p-ISSN : 2443-2210
e-ISSN : 2443-2229

Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan *Pre-Trained CNN Model*

http://dx.doi.org/10.28932/jutisi.v4i2.828

Muftah Afrizal Pangestu[1], Hendra Bunyamin[2]

Teknik Informatika, Universitas Kristen Maranatha
Jl. Prof. drg. Surya Sumantri No. 65 Bandung
[1]afrizal998@gmail.com
[2]hendra.bunyamin@it.maranatha.edu

Abstract —**The main objective of this research is to develop an image recognition system for distinguishing dog breeds using Keras' pre-trained Convolutional Neural Network models and to compare the accuracy between those models. Specifically, the models utilized are ResNet50, Xception, and VGG16. The system that we develop here is a web application using Flask as its development framework. Moreover, this research also explains how the deep learning approaches, such as CNN, can distinguish an object in an image. After testing the system on a set of images manually, we learn that every model has different performance, and Xception came out as the best in term of accuracy. We also test the acceptance of the user interface we develop to the end-users.**

Keywords— **Convolutional Neural Network, Deep Learning, Flask, Image Recognition, Keras.**

I. PENDAHULUAN

Pertumbuhan data telah berkembang secara pesat pada era digital dewasa ini [1]. Dengan ribuan foto diunggah tiap harinya, model dan algoritma yang lebih baik untuk pengenalan objek di dalam gambar (*image recognition*) dapat dikembangkan dengan bantuan data tersebut [2]. *Image recognition* adalah kemampuan komputer dan mesin untuk mengenali objek dalam gambar. Komputer menerima *input* berupa gambar dan memberikan *output* berupa hasil klasifikasi yang sesuai dengan gambar [3]. *Image recognition* dapat berperan dalam otomatisasi klasifikasi gambar, sebagai alat *surveillance* dan sistem sekuritas.

Berbeda dengan manusia dan hewan, mesin dan komputer tidak dapat mengenali objek dengan mudah [3]. Metode-metode berbasis *machine learning* merupakan teknik yang umum digunakan untuk memecahkan masalah tersebut. Salah satu teknik yang terbukti menghasilkan hasil terbaik (*state-of-the-art*) adalah teknik dengan menggunakan *Convolutional Neural Network* atau lebih dikenal dengan ConvNet atau CNN [4]. *Convolutional Neural Network* (CNN) sendiri adalah pengembangan dari algoritma *Artificial Neural Network* (ANN) yang menerima input berupa gambar. Akan tetapi, pengembangan CNN masih terhitung mahal, mulai dari waktu yang digunakan untuk *training* model dan *hardware* yang dibutuhkan untuk menjalankan proses *training* tersebut.

Untuk menghemat proses pengembangan *Convolutional Neural Network* (CNN) dan memang fokus kebanyakan pengguna adalah menggunakan *Convolutional Neural Network* (CNN), model-model *Convolutional Neural Network* (CNN) yang sudah mengalami proses *training* disimpan dalam bentuk *pre-trained model*. Keras yang merupakan high-level API untuk jaringan saraf tiruan (*neural networks*) [5] menyediakan beberapa *Pre-trained model* seperti ResNet50 [6], Xception [7] dan VGG16 [8]. Ketiga *state-of-the-art pre-trained model* tersebut memiliki perbedaan pada arsitektur dan konfigurasi CNN.

Tujuan penelitian ini adalah membandingkan performa dari ketiga model *Convolutional Neural Network* (CNN) di atas dan mengembangkan tampilan antar muka untuk mempermudah penggunaan bagi *end user* dan menampilkan hasil prediksi ketiga model tersebut.

Model ResNet merupakan model yang menggunakan *deep residual learning framework*. Dengan menggunakan *framework* ini, setiap *layer network* memiliki referensi ke *layer network* sebelumnya; hal ini menjadikan proses optimasi menjadi lebih mudah daripada *layer network-layer network* yang tidak memiliki keterhubungan. Karena proses optimasi yang lebih mudah, *neural network* yang dibentuk dapat memiliki jumlah *layer* yang banyak sampai dengan 34 *layer* dan akibatnya, akurasi meningkat dari *neural network* yang tidak menggunakan *residual network* [6].

Umumnya model CNN melakukan proses *training* dengan membangun *filter* di dalam ruang berdimensi 3, dengan dua dimensi (panjang dan lebar) dan satu buah *channel*. Lebih lanjut, model Xception memanfaatkan konsep Inception, yaitu setiap *convolution layer* dari suatu *neural network* difaktorkan menjadi suatu barisan operasi yang memanfaatkan korelasi dari *cross-channel* dan korelasi

p-ISSN : 2443-2210
e-ISSN : 2443-2229

spasial (panjang dan lebar). Dengan konsep Inception ini, akurasi model meningkat disebabkan oleh penggunaan parameter model (korelasi) yang lebih efisien [7].

VGG16 merupakan model CNN yang memanfaatkan *convolutional layer* dengan spesifikasi *convolutional filter* yang kecil (3×3). Dengan ukuran *convolutional filter* tersebut, kedalaman *neural network* dapat ditambah dengan lebih banyak lagi *convolutional layer*. Hasilnya, model CNN menjadi lebih akurat daripada model-model CNN sebelumnya. Model VGG16 mempunyai 19 *layer* yang terdiri dari 16 *convolutional layer* dan 3 *fully-connected layer* [8].

Ketiga model yang dipilih merupakan model-model CNN *state-of-the-art*, yaitu keakuratan ketiga model tersebut menjadi tolok ukur dalam pengenalan objek di dalam gambar untuk saat ini.

Adapun data latih untuk model CNN berasal dari basis data gambar, ImageNet dan data uji adalah sekumpulan gambar anjing yang sudah diberi label dari *dog-project* oleh Udacity [9].

II. KAJIAN TEORI

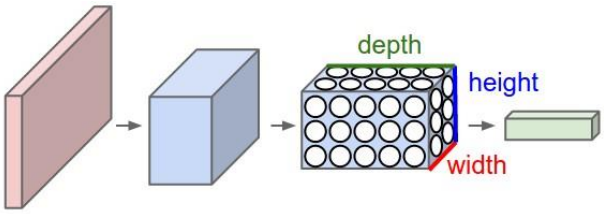
Teori yang dijelaskan meliputi: *Convolutional Neural Network*, Keras sebagai *library* untuk membangun *Artificial Neural Network* (ANN), Tensorflow, ImageNet, dan Flask.

A. Convolutional Neural Network

Convolutional Neural Network yang juga dikenal dengan CNN atau ConvNet adalah pengembangan dari *Multilayer Perceptron* (MLP) untuk mengolah data dua dimensi [10]. *Multilayer Perceptron* sendiri adalah pengembangan dari *Artificial Neural Network* (ANN) yang ditujukan untuk menutupi kekurangan *Artificial Neural Network* (ANN) dengan *Single-layer Perceptron* dalam menyelesaikan operasi logika yang kompleks [11]. Hal tersebut dimungkinkan dengan menambahkan *hidden layer* yang membuat ANN *powerful* untuk memecahkan operasi logika yang kompleks (*universal approximation*) [12] dan sering digunakan untuk permasalahan-permasalahan klasifikasi, *recognition*, dan prediksi.

CNN dapat dibilang sangat sukses dalam video dan *image recognition* berskala besar [8], Google bahkan menggunakan CNN untuk mengambil nomor rumah dari gambar StreetView [13]. Akan tetapi, dari sekian banyaknya keunggulan yang ditawarkan CNN, untuk mengaplikasikan CNN pada sekumpulan gambar dalam skala besar dan berkualitas tinggi masih terhitung mahal secara waktu, *dataset* yang digunakan, dan *hardware* [4]. Namun hal tersebut dapat diatasi saat ini dengan *dataset* gambar berukuran besar yang telah disediakan seperti ImageNet dan *pre-trained model* seperti yang disediakan oleh Keras.

Berbeda dengan manusia, komputer mengenali gambar dalam bentuk *array* dari nilai piksel-pikselnya. Bayangkan untuk kasus input gambar dengan resolusi pixel 260×260, akan terdapat 260×260×3 *array* dari angka. Angka 3 yang merupakan suku ketiga dari perkalian melambangkan 3 nilai RGB pada gambar. Selanjutnya, *array* angka-angka tersebut akan diproses oleh CNN untuk dijadikan nilai kemungkinan suatu gambar berada pada kelas tertentu, misalnya 96% untuk jenis Poodle, 8% untuk jenis Spaniel, dan 0.3% untuk Pomeranian.

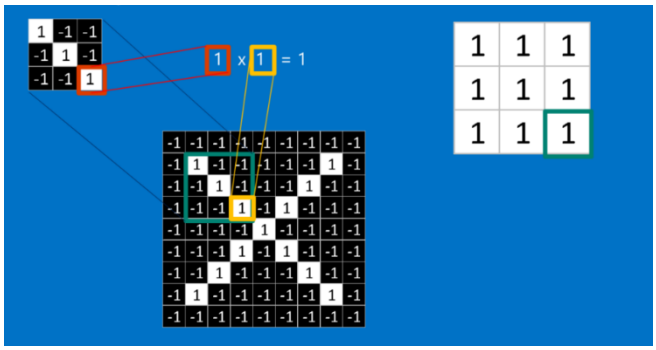


Gambar 1 Layer pada Convolutional Neural Network [14].

Pada CNN, data yang dikirimkan ke dalam jaringan adalah data dua dimensi, sehingga bobot pada hubungan antar *neuron* akan berbeda [10]. Operasi *linear* yang dilakukan pada CNN pun akan berbeda yaitu akan digunakan operasi konvolusi pada setiap *layer*. Tidak seperti ANN pada umumnya, *neuron* pada *layer* CNN diatur dalam 3 dimensi yaitu: lebar, tinggi, dan kedalaman seperti yang dijelaskan pada Gambar 1.

Layer kedua setelah *layer input* pada Gambar 1 dapat dibagi menjadi beberapa bagian yaitu *convolution layer*, *pooling layer*, dan *fully-connected layer* [14].

Pada *convolution layer*, dilakukan operasi konvolusi yang merupakan proses utama pada CNN. Konvolusi adalah istilah matematis yang berarti mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang [10]. Ketika menguji keberadaan fitur pada gambar baru, CNN akan mencoba semua kemungkinan posisi pada gambar [15]. Filter dibuat untuk menghitung kecocokan fitur pada keseluruhan gambar.

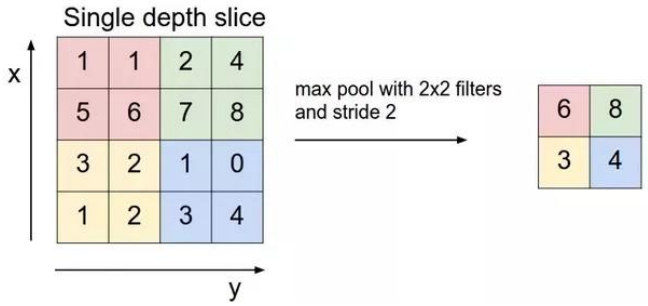


Gambar 2 Contoh operasi konvolusi pada *convolution layer* [15].

Gambar 2 adalah contoh operasi konvolusi yang membaca matriks 3×3 dan mengalikan matriks tersebut secara *element-wise* dengan dirinya sendiri.

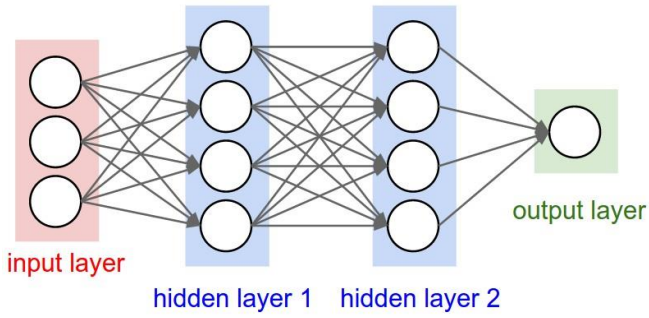
Pada *pooling layer* dilakukan operasi *subsampling*, yaitu proses mereduksi ukuran sebuah data citra [10]. *Pooling* bertujuan untuk memperkecil gambar berukuran besar tetapi

masih menyimpan informasi-informasi pentingnya. Salah satu pendekatan yang banyak digunakan untuk CNN adalah *max pooling*, *max pooling* membagi matriks gambar ke dalam beberapa bagian kecil dan memilih nilai paling besar di dalamnya untuk digunakan ketika matriks gambar baru yang sudah direduksi dibentuk. Gambar 3 menunjukkan konsep *max pooling* dengan filter 2×2 dan pergeseran (*stride*) 2.



Gambar 3 Operasi *max pooling* untuk mereduksi gambar [14].

Fully-connected layer adalah *layer* yang biasa digunakan pada *Multilayer Perceptron* (MLP) dan bertujuan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasi secara linear [10]. Gambar 4 menggambarkan suatu MLP dengan 2 *hidden layer* yang *fully-connected*. *Output* dari *layer* ini adalah *array* dengan panjang jumlah kelas yang model harus pilih. Pada *fully connected layer*, bobot yang paling besar dari *layer* sebelumnya akan menentukan fitur mana yang paling berhubungan dengan kelas atau label tersebut.



Gambar 4 *Multi-layer Perceptron* sederhana dengan 2 *hidden layer* [14].

Proses *training* pada CNN untuk mendapatkan fitur-fitur serta *weight* dari setiap kelas adalah dengan menggunakan metode yang disebut dengan *backpropagation*.

B. Hubungan dengan Penelitian Sebelumnnya

Hal yang membedakan penelitian ini dengan penelitian-penelitian sebelumnya [4,6,7,8] adalah *dataset* yang digunakan sebagai data uji. Penelitian ini bermaksud untuk mengekplorasi lebih jauh kinerja ketiga model terpilih di dalam mendeteksi suatu objek, dalam hal ini anjing, di dalam gambar.

C. Keras

Keras adalah *High Level Neural Network* API yang dapat dijalankan di atas *framework-framework machine learning* seperti TensorFlow, CNTK, atau Theano. Keras ditulis dalam bahasa Python [5]. Lebih lanjut, Keras menyediakan API yang mempermudah *user* dalam membangun arsitektur ANN.

Keras juga menyediakan Keras *Applications* yang merupakan *deep learning models* yang digunakan bersama dengan *pre-trained weights.* Model-model tersebut dapat digunakan untuk memprediksi, melakukan *feature extraction* atau *fine-tuning*.

Model-model tersebut di antaranya adalah Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet & NasNet. Penelitian ini akan membandingkan tingkat akurasi dari tiga *Pre-trained model* Keras yaitu ResNet50, Xception, dan VGG16.

D. Tensorflow

TensorFlow adalah *framework machine learning* yang bekerja dalam skala besar dan dalam *environment* yang *heterogeneous* [16]. TensorFlow digunakan untuk melakukan eksperimen model *deep learning*, melatih model pada *dataset* yang berukuran besar, dan membuatnya layak diproduksi.

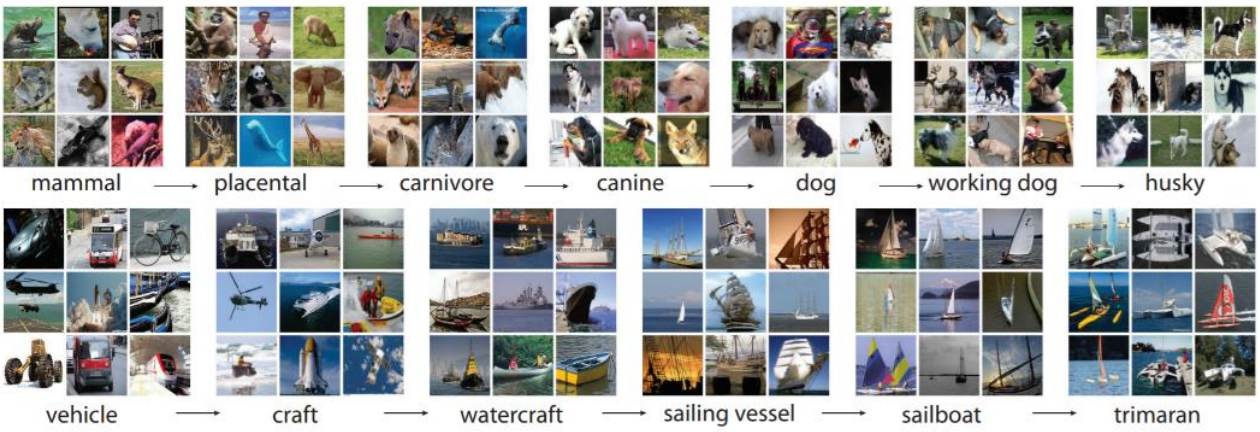
Selain itu TensorFlow juga mendukung *training* dan *inference* berskala besar dengan menggunakan ratusan *server* yang menggunakan *Graphic Processing* Unit (GPU) untuk *training* secara efisien. Penelitian ini menggunakan TensorFlow sebagai *backend framework* untuk Keras.

E. ImageNet

Penelitian ini menggunakan *pre-trained model* dari Keras yang dilatih pada *dataset* gambar berukuran besar yaitu ImageNet. ImageNet adalah *dataset* gambar yang diorganisir berdasarkan hirarki untuk setiap *node* dari hirarkinya terdiri dari ratusan hingga ribuan gambar [2]. Sebagai contoh untuk *subtree* anjing maka akan memiliki banyak *branch* yang bisa berupa gambar anjing yang telah dikelompokkan berdasarkan rasnya.

Gambar yang terdapat dalam ImageNet dikumpulkan dari Web dan diberikan label atau kelas oleh manusia dengan memanfaatkan Amazon's Mechanical Turk. Hingga pada tahun 2009, telah terdapat 12 *subtrees* yang terdiri dari 3,2 juta gambar yang sudah diberikan anotasi dan terbagi dalam 5247 kategori yang berbeda. Gambar 5 adalah contoh hierarki *subtree* dari kelas mamalia dan kendaraan.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 5 Contoh hirarki dari *subtree* ImageNet untuk mamalia dan kendaraan [2].

F. Flask

Flask adalah *Web Application Framework* yang ditulis dalam bahasa pemrograman Python. Flask digunakan untuk mempersingkat dan mempermudah pengembangan *Web Application* [17]. Flask bergantung pada dua *external libraries* yaitu Jinja2 Template Engine dan WSGI Toolkit [18]. *Extensions* dapat digunakan untuk menambahkan fitur-fitur yang tidak disediakan pada Flask dapat ditambahkan seperti *Form Validation* atau *Upload Handling*. Flask tersedia dalam bentuk *library* dan dapat di-*install* melalui Package Manager Python [19].

Penelitian ini menggunakan Flask untuk mengembangkan *user interface* yang akan dihubungkan dengan sistem pemrosesan *image recognition* menggunakan Keras. Flask dipilih karena adanya kesamaan antara Keras dan Flask yang sama-sama ditulis dalam bahasa pemrograman Python sehingga proses integrasi *user interface* (UI) dan *backend* menjadi mudah. Pada tampilan antar muka dari *web application* yang dikembangkan, digunakan juga bahasa pemrograman HTML untuk elemen dasar dari *web*, CSS, dan Bootstrap untuk memberikan *style* pada tampilan, dan juga JavaScript untuk validasi *form*.

III. DESAIN DAN IMPLEMENTASI RANCANGAN SISTEM

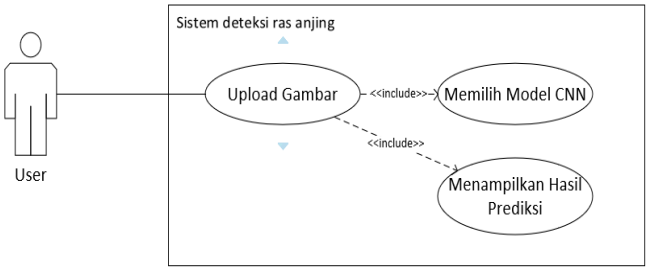
A. Gambaran Besar Sistem

Sistem yang akan dikembangkan adalah suatu aplikasi *web*. Pengembangan aplikasi *web* ini akan menggunakan *framework* Flask dan Keras sebagai *backend*. Sistem yang dikembangkan akan mengirimkan gambar yang sebelumnya sudah di *preprocessing* terlebih dahulu ke *Application Programming Interface* (API) Keras untuk kemudian diprediksi apakah gambar memiliki objek di dalamnya atau tidak. Kode program dan dokumentasi dapat diakses secara online[1].

Gambar 6 Overview Sistem yang Dikembangkan

Gambar 6 merupakan sistem keseluruhan yang terdiri dari dua buah *file* Python, yaitu `template_handler.py` dan `request_handler.py`.

File `template_handler.py` bertugas untuk mengatur navigasi dari halaman web dan mengunggah gambar. Sedangkan `request_handler.py` bertugas untuk memanggil fungsi-fungsi Keras untuk melakukan *pre-processing* gambar sehingga gambar siap untuk dikenakan proses prediksi dan *file* ini juga melakukan proses pengiriman *request* untuk memprediksi hasil gambar yang sudah dikenakan *pre-processing* tersebut.



Gambar 7 *Use case diagram* sistem deteksi ras anjing.

Gambar 7 menggambarkan bahwa *user* dapat mengunggah gambar pada tampilan *web* yang kemudian akan disimpan di dalam *server*. *User* juga dapat memilih salah satu dari tiga pilihan model yang akan digunakan dalam melakukan prediksi, yaitu ResNet50, Xception, dan

[1] https://bitbucket.org/MuftahAfrizal/kp2017-1572054-muftahafrizal-deteksirasanjing/src/master/

VGG16. Setelah *user* mengunggah gambar dan memilih model, sistem akan menampilkan hasil prediksi objek yang terdapat pada gambar.

Gambar 8 menunjukkan *activity diagram* dari sistem. Pada saat sistem dijalankan, `template_handler` akan menampilkan halaman `index.html` yang merupakan tampilan awal untuk mengunggah gambar dan memilih model. Saat *user* mengklik tombol *upload*, sistem akan melakukan validasi. Ketika untuk *user* tidak memilih satu pun *file*, sistem akan menampilkan peringatan bahwa *user* tidak mengunggah *file* apapun. Apabila *user* mengunggah *file* dengan format yang tidak diterima oleh sistem, sistem akan menampilkan halaman `exception.html`.

Apabila format diterima oleh sistem, gambar akan disalin ke dalam *server* dan sistem akan mengirimkan *request* kepada `request_handler.py`. Setelah itu, `request_handler.py` akan melakukan *preprocessing* terhadap gambar yang diunggah oleh *user* sesuai dengan model yang juga dipilih oleh *user* karena tiap model menggunakan metode *preprocessing* gambar yang berbeda. Kemudian sistem akan melakukan prediksi pada gambar yang sudah di-*preprocess* sesuai model yang dipilih.

Hasil prediksi kemudian akan di-*decode* untuk dicek apakah hasil prediksi merupakan anjing atau bukan. Kemudian hasil prediksi akan dikirim kepada `template_handler.py` untuk diproses ke dalam halaman. Apabila hasil kembalian menghasilkan prediksi anjing, `template_handler.py` akan menampilkan `dog.html` dan apabila hasil prediksi bukanlah anjing, `template_handler.py` akan menampilkan halaman `non_dog.html`.

Gambar 9 merupakan halaman utama dari sistem. Halaman utama terdiri dari 3 bagian, yaitu (a), (b), dan (c). Bagian (a) adalah petunjuk penggunaan sistem. Bagian (b) adalah tempat *user* mengunggah file gambar dan bagian (c) merupakan penjelasan konsep singkat mengenai *Convolutional Neural Network*. Selain itu, Gambar 10 menggambarkan halaman hasil prediksi untuk model Xception dan file gambar yang di-*upload*, yaitu ras anjing *Labrador Retriever* dengan persentase 84%.

IV. PENGUJIAN

A. Pengujian Tingkat Akurasi Pada Ketiga Model Keras

Pengujian dilakukan pada ketiga model yang sudah diterapkan pada sistem yang telah dikembangkan, yaitu ResNet50, XCeption, dan VGG16. Ketiga model tersebut akan diuji pada sekumpulan gambar yang sama dan masing-masing akan dihitung keakuratan prediksinya. Kumpulan gambar yang akan digunakan untuk pengujian adalah 79 gambar anjing dari berbagai ras yang telah diberi label untuk memudahkan pengujian.

Dalam pengujian, dihitung 2 faktor penentu yaitu akurat atau tidaknya model dalam menentukan apakah terdapat anjing atau tidak dalam gambar dan dalam menentukan ras anjing pada gambar. Apabila model mengeluarkan prediksi yang sesuai maka akan diberikan nilai 1 dan 0 apabila prediksinya salah. Tabel I menunjukkan hasil pengujian tingkat akurasi tiap model terhadap 79 gambar anjing yang telah diberi label. Terlihat bahwa keakuratan model Xception lebih baik daripada kedua model lainnya.

TABEL I

HASIL PENGUJIAN TINGKAT AKURASI MODEL

Model	Dog/Not	Dog Race
Xception	**98.7%**	**67.1%**
ResNet50	97.5%	62.0%
VGG16	97.5%	58.2%

B. Error Analysis dengan Metode Eyeball Test

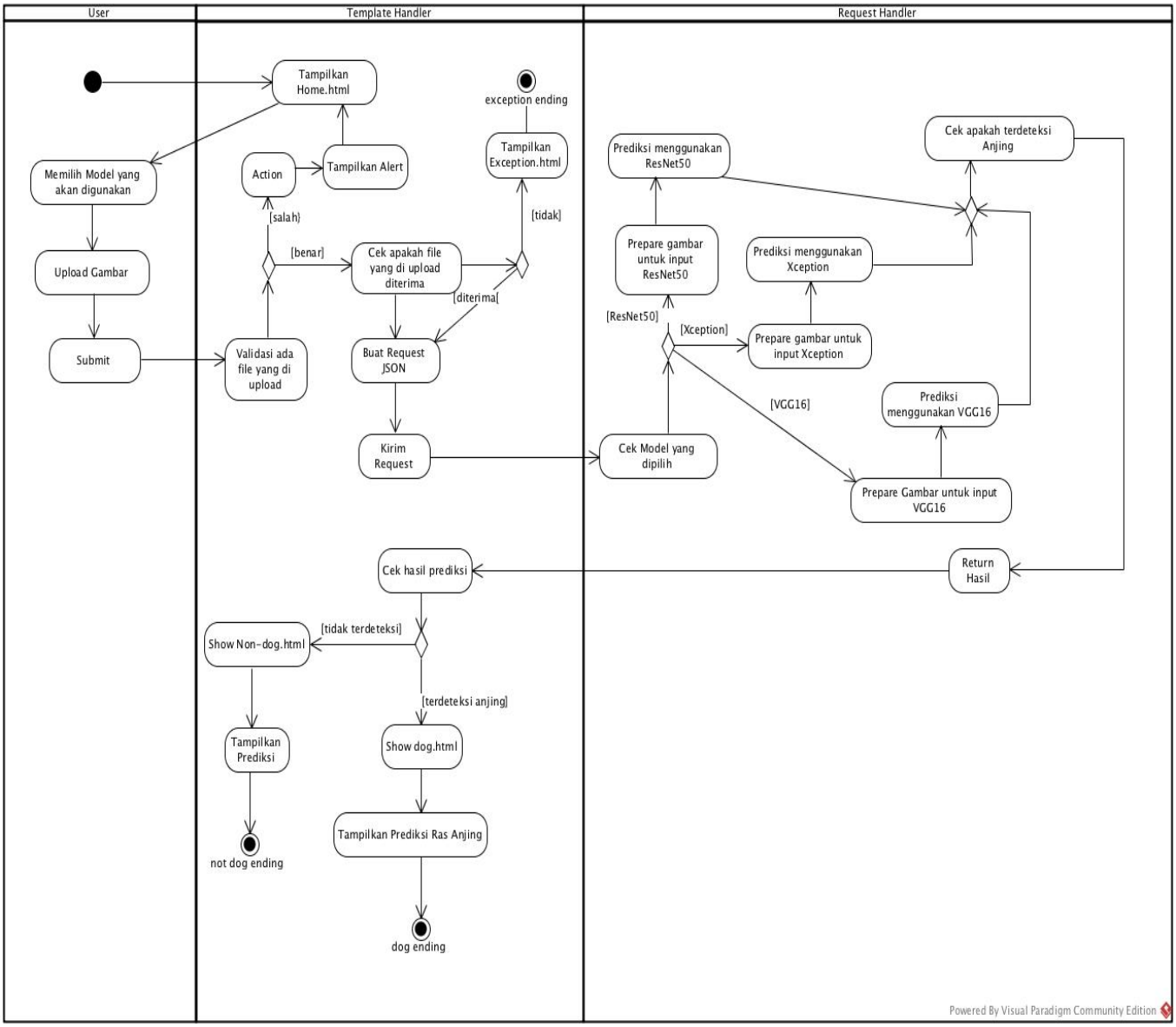
Error Analysis adalah proses untuk menyelidiki kembali contoh-contoh hasil klasifikasi salah yang dilakukan oleh algoritma yang dikembangkan untuk mengetahui hal-hal yang dapat memungkinkan terjadinya *error* [1]. *Error analysis* dapat digunakan sebagai acuan untuk pengembangan model selanjutnya. Metode yang digunakan dalam penelitian ini untuk *error analysis* adalah *Eyeball Test.*

TABEL II

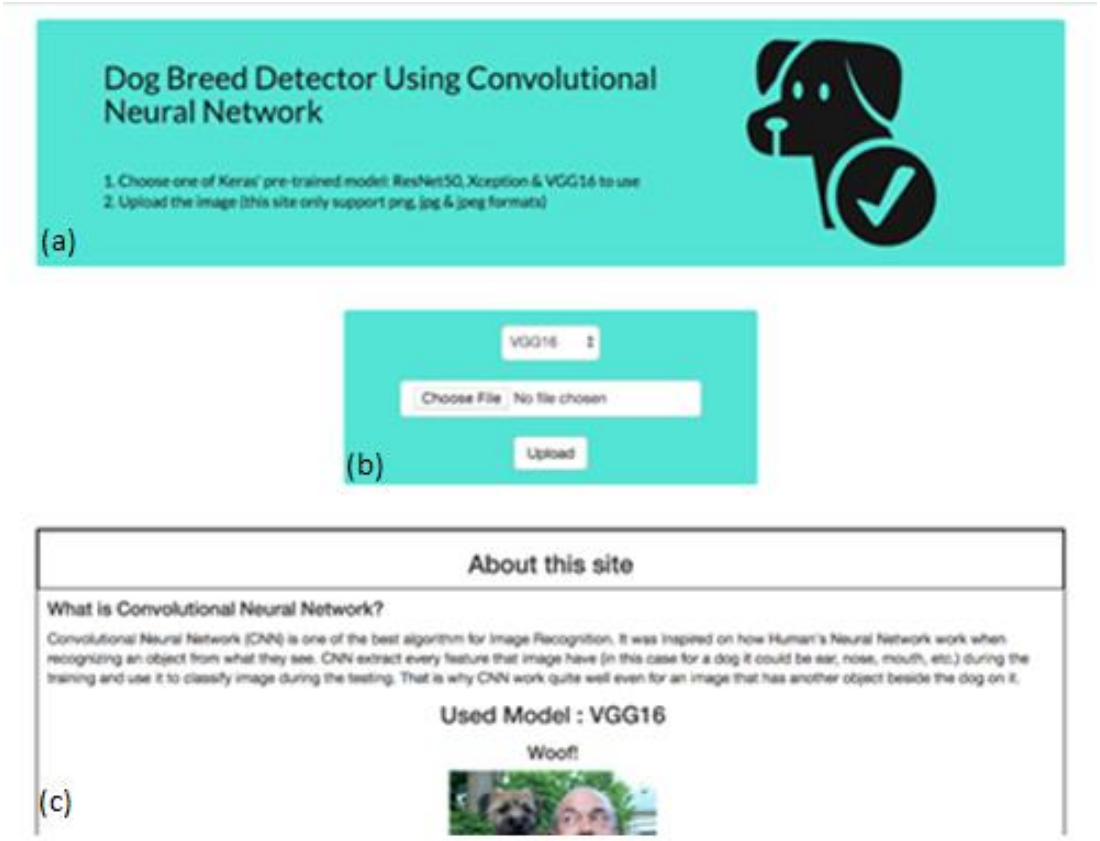
BEBERAPA SAMPEL ERROR ANALYSIS

Gambar	Exception	ResNet50	VGG16
American Eskimo Dog	Pomerian	Pomerian	Pomerian
Keterangan	American Eskimo Dog pada gambar masih berumur muda, sehingga terlihat mirip dengan pomeranian.		
Alaskan Malamute	Eskimo Dog	Alaskan Malamute	Alaskan Malamute
Keterangan	Pada gambar tidak terlalu mirip, tetapi malamute berada pada posisi ranking nomor dua dari hasil prediksi Xception		
American Water Spaniel	Sussex Spaniel	Gordon Setter	Afghan Hound
Keterangan	Gambar gelap, buram & terdapat manusia. Ras hasil prediksi cukup mirip dengan hasil asli.		
Anatolian Shepherd Dog	Leonberg	Leonberg	Irish Wolfhound
Keterangan	Terdapat manusia pada gambar. Ras cukup mirip dengan Leonberg tetapi tidak mirip dengan Irish Wolfhound.		

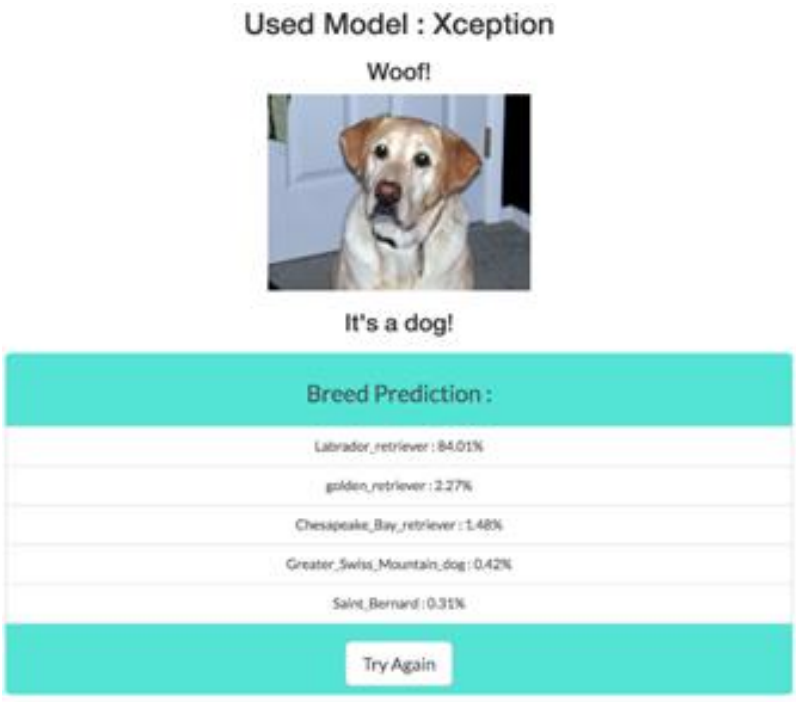
Tabel II menunjukkan beberapa sampel error analisis. Baris pertama menunjukkan bahwa prediksi ketiga model adalah *Pomerian* sedangkan ras anjing yang benar adalah *American Eskimo Dog*. Baris kedua menunjukkan bahwa hanya model Xception yang memberikan prediksi yang salah (*Eskimo Dog*, yang benar adalah *Alaskan Malamute*) dan baris ketiga menunjukkan bahwa semua model memberikan prediksi yang salah.



Gambar 8 *Activity diagram* sistem deteksi ras anjing



Gambar 9 Halaman utama sistem deteksi ras anjing



Gambar 10 Hasil prediksi sistem deteksi ras anjing.

V. SIMPULAN DAN SARAN

Dari tiga *pre-trained model* dari Keras yang diujikan (ResNet50, Xception dan VGG16), Xception memiliki performa keakuratan yang lebih baik daripada dua model lainnya. Selain itu, tersedianya tampilan antar muka juga akan membantu *user* dalam menggunakan sistem deteksi ras anjing ini.

Adapun saran yang dapat dikembangkan lebih lanjut adalah menambah model yang diuji [20], menambah jumlah data yang digunakan, membuat sistem ini *online* sehingga sistem dapat digunakan oleh kalangan luas, dan meningkatkan akurasi model untuk mendeteksi wajah manusia.

DAFTAR PUSTAKA

[1] A. Ng, *Machine Learning Yearning: Technical strategy for AI engineers, in the era of deep learning*, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li dan L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, *CVPR 2009*, Miami, 2009.

[3] A. Deshpande, *A Beginner's Guide To Understanding Convolutional Neural Networks*, 20 July 2016. [Online]. Available: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/. [Diakses 28 April 2018].

[4] A. Krizhevsky, I. Sutskever dan G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, *Neural Information Processing System 2012*, Stateline, 2012.

[5] Keras, *Keras Documentation*, [Online]. Available: https://keras.io. [Diakses 8 April 2018].

[6] K. He, X. Zhang, S. Ren dan J. Sun, *Deep Residual Learning for Image Recognition*, *CVPR 2016*, Boston, 2016.

[7] F. Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions*, *CVPR 2017*, Honolulu, 2017.

[8] K. Simonyan dan A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, *ICLR*, San Diego, 2015.

[9] udacity, Github, Inc., 2017. [Online]. Available: https://github.com/udacity/dog-project. [Diakses 11 May 2018].

[10] I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, *Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101*, *Jurnal Teknik ITS*, vol. 5, no. 1, pp. 65-69, 2016

[11] I. Krisnadi, A. Rakhmatsyah dan T. A. B. Wirayuda, *Perbandingan Antara Jaringan Saraf Tiruan Multilayer Perceptron dan Jaringan Saraf Tiruan Multilayer Perceptron*, 2008.

[12] E. Alpaydin, *Introduction to Machine Learning 2nd Edition*, London: MIT Press, 2010.

[13] G. Developer, *Google Developer Blog*, 21 September 2017. [Online]. Available: https://developers.googleblog.com/2017/09/how-machine-learning-with-tensorflow.html. [Diakses 26 March 2018].

[14] A. Karpathy dan J. Johnson, *C231n Convolutional Neural Networks for Visual Recognition*, Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 15 April 2018]

[15] B. Rohrer, *Data Science and Robot Blog*, 18 August 2016. [Online]. Available: https://brohrer.github.io/how_convolutional_neural_networks_work.html. [Diakses 28 April 2018]

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane , R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke , Y. Yu dan X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous* Distributed Systems, 2015.

[17] A. F. Aslam, N. H. Mohammed dan P. Lokhande, *Efficient Way of Web Development Using Python and Flask*, *International Journal of Advanced Research in Computer Science*, vol. 6, 2015.

[18] Pallets Team, *Flask Documentation*, Pallets Team , 2010. [Online]. Available: http://flask.pocoo.org/docs/1.0/. [Diakses 24 April 2018].

[19] Python Software Foundation, *Python Documentation*, Python Software Foundation, 30 April 2018. [Online]. Available: https://docs.python.org/3/tutorial/venv.html. [Diakses 1 May 2018]

[20] J. Rey, Object Detection: An Overview in the age of Deep Learning, Tryo Labs, 30 Agustus 2017. [Online]. Available: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning. [Diakses 18 Juli 2018]