adalah ViT. Penelitian ini dirancang menggunakan kelas dan jumlah dataset dipertahankan dari dua penelitian sebelumnya. Sedangkan dataset gambar ikan koi yang digunakan pada penelitian ini dikumpulkan dari internet dan telah divalidasi. Implementasi ViT sebagai *classifier* dalam klasifikasi koi pada penelitian ini menghasilkan tingkat akurasi yang mencapai rata-rata 89% pada seluruh kelas data pengujian.
Kata kunci: *deep learning, ViT, pengolahan citra, klasifikasi citra, ikan koi.*

Vision Transformer for 15 Variations of Koi Fish Identification

Abstract

This research aims to classify various types of koi fish using Vision Transformer (ViT). There is previous research [1] using Support Vector Machine (SVM) as a classifier to identify 15 types of koi fish with training and testing datasets respectively of 1200 and 300 images. This research was continued by research [2] which implemented a Convolutional Neural Network (CNN) as a classifier to identify 15 types of koi fish with the same amount dataset. As a result, the research achieved a classification accuracy rate of 84%. Although the accuracy obtained from using CNN is quite high, there is still room for improvement in classification accuracy. Overcoming obstacles such as limitations in classification accuracy in previous studies and further exploration of the use of new algorithms and techniques, this study proposes a ViT architecture to improve accuracy in Koi fish classification. ViT is a deep learning algorithm adopted from the Transformer algorithm which works by relying on self-attention mechanism tasks. Because the power of data representation is better than other deep learning algorithms including CNN, researchers have applied this Transformer task in the field of computer vision, one of the results of this application is ViT. This study was designed using class and number datasets retained from two previous studies. Meanwhile, the koi fish image dataset used in this research was collected from the internet and has been validated. The implementation of ViT as a classifier in koi classification in this research resulted in an accuracy level that reached an average of 89% in all classes of test data.
Keywords: *deep learning, ViT, image processing, image classification, koi fish*

1. PENDAHULUAN

Salah satu jenis ikan yang sangat digemari pengoleksi ikan hias adalah ikan koi. Ikan koi merupakan salah satu jenis ikan hias yang populer di seluruh dunia, terutama di Jepang. Sejauh ini penelitian tentang klasifikasi citra ikan koi telah dilakukan menggunakan berbagai teknik pengenalan pola citra, termasuk metode klasifikasi seperti *Support Vektor Machine* (SVM) [1]. Penelitian Sapphira mengidentifikasi varietas ikan koi berdasarkan gambar, tetapi Sapphira menyatakan bahwa dataset yang diambil dari internet kualitasnya tidak konsisten. Kondisi gambar yang didapat memiliki perbedaan pantulan cahaya, bayangan, pantulan air, atau riak air yang berbeda.

Penelitian [2] mengimplementasikan *Convolutional Neural Network* (CNN) untuk mengklasifikasikan 15 variasi koi dengan akurasi 84%, dengan alasan bahwa CNN dapat mendeteksi pola kompleks dan memeriksa seluruh fitur untuk prediksi yang lebih akurat. masih ada ruang untuk peningkatan dalam akurasi klasifikasi. Mengatasi kendala seperti batasan dalam akurasi klasifikasi pada penelitian sebelumnya dan untuk eksplorasi lebih lanjut terhadap penggunaan algoritma dan teknik baru, penelitian ini mengusulkan arsitektur ViT untuk meningkatkan akurasi dalam klasifikasi ikan koi.

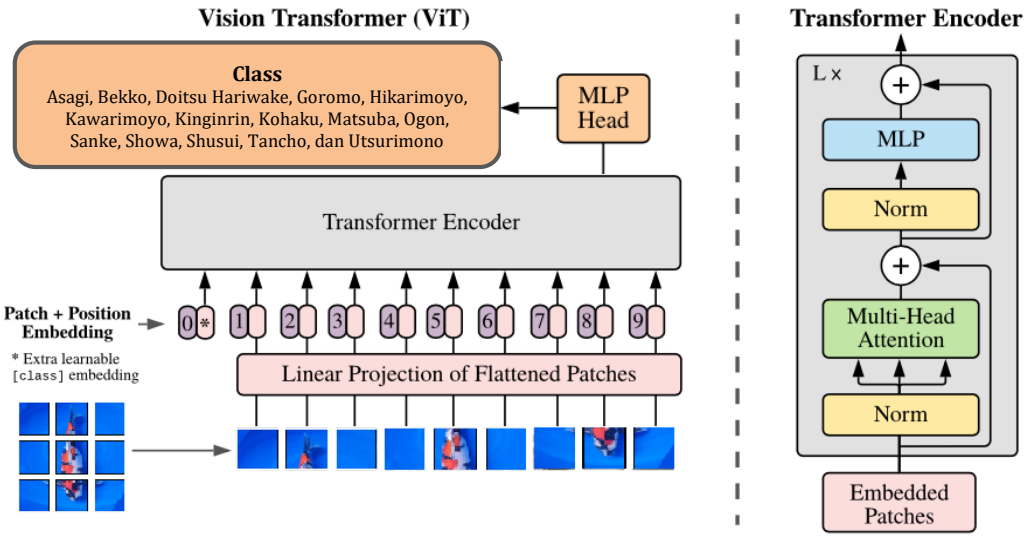
Terdapat penelitian lainnya yang menggunakan arsitektur pembeajaran mesin *deep learning* lainnya seperti halnya CNN. Sebagai contoh, penelitian [3] mengungkapkan bahwa salah satu kelebihan yang ada pada model *Transformer* adalah dalam memproses urutan data dengan cara yang lebih efisien dibanding arsitektur deep learning lainnya, termasuk CNN. Awalnya *Transformer* diawali penggunaannya dalam pemrosesan bahasa alami (NLP) dengan mengandalkan tugas mekanisme perhatian diri (self attention). Karena kekuatan representasi data yang lebih baik dibanding algoritma *deep learning* lainnya termasuk CNN, peneliti telah menerapkan tugas *Transformer* ini pada bidang *computer vision*, yang salah satu hasil dari penerapan tersebut adalah ViT[4].

Arsitektur Vision *Transformer* (ViT) merupakan adaptasi dari arsitektur *Transformer* yang saat ini menjadi *state-of-the-art* (SOTA) dalam pemrosesan bahasa alami. ViT terdiri dari beberapa tahap, termasuk *patch embeddings*, *multi-head attention*, dan *multi-layer perceptron* [5]. Beberapa penelitian telah dipublikasikan dengan mengaplikasikan ViT sebagai arsitektur pada dataset publik. Contoh penelitian tersebut diantaranya penelitian [6] pada dataset Imagenet1K, kemudian penelitian [7] pada dataset kecil seperti CIFAR10/100, CINIC10, SVHN, Tiny-ImageNet, Aircraft dan Cars, dan penelitian [8] pada dataset CheXpert dan Pediatric Pneumonia. Sedangkan aplikasi ViT sebagai arsitektur yang digunakan dalam mengklasifikasi citra dari dataset pribadi sudah digunakan di beberapa penelitian. Sebagai contoh, penelitian [9] yang mengklasifikasikan interpretasi emfisema dengan ViT, penelitian [10] dalam klasifikasi gender berdasarkan citra wajah, dan penelitian [11] dalam segmentasi gambar pada data yang berkaitan dengan kebakaran hutan.

Arsitektur *Vision Transformer* (ViT) digunakan untuk dinilai tingkat akurasi hasil klasifikasinya pada citra ikan koi dalam penelitian ini. Tujuan utama mengimplementasikan ViT dalam penelitian ini adalah bagaimana merancang model klasifikasi ViT, menemukan hasil klasifikasinya, dan memperoleh performa ViT dalam mengidentifikasi variasi ikan koi. Kemudian melakukan penelitian agar memperoleh hasil penelitian yang dapat memberikan kemudahan identifikasi bagi penghobi koi, memberikan kontribusi pada pengembangan teknologi klasifikasi citra koi, dan menyediakan informasi bagi peneliti lain dalam pemilihan arsitektur klasifikasi gambar. Penelitian ini juga diharapkan memberikan kontribusi signifikan pada peningkatan akurasi dan efektivitas klasifikasi variasi-variasi ikan koi.

Penelitian ini bertujuan untuk mengadopsi arsitektur *Vision Transformer* (ViT) dalam mengidentifikasi variasi ikan koi berdasarkan citra. Dengan merancang kelas dan jumlah dataset masing-masing variasi koi yang dipertahankan seperti halnya pada penelitian sebelumnya, sebanyak 1500 citra ikan koi akan digunakan dan dibagi ke dalam 15 kelas, sesuai dengan variasi ikan koi. Kelima belas (15) jenis ikan koi tersebut adalah, Asagi, Bekko, Doitsu Hariwake, Goromo, Hikarimoyo, Kawarimoyo, Kinginrin, Kohaku, Matsuba, Ogon, Sanke, Showa, Shusui, Tancho, dan Utsurimono. Yang mana setiap kelas dataset terdiri dari 80 citra latih dan 20 citra uji (*testing*).

Penulis memilih model ViT-B/16 untuk dataset penelitian ini yang berukuran 100 citra x 15 kelas. Gambar 1 menunjukkan arsitektur ViT secara umum.

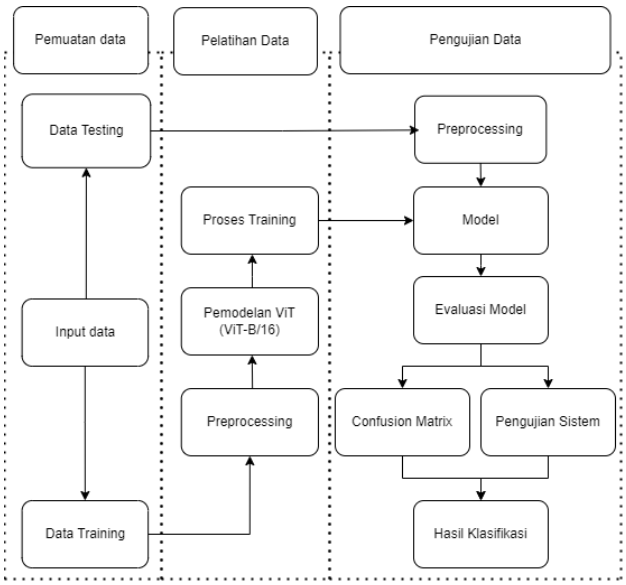


Gambar 1. Arsitektur *Vision Transformer*

Gambar 1 menunjukkan proses pada ViT dimulai dari kanan bawah, di mana citra *input* diubah menjadi beberapa *patch* dengan *patch* embeddings. Lalu urutan *patch* memasuki blok *encoder* yang terdiri dari lapisan *multi-head Attention* dan blok MLP yang bergantian, serta lapisan normalisasi diterapkan setiap sebelum blok.

2. METODE PENELITIAN

Metodologi penelitian berperan penting untuk membantu penelitian dan penulisan agar sesuai dengan fokus masalah yang tengah diselidiki. Metodologi berisi langkah-langkah yang dilakukan dalam penelitian yang disebut dengan Algoritma penelitian. Berikut adalah algoritma penelitian seperti yang ditunjukkan pada Gambar 2.



Gambar 2. Algoritma Penelitian

2.1 Pemuatan Data

Penelitian klasifikasi variasi ikan koi ini menggunakan gambar dari situs indonesiakoishow.com, situs agregator kejuaraan ikan koi di Indonesia. Data masing-masing kelas tersebut berjumlah 100 gambar dan dialokasikan menjadi dataset train dan test seperti pada Tabel 1 sehingga total data yang akan diproses adalah 1500 gambar ikan koi.

Tabel 1 Pembagian Data *Input* Citra

No	Kelas	Train	Test
1	Asagi	80	20
2	Bekko	80	20
3	Doitsu Hariwake	80	20
4	Goromo	80	20
5	Hikarimoyo	80	20
6	Kawarimono	80	20
7	Kinginrin	80	20
8	Kohaku	80	20
9	Matsuba	80	20
10	Ogon	80	20
11	Sanke	80	20
12	Showa	80	20
13	Shusui	80	20
14	Tancho	80	20
15	Utsurimono	80	20
	Total Data:	1200	300

Semua gambar ikan koi diunduh dalam format (.jpg) Data berisikan dataset gambar yang digunakan untuk proses *training* agar sistem dapat memproses data tanpa kendala yang disebabkan perbedaan format. Langkah validasi data, gambar diserahkan kepada M Rizky Ramadan Keman, seorang ahli ikan koi yang mengelola pusat budidaya dan ternak "Berkah Koi" di Kota Pekanbaru, Provinsi Riau.

2.2 Pelatihan Data

Proses pelatihan dimulai dengan *preprocessing* data citra ikan koi, dilanjutkan dengan *transfer learning* pada model *Vision Transformer* (ViT). Selama pelatihan, ViT disesuaikan dengan data pelatihan untuk mengenali variasi dan karakteristik citra. Pengukuran *loss* dan akurasi dilakukan untuk mengevaluasi dan memperbaiki kinerja model hingga mencapai tingkat akurasi yang diinginkan.

I. *Preprocessing*

Bagian *preprocessing* adalah bagian yang cukup penting dalam pengolahan data citra sebelum memasuki model. Model ViT-B/16 mempersyaratkan *input* gambar berukuran 224 dan ukuran *patch* masukan 16x16. Sehingga terdapat rangkaian teknik *pre-processing* berjenis *data transform* yang digunakan pada citra *input*, yaitu mulai dari *resize*, *patchification*, *Conv2D Projection*, *Flattened feature maps*, hingga diperoleh *array* dari *flattened feature maps* [12].

II. Pemodelan

Tahap pemodelan ViT adalah tahap membangun model penelitian. Arsitektur ViT pada model ViT-B/16 yang diimplementasikan terdiri dari beberapa blok dan *layer* utama, diantaranya adalah lapisan *Conv2d*, Blok *Encoder Transformer*, Normalisasi Lapisan, dan Lapisan *Linear*. Gambar 3 adalah ringkasan ukuran dan parameter yang digunakan pada ViT-B/16.

```
==========================================================================================================================================
Layer (type (var_name))                                 Input Shape          Output Shape         Param #              Trainable
==========================================================================================================================================
VisionTransformer (VisionTransformer)                   [1, 3, 224, 224]     [1, 15]              768                  Partial
├─Conv2d (conv_proj)                                    [1, 3, 224, 224]     [1, 768, 14, 14]     (590,592)            False
├─Encoder (encoder)                                     [1, 197, 768]        [1, 197, 768]        151,296              False
│    └─Dropout (dropout)                                [1, 197, 768]        [1, 197, 768]        --                   --
│    └─Sequential (layers)                              [1, 197, 768]        [1, 197, 768]        --                   False
│    │    └─EncoderBlock (encoder_layer_0)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_1)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_2)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_3)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_4)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_5)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_6)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_7)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_8)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_9)              [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_10)             [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    │    └─EncoderBlock (encoder_layer_11)             [1, 197, 768]        [1, 197, 768]        (7,087,872)          False
│    └─LayerNorm (ln)                                   [1, 197, 768]        [1, 197, 768]        (1,536)              False
├─Sequential (heads)                                    [1, 768]             [1, 15]              --                   True
│    └─Linear (0)                                       [1, 768]             [1, 15]              11,535               True
==========================================================================================================================================
Total params: 85,810,191
Trainable params: 11,535
Non-trainable params: 85,798,656
Total mult-adds (M): 172.48
==========================================================================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 104.09
Params size (MB): 229.24
Estimated Total Size (MB): 333.93
==========================================================================================================================================
```

Gambar 3 Ringkasan Parameter dan *Output* dari Model ViT-B/16

Salah satu metode *patch embeddings* khususnya *linear projection* seperti yang ditunjukkan pada awal arsitektur di Gambar 3 dapat diterapkan pada *patch* yang diekstraksi dari peta fitur CNN dengan menggunakan jaringan CNN [12]. Proses yang disebut *patch embeddings* ini mengubah citra *input* dengan membagi menjadi sejumlah *patch* berukuran (n x n), yang selanjutnya diubah menjadi vektor satu dimensi yang berisi nilai piksel dari citra tersebut. Proses pelatihan pada arsitektur *deep learning* biasanya membutuhkan sumber daya komputasi yang besar. Untuk mengurangi penggunaan daya dan waktu, kita bisa menggunakan normalization layer. Teknik *normalization layer* mengurangi nilai *input* dengan menggunakan rata-rata dan standar deviasi.

Setelah *normalization layer*, lapisan berikutnya adalah *multi-head attention layer* yang mencari informasi jangka panjang dari vektor masukan. Lapisan ini menduplikasi vektor masukan menjadi vektor *query*, *key*, dan *value* yang diproses menggunakan fungsi *attention* untuk menghasilkan matriks *output* dan diulangi sesuai jumlah *head* secara paralel. Hasilnya di-*concat* lalu dikompres menjadi ukuran vektor tertentu. Setelah proses *normalization layer*, proses *training* memasuki akhir tahapan pelatihan, yaitu *layer linear* yang di dalamnya terdapat MLP *head*. MLP *head* merupakan salah satu inovasi utama dalam ViT yang mengambil *input* dari *output Transformer* yang terkait dengan embedding kelas khusus dan mengabaikan *output* lainnya. MLP

III. Proses *Training*

Proses *training* menggunakan *transfer learning*, memanfaatkan model yang ada untuk melatih kembali dengan dataset baru. Pendekatan ini efisien waktu. *Training* menyesuaikan *parameter internal* model dengan data pelatihan, memahami variasi dan karakteristik citra ikan koi. Pengukuran *loss* dan akurasi pada data *train* dan *testing* dilakukan selama *training*. Gambar 4 menunjukkan proses *training* model ViT-B/16.

```
100% ████████████████████ 15/15 [13:05<00:00, 40.94s/it]
Epoch: 1 | train_loss: 1.9647 | train_acc: 0.4515 | test_loss: 1.4202 | test_acc: 0.6333
Epoch: 2 | train_loss: 1.0725 | train_acc: 0.7533 | test_loss: 1.0050 | test_acc: 0.7802
Epoch: 3 | train_loss: 0.7570 | train_acc: 0.8339 | test_loss: 0.8161 | test_acc: 0.7854
Epoch: 4 | train_loss: 0.6017 | train_acc: 0.8684 | test_loss: 0.7116 | test_acc: 0.8354
Epoch: 5 | train_loss: 0.4980 | train_acc: 0.8972 | test_loss: 0.6391 | test_acc: 0.8417
Epoch: 6 | train_loss: 0.4265 | train_acc: 0.9194 | test_loss: 0.5967 | test_acc: 0.8448
Epoch: 7 | train_loss: 0.3641 | train_acc: 0.9326 | test_loss: 0.5544 | test_acc: 0.8385
Epoch: 8 | train_loss: 0.3217 | train_acc: 0.9449 | test_loss: 0.5256 | test_acc: 0.8823
Epoch: 9 | train_loss: 0.2865 | train_acc: 0.9548 | test_loss: 0.4987 | test_acc: 0.8635
Epoch: 10 | train_loss: 0.2586 | train_acc: 0.9646 | test_loss: 0.4831 | test_acc: 0.8792
Epoch: 11 | train_loss: 0.2357 | train_acc: 0.9679 | test_loss: 0.4712 | test_acc: 0.8948
Epoch: 12 | train_loss: 0.2111 | train_acc: 0.9778 | test_loss: 0.4560 | test_acc: 0.8917
Epoch: 13 | train_loss: 0.1927 | train_acc: 0.9762 | test_loss: 0.4442 | test_acc: 0.8979
Epoch: 14 | train_loss: 0.1757 | train_acc: 0.9827 | test_loss: 0.4378 | test_acc: 0.9010
Epoch: 15 | train_loss: 0.1628 | train_acc: 0.9844 | test_loss: 0.4232 | test_acc: 0.8917
```

Gambar 4. Proses Iterasi Pelatihan

Proses training dijalankan sebagai program *training* pada *cell code* yang ada di dalam interface Colab. Proses *training* dihentikan setelah 15 *epoch* karena proses training setelah *epoch* 15 *epoch* diketahui mencapai tren konvergensi atau tidak terdapat peningkatan akurasi pengujian yang signifikan.

2.3 Pengujian Data

Evaluasi model diawali dari informasi hasil simulasi klasifikasi dapat dilihat pada tabel *confusion matrix*, lalu berdasarkan informasi hasil klasifikasi yang ada pada *confusion matrix*, dapat di ketahui metriks performa setelah dilakukan perhitungan pada bagian selanjutnya. Tahap evaluasi model ViT yang telah dilatih adalah sebagai berikut:

I. *Confusion Matrix*

Confusion Matrix adalah pengujian yang dapat mencatat seberapa banyak benar dan salahnya hasil prediksi suatu algoritma dalam melakukan klasifikasi [13]. *Confusion matrix* menampilkan hasil prediksi terhadap 300 data uji pada penelitian ini yang diperlukan untuk menghitung nilai keempat metrics performa model. Ada 4 metrik performa yang dapat dihitung dari *confusion matrix* [14], diantaranya:

a. Akurasi menunjukkan seberapa akurat suatu kelas diprediksi dengan benar.

$$Accuracy = (\mathrm{TP} + \mathrm{TN}) / (\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}) \quad (1)$$

b. *Precision* menggambarkan hubungan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model.

$$Precision = (TP) / (TP + FP) \quad (2)$$

c. *Recall* atau *sensitivity* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi.

$$Recall = TP / (TP + FN) \quad (3)$$

d. *F1-score* menggambarkan perbandingan rata-rata harmonik *precision* dan *recall*

$$F - 1\ Score = (2 * Recall * Precision) / (Recall + Precision) \quad (4)$$

II. Pengujian Sistem

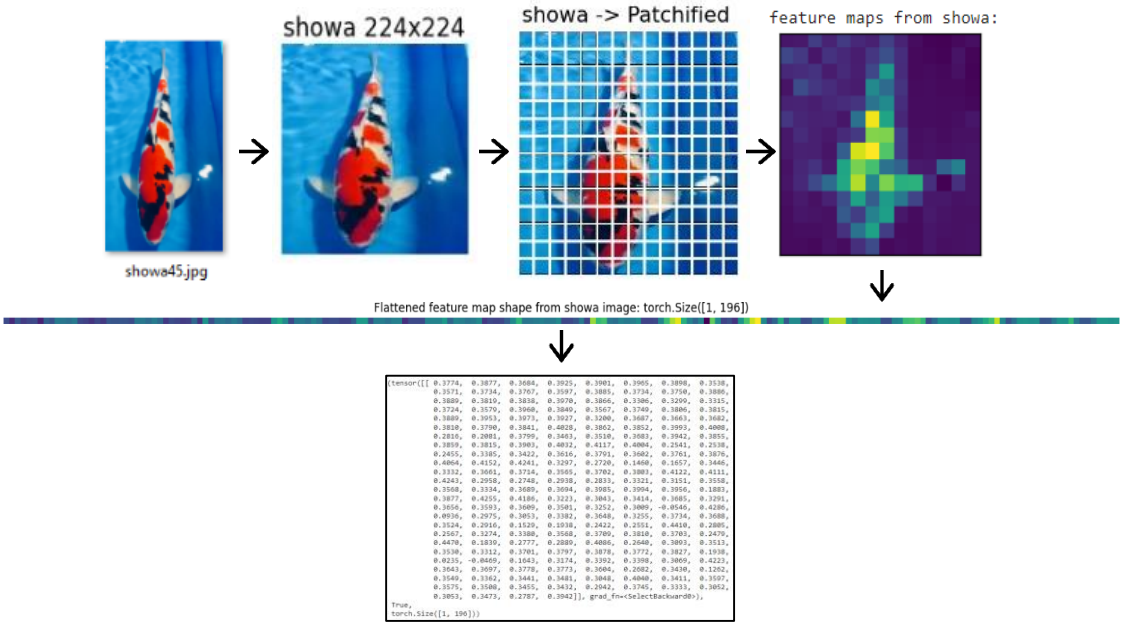
Pengujian sistem merupakan tahap pengujian yang berlaku setelah model berhasil dilatih. Metriks akurasi dari model ViT menunjukkan bahwa model dapat melakukan simulasi klasifikasi dengan baik. Pengujian sistem dilakukan dengan implementasi sistem klasifikasi gambar yang sama sekali tidak ada pada direktori *drive*.

3. HASIL DAN PEMBAHASAN

Bagian ini menjelaskan hasil implementasi, meliputi *preprocessing*, *training*, evaluasi, hingga pengujian sistem

3.1. Hasil Preprocessing

Bagian *preprocessing* menyiapkan data citra sebelum proses *training* dengan rangkaian teknik *data transform* yang diantaranya adalah *resize*, *patchification*, *Conv2D Projection*, *flattened feature maps*, dan hasil *array* dari *feature maps*. Hasil dari teknik *preprocessing* yang digunakan adalah sebagai berikut:



Gambar 5. Hasil Teknik Preprocessing

a. *Resizing*, dilakukan untuk memperoleh gambar ukuran 224x224 untuk memenuhi ekspektasi model ViT-B/16 dapat membagi *patch* berukuran 16x16.

b. *Patchification*, dilakukan dengan membagi gambar menjadi potongan-potongan kecil yang disebut "*patch*" sebelum dimasukkan ke dalam model agar model ViT menangkap hubungan antar-*patch* dalam representasi urutan data.

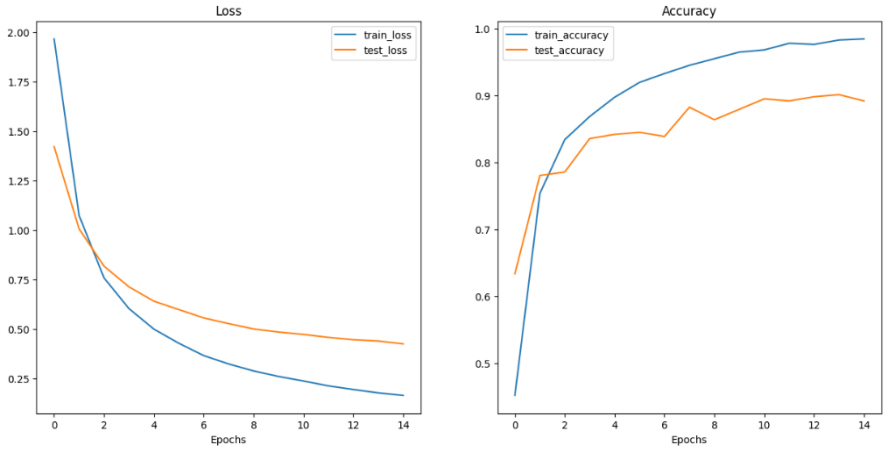
c. *Conv2D Projection*, adalah tahap *linear projection* yang dilakukan dengan penggunaan lapisan konvolusi 2D sebagai langkah awal penyesuaian urutan *patch* menjadi dimensi *input* yang dibutuhkan model. Hasil dari *linear projection* disebut *feature maps*.

d. *Flattened feature maps*, dilakukan dengan meratakan *feature maps*, atau mengubah bentuk matriks menjadi satu dimensi. Proses *flattened feature maps* merupakan tahap lanjut dari *Conv2D Projection*.

e. *Array* dari *feature maps*, terdiri dari nilai-nilai dari semua piksel di semua *feature maps*. Ini adalah *array* satu dimensi yang dapat digunakan sebagai *input* untuk blok *Transformer* dalam arsitektur *Vision Transformer*.

3.2. Hasil *Training*

Perkembangan metrik akurasi dan *loss* dapat dilihat pada gambar berikut:



Gambar 6. Hasil *Training*

Program *training* diatur untuk diselesaikan pada epoch 15 karena konvergensi dihasilkan pada jumlah *epoch* 15. Program *training* juga menunjukkan hasil perkembangan metrik akurasi dan *loss* yang semakin baik, sehingga pada akhir *epoch* proses pelatihan dapat diselesaikan dengan metrik akurasi dan *loss* yang baik pula.

3.3. Evaluasi Model

Salah satu metode evaluasi yang umum digunakan adalah *confusion matrix*, yang memberikan gambaran menyeluruh tentang kinerja model. *Confusion matrix* merinci jumlah prediksi yang benar dan salah untuk setiap kelas dalam suatu dataset. Berikut adalah tabel *confusion matrix* yang menampilkan hasil prediksi kelas terhadap 300 data uji pada penelitian ini, dapat dilihat pada Tabel 2.

Tabel 2. Hasil *Confusion Matrix*

No	Kelas		Prediksi															
			Asagi	Bekko	Doitsu Hariwake	Goromo	Hikarimoyo	Kawarimono	Kinginrin	Kohaku	Matsuba	Ogon	Sanke	Showa	Shusui	Tancho	Utsurimono	Total
1	Aktual	Asagi	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20
2		Bekko	0	19	0	0	0	1	0	0	0	0	0	0	0	0	0	20
3		Doitsu Hariwake	0	0	15	0	5	0	0	0	0	0	0	0	0	0	0	20
4		Goromo	0	0	0	18	0	0	0	1	0	0	0	0	0	0	1	20
5		Hikarimoyo	0	0	0	0	17	1	0	1	0	1	0	0	0	0	0	20
6		Kawarimono	1	0	0	0	1	13	0	0	1	1	0	1	0	0	2	20
7		Kinginrin	0	0	0	1	0	0	16	1	0	0	1	0	0	0	1	20
8		Kohaku	0	0	2	1	0	0	0	17	0	0	0	0	0	0	0	20
9		Matsuba	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	20
10		Ogon	0	0	0	0	0	1	0	0	0	19	0	0	0	0	0	20
11		Sanke	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	20
12		Showa	0	0	0	0	0	0	0	0	0	0	2	18	0	0	0	20
13		Shusui	0	0	0	0	0	1	0	0	0	0	0	0	19	0	0	20
14		Tancho	0	1	0	0	0	0	0	0	1	0	0	0	0	18	0	20
15		Utsurimono	0	0	0	0	0	0	0	0	0	0	0	1	0	1	18	20

Seperti yang terlihat pada tabel *confusion matrix*, Sebagian besar kesalahan klasifikasi masih didominasi kelas kawarimono yang diklasifikasikan sebagai beberapa dari kelas lainnya, lalu pada kelas kinginrin yang diklasifikasikan sebagai beberapa kelas lainnya, dan doitsu hariwake yang diklasifikasikan sebagai kelas hikarimoyo.

Kesalahan klasifikasi pada kelas kawarimono mungkin disebabkan oleh kurangnya generalisasi ViT terhadap variasi pola dan warna, dikarenakan terbatasnya jumlah data di dataset (80 sampel). Pada kelas kinginrin kesalahan terjadi karena ViT seharusnya lebih menonjolkan generalisasi kilauan silver pada citra koi untuk mengenali kinginrin melainkan lebih mengandalkan pengenalan berdasarkan warna dan pola saja yang menyebabkan kelas kinginrin sering diklasifikasikan sebagai jenis koi lainnya. Sedangkan pada kelas doitsu hariwake adalah karena ketidakmampuan ViT mengenali fitur ketiadaan sisik sehingga doitsu hariwake diklasifikasikan sebagai hikarimoyo.

Terdapat peningkatan hasil klasifikasi yang signifikan pada kelas matsuba dibanding penelitian sebelumnya (akurasi 70%) yang mana tidak terdapat lagi kesalahan klasifikasi pada kelas matsuba di penelitian ini yang dapat dilihat pada Tabel 3.

Tabel 3. Hasil Klasifikasi

No	Kelas	Hasil Confusion Matrix				Akurasi
		True Positive	True Negative	False Positive	False Negative	
1	Asagi	20	279	1	0	1.00
2	Bekko	19	279	1	1	0.95
3	Doitsu Hariwake	15	278	2	5	0.75
4	Goromo	18	278	2	2	0.90
5	Hikarimoyo	17	274	6	3	0.85
6	Kawarimono	13	276	4	7	0.65
7	Kinginrin	16	280	0	4	0.80
8	Kohaku	17	277	3	3	0.85
9	Matsuba	20	278	2	0	1.00
10	Ogon	19	278	2	1	0.95
11	Sanke	20	277	3	0	1.00
12	Showa	18	278	2	2	0.90
13	Shusui	19	280	0	1	0.95
14	Tancho	18	279	1	2	0.90
15	Utsurimono	18	276	4	2	0.90

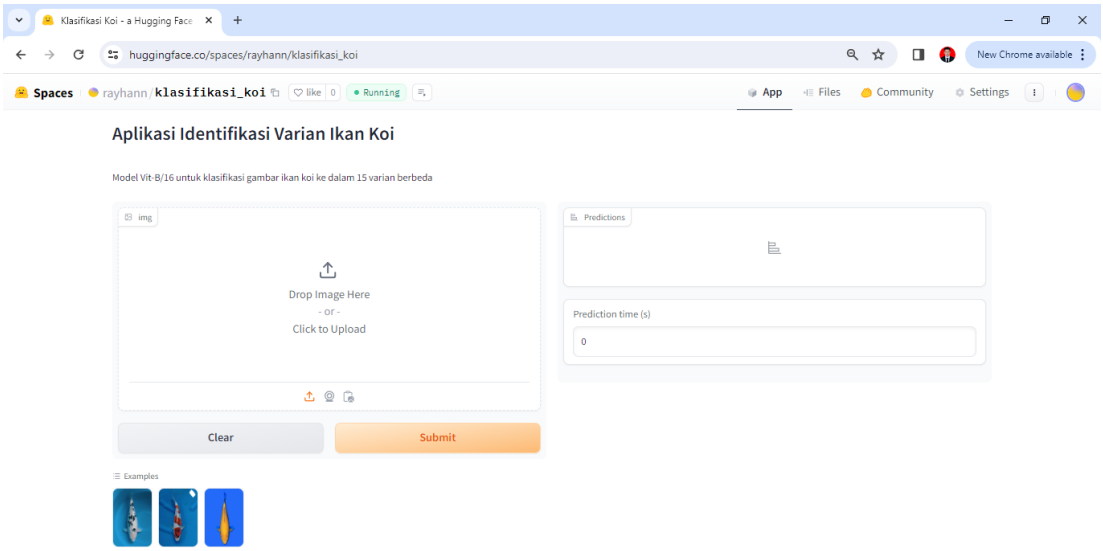
Hasil klasifikasi ViT pada Tabel 2 memetakan akurasi prediksi per kelas, hasilnya mencapai 100% untuk kelas asagi, 95% untuk kelas bekko, 75% untuk kelas doitsu hariwake, 90% untuk kelas goromo, 85% untuk kelas hikarimoyo, 65% untuk kelas kawarimono, 80% untuk kelas kinginrin, 85% untuk kelas kohaku, 100% untuk kelas matsuba, 95% untuk kelas ogon, 100% untuk kelas sanke, 90% untuk kelas showa, 95% untuk kelas shusui, 90% untuk kelas tancho, dan 90% untuk kelas utsurimono dari total 20 data uji per kelas.

3.4 Pengujian Sistem

Implementasi sistem klasifikasi ikan koi dilakukan menggunakan sistem aplikasi berbasis *graphic user interface* (GUI) yang dapat diakses di url https://huggingface.co/spaces/rayhann/klasifikasi_koi. Hasil implementasi yang telah dibuat adalah sebagai berikut:

I. Hasil Perancangan Halaman Jelajah Gambar

Penggunaan sistem klasifikasi gambar ikan koi, dilakukan dengan menggunakan file gambar yang dimasukkan ke dalam sistem melalui aplikasi menggunakan *window* 'img', dengan cara *drag and drop*, atau pilih salah satu gambar sampel di *window* 'examples'. Struktur halaman untuk menjelajah gambar dapat ditemukan di Gambar 8.

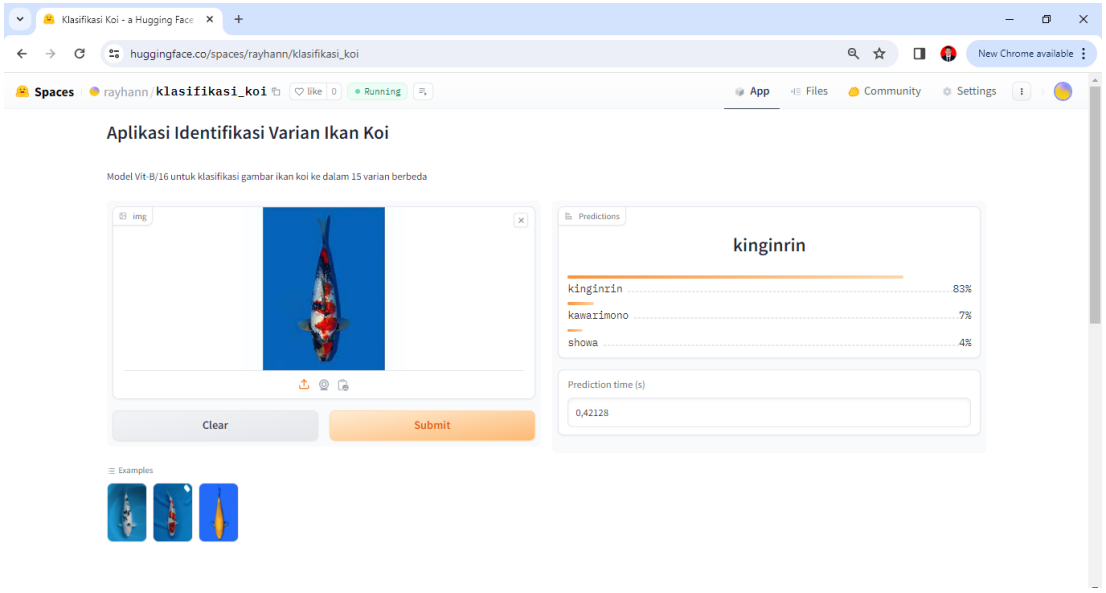


Gambar 7. Tampilan Halaman Jelajah Gambar

Jika gambar berhasil di tampilkan, maka proses klasifikasi dapat dijalankan dengan cara klik ‘submit’. Setelah gambar di *submit*, proses mulai melakukan klasifikasi dan memerlukan waktu beberapa saat hingga hasil prediksi kelas muncul.

II. Hasil Perancangan Halaman Hasil Prediksi

Halaman hasil prediksi menampilkan variasi prediksi untuk gambar ikan koi yang dipilih. Contoh pada Gambar 9 menunjukkan hasil prediksi yang akurat untuk varian koi kinginrin. Persentase probabilitas teratas dari fungsi klasifikasi pada sistem ditampilkan dalam *window* 'predictions' di sebelah kanan



Gambar 8. Tampilan Halaman

Tampilan halaman hasil prediksi menunjukkan nilai 83% pada kelas kinginrin, yang membuat hasil prediksi kelas dari gambar yang di-*upload* adalah kinginrin. Pengujian klasifikasi dapat dilakukan ulang dengan cara menghapus terlebih dahulu gambar yang ditampilkan dengan cara menekan tombol ‘Clear’, lalu dapat dilakukan kembali proses yang ada pada butir I.

4. KESIMPULAN

Penelitian yang berjudul *Vision Transformer* untuk Identifikasi 15 Variasi Citra Ikan Koi ini telah dilakukan. Melalui hasil penelitian dapat diambil beberapa kesimpulan sebagai berikut:

1. Model dengan arsitektur ViT untuk klasifikasi 15 variasi ikan koi telah dibuat menggunakan model ViT-B/16 *transfer learning*. Sistem ViT dapat memprediksi berbagai varian ikan koi berdasarkan warna dan polanya. Sebagian besar kesalahan dalam klasifikasi kemungkinan disebabkan oleh kurangnya generalisasi ViT terhadap variasi pola dan warna koi kelas kawarimono, dikarenakan terbatasnya jumlah data di dataset (80 sampel), ketidakmampuan ViT menggeneralisasi kilauan silver pada citra koi untuk mengenali kelas kinginrin, serta ketidakmampuan ViT mengenali fitur ketiadaan sisik pada doitsu hariwake.
2. Hasil klasifikasi untuk 15 variasi ikan koi menunjukkan akurasi yang signifikan. Dari pengujian pada 300 data uji, ViT mencapai akurasi keseluruhan sebesar 89%, menunjukkan performa yang baik secara keseluruhan. Akurasi yang diperoleh lebih tinggi dari metode yang dipakai pada penelitian sebelumnya dimana mereka memperoleh akurasi sebesar 84% pada metode CNN.
3. Saat melihat akurasi mikro, atau akurasi prediksi per kelas, hasilnya mencapai 100% untuk kelas asagi, 95% untuk kelas bekko, 75% untuk kelas doitsu hariwake, 90% untuk kelas goromo, 85% untuk kelas hikarimoyo, 65% untuk kelas kawarimono, 80% untuk kelas kinginrin, 85% untuk kelas kohaku, 100% untuk kelas matsuba, 95% untuk kelas ogon, 100% untuk kelas sanke, 90% untuk kelas showa, 95% untuk kelas shusui, 90% untuk kelas tancho, dan 90% untuk kelas utsurimono dari total 20 data uji per kelas.

DAFTAR PUSTAKA

[1] A. Sapphira, A. Setiawan, and E. Setyati, "Identifikasi Varietas Koi Berdasarkan Gambar Menggunakan Zero Parameter Simple Linear Iterative Clustering dan Support Vector Machine," no. 1, pp. 1–7, 2020.
[2] M. S. Cueto, J. M. B. Diangkinay, K. W. B. Melencion, T. P. Senerado, H. L. P. Taytay, and E. R. E. Tolentino, "Classification of different types of koi fish using convolutional neural network," *Proc. - 5th Int. Conf. Intell. Comput. Control Syst. ICICCS 2021*, no. Iciccs, pp. 1135–1142, 2021, doi: 10.1109/ICICCS51141.2021.9432358.
[3] A. A. Khan *et al.*, "An Efficient Text-Independent Speaker Identification Using Feature Fusion and *Transformer* Model," *Comput. Mater. Contin.*, vol. 75, no. 2, pp. 4085–4100, 2023, doi: 10.32604/cmc.2023.036797.
[4] K. Han *et al.*, "A survey on vision *Transformer*," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, 2022.
[5] J. A. Figo, N. Yudistira, and A. W. Widodo, "Deteksi Covid-19 dari Citra X-ray menggunakan Vision *Transformer*," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2020.
[6] X. Chen, C.-J. Hsieh, and B. Gong, "When vision *Transformer*s outperform resnets without pre-training or strong data augmentations," *arXiv Prepr. arXiv2106.01548*, 2021.
[7] C. F. Chen, Q. Fan, and R. Panda, "CrossViT: Cross-Attention Multi-Scale Vision *Transformer* for Image Classification," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 347–356, 2021, doi: 10.1109/ICCV48922.2021.00041.
[8] M. Usman, T. Zia, and A. Tariq, "Analyzing transfer learning of vision *Transformer*s for interpreting chest radiography," *J. Digit. Imaging*, vol. 35, no. 6, pp. 1445–1462, 2022.
[9] Y. Wu, S. Qi, Y. Sun, S. Xia, Y. Yao, and W. Qian, "A vision *Transformer* for emphysema classification using CT images," *Phys. Med. Biol.*, vol. 66, no. 24, p. 245016, 2021.
[10] G. G. Tahyudin, E. Rachmawati, and M. D. Sulistiyo, "Klasifikasi Gender Berdasarkan Citra Wajah Menggunakan Vision *Transformer*," *eProceedings Eng.*, vol. 10, no. 2, 2023.
[11] R. Ghali, M. A. Akhloufi, M. Jmal, W. Souidene Mseddi, and R. Attia, "Wildfire segmentation using deep vision *Transformer*s," *Remote Sens.*, vol. 13, no. 17, p. 3527, 2021.
[12] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: *Transformer*s for image recognition at scale," *arXiv Prepr. arXiv2010.11929*, 2020.
[13] R. Firdaus, J. Satria, and B. Baidarus, "Klasifikasi Jenis Kelamin Berdasarkan Gambar Mata Menggunakan Algoritma Convolutional Neural Network (CNN)," *J. CoSciTech (Computer Sci. Inf. Technol.*, vol. 3, no. 3, pp. 267–273, 2022.
[14] H. Mukhtar, E. Aryanto, and Y. S. Sy, "Deep Learning untuk mendeteksi gangguan lambung melalui citra iris mata," *J. CoSciTech (Computer Sci. Inf. Technol.*, vol. 4, no. 3, pp. 580–589, 2023.