TESIS

PENINGKATAN PERFORMA KLASIFIKASI RAS KUCING MENGGUNAKAN HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK ARSITEKTUR XCEPTION

Disusun oleh:

Nama : **D. Diffran Nur Cahyo**
NIM : **22.51.1254**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

TESIS

PENINGKATAN PERFORMA KLASIFIKASI RAS KUCING MENGGUNAKAN HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK ARSITEKTUR XCEPTION

IMPROVING CAT BREED CLASSIFICATION PERFORMANCE USING HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK XCEPTION ARCHITECTURE

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama : **D. Diffran Nur Cahyo**
NIM : **22.51.1254**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

HALAMAN PENGESAHAN

PENINGKATAN PERFORMA KLASIFIKASI RAS KUCING MENGGUNAKAN HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK ARSITEKTUR XCEPTION

IMPROVING CAT BREED CLASSIFICATION PERFORMANCE USING HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK XCEPTION ARCHITECTURE

Dipersiapkan dan Disusun oleh

D. Diffran Nur Cahyo

22.51.1254

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Selasa, 2 Januari 2024

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 2 Januari 2024
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

PENINGKATAN PERFORMA KLASIFIKASI RAS KUCING MENGGUNAKAN HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK ARSITEKTUR XCEPTION

IMPROVING CAT BREED CLASSIFICATION PERFORMANCE USING HYPERPARAMETER TUNING CONVOLUTIONAL NEURAL NETWORK XCEPTION ARCHITECTURE

Dipersiapkan dan Disusun oleh

D. Diffran Nur Cahyo

22.51.1254

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Selasa, 2 Januari 2024

Pembimbing Utama	**Anggota Tim Penguji**
Dr. Andi Sunyoto, M.Kom.	**Hanif Al Fatta, M.Kom., Ph.D.**
NIK. 190302052	**NIK. 190302096**
	Alva Hendi Muhammad, S.T., M.Eng., Ph.D.
Pembimbing Pendamping	**NIK. 190304293**
Dhani Ariatmanto, M.Kom., Ph.D.	**Dr. Andi Sunyoto, M.Kom.**
NIK. 190302197	**NIK. 190302052**

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 2 Januari 2024
Direktur Program Pascasarjana

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : D. Diffran Nur Cahyo
NIM : 22.51.1254
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Peningkatan Performa Klasifikasi Ras Kucing menggunakan Hyperparameter Tuning Convolutional Neural Network Arsitektur Xception

Dosen Pembimbing Utama : Dr. Andi Sunyoto, M.Kom.
Dosen Pembimbing Pendamping : Dhani Ariatmanto, M.Kom., Ph.D.

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 2 Januari 2024
Yang Menyatakan,

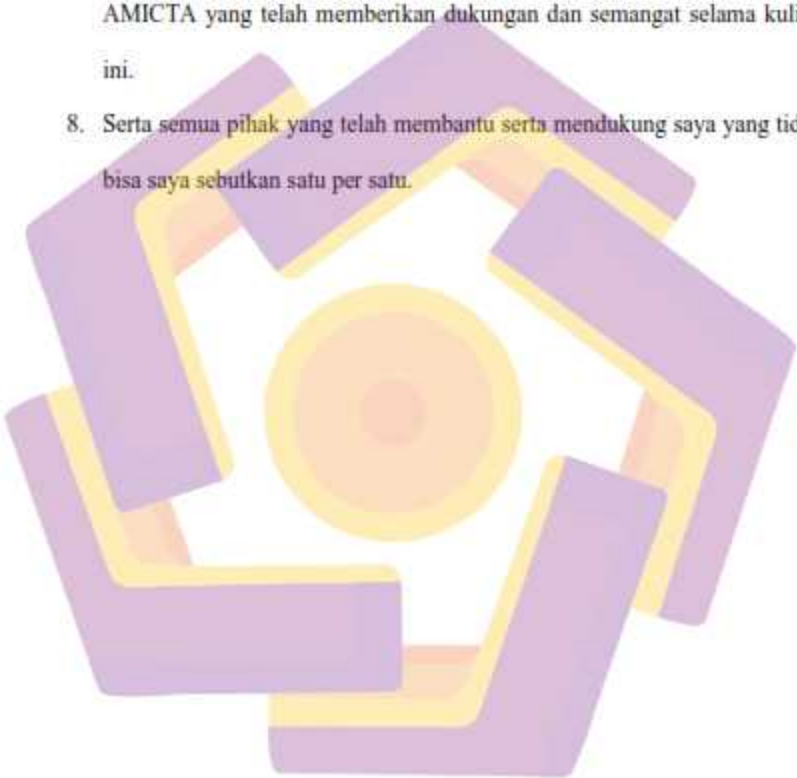
D. Diffran Nur Cahyo

HALAMAN PERSEMBAHAN

Puji Syukur saya panjatkan kehadirat Allah SWT yang telah memberikan nikmat dan berkat yang luar biasa kepada saya, sehingga saya bisa menyelesaikan tesis ini dengan baik. Saya juga sangat berterima kasih kepada orang-orang yang secara langsung maupun tidak langsung telah membantu saya dalam menyelesaikan tesis ini. Tesis ini saya persembahkan kepada:

1. Ayah dan Ibu saya yang selalu mendokakan saya, selalu mendukung baik finansial maupun dukungan lainnya. Terima kasih sudah mengorbankan banyak hal untuk keberhasilan ini.
2. Bapak Dr. Andi Sunyoto, M.Kom dan Dhani Ariatmanto, M.Kom, Ph.D selaku dosen pembimbing yang selalu memberikan masukan serta bimbingan yang positif dalam menyelesaikan tesis ini. Terima kasih juga atas ilmu yang telah diberikan.
3. Bapak dan Ibu Dosen yang selalu memberikan ilmu yang bermanfaat selama saya kuliah.
4. Teman-teman Angkatan 28 untuk memori indah yang pernah kita rajut bersama selama perkuliahan. Terima kasih atas bantuan dan ilmu yang pernah kalian bagi.
5. Saudari Silpi Dwine Rahayu, S.Ak yang telah membantu dan memberikan semangat serta dukungan selama saya kuliah.

6. Teman-teman Burjo Animation sekaligus alumni S1 Teknologi Informasi 18 Universitas Amikom Yogyakarta yang telah membantu serta mendukung saya untuk menyelesaikan kuliah ini.
7. Rekan satu tim lomba Indoneris 2022, PHR Virtual Challenge 2023, dan AMICTA yang telah memberikan dukungan dan semangat selama kuliah ini.
8. Serta semua pihak yang telah membantu serta mendukung saya yang tidak bisa saya sebutkan satu per satu.



HALAMAN MOTTO

"Don't compare yourself with anyone in this world. If you do so, you are insulting yourself"

Jangan membandingkan diri Anda dengan siapa pun di dunia ini. Jika Anda melakukannya, Anda sedang menghina diri sendiri

Bill Gates

"The only way to prove yourself is to perform. Always be the hardest worker in the room"

Satu-satunya cara untuk membuktikan diri adalah dengan berkinerja. Selalu menjadi pekerja yang paling keras di ruangan itu

Sir Alex Ferguson

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah sehingga penulis dapat menyelesaikan tugas akhir dengan judul **"Peningkatan Performa Klasifikasi Ras Kucing menggunakan Hyperparameter Tuning Convolutional Neural Network Arsitektur Xception"** dengan bauk dan sesuai waktu yang diharapkan.

Tesis ini untuk memenuhi salah satu syarat untuk memperoleh gelar Magister Informatika di Universitas Amikom Yogyakarta. Penulis menyadari bahwa penulisan ini tidak akan terwujud tanpa adanya dukungan, bantuan dari berbagai pihak. Untuk itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada yang terhormat:

1. Kedua Orang tua saya, Ayah Nur Cahyono dan Ibu Dwi Astuti Widyaningrum yang telah memberikan saya semangat untuk segera menyelesaikan kuliah S2 saya ini.
2. Adik saya, Daray Nur Cahyo yang telah memberikan saya dukungan untuk menyelesaikan kuliah S2 ini.
3. Ibu Prof. Dr. Kusrini, M.Kom selaku Direktur Program Pascasarjana.
4. Bapak Dr. Andi Sunyoto, M.Kom selaku Pembimbing Utama saya yang telah membimbing saya sehingga terselesaikannya naskah tesis ini.
5. Bapak Dhani Ariatmanto, M.Kom, Ph.D selaku Pembimbing Pendamping saya yang telah membimbing saya sehingga terselesaikannya naskah tesis ini.

6. Ibu Prof. Ema Utami, S.Si., M.Kom dan Alva Hendi Muhammad, S.T., M.Eng., Ph.D selaku Penguji Seminar Proposal Tesis.
7. Bapak Hanif Al Fatta, M.Kom, Ph.D dan Alva Hendi Muhammad, S.T., M.Eng., Ph.D selaku Penguji Seminar Hasil Proposal Tesis dan Ujian Tesis.
8. Bapak dan Ibu Dosen Universitas Amikom Yogyakarta yang telah memberikan ilmu, pengetahuan, motivasi, dan pengalaman setiap mengajar selama penulis menempuh kuliah.
9. Saudari Silpi Dwine Rahayu, S.Ak yang selalu memberikan dukungan dan semangat untuk menyelesaikan kuliah S2 ini.
10. Teman-teman Angkatan 28 yang telah menemani penulis selama masa perkuliahan dan berbagi canda tawa bersama.

Penulis menyadari bahwa dalam pembuatan tesis ini masih banyak kekurangan dan kelemahannya. Oleh karena itu penulis berharap kepada semua pihak agar dapat menyampaikan kritik dan saran yang membangun untuk menambah kesempurnaan tesis ini. Namun, penulis tetap berharap tesis ini akan bermanfaat bagi semua pihak yang membacanya.

Yogyakarta, 2 Januari 2024

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Klasifikasi ras kucing, dalam bidang pengolahan citra dan machine larning menjadi sebuah tantangan karena adanya kesamaan karakteristik fisik di antara berbagai ras. Fokus utama penelitian ini adalah meningkatkan akurasi dalam klasifikasi ras kucing dengan menggunakan algoritma Convolutional Neural Network (CNN) dengan arsitektur Xception. Tujuan dari penelitian ini adalah untuk memperbaiki tingkat akurasi dalam klasifikasi ras kucing dengan menggabungkan algoritma CNN dan SVM yang dimana model CNN yang menggunakan hyperparameter yang telah ditentukan oleh peteliti.

Metodologi yang diterapkan dalam penelitian ini mencakup beberapa tahap penting. Tahap pertama adalah pengumpulan data yang kemudian diikuti oleh preprocessing dan augmentasi. Selanjutnya, penelitian ini melakukan implementasi berbagai skenario yang menggabungkan CNN dan SVM, dengan tujuan menemukan kombinasi yang paling efektif. Evaluasi dari model-model ini dilakukan dengan menggunakan confusion matrix seperti akurasi dan perbandingan kinerja.

Hasil yang diperoleh dari penelitian ini menunjukkan skenario optimal dengan nilai akurasi yang tinggi, yaitu sebesar 94.58% Hal ini menandakan efektivitas dari pendekatan yang diambil dalam penelitian. Kesimpulan dari penelitian ini menekankan keberhasilan model dalam mengklasifikasikan ras kucing dengan tingkat akurasi yang sangat tinggi. Saran untuk penelitian lebih lanjut mencakup pengaplikasian model ini dalam situasi waktu nyata dan eksplorasi lebih dalam terhadap arsitektur atau dataset yang berbeda untuk melihat kemungkinan peningkatan kinerja.

Kata kunci: Klasifikasi ras kucing, CNN, Deep Learning, Xception

ABSTRACT

Cat breed classification, in the field of image processing and machine learning, presents a challenge due to the similarity in physical characteristics among various breeds. The primary focus of this research is to enhance the accuracy of cat breed classification using the Convolutional Neural Network (CNN) algorithm with Xception architecture. The aim of this study is to improve the accuracy level in cat breed classification by combining the CNN and SVM algorithms, where the CNN model employs hyperparameters predetermined by the researchers.

The methodology applied in this research involves several crucial stages. The first stage is data collection, followed by preprocessing and augmentation. Subsequently, the study implements various scenarios combining CNN and SVM, aiming to find the most effective combination. The evaluation of these models is conducted using a confusion matrix, including accuracy and performance comparison.

The results obtained from this research indicate an optimal scenario with a high accuracy value, specifically 94.58%. This demonstrates the effectiveness of the approach taken in the study. The conclusion of this research emphasizes the success of the model in classifying cat breeds with a very high accuracy level. Recommendations for further research include applying this model in real-time situations and deeper exploration of different architectures or datasets to explore potential performance improvements.

Keywords: Cat Breed Classification, CNN, Deep Learning, Xception

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Kucing dalam bahasa latin Felis silvestris catus, adalah sejenis karnivora. Kata "kucing" biasanya mengacu pada "kucing" peliharaan, tetapi juga dapat merujuk pada "kucing besar" seperti singa, harimau, dan macan tutul. Kucing merupakan salah satu hewan peliharaan terpopuler di dunia. Kucing (Felis catus) adalah mamalia karnivora dari keluarga Felidae. Kucing telah diintegrasikan ke dalam kehidupan manusia setidaknya sejak 6.000 SM, dimulai dengan kerangka kucing di pulau Siprus. Orang Mesir Kuno dari 3500 SM. gunakan kucing untuk menjauhkan tikus atau hewan pengerat lainnya dari gudang tempat penyimpanan hasil panen. Pada 6000 SM, kucing diketahui telah berbaur dengan manusia dan menyebar ke berbagai belahan dunia (Fossati & Giancarlo, 2021). Jumlah kucing ras di dunia hanya sekitar 1%, sehingga sebagian besar didominasi oleh kucing hibrida atau domestik. Kurangnya kucing ras membuat kucing ras jauh lebih mahal. Setiap ras kucing memiliki karakteristiknya sendiri, tetapi karena banyaknya persilangan, menjadi lebih sulit untuk mengidentifikasi ras kucing (Azahro Choirunisa et al., 2021). Kucing adalah hewan pintar, menggemaskan, dan dapat menghibur serta dapat dijadikan teman atau kerabat pada saat kesepian (Samsugi & Naufal Falikh Suprapto, 2021). Tampilannya yang menarik dengan berbagai jenisnya membuat kucing menjadi salah satu hewan peliharaan terpopuler di dunia. Banyak orang memelihara kucing karena kelebihannya seperti mengusir tikus,

kenyamanan karena tidak berisik, dan tidak membutuhkan ruangan yang begitu luas. Selain itu, memiliki kucing juga dapat memberikan dampak positif bagi pemiliknya, salah satunya adalah menghilangkan stress (Karlita et al., 2022).

Namun, ada berbagai ras kucing di seluruh dunia dan setiap ras kucing memiliki ciri, penampilan, dan kepribadian yang berbeda. Tidak semua kucing kucing di seluruh dunia memiliki ciri fisik atau ciri perilaku yang sama, mereka juga tidak mirip dengan satu sama lain dalam segala hal. Tergantung pada ras mereka, biasanya ras kucing menentukan kapasitas fisik, serta kepribadian dan preferensi mereka. Sebagai pemilik atau pemelihara kucing harus mengetahui ras kucing, alasannya karena ini akan memberi gambaran yang jelas tentang keperluan dan cara perawatan mereka, agar mereka dapat menikmati hidup mereka dengan umur yang panjang. Umumnya, ada beberapa faktor yang dapat menentukan jenis ras kucing. Diantaranya yaitu ukuran, pemeliharaan, aktivitas, mantel bulu, kepribadian dan temperamen.

Menurut Whiskas Indonesia, untuk bisa mengetahui jenis ras kucing, pemilik atau pemelihara kucing dapat melakukannya dengan tes DNA ke dokter hewan. Tes DNA tidak hanya untuk mengetahui jenis ras kucing saja, tetapi dapat mengetahui nenek moyang dari kucing tersebut. Sayangnya, untuk tes DNA kucing sendiri sangatlah mahal dan membutuhkan waktu yang lama. Untuk bisa mengetahui jenis ras kucing diperlukan sebuah metode klasifikasi.

Secara umum, klasifikasi berarti proses pengelompokan. Dalam istilah ilmu data, klasifikasi kemudian dipahami sebagai proses pengelompokan data ke dalam beberapa kategori untuk memudahkan pengolahan dan analisis (Audebert et al.,

2019). Proses klasifikasi pada dasarnya dilakukan agar analisis data menjadi lebih mudah dan tentunya memberikan hasil yang akurat. Agar bisa memberikan suatu informasi yang bermanfaat, data memang memerlukan proses panjang (Audebert et al., 2019). Klasifikasi gambar digunakan hampir di semua aspek, penggunakan klasifikasi tidak sepenuhnya tercapai di bidang tertentu. Salah satu bidang tersebut yaitu klasifikasi spesies hewan. Klasifikasi gambar hewan tetap menjadi masalah yang belum terpecahkan karea tantangan dalam gambar. Dalam hal klasifikasi dan pengenalan citra digital, hewan adalah paling sulit (Kong et al., 2019).

Munculnya era kecerdasan buatan pengenalan gambar berkembang sangat pesat. Seringkali koleksi gambar yang digunakan pada beberapa tahap sistem klasifikasi memiliki kekurangan, seperti banyak noise, shading, kontras rendah, dan gambar buram. Salah satu pendekatan yang dapat digunakan adalah teknik deep learning yang dapat menangkap dan memukul suatu item dalam sebuah citra virtual. Salah satu teknik deep learning yang dapat mengenali dan mengklasifikasikan kategori citra adalah Convolutional Neural Networks (CNN) (Purnama, 2020).

Convolutional Neural Network (CNN) adalah jenis deep learning neural network (Aloysius & Geetha, 2018). CNN adalah terobosan besar dalam pengenalan gambar. Mereka umumnya digunakan untuk analisis gambar visual dan sering bekerja di belakang klasifikasi gambar. CNN memiliki teknik yang sangat baik untuk deteksi dan pengenalan objek (Q. Zhang et al., 2019). Metode CNN mirip dengan neuron pada otak manusia, dilengkapi dengan fungsi weight, bias, dan activation. CNN mempunyai beberapa lapisan konvolusional yang ditambahkan untuk mendapatkan fitur gambar secara otomatis (X. Zhang et al., 2019). Kelebihan

dari CNN dalam pengolahan citra digital yaitu arsitekturnya yang mampu mengenali informasi prediktif dari suatu objek. Namun, metode CNN memiliki kelemahan yaitu membutuhkan waktu komputasi yang sangat lama untuk menyelesaikan tugasnya (Albawi et al., 2018). Sedangkan Support Vector Machine (SVM) adalah algoritma machine learning dengan pendekatan supervised learning yang diawasi yang dapat digunakan untuk klasifikasi dan regresi. Cara kerja dari SVM didasarkan pada SRM atau Structural Risk Minimization yang dirancang untuk mengolah data menjadi Hyperplane yang mengklasifikasikan ruang input menjadi dua kelas (Cervantes et al., 2020). Pada konsep SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. Teori SVM diawali dengan pengelompokan kasus-kasus linier yang dapat dipisahkan dengan hyperplane dan dibagi menurut kelasnya. SVM juga dapat mengatasi masalah klasifikasi dan regresi dengan linear maupun nonlinear (Ebrahimi et al., 2022).

Secara umum arsitektur CNN dibagi menjadi tiga layer dengan kontribusi CNN di feature extraction layer adalah pada convolution dan pooling layer. Lalu, pada fully-connected layer yang digunakan untuk melakukan klasifikasi hanya dapat digunakan pada akhir neural network. Karena untuk masuk ke sebuah fully-connected layer data dari convolution layer perlu ditransformasi mejadi data satu dimensi terlebih dahulu sehingga menyebabkan data kehilangan informasi spasialnya (Pan et al., 2022). CNN memiliki beberapa arsitektur (Khan et al., 2020) yang digunakan seperti AlexNet, GoogLeNet, NASNetMobile, VGG16, VGG19, Inception-V3, Xception, dsb (Elhassouny & Smarandache, 2019). Dalam

penggunaan CNN, perlu penentuan besar learning rate yang akan digunakan dalam melakukan training dataset. Salah satu tantangan kritis dalam machine learning, khususnya dalam penerapan CNN adalah tuning atau penyetelan hyperparameter (T. Yu & Zhu, 2020). Hyperparameter adalah parameter yang nilai-nya ditetapkan sebelum proses pelatihan model dan tidak berubah selama pelatihan berlangsung (Yang & Shami, 2020). Hal ini termasuk dalam learning rate, jumlah epoch, ukuran batch, dan parameter regularisasi yang memiliki peran penting dalam menentukan kinerja model (Bischl et al., 2023). Penyetelan hyperparameter yang tidak tepat dapat menyebabkan model tidak mampu menggeneralisasi data dengan baik, yang pada akhirnya mempengaruhi akurasi klasifikasi (Bischl et al., 2023). Dalam konteks klasifikasi ras kucing, penyetelan hyperparameter menjadi aspek kunci untuk mencapai tingkat akurasi yang tinggi. Oleh karena itu, penelitian ini berfokus pada eksplorasi strategi penyetelan hyperparameter yang optimal pada arsitektur CNN. Penggunaan learning rate yang tepat dalam algoritma pembelajaran mesin sangat penting untuk mencapai konvergensi yang baik dan memperoleh hasil yang optimal. Learning rate adalah parameter yang mengontrol seberapa besar penyesuaian bobot dan bias dalam proses pelatihan model. Memilih learning rate yang sesuai dapat mempengaruhi kecepatan konvergensi dan kestabilan model (Smith, 2017).

Pada penelitian (Ismail et al., 2019; Ruder, 2016; Smith, 2017) telah melakukan untuk memahami pengaruh learning rate pada pelatihan model dan pemilihan learning rate yang optimal. Namun, tidak ada aturan tetap yang berlaku

untuk semua jenis masalah dan arsitektur model. Pemilihan learning rate yang tepat sering kali merupakan proses empiris yang melibatkan percobaan dan iterasi.

Menyiasati hal yang berkaitan dengan klasifikasi ras kucing, pada penelitian (Afif et al., 2020) melakukan penelitian dengan menggunakan algoritma CNN dengan yang diklasifikasikan menjadi 12 kelas. Akurasi yang didapatkan dengan hasil yang terbaik yaitu sebesar 93,75% dengan menggunakan Xception dan transfer learning. Pada penelitian (Imanuel & Setiabudi, 2022) melakukan penelitian dengan menggunakan algoritma CNN dengan yang diklasifikasikan menjadi 12 kelas. Akurasi yang didapatkan yaitu sebesar 89.59% dengan menggunakan Pre-trained model xception. Namun, dalam penelitian klasifikasi ras kucing yang menggunakan algoritma kombinasi CNN dan SVM masih belum ada sehingga menggunakan acuan penelitian dengan objek yang berbeda.

Kombinasi CNN-SVM memanfaatkan keunggulan masing-masing metode. CNN digunakan untuk menghasilkan fitur-fitur yang relevan dan mengurangi dimensi data, sedangkan SVM digunakan untuk melakukan klasifikasi berdasarkan fitur-fitur tersebut. Dalam kombinasi ini, output dari lapisan sebelumnya dalam CNN diambil sebagai fitur input untuk SVM (Agarap, 2017).

Pada penelitian (Agustin et al., 2022; Koklu et al., 2022; Tao & Wei, 2022) menunjukkan bahwa kombinasi CNN-SVM dapat menghasilkan hasil yang lebih baik daripada menggunakan CNN atau SVM secara terpisah. Dalam beberapa kasus, SVM dapat membantu meningkatkan akurasi dan mengurangi overfitting yang mungkin terjadi dalam CNN. Namun, perlu dicatat bahwa efektivitas kombinasi ini tergantung pada dataset dan tugas yang sedang dihadapi.

Menyiasati hal tersebut (Koklu et al., 2022) melakukan penelitian dengan menggabungkan dua metode klasifikasi CNN dengan SVM untuk mengklasifikasi daun anggur dengan yang diklasifikasikan menjadi 5 kelas. Akurasi yang didapatkan yaitu sebesar 97.6 % dengan menggunakan arsitektur MobileNetv2. Pada penelitian (Tao & Wei, 2022) melakukan penelitian dengan menggabungkan dua metode klasifikasi CNN dengan SVM untuk mengklasifikasikan gulma pada musim dingin dengan yang diklasifikasikan menjadi 3 kelas. Akurasi yang didapatkan yaitu 92.1 % dengan menggunakan arsitektur VGG. Pada penelitian (Agustin et al., 2022) yang dimana penelitian ini menggunakan 4 kelas untuk mendeteksi otomatis non-proliferative diabetic retinopathy. Akurasi yang didapatkan yaitu sebesar 99.77% dengan menggunakan arsitektur Resnet50. Secara keseluruhan, kombinasi CNN dan SVM adalah pendekatan yang menjanjikan untuk meningkatkan akurasi dalam tugas-tugas pengenalan pola dan klasifikasi. Namun, seperti halnya dengan pendekatan lain dalam machine learning, penting untuk melakukan eksperimen dan evaluasi yang cermat terhadap data yang spesifik sebelum mengadopsi metode ini dalam suatu aplikasi nyata.

Dari penelitian diatas menjelaskan bahwa pada penelitian-penelitian sebelumnya yang menggunakan algoritma kombinasi CNN dan SVM bisa disimpulkan hasil akurasinya lebih baik daripada yang hanya menggunakan algoritma CNN. Untuk mengembangkan penelitian klasifikasi ras kucing dengan menggunakan algoritma CNN dan kombinasi CNN-SVM, maka diperlukan suatu penelitian yang membahas tentang Peningkatan Performa Klasifikasi Ras Kucing

menggunakan Hyperparameter Tuning Convolutional Neural Network Arsitektur Xception.

1.2. Rumusan Masalah

Bagian ini memuat penjelasan tentang permasalahan sehingga memerlukan solusi penelitian. Permasalahan yang diuraikan dalam latar belakang masalah dirumuskan kembali secara tegas dan jelas dalam bentuk poin-poin yang terinci yang berisi masalah-masalah yang akan dikaji pada penelitian.

a. Apakah algoritma CNN dapat meningkatkan *accuracy, precision, recall,* dan *f1-score* untuk mengklasifikasi ras kucing ?
b. Apakah algoritma kombinasi CNN-SVM dapat meningkatkan meningkatkan *accuracy, precision, recall,* dan *f1-score* untuk mengklasifikasi ras kucing ?
c. Apakah terdapat perbedaan kinerja dari algoritma CNN dengan kombinasi CNN-SVM?

1.3. Batasan Masalah

Bagian ini memuat penjelasan tentang:

a. Metode yang digunakan untuk klasifikasi ras kucing adalah Convutional Neural Network (CNN) dan Support Vector Machine (SVM)
b. Menggunakan *Google Colab Pro* untuk mengimplementasikan
c. Dataset menggunakan 12 jenis ras kucing yaitu Abyssinian, Bengal, Bombay, Birman, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx dengan jumlah dataset sebanyak 2.400

gambar. Dataset diambil dari Oxford-IIIT Pet Dataset (https://www.robots.ox.ac.uk/~vgg/data/pets/).

d. Augmentasi data yang digunakan *adalah rescale, rotation range, zoom range, width and height shift range, fill mode,* dan *horizontal flip*.
e. Arsitektur menggunakan Inception-V3 dan Xception.
f. Optimasi menggunakan RMSprop dengan learning rate 0.0001.
g. Evaluasi model klasifikasi menggunakan *confusion matrix* dan *classication report*.
h. Perbandingan yang dilakukan dalam penelitian ini dari segi nilai akurasi dan performa.

1.4. Tujuan Penelitian

Bagian ini memuat penjelasan secara spesifik:

a. Mengetahui tingkat *accuracy, precision, recall,* dan *f1-score* pada algoritma CNN untuk mengklasifikasikan ras kucing.
b. Mengetahui tingkat *accuracy, precision, recall,* dan *f1-score* pada algoritma kombinasi CNN dan SVM untuk mengklasifikasikan ras kucing
c. Mengetahui adanya perbedaan kinerja dari algoritma CNN dengan kombinasi CNN-SVM

1.5. Manfaat Penelitian

Bagian ini memuat penjelasan tentang:

a. Mengklasifikasikan Ras Kucing menggunakan Algoritma CNN dengan kombinasi CNN-SVM.
b. Mengetahui kelebihan dan kekurangan antara Algoritma CNN dengan Kombinasi CNN-SVM
c. Mengetahui perbedaan kinerja antara Algoritma CNN dengan kombinasi CNN-SVM
d. Dapat dijadikan referensi untuk penelitian lain dan bisa dijadikan wawasan untuk pengembangan penelitian selanjutnya.

1.6. Hipotesa

Secara teoretis, Support Vector Machine (SVM) sangat efektif untuk klasifikasi, baik untuk dua kelas maupun multi-kelas. Berdasarkan studi-studi sebelumnya yang mengeksplorasi kombinasi antara Convolutional Neural Networks (CNN) dan SVM, hasil yang diperoleh menunjukkan variasi yang signifikan. Arsitektur serta proses optimasi yang diterapkan pada CNN memiliki pengaruh yang substansial terhadap tingkat akurasi yang dicapai.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian tentang Klasifikasi Ras Kucing telah diterapkan oleh beberapa peneliti. Namun, dalam penelitian tersebut terdapat perbedaan baik metodologinya maupun hasilnya. Pada penelitian (Afif et al., 2020) telah melakukan training dan testing dengan menggunakan dataset Oxford-IIIT Pet Dataset dengan jumlah 2.393 gambar dengan 12 kelas. Kelas-kelas tersebut adalah Abyssinian, Bengal, Birman, Bombay, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx. Pada peneltian ini menggunakan algoritma CNN dengan arsitektur VGG16, Inception-V3, ResNet50, dan Xception serta optimasi menggunakan RMSprop dengan learning rate 0.0001. Hasil akurasi testing yang didapatkan masing-masing arsitektur yaitu 60.85%, 84,94%, 71,39%, dan 93,75%.

Masih sama dengan objek ras kucing, pada penelitian (Imanuel & Setiabudi, 2022) penelitian ini menunjukkan hasil akurasi tertinggi yang dihasilkan dalam melakukan mendeteksi ras kucing yang menggunakan dataset dari The Oxford-IIIT Pet Dataset dengan jumlah 12 kelas Abyssinian, Bengal, Birman, Bombay, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx dengan total gambar 2.398 adalah 89.58%. Pada penelitian ini menggunakan Algoritma CNN dengan arsitektur Xception dan learning rate 0.0001.

Selanjutnya pada penelitian (X. Zhang et al., 2019) menggunakan dataset dari COCO datasets dengan jumlah dataset 80 kategori (jenis ras kucing). Penelitian

ini bertujuan untuk membandingkan berbagai model deep learning dan menyajikan aplikasi Android yang digunakan untuk memprediksi lokasi dan ras kucing tertentu menggunakan kamera ponsel. Penelitian ini berfokus pada pengembangan model pendeteksian kucing dengan deep learning dan pengiriman melalui aplikasi seluler. Aplikasi ini telah dilatih untuk mengenali 14 jenis kucing dengan akurasi rata-rata 81,74%. Dalam melakukan ini, penelitian membandingkan enam model populer dengan pilihan terakhir SSD Mobilenet_v1 FPN.

Selanjutnya pada penelitian (Qatrunnada et al., 2022) menggunakan dataset dari Kaggle (https://www.kaggle.com/ma7555/cat-breeds-dataset). Penelitian ini bertujuan untuk mengenali dan melatih dengan menggunakan CNN multi-object yang dimana pada penelitian sebelumnya adalah citra objek tunggal karena memiliki hasil klasifikasi yang baik pada penelitian-penelitian sebelumnya. Hasil pengujian diukur dengan menggunakanfusion matrix, diperoleh presisi, recall, skor f1 dan akurasi 100% pada citra multi objek dengan 2 objek dan 3 objek. Pada citra dengan 4 objek diperoleh nilai presisi, recall, f1 dan akurasi sebesar 89%, 87%, 87% dan 95%. Sedangkan nilai presisi, recall, skor f1 dan akurasi pada citra dengan 5 objek masing-masing mendapatkan 87%, 86%, 86% dan 94%. Penelitian ini menggunakan 5 ras kucing dengan setiap ras memiliki 200-3200 gambar untuk pelatihan. Jadi, Semakin banyak objek pada suatu citra maka nilai akurasi akan mulai berkurang. Begitu juga sebaliknya, semakin sedikit objek maka nilai akurasi akan semakin meningkat. Waktu komputasi juga dipengaruhi oleh banyaknya objek yang terdapat pada citra. Semakin banyak objek yang ada, semakin lama waktu komputasi.

Selanjutnya pada penelitian (Dai et al., 2021) menggunakan dataset dari Kaggle Dogs vs Cats. Penelitian ini bertujuan pada akurasi rendah dari Mask R-CNN tradisional yang diterapkan pada segmentasi gambar kucing yang berbeda, algoritma pengenalan dan segmentasi Mask R-CNN yang ditingkatkan diusulkan. Hasil eksperimen menunjukkan bahwa metode ini mencapai akurasi segmentasi 87,54% pada dataset deteksi klasifikasi anjing dan kucing. Mengusulkan metode Mask R-CNN yang ditingkatkan dan menerapkannya pada tugas segmentasi deteksi kucing. Untuk kekhususan tugas spesies individu kucing yang berbeda di berbagai lingkungan. Selama proses pengujian, ditemukan bahwa efek segmentasi daerah tepi kucing individu tidak memuaskan, misalnya, daerah perbatasan gambar kucing individu tidak sepenuhnya dikelilingi oleh bingkai deteksi, sehingga menghasilkan kesalahan tertentu, dan bagaimana meningkatkan efek ini akan menjadi tujuan utama penelitian di masa depan.

Terakhir, pada penelitian (Mahardi et al., 2020) menggunakan dataset dari CDC. Penelitian ini bertujuan membangun pengklasifikasi gambar untuk mengenali berbagai ras anjing dan kucing (CDC) menggunakan model VGG yang disesuaikan. Dua model umum, VGG16 dan VGG19 digunakan untuk membangun classifier. Model yang dihasilkan dari VGG16 memiliki akurasi pelatihan 98,47%, akurasi validasi 98,56%, dan akurasi pengujian 83,68%. Model dari VGG19 memiliki akurasi pelatihan 98,59%, akurasi validasi 98,56%, dan akurasi pengujian 84,07%.

Dengan adanya tinjauan Pustaka yang telah ditinjau, maka penulis ingin membuat penelitian tentang klasifikasi ras kucing. Oleh karena itu diperlukan suatu

sistem yang dapat mengidentifikasi jenis ras kucing. Beberapa peneliti telah melakukan penelitian untuk mendeteksi ras kucing, tetapi hasil akhir dari penelitian ini tidak terlalu akurat. Memang, ada beberapa ras kucing yang memiliki perbedaan minimal satu sama lain, sehingga sulit untuk membedakannya (Afif et al., 2020; Dai et al., 2021; Imanuel & Setiabudi, 2022; Mahardi et al., 2020; Qatrunnada et al., 2022; X. Zhang et al., 2019).

2.2. Keaslian Penelitian

Tabel 2. 1 Matriks literature review Peningkatan Performa Klasifikasi Ras Kucing menggunakan Hyperparameter Tuning Convolutional Neural Network Arsitektur Xception

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	Klasifikasi Ras pada Kucing menggunakan Algoritma Convolutional Neural Network (CNN)	Muhammad Afif Amanullah Fawwaz, Kurniawan Nur Ramadhani, Febriyanti Sthevanie, E-Proceeding of Engineering, 2021	Mencari model yang terbaik antara VGG16, InceptionV3, ResNet50, dan Xception. Evaluasi dilakukan dengan menggunakan metrik akurasi.	Model Xception + Transfer Learning + Fine Tuning adalah model yang terbaik. Model ini menghasilkan akurasi, precision, recall, dan f1-score dengan nilai 93.75%, 93.74%, 93.56%, dan 93.64%.	Dalam proses pelatihan, disarankan menggunakan perangkat berkapasitas tinggi karena model yang diuji memerlukan sumber daya komputasi yang besar.	Pada penelitian kami menggunakan algoritma kombinasi CNN dan SVM dan serta optimizer Adamax. Hasil akhir akan diuji secara realtime
2	Penerapan Convolutional Neural Network dengan Pre-Trained Model Xception untuk Meningkatkan Akurasi dalam Mengidentifikasi Jenis Ras Kucing	Abraham Imanuel, Djoni Haryadi Setiabudi, Jurnal Infra, 2022	Sistem identifikasi ras kucing dibutuhkan untuk memilih ras yang cocok. sebelumnya dengan SSD Mobilenet_v1 FPN mencapai akurasi 81.74%.	Hasil yang terbaik dengan akurasi 89.59% pada pengujian yang ketujuh dengan menggunakan dataset dari Oxford-IIIT Pet dan pre-procesing menggunakan resize 299x299 dan colorspace RGB.	Untuk memperbaiki penelitian, gunakan dataset yang beragam dengan perbedaan kelas yang signifikan dan tambahkan lapisan Dropout untuk meningkatkan kinerja model.	Pada penelitian kami menggunakan algoritma kombinasi CNN dan SVM. Ada perbandingan untuk arsitekturnya yaitu Inception-V3 dengan Xception.

Tabel 2. 1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
3	A Mobile Application for Cat Detection and Breed Recognition Based on Deep Learning	Xiaolu Zhang, Luyang Yang, dan Richard Sinnott, AI4Mobile 2019 - 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile, 2019	Penelitian ini membandingkan berbagai model deep learning dan menyajikan aplikasi Android yang digunakan untuk memprediksi lokasi dan ras kucing tertentu menggunakan kamera ponsel.	Penelitian ini mengembangkan model deteksi kucing berbasis pembelajaran mendalam dengan integrasi aplikasi seluler, dilatih untuk mengidentifikasi 14 jenis kucing dengan akurasi 81,74%. Studi ini melakukan perbandingan antara enam model populer, memilih SSD Mobilenet_v1 FPN sebagai yang terbaik.	Perbaikan model melibatkan analisis gambar dengan fitur dan lapisan yang lebih efisien untuk mengurangi waktu pemrosesan. Namun, penyesuaian fitur dan pengurangan lapisan dapat berdampak pada akurasi, sehingga menyeimbangkan waktu pemrosesan dengan akurasi menjadi tantangan.	Penelitian kami akan berfokus pada mengklasifikasikan ras kucing dengan menggunakan algoritma kombinasi CNN-SVM. Namun, dalam penelitian kami terdapat pengujian secara realtime yang dapat digunakan sebagai perbandingan dengan penelitian ini.

Tabel 2. 1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
4	Cat Breeds Classification using Convolutional Neural Network for Multi-Object Image	Naura Qafrunnada, M. Fachrurrozi, Alvi Syahrini Utami, Sriwijaya Journal of Informatics and Applications, 2022	Bertujuan untuk mengenali dan melatih dengan menggunakan CNN multi-object yang dimana pada penelitian sebelumnya adalah citra objek tunggal karena memiliki hasil klasifikasi yang baik pada penelitian-penelitian sebelumnya.	Hasil pengujian diukur dengan menggunakanfusion matrix, diperoleh presisi, recall, skor f1 dan akurasi 100% pada citra multi objek dengan 2 objek dan 3 objek. Pada citra dengan 4 objek diperoleh nilai presisi, recall, f1 dan akurasi sebesar 89%, 87%, 87% dan 95%. Sedangkan nilai presisi, recall, skor f1 dan akurasi pada citra dengan 5 objek masing-masing mendapatkan 87%, 86%, 86% dan 94%	Penelitian ini melibatkan 5 ras kucing dengan 200-3200 gambar per ras untuk pelatihan, menunjukkan bahwa jumlah objek dalam citra berbanding terbalik dengan akurasi dan berbanding lurus dengan waktu komputasi.	Pada penelitian kami menggunakan algoritma kombinasi CNN dan SVM serta dataset yang digunakan pada penelitian ini adalah dari Oxford-IIIT Pet yang memiliki 12 kelas.

Tabel 2. 1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
5	Mask R-CNN-based Cat Class Recognition and Segmentation	Yile Dai, Yunqing Liu, Siyuan Zhang, Journal of Physics: Conference Series, 2021	Bertujuan pada akurasi rendah dari Mask R-CNN tradisional yang diterapkan pada segmentasi gambar kucing yang berbeda, algoritma pengenalan dan segmentasi Mask R-CNN yang ditingkatkan diusulkan.	Eksperimen mencapai akurasi segmentasi 87,54% pada dataset anjing dan kucing menggunakan Mask R-CNN yang ditingkatkan untuk segmentasi spesies kucing dalam beragam lingkungan.	Selain itu, peningkatan pada segmentasi tepi kucing menjadi prioritas utama untuk penelitian selanjutnya, akibat kurangnya keakuratan dalam melingkupi tepi gambar kucing yang menyebabkan kesalahan deteksi.	Pada penelitian kami menggunakan metode perbandingan kombinasi CNN dan SVM. Dataset diambil dari Oxford-IIIT Pet yang memiliki 12 kelas.

Tabel 2. 1 Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
6	Images Classification of Dogs and Cats using Fine-Tuned VGG Models	Mahardi, I-Hung Wang, Kuang-Chyi Lee, Shinn-Liang Chang, 2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020, ECICE 2020, 2020	Membangun pengklasifikasi gambar untuk mengenali berbagai ras anjing dan kucing (CDC) menggunakan model VGG yang disesuaikan.	Dua model umum, VGG16 dan VGG19 digunakan untuk membangun classifier. Model yang dihasilkan dari VGG16 memiliki akurasi pelatihan 98,47%, akurasi validasi 98,56%, dan akurasi pengujian 83,68%. Model dari VGG19 memiliki akurasi pelatihan 98,59%, akurasi validasi 98,56%, dan akurasi pengujian 84,07%.	Percobaan mengembangkan pengklasifikasi gambar anjing dan kucing menggunakan VGG16 dan VGG19, dengan akurasi validasi kedua model sebesar 98,56%. VGG19 lebih unggul, dengan akurasi pelatihan 98,59% dan pengujian 84,07%, dibandingkan VGG16 yang memiliki akurasi pelatihan 98,47% dan pengujian 83,68%. Kekurangan dan saran belum dijelaskan oleh penulis.	Pada penelitian kami menggunakan metode kombinasi CNN dan SVM. Pada arsitektur menggukanan Inception-V3 dan Xception. Dataset menggunakan dari Oxford-IIIT Pet yang terdiri dari 12 kelas.

2.3. Landasan Teori

2.3.1. Klasifikasi

Secara umum, klasifikasi berarti proses pengelompokan. Dalam istilah ilmu data, klasifikasi kemudian dipahami sebagai proses pengelompokan data ke dalam beberapa kategori untuk memudahkan pengolahan dan analisis (Audebert et al., 2019). Proses classification pada dasarnya dilakukan agar analisis data menjadi lebih mudah dan tentunya memberikan hasil yang akurat. Agar bisa memberikan suatu informasi yang bermanfaat, data memang memerlukan proses panjang (Audebert et al., 2019).

Dalam konteks citra, klasifikasi adalah tugas dalam ilmu pengolahan citra dan pembelajaran mesin di mana komputer atau algoritma pembelajaran mesin diajarkan untuk mengidentifikasi dan mengklasifikasikan objek atau entitas dalam sebuah citra digital ke dalam kategori atau kelas tertentu. Tujuan utama dari image classification adalah untuk mengambil sebuah citra dan menentukan kategori atau kelas yang mewakili objek atau fitur yang ada dalam citra tersebut (Nath et al., 2014).

2.3.2. Preprocessing Data

Dalam konteks image classification, preprocessing data mengacu pada serangkaian langkah atau teknik yang diterapkan pada citra (gambar) sebelum citra tersebut digunakan sebagai input untuk model pembelajaran mesin atau jaringan saraf tiruan (neural networks) untuk tugas klasifikasi. Tujuan dari preprocessing data pada image classification adalah untuk mempersiapkan citra sehingga

informasi yang penting dapat diekstraksi dengan lebih efektif dan akurat oleh model.

2.3.3. Augmentasi Data

Augmentasi data pada citra (image data augmentation) adalah teknik yang digunakan dalam pemrosesan citra dan machine learning untuk menciptakan variasi baru dari citra-citra pelatihan dengan mengaplikasikan transformasi atau modifikasi tertentu pada citra asli. Tujuan utama dari augmentasi data adalah untuk meningkatkan keragaman data pelatihan, mencegah overfitting, dan meningkatkan kemampuan model untuk menggeneralisasi dengan baik terhadap data yang belum pernah dilihat sebelumnya. Dengan cara ini, augmentasi data membantu meningkatkan kinerja model dalam tugas-tugas seperti image classification (Shorten & Khoshgoftaar, 2019).

2.3.4. Machine Learning

Machine learning adalah cabang dari artificial intelligence yang fokus pada pengembangan algoritma komputer yang dapat memungkinkan sistem untuk mempelajari pola atau perilaku dari data (biasanya data historis) dan mengambil keputusan atau membuat prediksi berdasarkan pemahaman yang diperoleh dari data tersebut, tanpa perlu pemrograman eksplisit untuk setiap tugas tertentu (Mahesh, 2020).

2.3.5. Supervised Learning

Supervised learning adalah jenis machine learning yang dimana model diinstruksikan atau diberi pengawasan dengan data pelatihan yang sudah diberi label. Dalam kata lain, model learning dari contoh-contoh data yang sudah memiliki label atau kelas yang benar. Tujuan utama dari supervised learning adalah untuk menghasilkan model yang dapat memetakan input ke output yang benar, atau dengan kata lain, mempelajari hubungan antara input dan output berdasarkan data pelatihan yang ada. Beberapa contoh tugas supervised learning meliputi klasifikasi (mengelompokkan data ke dalam kategori atau kelas yang sudah ditentukan), regresi (memprediksi nilai berkelanjutan), dan deteksi anomali (mengidentifikasi data yang tidak biasa) (Bradley C. Love, 2022).

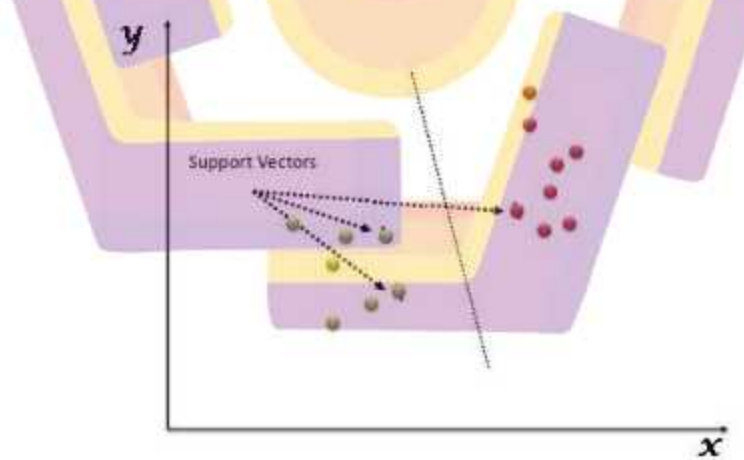
2.3.6. Unsupervised Learning

Unsupervised learning adalah jenis machine learning yang dimana model berusaha untuk menemukan pola, struktur, atau hubungan dalam data tanpa bantuan label. Dalam kata lain, tidak ada pengawasan atau petunjuk yang diberikan kepada model. Tujuan utama dari unsupervised learning adalah untuk mengelompokkan atau menggali struktur dalam data yang dapat membantu dalam pemahaman data lebih lanjut atau untuk tugas seperti reduksi dimensi. Beberapa contoh tugas unsupervised learning meliputi pengelompokan (clustering), reduksi dimensi (seperti Principal Component Analysis atau PCA), dan pencarian asosiasi (seperti analisis apriori dalam data mining) (Bradley C. Love, 2022).

2.3.7. Support Vector Machine

Support Vector Machine (SVM) adalah salah satu algoritma machine learning supervised learning yang dapat digunakan untuk klasifikasi dan regresi. Cara kerja SVM adalah berdasarkan SRM atau Structural Risk Reduction dirancang untuk memproses data dalam hyperplane yang mengklasifikasikan ruang input menjadi dua kelas. Teori SVM dimulai dengan sekelompok kasus linier yang dapat diuraikan oleh hyperplane dan dibagi lagi berdasarkan kelas (Cervantes et al., 2020).

Konsep SVM dimulai dengan masalah klasifikasi dua kelas yang membutuhkan set pelatihan positif dan negatif. SVM akan mencoba untuk mendapatkan hyperplane (pembatas) terbaik untuk memisahkan dua lapisan dan memaksimalkan margin dari dua lapisan (Ebrahimi et al., 2022).



Gambar 2. 1. Support Vector Machine

2.3.8. Deep Learning

Deep learning adalah cabang dari machine learning dan kecerdasan buatan (artificial intelligence) yang berfokus pada pengembangan dan pelatihan model jaringan saraf tiruan (neural networks) yang dalam (dikenal sebagai deep neural networks) untuk memahami dan menggeneralisasi pola yang rumit dari data yang lebih kompleks. Deep learning menggunakan arsitektur jaringan saraf tiruan yang terdiri dari banyak lapisan (deep layers) yang berisi neuron atau unit pemrosesan informasi yang saling terhubung (Lecun et al., 2015).

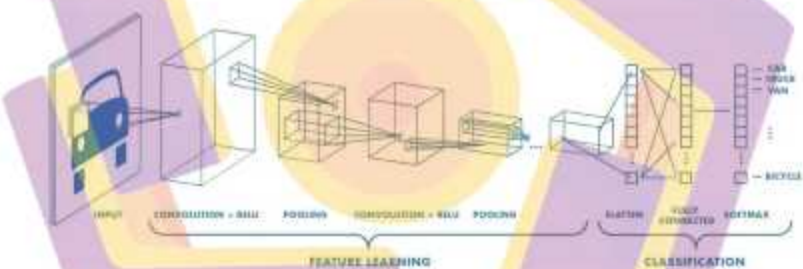
Deep learning mengacu pada penggunaan jaringan saraf tiruan yang memiliki banyak lapisan (biasanya lebih dari satu atau dua) antara lapisan input dan lapisan output. Jaringan saraf ini disebut deep neural network. Deep learning memungkinkan model untuk secara otomatis memahami fitur-fitur atau representasi yang relevan dari data yang diberikan, tanpa perlu ekstraksi fitur manual oleh manusia (Miikkulainen et al., 2018). Deep learning sering memerlukan jumlah data yang besar untuk pelatihan yang efektif. Data yang melimpah memungkinkan model untuk mengidentifikasi pola yang lebih kompleks dan meningkatkan kemampuannya untuk menggeneralisasi. Deep learning telah digunakan dalam berbagai aplikasi, termasuk pengenalan wajah, klasifikasi citra, natural language processing (NLP), rekomendasi, permainan video, kendaraan otonom, dan banyak lagi (Shinde & Shah, 2018).

Meskipun deep learning memiliki kemampuan untuk menghasilkan hasil yang luar biasa dalam banyak tugas, pemeliharaan, pelatihan, dan tuning model

yang rumit memerlukan sumber daya komputasi yang besar dan keahlian dalam pengelolaan model (M. Zhang et al., 2020).

2.3.9. Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu jenis neural network yang biasa digunakan pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali object pada sebuah image (Albawi et al., 2018). Secara garis besar CNN tidak jauh beda dengan neural network biasanya. CNN terdiri dari neuron yang memiliki weight, bias dan activation function seperti yang sudah kita pelajari pada part sebelumnya (Q. Zhang et al., 2019).



Gambar 2. 2. Convolutional Neural Network

Pada gambar 2.2 menjelaskan bahwa sebuah model CNN pada dasarnya memiliki beberapa layer untuk proses penyelesaian klasifikasi pada gambar sebagai berikut: (D. Yu et al., 2014)

A. Layer Input

Layer input dalam CNN adalah lapisan pertama dalam arsitektur CNN yang berfungsi sebagai pintu masuk atau antarmuka yang menerima citra atau data input yang akan diproses oleh jaringan. Fungsi utama dari layer input adalah menerima

citra-citra atau data input dalam format yang sesuai dan mengirimkannya ke lapisan-lapisan berikutnya dalam jaringan untuk diproses.

Layer input memiliki dimensi atau ukuran yang sesuai dengan data input. Misalnya, jika jaringan CNN digunakan untuk mengklasifikasikan citra 224x224 piksel, maka layer input akan memiliki dimensi 224x224 (lebar x tinggi). Jika citra adalah citra berwarna (RGB), maka layer input akan memiliki tiga saluran warna (merah, hijau, biru) yang mewakili komponen warna citra. Sedangkan jika citra adalah citra hitam putih (grayscale), maka layer input akan memiliki satu saluran warna yang tidak mewakili komponen warna citra.

B. Convolutional

Convolutional layer adalah operasi matematis yang digunakan dalam CNN untuk menggabungkan citra input dengan filter atau kernel. Layer melibatkan perhitungan dot product antara bagian-bagian kecil dari citra input dan filter yang bergerak melintasi citra. Hasil konvolusi menghasilkan fitur-fitur yang dapat digunakan untuk mengenali pola atau fitur-fitur dalam citra. Operasi konvolusi digunakan untuk mengekstraksi fitur-fitur berhierarki dari citra, seperti tepi, sudut, dan tekstur, dan untuk menghasilkan representasi yang semakin kompleks melalui lapisan-lapisan konvolusi yang dalam.

C. ReLU

ReLU atau *Rectified Linear Activation* adalah fungsi aktivasi yang digunakan dalam CNN. Fungsi ini sederhana, yaitu f(x) = max(0, x), artinya jika nilai input (x) lebih kecil dari nol, maka keluarannya adalah nol, dan jika lebih besar dari nol, keluarannya adalah nilainya sendiri. ReLU memasukkan non-linearitas ke

dalam jaringan CNN, yang penting untuk memungkinkan jaringan untuk memahami pola yang kompleks dalam data. Ini membantu mengatasi masalah vanishing gradient yang bisa terjadi dalam jaringan yang dalam.

D. Pooling

Lapisan pooling (bMax Pooling dan Global Average Pooling) digunakan untuk mengurangi dimensi spasial citra dan mengurangi jumlah parameter dalam model. Ini mengambil nilai maksimum atau rata-rata dari sekelompok piksel dalam citra. Max pooling adalah jenis operasi pooling yang paling umum digunakan dalam CNN. Operasi ini melibatkan pemilihan nilai maksimum dari sekelompok piksel atau nilai dalam matriks di setiap area yang disebut jendela pooling. Sedangkan, Global Average Pooling (GAP) adalah jenis operasi pooling khusus yang digunakan dalam beberapa arsitektur CNN. Dalam GAP, untuk setiap fitur dalam lapisan, nilai rata-rata dihitung dari semua nilai di seluruh area atau peta fitur. Tujuan dari jenis kedua pooling tersebut adalah membantu mengurangi jumlah parameter dalam model dan mengurangi beban komputasi. Ini juga membantu dalam mempertahankan informasi penting dan menekan noise dalam citra.

E. Flatten

Lapisan Flatten adalah lapisan yang digunakan untuk mengubah struktur tiga dimensi (biasanya hasil dari lapisan konvolusi dan pooling) menjadi vektor satu dimensi. Lapisan ini diperlukan karena lapisan-lapisan konvolusi dan pooling menghasilkan fitur-fitur berdimensi tinggi, tetapi lapisan-lapisan penuh terhubung (fully connected) memerlukan input berbentuk vektor satu dimensi. Jika hasil dari lapisan konvolusi adalah matriks 3D dengan dimensi (tinggi, lebar, kedalaman),

lapisan Flatten akan mengubahnya menjadi vektor satu dimensi dengan panjang (tinggi * lebar * kedalaman).

F. Fully Connected

Lapisan Fully Connected adalah lapisan dalam CNN yang menghubungkan setiap neuron di lapisan ini dengan setiap neuron di lapisan sebelumnya, menciptakan hubungan terhubung sepenuhnya (fully connected). Lapisan ini bertanggung jawab untuk menggabungkan fitur-fitur yang telah diekstraksi oleh lapisan konvolusi dan pooling ke dalam representasi yang lebih abstrak dan kompleks. Ini juga bertanggung jawab untuk melakukan klasifikasi atau regresi berdasarkan representasi tersebut. Jika output dari lapisan Flatten adalah vektor satu dimensi, lapisan Fully Connected dapat mengandung beberapa neuron untuk tugas klasifikasi. Jaringan ini mempelajari hubungan antara fitur-fitur tersebut dan menghasilkan output yang sesuai dengan kelas-kelas yang mungkin.

G. Softmax

Lapisan Softmax adalah lapisan terakhir dalam banyak model CNN yang digunakan dalam tugas klasifikasi multikelas. Ini adalah fungsi aktivasi yang mengubah keluaran dari lapisan Fully Connected menjadi distribusi probabilitas atas kelas-kelas yang mungkin. Lapisan ini membantu menghasilkan probabilitas untuk setiap kelas sehingga kita dapat menentukan kelas mana yang paling mungkin untuk data input yang diberikan. Jika model CNN digunakan untuk mengklasifikasikan gambar menjadi beberapa kategori (kelas), lapisan Softmax akan mengubah output dari lapisan Fully Connected menjadi probabilitas bahwa gambar tersebut termasuk dalam setiap kategori.

2.3.10. Hyperparameter Tuning

Hyperparameter tuning dalan CNN adalah proses kritis dalam pembelajaran mesin, khususnya dalam pengembangan model deep learning seperti CNN. Hiperparameter adalah parameter yang nilai-nya ditetapkan sebelum proses pelatihan model dan tidak berubah selama pelatihan berlangsung. Proses penyetelan ini melibatkan penyesuaian variabel-variabel seperti laju pembelajaran (learning rate), jumlah epoch, dan ukuran batch, yang semuanya memiliki dampak signifikan terhadap kinerja model (Awang Pona & K K, 2021).

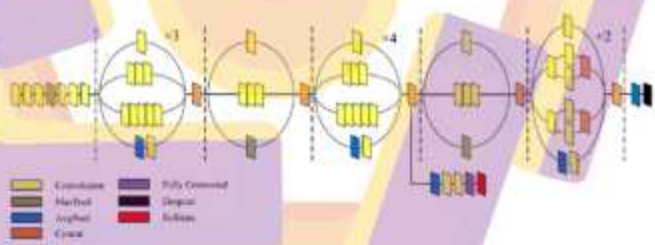
2.3.11. Transfer Learning

Transfer learning adalah teknik yang umum digunakan dalam Convolutional Neural Network (CNN) di mana model yang telah dilatih sebelumnya pada tugas tertentu (misalnya, pengenalan citra pada dataset besar) digunakan kembali sebagai titik awal atau basis untuk melatih model baru pada tugas serupa atau berbeda. Dengan kata lain, model yang telah memahami fitur-fitur umum dalam data dari tugas sebelumnya dapat diadaptasi untuk tugas baru (Shaha & Pawar, 2018).

A. Inception-V3

Inception-V3 adalah model arsitektur neural network yang dirancang khusus untuk masalah pengenalan gambar (Dong et al., 2020). Model meniru proses multi-lapisan pengenalan gambar oleh manusia, pemrosesan awal sel-sel tertentu di korteks serebral, menemukan tepi dan arah bentuk, menentukan bentuk secara abstrak (seperti lingkaran dan kotak) dan keputusan abstrak lebih

lanjut (seperti menilai objek sebagai balon). CNN biasanya mencakup lima lapisan, yaitu input, convolutional, pooled, fully connected, dan output. Inception V3 adalah arsitektur convolutional neural network (CNN) yang dikembangkan dari GoogleNet (Inception) versi sebelumnya dan Inception V2. Googlenet pertama kali diperkenalkan pada tahun 2014 dan dinobatkan sebagai Arsitektur Terbaik dalam kompetisi ImageNet Large Scale Image Recognition Challenge (ILSVRC) 2014. Inception memperkenalkan teknik baru dalam arsitektur. CNN, khususnya pengganda lapisan konvolusional multilayer dengan ukuran kernel yang lebih kecil (Dong et al., 2020). Penggunaan teknik ini mengurangi jumlah parameter dengan membagi bobot pada beberapa kelas. Selain itu, menggunakan teknik ini juga memiliki potensi untuk menciptakan arsitektur yang lebih dalam tanpa meningkatkan beban komputasi.



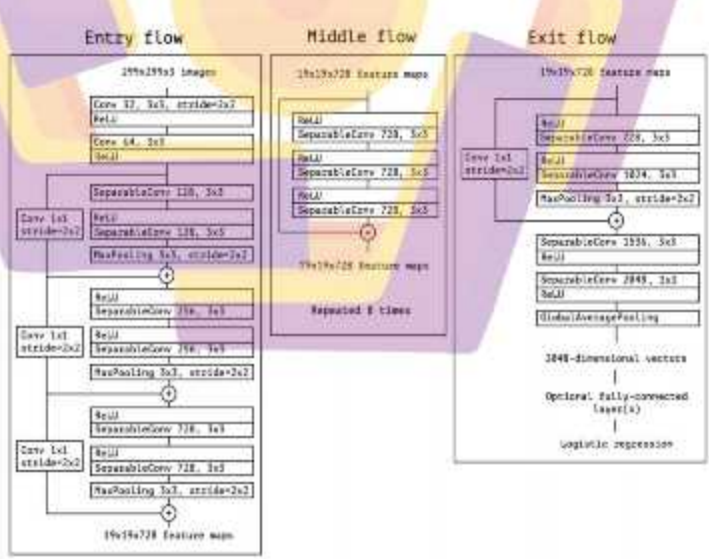
Gambar 2. 3. Inception-V3

B. Xception

Xception merupakan deep convolutional neural network architecture yang terinspirasi dari Model Inception, hanya saja di Model Xception, modul Inception diganti dengan depthwise separable convolutions. Depthwise separable convolution terdiri dari 2 layers, yaitu depthwise convolutions dan

pointwise convolutions. Depthwise convolutions digunakan untuk mengaplikasikan sebuah single filter pada setiap input channel. Pointwise convolutions, yang merupakan sebuah 1 x 1 convolutional yang sederhana, digunakan untuk membuat sebuah kombinasi linear dari output depthwise layer (Chollet, 2017).

Arsitektur dari Xception memiliki 36 convolutional layers yang membentuk feature extraction base dari sebuah network. Dengan Covolutional layer tang berjumlah 36 ini kemudian direstrukturisasikan menjadi 14 modul yang dimana yang disetiap modulnya memiliki linear residual connections di sekitarnya, kecuali modul pertama dan terakhir. Konklusinya, Xception ini merupakan stack linear yang terdiri dari depthwis separable convolution layers' yang dilengkapi dengan residual connections (Chollet, 2017).



Gambar 2. 4. Xception

2.3.12. Fine-tuning

Fine-tuning adalah teknik dalam machine learning, terutama dalam konteks deep learning, di mana model yang telah dilatih sebelumnya (model dasar atau pre-trained model) digunakan sebagai titik awal atau dasar untuk melatih model baru pada tugas yang serupa atau terkait. Proses fine-tuning ini mencakup penyesuaian parameter-model dasar agar lebih cocok dengan data atau tugas baru yang spesifik (Kading et al., 2017).

Tujuan utama fine-tuning adalah untuk memanfaatkan pengetahuan yang sudah ada dalam model dasar dan mengadaptasikannya ke tugas atau data baru. Ini dapat menghemat waktu dan sumber daya yang diperlukan untuk melatih model dari awal. Fine-tuning sering digunakan ketika kita memiliki dataset yang terbatas atau ketika kita ingin meningkatkan kinerja model pada tugas spesifik. Dengan mengubah model dasar, kita dapat mencapai kinerja yang lebih baik pada tugas baru daripada memulai dari awal.

Selain penyesuaian parameter, fine-tuning juga dapat melibatkan penyesuaian hyperparameter, seperti tingkat pembelajaran (learning rate), jumlah epok pelatihan, dan lainnya, agar sesuai dengan tugas dan data baru.

2.3.13. Confusion Matrix

Confusion Matrix adalah tabel dengan 4 kombinasi berbeda dari nilai prediksi dan nilai actual (Narkhede, 2018). Ada empat istilah yang merupakan representasi hasil proses klasifikasi pada confusion matrix yaitu True Positive (TP),

True Negative (TN), False Positive (FP), dan False Negative (FN) yang bisa dilihat pada tabel 2.2. (Xu et al., 2020).

Tabel 2. 2 Confusion Matrix

Actually / Predicted	Actually Positive	Actually Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

A. Akurasi

Akurasi merupakan seberapa akurat model dalam mengklasifikasikan dengan benar.

$$Accuracy = \frac{TPs+TNs}{TPs+TNs+FPs+FNs} \quad (1)$$

B. Presisi

Presisi merupakan kurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model.

$$Precision = \frac{TPs}{TPs+FPs} \quad (2)$$

C. Recall

Recall merupakan keberhasilan model dalam menemukan kembali sebuah informasi.

$$Recall = \frac{TPs}{TPs+FNs} \quad (3)$$

D. F1-Score

F1-Score merupakan perbandingan rata-rata precision dan recall yang dibobotkan. Accuracy tepat kita gunakan sebagai acuan performansi algoritma jika dataset kita memiliki jumlah data False Negatif dan False Positif yang sangat mendekati (symmetric). Namun jika jumlahnya tidak mendekati, maka sebaiknya kita menggunakan F1-Score sebagai acuan.

$$F1\ Score = \frac{2\ x\ Presisi\ x\ Recall}{Presisi + Recall} \quad (4)$$

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Dalam penelitian ini merupakan penelitian experimental. Penelitian ini yang dilakukan adalah dengan mengkombinasikan algoritma CNN dan SVM untuk klasifikasi ras kucing.

Sifat dari penelitian ini adalah deskriptif yang dimana penelitian ini akan menjelaskan dampak penggunaan metode SVM terhadap CNN untuk klasifikasi ras kucing.

Pendekatan pada penelitian ini adalah pendekatan kuantitatif yang dimana penelitian ini akan menghitung akurasi dan performa pada algoritma kombinasi CNN dan SVM dalam klasifikasi ras kucing. Data yang diambil merupakan dataset ras kucing yang didapat dari Oxford-IIIT Pet. Setelah didapatkan adalah melakukan pembagian dataset seperti data training, testing, dan validasi. Kemudian data tersebut digunakan sebagai dasar untuk menentukan klasifikasi ras kucing.

3.2. Metode Pengumpulan Data

Pengumpulan data adalah kegiatan yang dilakukan pertama kali sebelum melakukan kegiatan analisis data. Data yang diperlukan pada penelitian ini berupa citra ras kucing dengan 12 jenis ras kucing yaitu Abyssinian, Bengal, Bombay, Birman, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx. Metode pengumpulan data yang dilakukan untuk

mendapatkan gambar ras kucing yaitu dengan men-download melalui Oxford-IIIT Pet Dataset.

3.3. Metode Analisis Data

Analisis data dilakukan setelah melakukan pengujian dengan menghitung akurasi dan performa dengan menggunakan algoritma kombinasi CNN dan SVM dalam klasifikasi ras kucing.

3.4. Alur Penelitian

Gambar 3. 1. Alur Penelitian

Alur penelitian dapat dilihat pada gambar 3.1. Alur penelitian dimulai dari pengumpulan data dan berakhir jika klasifikasi ras kucing berhasil dilakukan.

a. Permasalahan

Tahap ini merupakan tahap awal untuk memulai sebuah penelitin dengan melihat permasalahan sebelumnya.

b. Studi Literatur

Setelah menentukan permasalahan, proses selanjutnya adalah mencari informasi mengenai hal yang berhubungan dengan masalah tersebut melalui studi literatur. Studi literatur ini dilakukan dengan membaca penelitian-penelitian sebelumnya yang dianggap relevan dengan permasalahan yang diangkat. Proses ini juga sebagai bahan rujukan penelitian untuk memilih metode yang dianggap sesuai dengan permasalahan.

c. Pengumpulan Data

Data pada penelitian ini adalah berupa citra ras kucing dengan 12 jenis ras kucing yaitu Abyssinian, Bengal, Bombay, Birman, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx. Metode pengumpulan data yang dilakukan untuk mendapatkan gambar ras kucing yaitu dengan mendownload melalui Oxford-IIIT Pet Dataset.

d. Pembagian data (data latih, validasi, dan uji)

Pada proses ini dataset dibagi menjadi data latih, data validasi, dan data uji dengan pembagiannya yang sudah ditentukan.

e. Preprocessing dan Agumentasi

Tahap preprocessing diperlukan untuk memperbaiki gambar. Dalam tahap proses preprocessing ini hanya memerlukan resize yang bertujuan untuk menyamakan ukuran dari gambar yang akan diolah agar cocok untuk menjadi input persegi dalam proses untuk membuat model baik CNN maupun CNN-SVM. Sedangkan dalam tahap augmentasi diperlukan untuk menambah variasi pada semua gambar yang telah diolah pada tahap preprocessing. Augmentasi yang digunakan adalah rescale, rotation, zoom, fill mode, width dan height shift, dan horizontal flip.

f. Skenario penelitian

Pada jenis penelitian ini bersifat eksperimen, maka alur penelitian terdapat langkah skenario. Di dalam skenario terdapat perbandingan dengan menggunakan CNN dengan transfer learning dan fine-tuning dan kombinasi CNN-SVM. Dalam perbandingan tersebut terdapat arsitektur, learning rate, dan activation fuction. Arsitektur yang digunakan yaitu Inception-V3 dan Xception, lalu untuk learning rate yang digunakan yaitu 0.0001, sedangkan untuk activation yang digunakan yaitu softmax. Setelah menentukan arsitektur, learning rate, dan activation fuction, langkah selanjutnya yaitu mengkombinasikan SVM dengan model CNN yang telah ditentukan. Jadi, untuk skenario terdapat empat skenario diantaranya yaitu:

a) CNN dengan arsitektur Inception-V3, transfer learning, dan learning rate 0.0001.

b) CNN dengan arsitektur Inception-V3, transfer learning, fine-tuning, dan learning rate 0.0001.

c) Kombinasi CNN-SVM dengan arsitektur Inception-V3, transfer learning, fine-tuning dan learning rate 0.0001.

d) CNN dengan arsitektur Xception, transfer learning, dan learning rate 0.0001.

e) CNN dengan arsitektur Xception, transfer learning, fine-tuning, dan learning rate 0.0001.

f) Kombinasi CNN-SVM dengan arsitektur Xception, transfer learning, fine-tuning dan learning rate 0.0001.

g. Data hasil training

Setelah mementukan model mana yang terbaik dari skenario yang ada, selanjutnya melakukan percobaan menggunakan dataset citra yang telah diolah pada preprocessing dan augmentasi sebelumnya. Data hasil training merupakan hasil skenario terbaik untuk menentukan akurasi dan performa. Model klasifikasi nantinya menjadi model yang direkomendasikan pada tahap evaluasi.

h. Evaluasi

Pada proses ini melakukan perbandingan akurasi pada semua skenario yang telah dilakukan. Perhitungan nilai akurasi dan performa menggunakan confusion matrix dan classification report.

i. Kesimpulan

Pada tahap ini menyajikan hasil dari penelitian. Hasil penelitian berupa data yang dihasilkan dari nilai confusion matrix dan classification report terkait dengan algoritma CNN dengan kombinasi CNN-SVM. Apakah metode kombinasi efektif digunakan dengan mendapatkan akurasi yang tinngi melebihi akurasi metode CNN.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Membangun Dataset

4.1.1. Pengumpulan Data

Data yang diperlukan pada penelitian ini berupa citra ras kucing dengan 12 jenis ras kucing yaitu Abyssinian, Bengal, Bombay, Birman, British Shorthair, Egyptian Mau, Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, dan Sphynx. Dataset ini diperoleh dari University of Oxford yang kemudian diberi nama Oxford-IIIT Pet yang dapat diakses secara publik (https://www.robots.ox.ac.uk/~vgg/data/pets/).

Tabel 4. 1 Detail dataset ras kucing

Jenis ras	Contoh citra dari ras kucing	Jumlah data
Abyssinian		200
Bengal		200
Bombay		200
Birman		200

Tabel 4. 1 Lanjutan

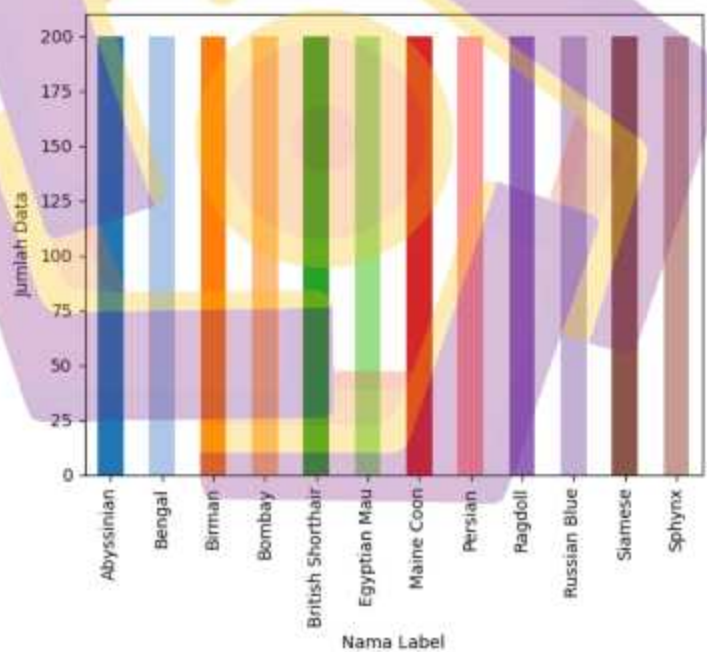
Jenis ras	Contoh citra dari ras kucing	Jumlah data
British Shorthair		200
Egyptian Mau		200
Maine Coon		200
Persian		200
Ragdoll		200
Russian Blue		200
Siamese		200
Sphynx		200
Total	**2400**	

Pada awalnya dataset tersebut terdiri dari 2 jenis hewan yaitu anjing dan kucing. Kemudian dataset yang dipilih hanya jenis kucing saja secara manual

karena sesuai dengan penelitiannya. Detail dari data yang digunakan pada penelitian ini ditampilkan pada tabel 4.1 dan 4.2, total dari keseluruhan data yang digunakan adalah 2400 data citra. Pada dataset ini dibagi ke dalam 12 kelas dengan masing-masing kelas berjumlah 200 data citra.

4.1.2. Persebaran Data

Data yang digunakan pada penelitian ini memiliki persebaran data yang seimbang, dapat dilihat pada gambar 4.1 adalah grafik persebaran data ras kucing yang digunakan.



Gambar 4. 1 Persebaran jumlah data ras kucing

4.2. Analisis Data

Sebelum dilakukan proses ke tahap implementasi, untuk menganalisisnya diperlukan suatu skenario agar mendapatkan perbandingan hasil dari nilai akurasi, presisi, recall, dan f1-score menggunakan confusion matrix dan classification report. Berikut skenario yang digunakan dalam penelitian ini adalah sebagai berikut:

a. CNN dengan arsitektur Inception-V3, transfer learning, dan learning rate 0.0001.
b. CNN dengan arsitektur Inception-V3, fine-tuning, dan learning rate 0.0001.
c. Kombinasi CNN-SVM dengan arsitektur Inception-V3, dan learning rate 0.0001.
d. CNN dengan arsitektur Xception, transfer learning, dan learning rate 0.0001.
e. CNN dengan arsitektur Xception, fine-tuning, dan learning rate 0.0001.
f. Kombinasi CNN-SVM dengan arsitektur Xception, dan learning rate 0.0001.

4.3. Implementasi pada Praproses Data

4.3.1. Implementasi Load Data

Langkah awal pada praproses merupakan load data yang digunakan untuk mengimpor data dari Google Drive ke Google Colab, mengakses direktori yang berisi dataset "Oxford-IIIT Pet Dataset," dan membangun DataFrame untuk digunakan dalam tugas machine learning berdasarkan penglihatan komputer. Berikut adalah script untuk load data.

```
import os
import pandas as pd

sdir=r'/content/drive/MyDrive/Dataset/Oxford-IIIT Pet Dataset'

filepaths=[]
labels=[]
classlist=os.listdir(sdir)
for class_ in classlist:
    classpath=os.path.join(sdir,class_)
    if os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(class_)
Fseries= pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())
```

Pada script diatas merupakan langkah awal sebelum ke pengelompokkan atau pembagian data. Setelah Google Drive terhubung menyediakan *path* ke direktori yang berisi dataset yang ingin digunakan, yaitu "Oxford-IIIT Pet Dataset." *Path* ini disimpan dalam variabel *sdir*. Kemudian, menyiapkan dua list kosong, *filepaths* dan *labels*, yang akan digunakan untuk menyimpan informasi tentang file gambar dan label kelas.

Untuk mengatur data dengan lebih baik menggunakan objek Series dalam *Pandas*. Pertama membuat *Fseries* yang berisi *path file* gambar dan diberi nama *'filepaths'*, serta *Lseries* yang berisi label kelas dan diberi nama *'labels'*. Kemudian menggabungkan kedua Series ini secara berdampingan untuk membentuk *DataFrame* yang disebut *df*. Hasilnya adalah *DataFrame* yang memiliki dua kolom, satu untuk *path file* gambar dan satu untuk label kelas.

4.3.2. Pengelompokkan Data

Pengelompokkan data dilakukan dengan menggunakan rasio perbandingan 70% data latih, 10% data validasi, dan 20% data uji. Pembagian dilakukan secara acak menggunakan *train_test_split* yang disediakan oleh *Sklearn* dari library *Python* yang menawarkan berbagai fitur untuk memroses data. Berikut adalah potongan dari script pembagian data pada ras kucing.

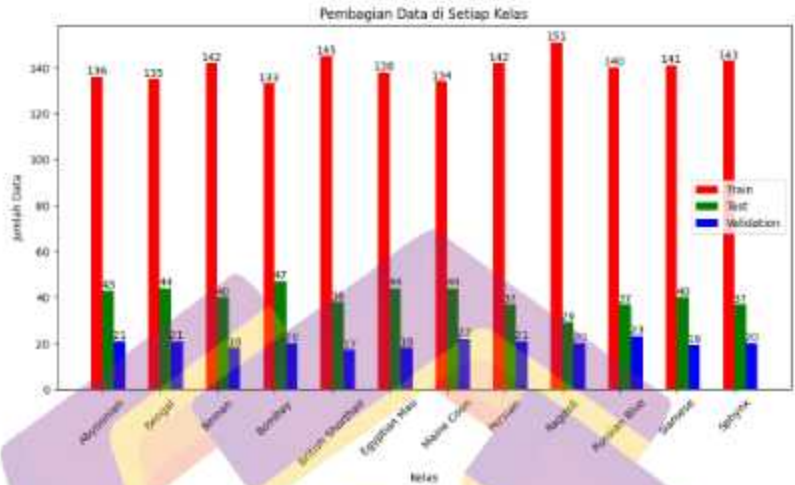
```
from sklearn.model_selection import train_test_split

train_split=.7
test_split=.2
dummy_split=test_split/(1-train_split)

train_df, dummy_df=train_test_split(df, train_size=train_split,
shuffle=True, random_state=0)
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split,
shuffle=True, random_state=0)

print ('train df length: ', len(train_df), ' test df length: ',
len(train_df), ' valid df length: ', len(valid_df))
```

Pada script diatas digunakan untuk membagi dataset menjadi tiga bagian yaitu data pelatihan, data pengujian, dan data validasi dengan menggunakan Teknik *train_test_split*. Teknik ini adalah cara umum dalam pembagian dataset ketika ingin membagi data. Selanjutnya pada script diatas juga menetapkan *train_split* sebesar 0.7 (70%) untuk data pelatihan dan *test_split* sebesar 0.2 (20%) untuk data pengujian. Ini berarti bahwa 70% data akan digunakan untuk pelatihan, 20% untuk pengujian, dan sisanya 10% untuk data validasi.



Gambar 4. 2 Pengelompokkan data pada kelas ras kucing

Hasil yang diperoleh dapat dilihat pada gambar 4.2 menunjukkan bahwa pada masing-masing kelas mendapatkan pembagian yang seimbang.

4.3.3. Preprocessing dan Augmentasi Data

Preprocessing data merupakan proses untuk menghindari data yang digunakan menjadi tidak ideal untuk digunakan pada model klasifikasi. Pada tahap ini, proses diawali dengan dilakukan resize dengan tujuan untuk menyamakan ukuran data menjadi 224x224 piksel untuk arsitektur Inception-V3 dan 299x299 piksel untuk arsitektur Xception dengan 3 channel warna (RGB) dengan batch size 32.

Augmentasi data merupakan teknik untuk menambah jumlah citra data yang digunakan dengan cara mengubah atau memodifikasi citra. Data masing-masing ras kucing yang digunakan pada penelitian ini memiliki jumlah yang seimbang. Tujuan

dari augmentasi ini untuk menambah variasi pada data pelatihan dan meningkatkan volume data. Augmentasi yang digunakan adalah rescale, rotation, zoom, width shift range, height shift range, horizontal flip, dan fill mode yang digunakan yaitu nearest.

Proses preprocessing dan augmentasi data ini dilakukan pada google colab dengan menggunakan library keras, berikut adalah script preprocessing dan augmentasi.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Inception-V3
height=299
width=299
channels=3
batch_size=32

# Xception
height=299
width=299
channels=3
batch_size=32

img_shape=(height, width, channels)
img_size=(height, width)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if
length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ('test batch size: ' ,test_batch_size, '  test steps: ',
test_steps)

gen=ImageDataGenerator(
    rescale=1./255,
    rotation_range = 20,
    zoom_range = 0.2,
    fill_mode = 'nearest',
    width_shift_range=0.2
    height_shift_range=0.2
    horizontal_flip = True,
    )
```

Lanjutan:

```
train_gen=gen.flow_from_dataframe( train_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True,
batch_size=batch_size)

validgen=ImageDataGenerator(rescale=1./255)

valid_gen=validgen.flow_from_dataframe( valid_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True,
batch_size=batch_size)

testgen=ImageDataGenerator(rescale=1./255)
test_gen=testgen.flow_from_dataframe( test_df, x_col='filepaths',
y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=False,
batch_size=test_batch_size)

classes=list(train_gen.class_indices.keys())
print (classes)
class_count=len(classes)
```

Pada script diatas menjelaskan tentang pendefinisian parameter dan ukuran citra. *Height, width,* dan *channels* adalah parameter yang digunakan untuk menentukan dimensi gambar yang akan diolah (tinggi, lebar, dan jumlah saluran warna). Lalu, *batch_size* adalah ukuran batch yang akan digunakan dalam pelatihan model. Kemudian, img_shape dan img_size adalah tuple yang menyimpan dimensi dan ukuran citra. Terakir, *length* adalah panjang (jumlah sampel) dari dataset uji (*test_df*).

ImageDataGenerator adalah yang digunakan untuk augmentasi citra. Augmentasi adalah proses mengubah citra asli dengan cara acak untuk meningkatkan variasi data pelatihan. Beberapa augmentasi yang digunakan termasuk rotasi, zoom, pergeseran horizontal dan vertikal, serta pemutaran citra. Hal ini membantu mencegah overfitting dan meningkatkan generalisasi model.

Setelah mementukan augmentasi yaitu melakukan pemrosesan data latih, validasi, dan uji. *Train_gen, valid_gen,* dan *test_gen* adalah objek generator gambar yang digunakan untuk memuat dan memproses gambar dari dataset pelatihan, validasi, dan pengujian. Data diproses sesuai dengan konfigurasi yang telah didefinisikan dalam *ImageDataGenerator* seperti *rescaling* (pengubahan rentang nilai piksel), augmentasi, dan lainnya. Fungsi *flow_from_dataframe* digunakan untuk memuat data dari data frame (seperti *pandas DataFrame*) dengan spesifikasi kolom gambar, kolom label, dan sebagainya.

Setelah melakukan pemrosesan, langkah terakhir yaitu membuat daftar kelas. *classes* adalah daftar kelas yang ditemukan dalam dataset pelatihan dan *class_count* adalah jumlah total kelas dalam dataset.

4.4. Tahap Membangun Model dan Implementasi

Pada tahap ini akan menjabarkan implementasi pada tahap pembangunan model CNN. Dalam penelitian ini, peneliti menggunakan model transfer learning Inception-V3 dan Xception lalu kemudian menggunakan klasifikasi dari CNN dan SVM. Input layer yang digunakan adalah data input dengan ukuran 224x224 piksel untuk Inception-V3 dan 299x299 piksel untuk Xception. Pada bagian output layer adalah layer klasifikasi yaitu lapisan Fully Connected. Pada lapisan ini model klasifikasi dapat disesuaikan dengan keinginan. Pada penelitian ini peneliti akan menggunakan softmax dari CNN atau bisa menggantinya dengan klasifikasi SVM.

Parameter yang digunakan pada membangun model ini dapat dilihat pada tabel 4.2 yang dimana masing-masing arsitektur memiliki parameter yang sama.

Pada penelitian ini lebih menekankan pada optimasi RMSprop dan Learning rate 0.0001.

Tabel 4. 2 Parameter yang digunakan pada moda model CNN baik dari Inception maupun Xception

Parameter	Rincian parameter yang digunakan
Optimasi	RMSprop
Learning rate	0.0001
Loss	Categorical Crossentropy
Metric	Accuracy
Aktivasi	Softmax
Dropout	0.5
Epoch	25
Batch Size	32

Untuk mendapatkan model klasifikasi yang diharapkan, skenario yang akan dibuat dengan membandingkan CNN transfer learning dan fine tuning dengan arsitektur Inception-V3 dan Xception yang dikombinasikan dengan SVM. Dalam penelitian ini akan melakukan 6 skenario percobaan yang dapat dilihat pada tabel 4.3.

Tabel 4. 3 Skenario Percobaan

Skenario	Model
Skenario pertama	CNN transfer learning arsitektur Inception-V3
Skenario kedua	CNN transfer learning dan fine-tuning arsitektur Inception-V3
Skenario ketiga	Kombinasi CNN arsitektur Inception-V3 dengan SVM
Skenario keempat	CNN transfer learning arsitektur Xception
Skenario kelima	CNN transfer learning dan fine-tuning arsitektur Xception
Skenario keenam	Kombinasi CNN arsitektur Xception dengan SVM

4.4.1. Tahap Implementasi Skenario Pertama

Pada tahap skenario pertama adalah melakukan perbandingan hasil dari transfer learning pada Inception-V3. Berikut adalah modul import yang digunakan pada arsitektur Inception-V3.

```
from tensorflow.keras.applications import InceptionV3
```

Kemudian memanggil fungsi model sesuai dengan arsitektur Inception-V3 dengan input yang telah ditentukan yaitu 224x224 piksel dan tanpa menyertakan model fully bawaan dari arsitketur transfer learning dengan parameter False pada include_top.

```
base_model = InceptionV3(weights='imagenet', include_top=False,
input_tensor=Input(shape=(224,224,3)))
```

Script selanjutnya adalah membuat lapisan fully connected yang baru sesuai dengan kebutuhan. Parameter pada *layer.trainable* disetting dengan *False* artinya yang dimana hanya membekukan semua lapisan kecuali lapisan fully connected yang baru dibuat.

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(2048, activation='relu')(x)
dropOut = Dropout(0.5)(x)
predictions = Dense(12, activation='softmax')(dropOut)
model = Model(inputs=base_model.input, outputs=predictions)

# Compile model
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.000
1),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Freeze semua layer di base model agar tidak di-train ulang
for layer in base_model.layers:
  layer.trainable = False
model.summary()
```

Setelah menambahkan fully conneced layer maka total parameter yang tidak dilatih berjumlah 21.802.784 dari total 26.023.724. Maka sisa yang dilatih berjumlah 4.220.940. Pada tabel 4.4 Menunjukkan total parameter, trainable dan non-trainable layer.

Tabel 4. 4 Total parameter, trainable dan non-trainable layer pada transfer learning Inception-V3

Transfer learning Inception-V3	
Total parameter	26.023.724
Trainable parameter	4.220.940
Non-trainable parameter	21.802.784

4.4.2. Tahap Implementasi Skenario Kedua

Pada tahap skenario kedua adalah melakukan perbandingan hasil dari fine-tuning pada Inception-V3. Fine-tuning berarti menggunakan kembali kemampuan klasifikasi pada Inception-V3 pada dataset yang berbeda. Untuk melakukan klasifikasi ras kucing menggunakan fine-tuning, maka beberapa convolutional layer ada yang dilatih kembali (trainable layer). Peneliti mencoba untuk melakukan skenario untuk melakukan pelatihan ulang pada tiap blok. Peneliti menggunakan fully connected layer sesuai dengan skenario yang pertama. Pada bagian output, peneliti membagi menjadi 12 kelas yang sesuai dengan jumlah kelas yang dibutuhkan.

Rincian dari setiap convolutional layer pada model fine-tuning Inception-V3 dengan input 224x224 piksel yang ditujukan pada tabel 4.5. Baris yang berwarna biru merupakan layer dibekukan (freeze layer), sedangkan baris berwarna hijau merupakan layer yang dilatih (trainable layer).

Tabel 4. 5 Arsitektur fine-tuning Inception-V3

Layer and (type)	Output Shape	Param#
input_1 (InputLayer)	(224, 224, 3)	0
conv2d (Conv2D)	(111, 111, 32)	864
batch_normalization (BatchNormalization)	(111, 111, 32)	96
activation (Activation)	(111, 111, 32)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_1 (Conv2D)	(109, 109, 32)	9216
batch_normalization_1 (BatchNormalization)	(109, 109, 32)	96
activation_1 (Activation)	(109, 109, 32)	0
conv2d_2 (Conv2D)	(109, 109, 64)	18432
batch_normalization_2 (BatchNormalization)	(109, 109, 64)	192
activation_2 (Activation)	(109, 109, 64)	0
max_pooling2d (MaxPooling2D)	(54, 54, 64)	0
conv2d_3 (Conv2D)	(54, 54, 80)	5120
batch_normalization_3 (BatchNormalization)	(54, 54, 80)	240
activation_3 (Activation)	(54, 54, 80)	0
conv2d_4 (Conv2D)	(52, 52, 192)	138240
batch_normalization_4 (BatchNormalization)	(52, 52, 192)	576
activation_4 (Activation)	(52, 52, 192)	0
max_pooling2d_1 (MaxPooling2D)	(25, 25, 192)	0
conv2d_8 (Conv2D)	(25, 25, 64)	12288
batch_normalization_8 (BatchNormalization)	(25, 25, 64)	192
activation_8 (Activation)	(25, 25, 64)	0
conv2d_6 (Conv2D)	(25, 25, 48)	9216
conv2d_9 (Conv2D)	(25, 25, 96)	55296
batch_normalization_6 (BatchNormalization)	(25, 25, 48)	144
batch_normalization_9 (BatchNormalization)	(25, 25, 96)	288
activation_6 (Activation)	(25, 25, 48)	0
activation_9 (Activation)	(25, 25, 96)	0
average_pooling2d (AveragePooling2D)	(25, 25, 192)	0
conv2d_5 (Conv2D)	(25, 25, 64)	12288

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_7 (Conv2D)	(25, 25, 64)	76800
conv2d_10 (Conv2D)	(25, 25, 96)	82944
conv2d_11 (Conv2D)	(25, 25, 32)	6144
batch_normalization_5 (BatchNormalization)	(25, 25, 64)	192
batch_normalization_7 (BatchNormalization)	(25, 25, 64)	192
batch_normalization_10 (BatchNormalization)	(25, 25, 96)	288
batch_normalization_11 (BatchNormalization)	(25, 25, 32)	96
activation_5 (Activation)	(25, 25, 64)	0
activation_7 (Activation)	(25, 25, 64)	0
activation_10 (Activation)	(25, 25, 96)	0
activation_11 (Activation)	(25, 25, 32)	0
mixed0 (Concatenate)	(25, 25, 256)	0
conv2d_15 (Conv2D)	(25, 25, 64)	16384
batch_normalization_15 (BatchNormalization)	(25, 25, 64)	192
activation_15 (Activation)	(25, 25, 64)	0
conv2d_13 (Conv2D)	(25, 25, 48)	12288
conv2d_16 (Conv2D)	(25, 25, 96)	55296
batch_normalization_13 (BatchNormalization)	(25, 25, 48)	144
batch_normalization_16 (BatchNormalization)	(25, 25, 96)	288
activation_13 (Activation)	(25, 25, 48)	0
activation_16 (Activation)	(25, 25, 96)	0
average_pooling2d_1 (AveragePooling2D)	(25, 25, 256)	0
conv2d_12 (Conv2D)	(25, 25, 64)	16384
conv2d_14 (Conv2D)	(25, 25, 64)	76800
conv2d_17 (Conv2D)	(25, 25, 96)	82944

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_18 (Conv2D)	(25, 25, 64)	16384
batch_normalization_12 (BatchNormalization)	(25, 25, 64)	192
batch_normalization_14 (BatchNormalization)	(25, 25, 64)	192
batch_normalization_17 (BatchNormalization)	(25, 25, 96)	288
batch_normalization_18 (BatchNormalization)	(25, 25, 64)	192
activation_12 (Activation)	(25, 25, 64)	0
activation_14 (Activation)	(25, 25, 64)	0
activation_17 (Activation)	(25, 25, 96)	0
activation_18 (Activation)	(25, 25, 64)	0
mixed1 (Concatenate)	(25, 25, 288)	0
conv2d_22 (Conv2D)	(25, 25, 64)	18432
batch_normalization_22 (BatchNormalization)	(25, 25, 64)	192
activation_22 (Activation)	(25, 25, 64)	0
conv2d_20 (Conv2D)	(25, 25, 48)	13824
conv2d_23 (Conv2D)	(25, 25, 96)	55296
batch_normalization_20 (BatchNormalization)	(25, 25, 48)	144
batch_normalization_23 (BatchNormalization)	(25, 25, 96)	288
activation_20 (Activation)	(25, 25, 48)	0
activation_23 (Activation)	(25, 25, 96)	0
average_pooling2d_2 (AveragePooling2D)	(25, 25, 288)	0
conv2d_19 (Conv2D)	(25, 25, 64)	18432
conv2d_21 (Conv2D)	(25, 25, 64)	76800
conv2d_24 (Conv2D)	(25, 25, 96)	82944
conv2d_25 (Conv2D)	(25, 25, 64)	18432
batch_normalization_19 (BatchNormalization)	(25, 25, 64)	192

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
batch_normalization_21 (BatchNormalization)	(25, 25, 64)	192
batch_normalization_24 (BatchNormalization)	(25, 25, 96)	288
batch_normalization_25 (BatchNormalization)	(25, 25, 64)	192
activation_19 (Activation)	(25, 25, 64)	0
activation_21 (Activation)	(25, 25, 64)	0
activation_24 (Activation)	(25, 25, 96)	0
activation_25 (Activation)	(25, 25, 64)	0
mixed2 (Concatenate)	(25, 25, 288)	0
conv2d_27 (Conv2D)	(25, 25, 64)	18432
batch_normalization_27 (BatchNormalization)	(25, 25, 64)	192
activation_27 (Activation)	(25, 25, 64)	0
conv2d_28 (Conv2D)	(25, 25, 96)	55296
batch_normalization_28 (BatchNormalization)	(25, 25, 96)	288
activation_28 (Activation)	(25, 25, 96)	0
conv2d_26 (Conv2D)	(12, 12, 384)	995328
conv2d_29 (Conv2D)	(12, 12, 96)	82944
batch_normalization_26 (BatchNormalization)	(12, 12, 384)	1152
batch_normalization_29 (BatchNormalization)	(12, 12, 96)	288
activation_26 (Activation)	(12, 12, 384)	0
activation_29 (Activation)	(12, 12, 96)	0
max_pooling2d_2 (MaxPooling2D)	(12, 12, 288)	0
mixed3 (Concatenate)	(12, 12, 768)	0
conv2d_34 (Conv2D)	(12, 12, 128)	98304
batch_normalization_34 (BatchNormalization)	(12, 12, 128)	384
activation_34 (Activation)	(12, 12, 128)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_35 (Conv2D)	(12, 12, 128)	114688
batch_normalization_35 (BatchNormalization)	(12, 12, 128)	384
activation_35 (Activation)	(12, 12, 128)	0
conv2d_31 (Conv2D)	(12, 12, 128)	98304
conv2d_36 (Conv2D)	(12, 12, 128)	114688
batch_normalization_31 (BatchNormalization)	(12, 12, 128)	384
batch_normalization_36 (BatchNormalization)	(12, 12, 128)	384
activation_31 (Activation)	(12, 12, 128)	0
activation_36 (Activation)	(12, 12, 128)	0
conv2d_32 (Conv2D)	(12, 12, 128)	114688
conv2d_37 (Conv2D)	(12, 12, 128)	114688
batch_normalization_32 (BatchNormalization)	(12, 12, 128)	384
batch_normalization_37 (BatchNormalization)	(12, 12, 128)	384
activation_32 (Activation)	(12, 12, 128)	0
activation_37 (Activation)	(12, 12, 128)	0
average_pooling2d_3 (AveragePooling2D)	(12, 12, 768)	0
conv2d_30 (Conv2D)	(12, 12, 192)	147456
conv2d_33 (Conv2D)	(12, 12, 192)	172032
conv2d_38 (Conv2D)	(12, 12, 192)	172032
conv2d_39 (Conv2D)	(12, 12, 192)	147456
batch_normalization_30 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_33 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_38 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_39 (BatchNormalization)	(12, 12, 192)	576
activation_30 (Activation)	(12, 12, 192)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
activation_33 (Activation)	(12, 12, 192)	0
activation_38 (Activation)	(12, 12, 192)	0
activation_39 (Activation)	(12, 12, 192)	0
mixed4 (Concatenate)	(12, 12, 768)	0
conv2d_44 (Conv2D)	(12, 12, 160)	122880
batch_normalization_44 (BatchNormalization)	(12, 12, 160)	480
activation_44 (Activation)	(12, 12, 160)	0
conv2d_45 (Conv2D)	(12, 12, 160)	179200
batch_normalization_45 (BatchNormalization)	(12, 12, 160)	480
activation_45 (Activation)	(12, 12, 160)	0
conv2d_41 (Conv2D)	(12, 12, 160)	122880
conv2d_46 (Conv2D)	(12, 12, 160)	179200
batch_normalization_41 (BatchNormalization)	(12, 12, 160)	480
batch_normalization_46 (BatchNormalization)	(12, 12, 160)	480
activation_41 (Activation)	(12, 12, 160)	0
activation_46 (Activation)	(12, 12, 160)	0
conv2d_42 (Conv2D)	(12, 12, 160)	179200
conv2d_47 (Conv2D)	(12, 12, 160)	179200
batch_normalization_42 (BatchNormalization)	(12, 12, 160)	480
batch_normalization_47 (BatchNormalization)	(12, 12, 160)	480
activation_42 (Activation)	(12, 12, 160)	0
activation_47 (Activation)	(12, 12, 160)	0
average_pooling2d_4 (AveragePooling2D)	(12, 12, 768)	0
conv2d_40 (Conv2D)	(12, 12, 192)	147456
conv2d_43 (Conv2D)	(12, 12, 192)	215040

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_48 (Conv2D)	(12, 12, 192)	215040
conv2d_49 (Conv2D)	(12, 12, 192)	147456
batch_normalization_40 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_43 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_48 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_49 (BatchNormalization)	(12, 12, 192)	576
activation_40 (Activation)	(12, 12, 192)	0
activation_43 (Activation)	(12, 12, 192)	0
activation_48 (Activation)	(12, 12, 192)	0
activation_49 (Activation)	(12, 12, 192)	0
mixed5 (Concatenate)	(12, 12, 768)	0
conv2d_54 (Conv2D)	(12, 12, 160)	122880
batch_normalization_54 (BatchNormalization)	(12, 12, 160)	480
activation_54 (Activation)	(12, 12, 160)	0
conv2d_55 (Conv2D)	(12, 12, 160)	179200
batch_normalization_55 (BatchNormalization)	(12, 12, 160)	480
activation_55 (Activation)	(12, 12, 160)	0
conv2d_51 (Conv2D)	(12, 12, 160)	122880
conv2d_56 (Conv2D)	(12, 12, 160)	179200
batch_normalization_51 (BatchNormalization)	(12, 12, 160)	480
batch_normalization_56 (BatchNormalization)	(12, 12, 160)	480
activation_51 (Activation)	(12, 12, 160)	0
activation_56 (Activation)	(12, 12, 160)	0
conv2d_52 (Conv2D)	(12, 12, 160)	179200
conv2d_57 (Conv2D)	(12, 12, 160)	179200

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
batch_normalization_52 (BatchNormalization)	(12, 12, 160)	480
batch_normalization_57 (BatchNormalization)	(12, 12, 160)	480
activation_52 (Activation)	(12, 12, 160)	0
activation_57 (Activation)	(12, 12, 160)	0
average_pooling2d_5 (AveragePooling2D)	(12, 12, 768)	0
conv2d_50 (Conv2D)	(12, 12, 192)	147456
conv2d_53 (Conv2D)	(12, 12, 192)	215040
conv2d_58 (Conv2D)	(12, 12, 192)	215040
conv2d_59 (Conv2D)	(12, 12, 192)	147456
batch_normalization_50 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_53 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_58 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_59 (BatchNormalization)	(12, 12, 192)	576
activation_50 (Activation)	(12, 12, 192)	0
activation_53 (Activation)	(12, 12, 192)	0
activation_58 (Activation)	(12, 12, 192)	0
activation_59 (Activation)	(12, 12, 192)	0
mixed6 (Concatenate)	(12, 12, 768)	0
conv2d_64 (Conv2D)	(12, 12, 192)	147456
batch_normalization_64 (BatchNormalization)	(12, 12, 192)	576
activation_64 (Activation)	(12, 12, 192)	0
conv2d_65 (Conv2D)	(12, 12, 192)	258048
batch_normalization_65 (BatchNormalization)	(12, 12, 192)	576
activation_65 (Activation)	(12, 12, 192)	0
conv2d_61 (Conv2D)	(12, 12, 192)	147456

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_66 (Conv2D)	(12, 12, 192)	258048
batch_normalization_61 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_66 (BatchNormalization)	(12, 12, 192)	576
activation_61 (Activation)	(12, 12, 192)	0
activation_66 (Activation)	(12, 12, 192)	0
conv2d_62 (Conv2D)	(12, 12, 192)	258048
conv2d_67 (Conv2D)	(12, 12, 192)	258048
batch_normalization_62 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_67 (BatchNormalization)	(12, 12, 192)	576
activation_62 (Activation)	(12, 12, 192)	0
activation_67 (Activation)	(12, 12, 192)	0
average_pooling2d_6 (AveragePooling2D)	(12, 12, 768)	0
conv2d_60 (Conv2D)	(12, 12, 192)	147456
conv2d_63 (Conv2D)	(12, 12, 192)	258048
conv2d_68 (Conv2D)	(12, 12, 192)	258048
conv2d_69 (Conv2D)	(12, 12, 192)	147456
batch_normalization_60 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_63 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_68 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_69 (BatchNormalization)	(12, 12, 192)	576
activation_60 (Activation)	(12, 12, 192)	0
activation_63 (Activation)	(12, 12, 192)	0
activation_68 (Activation)	(12, 12, 192)	0
activation_69 (Activation)	(12, 12, 192)	0
mixed7 (Concatenate)	(12, 12, 768)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
conv2d_72 (Conv2D)	(12, 12, 192)	147456
batch_normalization_72 (BatchNormalization)	(12, 12, 192)	576
activation_72 (Activation)	(12, 12, 192)	0
conv2d_73 (Conv2D)	(12, 12, 192)	258048
batch_normalization_73 (BatchNormalization)	(12, 12, 192)	576
activation_73 (Activation)	(12, 12, 192)	0
conv2d_70 (Conv2D)	(12, 12, 192)	147456
conv2d_74 (Conv2D)	(12, 12, 192)	258048
batch_normalization_70 (BatchNormalization)	(12, 12, 192)	576
batch_normalization_74 (BatchNormalization)	(12, 12, 192)	576
activation_70 (Activation)	(12, 12, 192)	0
activation_74 (Activation)	(12, 12, 192)	0
conv2d_71 (Conv2D)	(5, 5, 320)	552960
conv2d_75 (Conv2D)	(5, 5, 192)	331776
batch_normalization_71 (BatchNormalization)	(5, 5, 320)	960
batch_normalization_75 (BatchNormalization)	(5, 5, 192)	576
activation_71 (Activation)	(5, 5, 320)	0
activation_75 (Activation)	(5, 5, 192)	0
max_pooling2d_3 (MaxPooling2D)	(5, 5, 768)	0
mixed8 (Concatenate)	(5, 5, 1280)	0
conv2d_80 (Conv2D)	(5, 5, 448)	573440
batch_normalization_80 (BatchNormalization)	(5, 5, 448)	1344
activation_80 (Activation)	(5, 5, 448)	0
conv2d_77 (Conv2D)	(5, 5, 384)	491520
conv2d_81 (Conv2D)	(5, 5, 384)	1548288

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
batch_normalization_77 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_81 (BatchNormalization)	(5, 5, 384)	1152
activation_77 (Activation)	(5, 5, 384)	0
activation_81 (Activation)	(5, 5, 384)	0
conv2d_78 (Conv2D)	(5, 5, 384)	442368
conv2d_79 (Conv2D)	(5, 5, 384)	442368
conv2d_82 (Conv2D)	(5, 5, 384)	442368
conv2d_83 (Conv2D)	(5, 5, 384)	442368
average_pooling2d_7 (AveragePooling2D)	(5, 5, 1280)	0
conv2d_76 (Conv2D)	(5, 5, 320)	409600
batch_normalization_78 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_79 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_82 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_83 (BatchNormalization)	(5, 5, 384)	1152
conv2d_84 (Conv2D)	(5, 5, 192)	245760
batch_normalization_76 (BatchNormalization)	(5, 5, 320)	960
activation_78 (Activation)	(5, 5, 384)	0
activation_79 (Activation)	(5, 5, 384)	0
activation_82 (Activation)	(5, 5, 384)	0
activation_83 (Activation)	(5, 5, 384)	0
batch_normalization_84 (BatchNormalization)	(5, 5, 192)	576
activation_76 (Activation)	(5, 5, 320)	0
mixed9_0 (Concatenate)	(5, 5, 768)	0
concatenate (Concatenate)	(5, 5, 768)	0
activation_84 (Activation)	(5, 5, 192)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
mixed9 (Concatenate)	(5, 5, 2048)	0
conv2d_89 (Conv2D)	(5, 5, 448)	917504
batch_normalization_89 (BatchNormalization)	(5, 5, 448)	1344
activation_89 (Activation)	(5, 5, 448)	0
conv2d_86 (Conv2D)	(5, 5, 384)	786432
conv2d_90 (Conv2D)	(5, 5, 384)	1548288
batch_normalization_86 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_90 (BatchNormalization)	(5, 5, 384)	1152
activation_86 (Activation)	(5, 5, 384)	0
activation_90 (Activation)	(5, 5, 384)	0
conv2d_87 (Conv2D)	(5, 5, 384)	442368
conv2d_88 (Conv2D)	(5, 5, 384)	442368
conv2d_91 (Conv2D)	(5, 5, 384)	442368
conv2d_92 (Conv2D)	(5, 5, 384)	442368
average_pooling2d_8 (AveragePooling2D)	(5, 5, 2048)	0
conv2d_85 (Conv2D)	(5, 5, 320)	655360
batch_normalization_87 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_88 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_91 (BatchNormalization)	(5, 5, 384)	1152
batch_normalization_92 (BatchNormalization)	(5, 5, 384)	1152
conv2d_93 (Conv2D)	(5, 5, 192)	393216
batch_normalization_85 (BatchNormalization)	(5, 5, 320)	960
activation_87 (Activation)	(5, 5, 384)	0
activation_88 (Activation)	(5, 5, 384)	0
activation_91 (Activation)	(5, 5, 384)	0

Tabel 4. 5 Lanjutan

Layer and (type)	Output Shape	Param#
activation_92 (Activation)	(5, 5, 384)	0
batch_normalization_93 (BatchNormalization)	(5, 5, 192)	576
activation_85 (Activation)	(5, 5, 320)	0
mixed9_1 (Concatenate)	(5, 5, 768)	0
concatenate_1 (Concatenate)	(5, 5, 768)	0
activation_93 (Activation)	(5, 5, 192)	0
mixed10 (Concatenate)	(5, 5, 2048)	0

Total parameter untuk trainable parameter dan Non trainable parameter bergantung pada jumlah blok layer yang dilatih yang dimana sesuai dengan skenario yang telah dibuat. Jumlah parameter yang dilatih dengan fine-tuning ditunjukkan pada tabel 4.6.

Tabel 4. 6 Total parameter, trainable dan non-trainable layer pada fine-tuning Inception-V3

Fine-tuning Inception-V3	
Total parameter	21.802.784
Trainable parameter	19.228.288
Non-trainable parameter	2.574.496

Tahap selanjutnya adalah melakukan training model fine-tuning Inception-V3. Model fine-tuning Inception-V3 yang digunakan tanpa menggunakan fully connected layer.

```
model_ft_in = InceptionV3(weights='imagenet', include_top=False,
input_tensor=Input(shape=(224,224,3)))
```

Parameter trainable pada layer digunakan untuk membekukan bobot lapisan tertentu yang dapat diatur menjadi False yang artinya layer tersebut tidak boleh dilatih. Pada fine-tuning Inception-V3 ini, peneliti melatih layer mulai dari

Activation sampai dengan Concatenate yang bisa dilihat pada tabel 4.5 yang berwarna hijau.

Setiap layer memiliki parameter yang disebut trainable. Untuk membekukan lapisan tertentu harus mengatur parameter ke False yang menunjukkan bahwa layer tersebut tidak boleh dilatih. Pada Inception-V3, peneliti hanya melatih yang terdiri dari 200 layer.

```
# Fine-tuning (mengaktifkan beberapa layer di base model)
for layer in model_ft_in.layers[:-200]:
    layer.trainable = False
for layer in model_ft_in.layers:
    print(layer, layer.trainable)
```

Setelah menetapkan layer trainable pada model, pada tabel 4.7 menunjukkan parameter layer fully connected yang dibuat dengan 12 output sesuai dengan jumlah kelas yang dimiliki. Peneliti membuat fully connected layer lagi dengan menyamakan seperti pada skenario pertama. Pada tabel 4.8 menunjukkan jumlah parameter setelah menambahkan fully connection layer pada fine-tuning.

```
from tensorflow.keras.models import Sequential

# Create a Sequential model
model = Sequential()

# Add layers to the model
model.add(model_ft_in)
model.add(GlobalAveragePooling2D())
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(12, activation='softmax'))

# Compile model
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.000
1),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Summary of the model architecture
model.summary()
```

Tabel 4. 7. Fully Connection Layer yang ditambahkan

Layer and (type)	Output Shape	Param#
inception_v3 (Functional)	(5, 5, 2048)	21802784
global_average_pooling2d_1 (GlobalAveragePooling2D)	(2048,)	0
dense_2 (Dense)	(2048,)	4196352
dropout_1 (Dropout)	(2048,)	0
dense_2 (Dense)	(12,)	24588

Tabel 4. 8. Total parameter, trainable dan non-trainable layer pada fine-tuning Inception-V3 setelah menambahkan fully connection layer

Fine-tuning Inception-V3	
Total parameter	26.023.724
Trainable parameter	25.989.292
Non-trainable parameter	34.432

4.4.3. Tahap Implementasi Skenario ketiga

Pada tahap skenario ketiga adalah melakukan perbandingan hasil dari kombinasi CNN dan SVM menggunakan arsitektur dari Inception-V3. Langkah yang dilakukan pada tahap ini adalah dengan melakukan *feature extraxction* menggunakan CNN. Setelah mendapatkan hasil *feature extraction* langkah selanjutnya adalah mengubah ukuran dimensi dari *convolutional* lalu disesuaikan dengan dimensi pada SVM kemudian melakukan reshape. Berikut ini adalah script untuk mengambil *feature extraction* dari CNN dan reshape.

```
# Fungsi untuk ekstraksi fitur (sama seperti sebelumnya)
def extract_features(generator, model):
    features = []
    labels = []

    for i in range(generator.n // generator.batch_size + 1):
```

Lanjutan:

```
        x, y = generator.next()
        features_batch = model.predict(x)
        features.append(features_batch)
        labels.append(y)

        if i % 100 == 0:
            print(f'Processed {i} batches')

        if (i + 1) * generator.batch_size >= generator.n:
            break

    features = np.vstack(features)
    labels = np.vstack(labels)

    return features, labels

# Gunakan fungsi ekstraksi fitur dengan model yang dimuat
train_features, train_labels = extract_features(train_gen, model)
test_features, test_labels = extract_features(test_gen, model)

# Flatten fitur untuk SVM
train_features_flatten =
train_features.reshape(train_features.shape[0], -1)
test_features_flatten = test_features.reshape(test_features.shape[0], -
1)
```

Setelah mendapatkan input yang sesuai, langkah selanjutnya adalah melakukan klasifikasi menggunakan SVM. Model kernel SVM yang digunakan adalah kernel linear karena kelas output yang digunakan berupa *multiclass classification*. Hyperparameter yang digunakan pada klasifikasi menggunakan SVM ini bisa dilihat pada tabel 4.9.

Tabel 4. 9. Hyperparameter yang digunakan pada kombinasi CNN-SVM dari arsitektur Inception-V3

Hyperparameter	Rincian yang digunakan
Cost	50
gamma	0.0001
probability	True
toleransi	0.0001
Verbose	True
Random_state	0

Setelah itu melakukan pelatihan dan pengujian dengan menggunakan data latih dan data uji. Script yang digunakan untuk inisiasi sebagai berikut

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Membuat dan melatih model SVM
svm_model = SVC(kernel='linear', C=50, gamma=0.0001, probability=True
          ,tol=0.0001, verbose=True, random_state=0)
svm_model.fit(train_features_flatten, np.argmax(train_labels, axis=1))

# Melakukan prediksi pada data testing
predictions = svm_model.predict(test_features_flatten)

# Mengkonversi label testing ke format yang sesuai
test_labels_converted = np.argmax(test_labels, axis=1)

from sklearn.metrics import accuracy_score

# Menghitung dan mencetak accuracy score
accuracy = accuracy_score(test_labels_converted, predictions)
print("Accuracy Score:", accuracy)
```

4.4.4. Tahap Implementasi Skenario Keempat

Pada tahap skenario keempat adalah melakukan perbandingan hasil dari transfer learning pada Xception. Berikut adalah modul import yang digunakan pada arsitektur Xception.

```
from tensorflow.keras.applications import Xception
```

Kemudian memanggil fungsi model sesuai dengan arsitektur Xception dengan input yang telah ditentukan yaitu 299x299 piksel dan tanpa menyertakan model fully bawaan dari arsitketur transfer learning dengan parameter False pada include_top.

```
base_model = Xception(weights='imagenet', include_top=False,
input_tensor=Input(shape=(299,299,3)))
```

Script selanjutnya adalah membuat lapisan fully connected yang baru sesuai dengan kebutuhan. Parameter pada *layer.trainable* disetting dengan *False* artinya

yang dimana hanya membekukan semua lapisan kecuali lapisan fully connected yang baru dibuat.

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(2048, activation='relu')(x)
dropOut = Dropout(0.5)(x)
predictions = Dense(2, activation='softmax')(dropOut)
model = Model(inputs=base_model.input, outputs=predictions)

# Compile model
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.000
1),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Freeze semua layer di base model agar tidak di-train ulang
for layer in base_model.layers:
  layer.trainable = False

model.summary()
```

Setelah menambahkan fully conneced layer maka total parameter yang tidak dilatih berjumlah 20.861.480 dari total 25.082.420. Maka sisa yang dilatih berjumlah 4.220.940. Pada tabel 4.10 Menunjukkan total parameter, trainable dan non-trainable layer.

Tabel 4. 10 Total parameter, trainable dan non-trainable layer pada transfer learning Xception

	Transfer learning Xception
Total parameter	25.082.420
Trainable parameter	4.220.940
Non-trainable parameter	20.861.480

4.4.5. Tahap Implementasi Skenario Kelima

Pada tahap skenario kelima adalah melakukan perbandingan hasil dari fine-tuning pada Xception. Fine-tuning berarti menggunakan kembali kemampuan klasifikasi pada Xception pada dataset yang berbeda. Untuk melakukan klasifikasi

ras kucing menggunakan fine-tuning, maka beberapa convolutional layer ada yang dilatih kembali (trainable layer). Peneliti mencoba untuk melakukan skenario untuk melakukan pelatihan ulang pada tiap blok. Peneliti menggunakan fully connected layer sesuai dengan skenario yang pertama. Pada bagian output, peneliti membagi menjadi 12 kelas yang sesuai dengan jumlah kelas yang dibutuhkan.

Rincian dari setiap convolutional layer pada model fine-tuning Xception dengan input 299x299 piksel yang ditujukan pada tabel 4.11. Baris yang berwarna merah merupakan layer dibekukan (freeze layer), sedangkan baris berwarna kuning merupakan layer yang dilatih (trainable layer).

Tabel 4. 11 Arsitektur fine-tuning Xception

Layer and (type)	Output Shape	Param#
input_1 (InputLayer)	(299, 299, 3)	0
block1_conv1 (Conv2D)	(149, 149, 32)	864
block1_conv1_bn (BatchNormalization)	(149, 149, 32)	128
block1_conv1_act (Activation)	(149, 149, 32)	0
block1_conv2 (Conv2D)	(147, 147, 64)	18432
block1_conv2_bn (BatchNormalization)	(147, 147, 64)	256
block1_conv2_act (Activation)	(147, 147, 64)	0
block2_sepconv1 (SeparableConv2D)	(147, 147, 128)	8768
block2_sepconv1_bn (BatchNormalization)	(147, 147, 128)	512
block2_sepconv2_act (Activation)	(147, 147, 128)	0
block2_sepconv2 (SeparableConv2D)	(147, 147, 128)	17536
block2_sepconv2_bn (BatchNormalization)	(147, 147, 128)	512
conv2d (Conv2D)	(74, 74, 128)	8192
block2_pool (MaxPooling2D)	(74, 74, 128)	0
batch_normalization (BatchNormalization)	(74, 74, 128)	512
add (Add)	(74, 74, 128)	0

Tabel 4. 11 Lanjutan

Layer and (type)	Output Shape	Param#
block3_sepconv1_act (Activation)	(74, 74, 128)	0
block3_sepconv1 (SeparableConv2D)	(74, 74, 256)	33920
block3_sepconv1_bn (BatchNormalization)	(74, 74, 256)	1024
block3_sepconv2_act (Activation)	(74, 74, 256)	0
block3_sepconv2 (SeparableConv2D)	(74, 74, 256)	67840
block3_sepconv2_bn (BatchNormalization)	(74, 74, 256)	1024
conv2d_1 (Conv2D)	(37, 37, 256)	32768
block3_pool (MaxPooling2D)	(37, 37, 256)	0
batch_normalization_1 (BatchNormalization)	(37, 37, 256)	1024
add_1 (Add)	(37, 37, 256)	0
block4_sepconv1_act (Activation)	(37, 37, 256)	0
block4_sepconv1 (SeparableConv2D)	(37, 37, 728)	188672
block4_sepconv1_bn (BatchNormalization)	(37, 37, 728)	2912
block4_sepconv2_act (Activation)	(37, 37, 728)	0
block4_sepconv2 (SeparableConv2D)	(37, 37, 728)	536536
block4_sepconv2_bn (BatchNormalization)	(37, 37, 728)	2912
conv2d_2 (Conv2D)	(19, 19, 728)	186368
block4_pool (MaxPooling2D)	(19, 19, 728)	0
batch_normalization_2 (BatchNormalization)	(19, 19, 728)	2912
add_2 (Add)	(19, 19, 728)	0
block5_sepconv1_act (Activation)	(19, 19, 728)	0
block5_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block5_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block5_sepconv2_act (Activation)	(19, 19, 728)	0
block5_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block5_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block5_sepconv3_act (Activation)	(19, 19, 728)	0
block5_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536

Tabel 4. 11 Lanjutan

Layer and (type)	Output Shape	Param#
block5_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_3 (Add)	(19, 19, 728)	0
block6_sepconv1_act (Activation)	(19, 19, 728)	0
block6_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block6_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block6_sepconv2_act (Activation)	(19, 19, 728)	0
block6_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block6_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block6_sepconv3_act (Activation)	(19, 19, 728)	0
block6_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block6_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_4 (Add)	(19, 19, 728)	0
block7_sepconv1_act (Activation)	(19, 19, 728)	0
block7_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block7_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block7_sepconv2_act (Activation)	(19, 19, 728)	0
block7_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block7_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block7_sepconv3_act (Activation)	(19, 19, 728)	0
block7_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block7_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_5 (Add)	(19, 19, 728)	0
block8_sepconv1_act (Activation)	(19, 19, 728)	0
block8_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block8_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block8_sepconv2_act (Activation)	(19, 19, 728)	0
block8_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block8_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912

Tabel 4. 11 Lanjutan

Layer and (type)	Output Shape	Param#
block8_sepconv3_act (Activation)	(19, 19, 728)	0
block8_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block8_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_6 (Add)	(19, 19, 728)	0
block9_sepconv1_act (Activation)	(19, 19, 728)	0
block9_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block9_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block9_sepconv2_act (Activation)	(19, 19, 728)	0
block9_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block9_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block9_sepconv3_act (Activation)	(19, 19, 728)	0
block9_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block9_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_7 (Add)	(19, 19, 728)	0
block10_sepconv1_act (Activation)	(19, 19, 728)	0
block10_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block10_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block10_sepconv2_act (Activation)	(19, 19, 728)	0
block10_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block10_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block10_sepconv3_act (Activation)	(19, 19, 728)	0
block10_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block10_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_8 (Add)	(19, 19, 728)	0
block11_sepconv1_act (Activation)	(19, 19, 728)	0
block11_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block11_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block11_sepconv2_act (Activation)	(19, 19, 728)	0

Tabel 4. 11 Lanjutan

Layer and (type)	Output Shape	Param#
block11_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block11_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block11_sepconv3_act (Activation)	(19, 19, 728)	0
block11_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block11_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_9 (Add)	(19, 19, 728)	0
block12_sepconv1_act (Activation)	(19, 19, 728)	0
block12_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block12_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block12_sepconv2_act (Activation)	(19, 19, 728)	0
block12_sepconv2 (SeparableConv2D)	(19, 19, 728)	536536
block12_sepconv2_bn (BatchNormalization)	(19, 19, 728)	2912
block12_sepconv3_act (Activation)	(19, 19, 728)	0
block12_sepconv3 (SeparableConv2D)	(19, 19, 728)	536536
block12_sepconv3_bn (BatchNormalization)	(19, 19, 728)	2912
add_10 (Add)	(19, 19, 728)	0
block13_sepconv1_act (Activation)	(19, 19, 728)	0
block13_sepconv1 (SeparableConv2D)	(19, 19, 728)	536536
block13_sepconv1_bn (BatchNormalization)	(19, 19, 728)	2912
block13_sepconv2_act (Activation)	(19, 19, 728)	0
block13_sepconv2 (SeparableConv2D)	(19, 19, 1024)	752024
block13_sepconv2_bn (BatchNormalization)	(19, 19, 1024)	4096
conv2d_3 (Conv2D)	(10, 10, 1024)	745472
block13_pool (MaxPooling2D)	(10, 10, 1024)	0
batch_normalization_3 (BatchNormalization)	(10, 10, 1024)	4096
add_11 (Add)	(10, 10, 1024)	0
block14_sepconv1 (SeparableConv2D)	(10, 10, 1536)	1582080
block14_sepconv1_bn (BatchNormalization)	(10, 10, 1536)	6144

Tabel 4. 11 Lanjutan

Layer and (type)	Output Shape	Param#
block14_sepconv1_act (Activation)	(10, 10, 1536)	0
block14_sepconv2 (SeparableConv2D)	(10, 10, 2048)	3159552
block14_sepconv2_bn (BatchNormalization)	(10, 10, 2048)	8192
block14_sepconv2_act (Activation)	(10, 10, 2048)	0

Total parameter untuk trainable parameter dan Non trainable parameter bergantung pada jumlah blok layer yang dilatih yang dimana sesuai dengan skenario yang telah dibuat. Jumlah parameter yang dilatih dengan fine-tuning ditunjukkan pada tabel 4.12.

Tabel 4. 12 Total parameter, trainable dan non-trainable layer pada fine-tuning Xception

Fine-tuning Xception	
Total parameter	20.814.984
Trainable parameter	16.511.168
Non-trainable parameter	4.303.816

Tahap selanjutnya adalah melakukan training model fine-tuning Xception. Model fine-tuning Xception yang digunakan tanpa menggunakan fully connected layer.

```
model_ft_in = Xception(weights='imagenet', include_top=False,
input_tensor=Input(shape=(299,299,3)))
```

Parameter trainable pada layer digunakan untuk membekukan bobot lapisan tertentu yang dapat diatur menjadi False yang artinya layer tersebut tidak boleh dilatih. Pada fine-tuning Xception ini, peneliti melatih layer yang bisa dilihat pada tabel 4.11 yang berwarna kuning.

Setiap layer memiliki parameter yang disebut trainable. Untuk membekukan lapisan tertentu harus mengatur parameter ke False yang menunjukkan bahwa layer

tersebut tidak boleh dilatih. Pada Xception, peneliti hanya melatih yang terdiri dari 77 layer.

```
# Fine-tuning (mengaktifkan beberapa layer di base model)
for layer in model_ft_in.layers[:-77]:
    layer.trainable = False
for layer in model_ft_in.layers:
    print(layer, layer.trainable)
```

Setelah menetapkan layer trainable pada model, pada tabel 4.13 menunjukkan parameter layer fully connected yang dibuat dengan 12 output sesuai dengan jumlah kelas yang dimiliki. Peneliti membuat fully connected layer lagi dengan menyamakan seperti pada skenario pertama. Pada tabel 4.14 menunjukkan jumlah parameter setelah menambahkan fully connection layer pada fine-tuning.

```
from tensorflow.keras.models import Sequential

# Create a Sequential model
model = Sequential()

# Add layers to the model
model.add(model_ft_in)
model.add(GlobalAveragePooling2D())
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(12, activation='softmax'))

# Compile model
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.000
1),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Summary of the model architecture
model.summary()
```

Tabel 4. 13. Fully Connection Layer yang ditambahkan

Layer and (type)	Output Shape	Param#
xception (Functional)	(10, 10, 2048)	20861480
global_average_pooling2d_2 (GlobalAveragePooling2D)	(2048,)	0
dense_4 (Dense)	(2048,)	4196352
dropout_2 (Dropout)	(2048,)	0
dense_5 (Dense)	(12,)	24588

Tabel 4. 14. Total parameter, trainable dan non-trainable layer pada fine-tuning Xception setelah menambahkan fully connection layer

Fine-tuning Xception	
Total parameter	25.082.420
Trainable parameter	25.027.892
Non-trainable parameter	54.528

4.4.6. Tahap Implementasi Skenario Keenam

Pada tahap skenario keenam adalah melakukan perbandingan hasil dari kombinasi CNN dan SVM menggunakan arsitektur dari Xception. Langkah yang dilakukan pada tahap ini adalah dengan melakukan *feature extraxction* menggunakan CNN. Setelah mendapatkan hasil *feature extraction* langkah selanjutnya adalah mengubah ukuran dimensi dari *convolutional* lalu disesuaikan dengan dimensi pada SVM kemudian melakukan reshape. Berikut ini adalah script untuk mengambil *feature extraction* dari CNN dan reshape.

```
# Fungsi untuk ekstraksi fitur
def extract_features(generator, model):
    features = []
    labels = []

    for i in range(generator.n // generator.batch_size + 1):
        x, y = generator.next()
        features_batch = model.predict(x)
```

Lanjutan:

```
        features.append(features_batch)
        labels.append(y)

        if i % 100 == 0:
            print(f'Processed {i} batches')

        if (i + 1) * generator.batch_size >= generator.n:
            break

    features = np.vstack(features)
    labels = np.vstack(labels)

    return features, labels

# Gunakan fungsi ekstraksi fitur dengan model yang dimuat
train_features, train_labels = extract_features(train_gen, model)
test_features, test_labels = extract_features(test_gen, model)

# Flatten fitur untuk SVM
train_features_flatten =
train_features.reshape(train_features.shape[0], -1)
test_features_flatten = test_features.reshape(test_features.shape[0], -
1)
```

Setelah mendapatkan input yang sesuai, langkah selanjutnya adalah melakukan klasifikasi menggunakan SVM. Model kernel SVM yang digunakan adalah kernel linear karena kelas output yang digunakan berupa *multiclass classification*. Hyperparameter yang digunakan pada klasifikasi menggunakan SVM ini bisa dilihat pada tabel 4.15.

Tabel 4. 15. Hyperparameter yang digunakan pada kombinasi CNN-SVM dari arsitektur Xception

Hyperparameter	Rincian yang digunakan
Cost	50
gamma	0.0001
probability	True
tolcransi	0.0001
Verbose	True
Random_state	0

Setelah itu melakukan pelatihan dan pengujian dengan menggunakan data latih dan data uji. Script yang digunakan untuk inisiasi sebagai berikut

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Membuat dan melatih model SVM
svm_model = SVC(kernel='linear', C=50, gamma=0.0001, probability=True
          ,tol=0.0001, verbose=True, random_state=0)
svm_model.fit(train_features_flatten, np.argmax(train_labels, axis=1))

# Melakukan prediksi pada data testing
predictions = svm_model.predict(test_features_flatten)

# Mengkonversi label testing ke format yang sesuai
test_labels_converted = np.argmax(test_labels, axis=1)

from sklearn.metrics import accuracy_score

# Menghitung dan mencetak accuracy score
accuracy = accuracy_score(test_labels_converted, predictions)
print("Accuracy Score:", accuracy)
```

4.5. Hasil Analisis dan Pembahasan

Pada tahap ini proses klasifikasi ras kucing dilakukan dengan menggunakan skenario-skenario yang telah ditentukan pada penjelasan sebelumnya. Hasil dari skenario yang telah dilakukan akan dijelaskan secara rinci dan dianalis. Metode pengujian yang digunakan ini adalah *Confusion Matrix* dan *Classification Report*, dengan metode ini tidak hanya menunjukkan akurasi dari model klasifikasi saja akan tetap memberikan nilai precision, recall, dan f1-score yang bisa digunakan untuk menganalisa performa dari skenario-skenario diatas.

4.5.1. Hasil Preprocessing dan Augmentasi

Preprocessing yang digunakan adalah mengubah ukuran atau resize menjadi 299x299x3 secara menyeluruh. Lalu, untuk augmentasi terdapat beberapa yang

digunakan seperti rescale, rotation range, zoom range, fill mode, width and height shift range, dan horizontal flip. Pada tabel 4.16 merupakan parameter yang digunakan pada augmentasi.

Tabel 4. 16. Parameter pada augmentasi

Augmentasi	
Rescale	1./225
Rotation range	20
Zoom range	0.2
Fill Mode	Nearest
Width shift range	0.2
Height shift range	0.2
Horizontal flip	True

Pada gambar 4.3 merupakan contoh hasil pada preprocessing dan augmentasi yang sesuai dengan parameter yang digunakan.

Gambar 4. 3 Hasil preprocessing dan augmentasi

4.5.2. Hasil Pelatihan dari Model

Pada hasil pelatihan ini merupakan dari skenario-skenario yang telah dilatih yang dimana terdapat 6 skenario. Dalam penelitian ini tabel 4.17 menunjukkan hasil dari pelatihan pada data latih, validasi, dan uji pada masing-masing skenario.

Tabel 4. 17. Hasil pelatihan dari masing-masing skenario

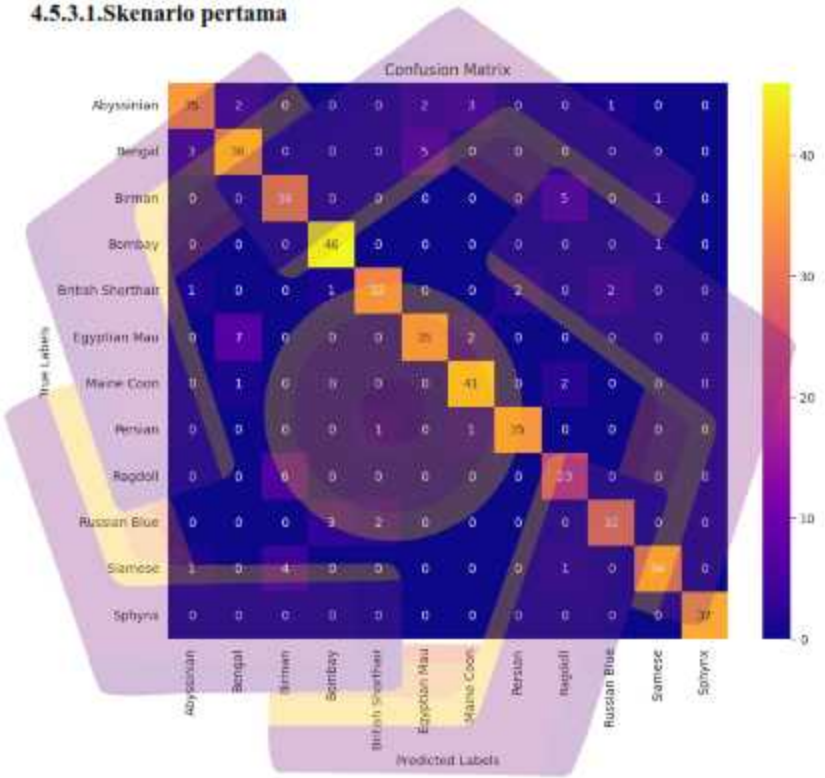
Skenario	Model	Akurasi yang didapatkan		
		Training	Validation	Testing
1	CNN Inception-V3	99.70%	89.17%	87.50%
2	CNN Inception-V3 dan Fine-Tuning	99.23%	91.67%	86.04%
3	Kombinasi CNN Inception-V3 dengan SVM	-	-	88.12%
4	CNN Xception	99.94%	92.50%	94.58%
5	CNN Xception dan Fine-Tuning	99.82%	93.33%	92.71%
6	Kombinasi CNN Xception dengan SVM	-	-	93.96%

Dari tabel 4.17 menunjukkan bahwa akurasi yang didapatkan dari tiap skenario tidak ada yang memiliki perbedaan yang cukup tinggi. Hasil akurasi yang tertinggi pada skenario keempat yang menunjukkan bahwa model ini mampu mempelajari fitur dengan sangat baik dan mungkin lebih baik daripada Inception-V3. Lalu, hasil akurasi yang terendah pada skenario kedua yang menunjukkan bahwa akurasi pada validasi meningkat dibanding skenario pertama, yang menunjukkan bahwa fine-tuning membantu model generalisasi lebih baik, namun masih terjadi penurunan pada data testing.

4.5.3. Hasil Perhitungan *Confusion Matrix*

Confusion Matrix adalah metode pengujian untuk menghitung performa dari sebuah model klasifikasi. Metode ini tersusun dari matrix hasil prediksi model klasifikasi dengan data aktual.

4.5.3.1.Skenario pertama



Gambar 4. 4. Confusion Matrix skenario pertama

Gambar 4.4 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 35 data di prediksi benar, 8 data di prediksi salah. Begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TN (True Negative), FP (False Positive), dan FN (False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

   Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 35 + 36 + 34 + 46 + 32 + 35 + 41 + 35 + 23 + 32 + 34 + 37 = 420, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

   $Akurasi = \frac{420}{480} = 0{,}8750.$

2. Precision

   Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario pertama untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

   a. Nilai precision kelas Abyssinian

      $$Precision = \frac{35}{35+3+0+0+1+0+0+0+0+0+1+0} = \frac{35}{40} = 0.88$$

   b. Nilai precision kelas Bengal

      $$Precision = \frac{36}{2+36+0+0+0+7+1+0+0+0+0+0} = \frac{36}{46} = 0.78$$

   c. Nilai precision kelas Birman

      $$Precision = \frac{34}{0+0+34+0+0+0+0+0+6+0+4+0} = \frac{34}{44} = 0.77$$

   d. Nilai precision kelas Bombay

$$Precision = \frac{46}{0+0+0+46+1+0+0+0+0+3+0+0} = \frac{46}{50} = 0.92$$

e. Nilai precision kelas British Shorthair

$$Precision = \frac{32}{0+0+0+0+32+0+0+1+0+2+0+0} = \frac{32}{35} = 0.91$$

f. Nilai precision kelas Egyptian Mau

$$Precision = \frac{35}{2+5+0+0+0+35+0+0+0+0+0+0} = \frac{35}{42} = 0.83$$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{41}{3+0+0+0+0+2+41+1+0+0+0+0} = \frac{41}{47} = 0.87$$

h. Nilai precision kelas Persian

$$Precision = \frac{35}{0+0+0+0+2+0+0+35+0+0+0+0} = \frac{35}{37} = 0.95$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{23}{0+0+5+0+0+0+2+0+23+0+1+0} = \frac{23}{31} = 0.74$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{32}{1+0+0+0+2+0+0+0+0+32+0+0} = \frac{32}{35} = 0.91$$

k. Nilai precision kelas Siamese

$$Precision = \frac{34}{0+0+1+1+0+0+0+0+0+0+34+0} = \frac{34}{36} = 0.94$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario pertama adalah:

$$\Sigma\, Precision = \frac{0.88+0.78+0.77+0.92+0.91+0.83+0.87+0.95+0.74+0.91+0.94+1.00}{0+0+0+0+0+0+0+0+0+0+0+37} =$$

$$\frac{10.5}{12} = 0.8750$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{35}{35+2+0+0+0+2+3+0+0+1+0+0} = \frac{35}{43} = 0.81$$

b. Nilai recall kelas Bengal

$$Recall = \frac{36}{3+36+0+0+0+5+0+0+0+0+0+0} = \frac{36}{44} = 0.82$$

c. Nilai recall kelas Birman

$$Recall = \frac{34}{0+0+34+0+0+0+0+0+5+0+1+0} = \frac{34}{40} = 0.85$$

d. Nilai recall kelas Bombay

$$Recall = \frac{46}{0+0+0+46+0+0+0+0+0+0+1+0} = \frac{46}{47} = 0.98$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{32}{1+0+0+1+32+0+0+2+0+2+0+0} = \frac{32}{38} = 0.84$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{35}{0+7+0+0+0+35+2+0+0+0+0+0} = \frac{35}{44} = 0.80$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{41}{0+1+0+0+0+2+41+0+2+0+0+0} = \frac{41}{44} = 0.93$$

h. Nilai recall kelas Persian

$$Recall = \frac{35}{0+0+0+0+1+0+1+35+0+0+0+0} = \frac{35}{37} = 0.95$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{23}{0+0+6+0+0+0+0+0+23+0+0+0} = \frac{23}{29} = 0.79$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{32}{0+0+0+3+2+0+0+0+0+32+0+0} = \frac{32}{37} = 0.86$$

k. Nilai recall kelas Siamese

$$Recall = \frac{34}{1+0+4+0+0+0+0+0+1+0+34+0} = \frac{34}{40} = 0.85$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario pertama adalah:

$$\Sigma Recall = \frac{0.81+0.82+0.85+0.98+0.84+0.80+0.83+0.95+0.79+0.86+0.85+1.00}{12} = \frac{10.48}{12} = 0.8733$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario pertama untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2\times0.88\times0.81}{0.88+0.81} = 0.84$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.78\times0.82}{0.78+0.82} = 0.80$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.77\times0.85}{0.77+0.85} = 0.81$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.92\times0.98}{0.92+0.98} = 0.95$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times0.91\times0.84}{0.91+0.84} = 0.88$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times0.92\times0.98}{0.92+0.98} = 0.81$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times0.87\times0.93}{0.87+0.93} = 0.90$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.95\times0.95}{0.95+0.95} = 0.95$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.74\times0.79}{0.74+0.79} = 0.77$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.91\times0.86}{0.91+0.86} = 0.89$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times0.94\times0.85}{0.94+0.85} = 0.89$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times1.00\times1.00}{1.00+1.00} = 1.00$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario pertama adalah:

$$\Sigma\, F1score = \frac{0.84+0.80+0.81+0.95+0.88+0.81+0.90+0.95+0.77+0.89+0.89+1.00}{12} = 0.8742$$

Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan *classification report* agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.18.

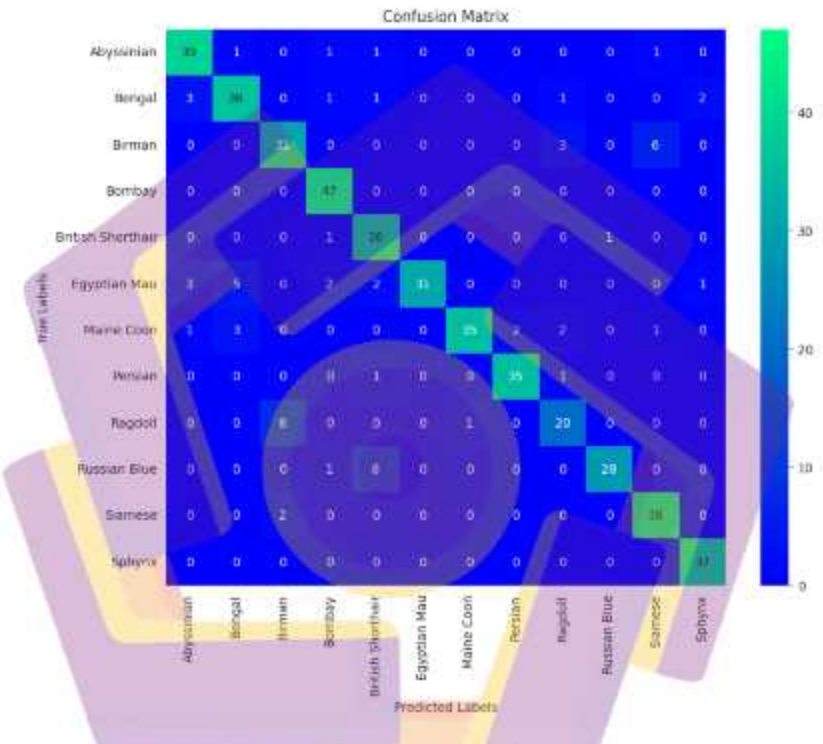
Tabel 4. 18. Classification Report pada Skenario Pertama

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	0.88	0.81	0.84
Bengal	0.78	0.82	0.80
Birman	0.77	0.85	0.81
Bombay	0.92	0.98	0.95
British Shorthair	0.91	0.84	0.88
Egyptian Mau	0.83	0.80	0.81
Maine Coon	0.87	0.93	0.90
Persian	0.95	0.95	0.95
Ragdoll	0.74	0.79	0.77
Russian Blue	0.91	0.86	0.89
Siamese	0.94	0.85	0.89
Sphynx	1.00	1.00	1.00
Accuracy			**0.88**

Classification report yang ditampilkan pada tabel 4.19 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan

diatas, pengujian pada skenario pertama diketahui menghasilkan nilai accuracy 0.8750, precision 0.8750, recall 0.8733, dan f1-score 0.8742.

4.5.3.2.Skenario kedua



Gambar 4. 5. Confusion Matrix skenario kedua

Gambar 4.5 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 39 data di prediksi benar dan 8 data di prediksi salah. Begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TN (True Negative), FP (False Positive), dan FN

(False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

   Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 39 + 36 + 31 + 47 + 36 + 31 + 35 + 35 + 20 + 28 + 38 + 37 = 413, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

   $Akurasi = \frac{413}{480} = 0.8604$

2. Precision

   Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario pertama untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

   a. Nilai precision kelas Abyssinian

      $Precision = \frac{39}{39+3+0+0+0+3+1+0+0+0+0+0} = \frac{39}{46} = 0.85$

   b. Nilai precision kelas Bengal

      $Precision = \frac{36}{1+36+0+0+0+5+3+0+0+0+0+0} = \frac{36}{45} = 0.80$

   c. Nilai precision kelas Birman

      $Precision = \frac{31}{0+0+31+0+0+0+0+0+8+0+2+0} = \frac{31}{41} = 0.76$

   d. Nilai precision kelas Bombay

      $Precision = \frac{47}{1+1+0+47+1+2+0+0+0+1+0+0} = \frac{47}{53} = 0.89$

   e. Nilai precision kelas British Shorthair

$$Precision = \frac{36}{1+1+0+0+36+2+0+1+0+8+0+0} = \frac{36}{49} = 0.73$$

f. Nilai precision kelas Egyptian Mau

$$Precision = \frac{31}{0+0+0+0+0+31+0+0+0+0+0+0} = \frac{31}{31} = 1.00$$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{35}{0+0+0+0+0+0+35+0+1+0+0+0} = \frac{35}{36} = 0.97$$

h. Nilai precision kelas Persian

$$Precision = \frac{35}{0+0+0+0+0+0+2+35+0+0+0+0} = \frac{35}{37} = 0.95$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{20}{0+1+3+0+0+0+2+1+20+0+0+0} = \frac{20}{27} = 0.74$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{28}{0+0+0+0+0+2+0+0+0+0+32+0+0} = \frac{32}{34} = 0.97$$

k. Nilai precision kelas Siamese

$$Precision = \frac{38}{1+0+6+0+0+0+1+0+0+0+38+0} = \frac{38}{46} = 0.83$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{37}{0+0+2+0+0+0+0+1+0+0+0+0+37} = \frac{37}{40} = 0.93$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario kedua adalah:

$$\Sigma\, Precision = \frac{0.85+0.80+0.76+0.89+0.73+1.00+0.97+0.95+0.74+0.97+0.83+0.93}{12} = \frac{10.42}{12} = 0.8683$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{39}{39+1+0+1+1+0+0+0+0+0+1+0} = \frac{39}{43} = 0.91$$

b. Nilai recall kelas Bengal

$$Recall = \frac{36}{3+36+0+1+1+0+0+0+1+0+0+2} = \frac{36}{44} = 0.82$$

c. Nilai recall kelas Birman

$$Recall = \frac{31}{0+0+31+0+0+0+0+0+3+0+6+0} = \frac{31}{40} = 0.78$$

d. Nilai recall kelas Bombay

$$Recall = \frac{47}{0+0+0+47+0+0+0+0+0+0+0+0} = \frac{47}{47} = 1.00$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{36}{0+0+0+1+36+0+0+0+0+1+0+0} = \frac{36}{38} = 0.95$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{31}{3+5+0+2+2+31+0+0+0+0+0+1} = \frac{31}{44} = 0.70$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{35}{1+3+0+0+0+0+35+2+2+0+1+0} = \frac{35}{44} = 0.80$$

h. Nilai recall kelas Persian

$$Recall = \frac{35}{0+0+0+0+1+0+0+35+1+0+0+0} = \frac{35}{37} = 0.95$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{20}{0+0+8+0+0+0+1+0+20+0+0+0} = \frac{20}{29} = 0.69$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{28}{0+0+0+1+8+0+0+0+0+28+0+0} = \frac{28}{37} = 0.76$$

k. Nilai recall kelas Siamese

$$Recall = \frac{38}{0+0+2+0+0+0+0+0+0+0+38+0} = \frac{38}{40} = 0.95$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario kedua adalah:

$$\Sigma\, Recall = \frac{0.91+0.82+0.78+1.00+0.95+0.70+0.80+0.95+0.69+0.76+0.95+1.00}{12} =$$

$$\frac{10.31}{12} = 0.8592$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario kedua untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2\times0.85\times0.91}{0.85+0.91} = 0.88$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.80\times0.82}{0.80+0.82} = 0.81$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.76\times0.78}{0.76+0.78} = 0.77$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.89\times1.00}{0.89+1.00} = 0.94$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times0.73\times0.95}{0.73+0.95} = 0.83$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times1.00\times0.70}{1.00+0.70} = 0.83$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times0.97\times0.80}{0.97+0.80} = 0.88$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.95\times0.95}{0.95+0.95} = 0.95$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.74\times0.69}{0.74+0.69} = 0.71$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.97\times0.76}{0.97+0.76} = 0.85$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times0.83\times0.95}{0.83+0.95} = 0.88$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times0.93\times1.00}{0.93+1.00} = 0.96$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario kedua adalah:

$$\Sigma\, F1score = \frac{0.88+0.81+0.77+0.94+0.83+0.83+0.88+0.95+0.71+0.85+0.88+0.96}{12} = 0.8575$$

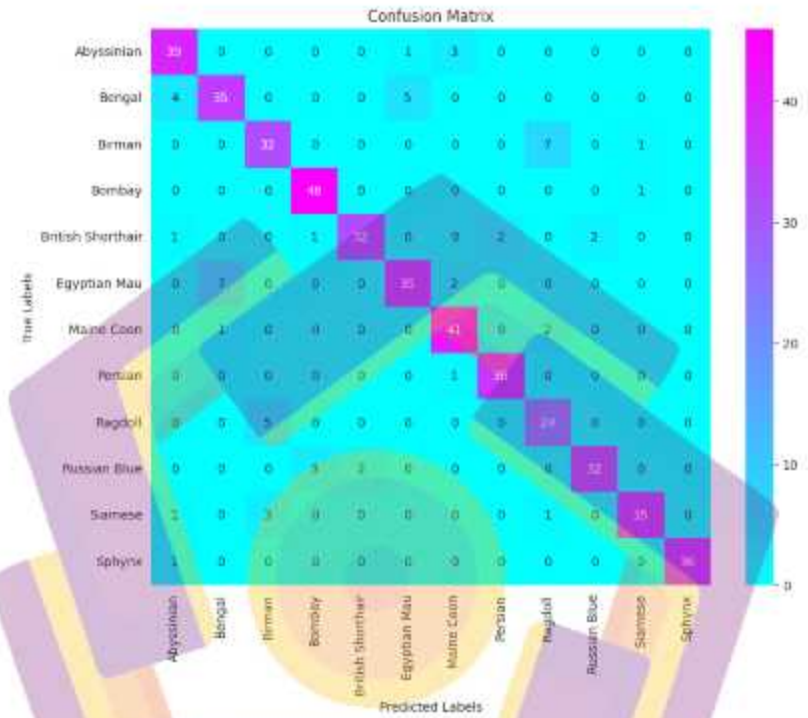
Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan *classification report* agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.19.

Tabel 4. 19. Classification Report pada Skenario Kedua

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	0.85	0.91	0.88
Bengal	0.80	0.82	0.81
Birman	0.76	0.78	0.77
Bombay	0.89	1.00	0.94
British Shorthair	0.73	0.95	0.83
Egyptian Mau	1.00	0.70	0.83
Maine Coon	0.97	0.80	0.88
Persian	0.95	0.95	0.95
Ragdoll	0.74	0.69	0.71
Russian Blue	0.97	0.76	0.85
Siamese	0.83	0.95	0.88
Sphynx	0.93	1.00	0.96
Accuracy			**0.86**

Classification report yang ditampilkan pada tabel 4.20 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan diatas, pengujian pada skenario kedua diketahui menghasilkan nilai accuracy 0.8604, precision 0.8683, recall 0.8592, dan f1-score 0.8575.

4.5.3.3.Skenario ketiga



Gambar 4. 6. Confusion Matrix skenario ketiga

Gambar 4.6 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 39 data di prediksi benar dan 4 data di prediksi salah. Begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TNs (True Negative), FP (False Positive), dan FN (False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 39 + 35+ 32 + 46 + 32 + 35 + 41 + 36 + 24 + 32 + 35 + 36 = 423, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

$Akurasi = \frac{423}{480} = 0.8812$

2. Precision

Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario keempat untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

a. Nilai precision kelas Abyssinian

$Precision = \frac{39}{39+4+0+0+1+0+0+0+0+0+1+1} = \frac{39}{46} = 0.85$

b. Nilai precision kelas Bengal

$Precision = \frac{35}{0+35+0+0+0+7+1+0+0+0+0+0} = \frac{35}{43} = 0.81$

c. Nilai precision kelas Birman

$Precision = \frac{32}{0+0+32+0+0+0+0+0+5+0+3+0} = \frac{32}{40} = 0.80$

d. Nilai precision kelas Bombay

$Precision = \frac{46}{0+0+0+46+1+0+0+0+0+3+0+0} = \frac{46}{50} = 0.92$

e. Nilai precision kelas British Shorthair

$Precision = \frac{32}{0+0+0+0+32+0+0+0+0+2+0+0} = \frac{32}{34} = 0.94$

f. Nilai precision kelas Egyptian Mau

$Precision = \frac{35}{1+5+0+0+0+35+0+0+0+0+0+0} = \frac{35}{41} = 0.85$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{41}{3+0+0+0+0+2+41+1+0+0+0+0} = \frac{41}{47} = 0.87$$

h. Nilai precision kelas Persian

$$Precision = \frac{36}{0+0+0+0+2+0+0+36+1+0+0+0} = \frac{36}{38} = 0.92$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{24}{0+0+7+0+0+0+2+0+24+0+1+0} = \frac{24}{34} = 0.71$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{32}{0+0+0+0+2+0+0+0+0+28+0+0} = \frac{32}{34} = 0.94$$

k. Nilai precision kelas Siamese

$$Precision = \frac{35}{0+0+1+1+0+0+0+0+0+0+35+0} = \frac{35}{37} = 0.95$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{36}{0+0+0+0+0+0+0+0+0+0+0+36} = \frac{36}{36} = 1.00$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario ketiga adalah:

$$\Sigma\, Precision = \frac{0.85+0.81+0.80+0.92+0.94+0.85+0.87+0.92+0.71+0.94+0.95+1.00}{12} =$$

$$\frac{10.56}{12} = 0.8800$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{39}{39+0+0+0+0+1+3+0+0+0+0+0} = \frac{39}{43} = 0.91$$

b. Nilai recall kelas Bengal

$$Recall = \frac{35}{4+35+0+0+0+5+0+0+0+0+0+0} = \frac{35}{44} = 0.80$$

c. Nilai recall kelas Birman

$$Recall = \frac{32}{0+0+32+0+0+0+0+0+7+0+1+0} = \frac{32}{40} = 0.80$$

d. Nilai recall kelas Bombay

$$Recall = \frac{46}{0+0+0+46+1+0+0+0+0+0+0+0} = \frac{46}{47} = 0.98$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{32}{1+0+0+1+32+0+0+2+0+2+0+0} = \frac{32}{38} = 0.84$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{35}{0+7+0+0+0+35+2+0+0+0+0+0} = \frac{35}{44} = 0.80$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{41}{0+1+0+0+0+0+41+0+2+0+0+0} = \frac{41}{44} = 0.93$$

h. Nilai recall kelas Persian

$$Recall = \frac{36}{0+0+0+0+0+0+1+36+0+0+0+0} = \frac{36}{37} = 0.97$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{24}{0+0+5+0+0+0+0+0+24+0+0+0} = \frac{24}{29} = 0.83$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{32}{0+0+0+3+2+0+0+0+0+32+0+0} = \frac{32}{37} = 0.86$$

k. Nilai recall kelas Siamese

$$Recall = \frac{35}{1+0+3+0+0+0+0+0+1+0+35+0} = \frac{35}{40} = 0.88$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{36}{1+0+0+0+0+0+0+0+0+0+0+36} = \frac{36}{37} = 0.97$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario ketiga adalah:

$$\Sigma\, Recall = \frac{0.91+0.80+0.80+0.98+0.84+0.80+0.93+0.97+0.83+0.86+0.88+0.97}{12} =$$

$$\frac{10.57}{12} = 0.8808$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario ketiga untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2\times0.85\times0.91}{0.85+0.91} = 0.88$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.81\times0.80}{0.81+0.80} = 0.80$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.80\times0.80}{0.80+0.80} = 0.80$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.92\times0.98}{0.92+0.98} = 0.95$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times0.94\times0.84}{0.94+0.84} = 0.89$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times0.85\times0.80}{0.85+0.80} = 0.82$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times0.87\times0.93}{0.87+0.93} = 0.90$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.95\times0.97}{0.95+0.97} = 0.96$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.71\times0.83}{0.71+0.83} = 0.77$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.94\times0.86}{0.94+0.86} = 0.90$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times0.95\times0.88}{0.95+0.88} = 0.91$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times1.00\times0.97}{1.00+0.97} = 0.99$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario ketiga adalah:

$$\Sigma\, F1score = \frac{0.88+0.80+0.80+0.95+0.89+0.82+0.90+0.96+0.76+0.90+0.91+0.99}{12} =$$

0.8800

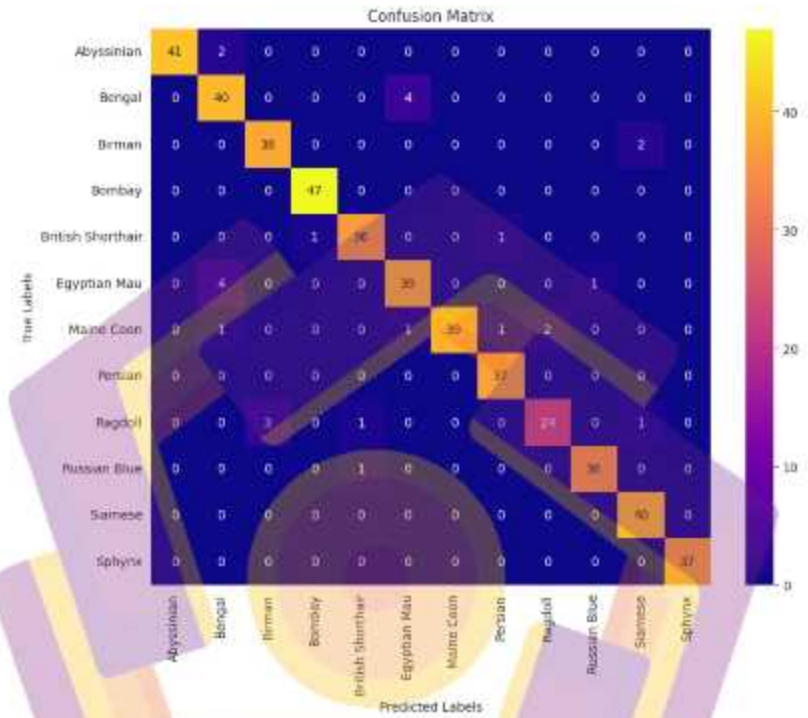
Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan classification report agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.20.

Tabel 4. 20. Classification Report pada Skenario Ketiga

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	0.85	0.91	0.88
Bengal	0.81	0.80	0.80
Birman	0.80	0.80	0.80
Bombay	0.92	0.98	0.95
British Shorthair	0.94	0.84	0.89
Egyptian Mau	0.85	0.80	0.82
Maine Coon	0.87	0.93	0.90
Persian	0.95	0.97	0.96
Ragdoll	0.71	0.83	0.76
Russian Blue	0.94	0.86	0.90
Siamese	0.95	0.88	0.91
Sphynx	1.00	0.97	0.99
Accuracy			**0.88**

Classification report yang ditampilkan pada tabel 4.21 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan diatas, pengujian pada skenario pertama diketahui menghasilkan nilai accuracy 0.8812, precision 0.8800, recall 0.8808, dan f1-score 0.8800.

4.5.3.4.Skenario keempat



Gambar 4. 7. Confusion Matrix skenario keempat

Gambar 4.7 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 41 data di prediksi benar dan 2 data di prediksi salah dan begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TN (True Negative), FP (False Positive), dan FN (False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 41 + 40+ 38 + 47 + 36 + 39 + 39 + 37 + 24 + 36 + 40 + 37 = 454, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

$Akurasi = \frac{454}{480} = 0.9458$

2. Precision

Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario keempat untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

a. Nilai precision kelas Abyssinian

$$Precision = \frac{41}{41+0+0+0+0+0+0+0+0+0+0} = \frac{41}{41} = 1.00$$

b. Nilai precision kelas Bengal

$$Precision = \frac{40}{2+40+0+0+0+4+1+0+0+0+0+0} = \frac{40}{47} = 0.85$$

c. Nilai precision kelas Birman

$$Precision = \frac{38}{0+0+38+0+0+0+0+0+3+0+0+0} = \frac{38}{41} = 0.93$$

d. Nilai precision kelas Bombay

$$Precision = \frac{47}{0+0+0+47+1+0+0+0+0+3+0+0} = \frac{47}{48} = 0.98$$

e. Nilai precision kelas British Shorthair

$$Precision = \frac{36}{0+0+0+0+36+0+0+0+1+1+0+0} = \frac{36}{38} = 0.95$$

f. Nilai precision kelas Egyptian Mau

$$Precision = \frac{39}{0+4+0+0+0+39+1+0+0+0+0+0} = \frac{39}{44} = 0.89$$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{39}{0+0+0+0+0+0+39+0+0+0+0+0} = \frac{39}{39} = 1.00$$

h. Nilai precision kelas Persian

$$Precision = \frac{37}{0+0+0+0+1+0+1+37+0+0+0+0} = \frac{37}{39} = 0.95$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{24}{0+0+0+0+0+0+2+0+24+0+0+0} = \frac{24}{26} = 0.92$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{36}{0+0+0+0+0+1+0+0+0+36+0+0} = \frac{36}{37} = 0.97$$

k. Nilai precision kelas Siamese

$$Precision = \frac{40}{0+0+2+0+0+0+0+0+1+0+40+0} = \frac{40}{43} = 0.93$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario keempat adalah:

$$\Sigma\, Precision = \frac{1.00+0.85+0.93+0.98+0.95+0.89+1.00+0.95+0.92+0.97+0.93+1.00}{12} = \frac{11.37}{12} = 0.9475$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{41}{41+2+0+0+0+0+0+0+0+0+0} = \frac{41}{43} = 0.95$$

b. Nilai recall kelas Bengal

$$Recall = \frac{40}{0+40+0+0+0+1+0+0+0+0+0} = \frac{40}{44} = 0.91$$

c. Nilai recall kelas Birman

$$Recall = \frac{38}{0+0+38+0+0+0+0+0+0+2+0} = \frac{38}{40} = 0.95$$

d. Nilai recall kelas Bombay

$$Recall = \frac{47}{0+0+0+47+0+0+0+0+0+0+0} = \frac{47}{47} = 1.00$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{36}{0+0+0+1+36+0+0+1+0+0+0} = \frac{36}{38} = 0.95$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{39}{0+4+0+0+0+39+0+0+0+1+0+0} = \frac{39}{44} = 0.89$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{39}{0+1+0+0+0+1+39+1+2+0+0+0} = \frac{39}{44} = 0.89$$

h. Nilai recall kelas Persian

$$Recall = \frac{37}{0+0+0+0+0+0+0+37+0+0+0+0} = \frac{37}{37} = 1.00$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{24}{0+0+3+0+1+0+0+0+24+0+1+0} = \frac{24}{29} = 0.83$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{36}{0+0+0+0+1+0+0+0+0+36+0+0} = \frac{36}{37} = 0.97$$

k. Nilai recall kelas Siamese

$$Recall = \frac{40}{0+0+0+0+0+0+0+0+0+0+40+0} = \frac{40}{40} = 1.00$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario keempat adalah:

$$\Sigma\, Recall = \frac{0.95+0.91+0.95+1.00+0.95+0.89+0.89+1.00+0.83+0.97+1.00+1.00}{12} = \frac{11.34}{12} = 0.9450$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario keempat untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2\times1.00\times0.95}{1.00+0.95} = 0.98$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.85\times0.91}{0.85+0.91} = 0.88$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.93\times0.95}{0.93+0.95} = 0.94$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.98\times1.00}{0.98+1.00} = 0.99$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times0.95\times0.95}{0.95+0.95} = 0.95$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times0.89\times0.89}{0.89+0.89} = 0.89$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times1.00\times0.89}{1.00+0.89} = 0.94$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.95\times1.00}{0.95+1.00} = 0.97$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.92\times0.83}{0.92+0.83} = 0.87$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.97\times0.97}{0.97+0.97} = 0.97$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times0.93\times1.00}{0.93+1.00} = 0.96$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times1.00\times1.00}{1.00+1.00} = 1.00$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario keempat adalah:

$$\Sigma\, F1score = \frac{0.98+0.88+0.94+0.99+0.95+0.89+0.94+0.97+0.87+0.97+0.96+1.00}{12} = 0.9450$$

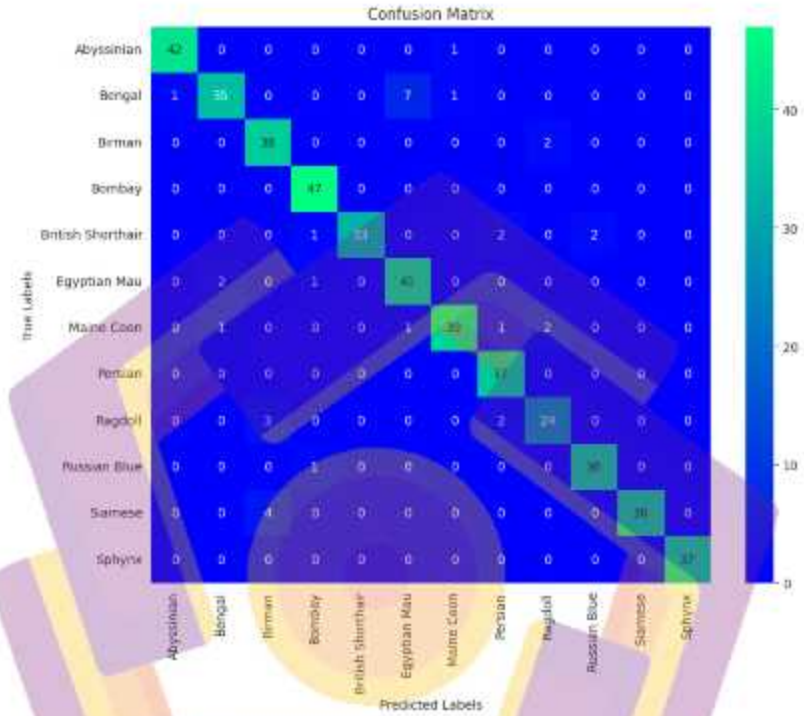
Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan classification report agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.21.

Tabel 4. 21. Classification Report pada Skenario Keempat

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	1.00	0.95	0.98
Bengal	0.85	0.91	0.88
Birman	0.93	0.95	0.94
Bombay	0.98	1.00	0.99
British Shorthair	0.95	0.95	0.95
Egyptian Mau	0.89	0.89	0.89
Maine Coon	1.00	0.89	0.94
Persian	0.95	1.00	0.97
Ragdoll	0.92	0.83	0.87
Russian Blue	0.97	0.97	0.97
Siamese	0.93	1.00	0.96
Sphynx	1.00	1.00	1.00
Accuracy			**0.95**

Classification report yang ditampilkan pada tabel 4.22 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan diatas, pengujian pada skenario keempat diketahui menghasilkan nilai accuracy 0.9458, precision 0.9475, recall 0.9450, dan f1-score 0.9450.

4.5.3.5.Skenario kelima



Gambar 4. 8. Confusion Matrix skenario keenam

Gambar 4.8 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 42 data di prediksi benar dan 1 data di prediksi salah. Begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TN (True Negative), FP (False Positive), dan FN (False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 42 + 35 + 38 + 47 + 33 + 41 + 39 + 37 + 24 + 36 + 36 + 37 = 445, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

$Akurasi = \frac{445}{480} = 0.9271$

2. Precision

Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario keempat untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

a. Nilai precision kelas Abyssinian

$$Precision = \frac{42}{42+1+0+0+0+0+0+0+0+0+0} = \frac{42}{43} = 0.98$$

b. Nilai precision kelas Bengal

$$Precision = \frac{35}{0+35+0+0+0+2+1+0+0+0+0} = \frac{35}{38} = 0.92$$

c. Nilai precision kelas Birman

$$Precision = \frac{38}{0+0+38+0+0+0+0+0+3+0+4+0} = \frac{38}{45} = 0.84$$

d. Nilai precision kelas Bombay

$$Precision = \frac{47}{0+0+0+47+1+1+0+0+0+1+0+0} = \frac{47}{50} = 0.94$$

e. Nilai precision kelas British Shorthair

$$Precision = \frac{33}{0+0+0+0+36+0+0+0+0+0+0+0} = \frac{33}{33} = 1.00$$

f. Nilai precision kelas Egyptian Mau

$$Precision = \frac{41}{0+7+0+0+0+41+1+0+0+0+0+0} = \frac{41}{49} = 0.84$$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{39}{1+1+0+0+0+0+39+0+0+0+0+0} = \frac{39}{41} = 0.95$$

h. Nilai precision kelas Persian

$$Precision = \frac{37}{0+0+0+0+2+0+1+37+2+0+0+0} = \frac{37}{42} = 0.88$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{24}{0+0+2+0+0+0+2+0+24+0+0+0} = \frac{24}{28} = 0.86$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{36}{0+0+0+0+2+0+0+0+0+36+0+0} = \frac{36}{38} = 0.95$$

k. Nilai precision kelas Siamese

$$Precision = \frac{36}{0+0+0+0+0+0+0+0+0+0+36+0} = \frac{36}{36} = 1.00$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario kelima adalah:

$$\Sigma\, Precision = \frac{0.98+0.92+0.84+0.94+1.00+0.84+0.95+0.88+0.86+0.95+1.00+1.00}{12} = \frac{11.16}{12} = 0.9300$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{42}{42+0+0+0+0+0+1+0+0+0+0+0} = \frac{42}{43} = 0.98$$

b. Nilai recall kelas Bengal

$$Recall = \frac{35}{1+35+0+0+0+7+1+0+0+0+0+0} = \frac{35}{44} = 0.80$$

c. Nilai recall kelas Birman

$$Recall = \frac{38}{0+0+38+0+0+0+0+0+2+0+0+0} = \frac{38}{40} = 0.95$$

d. Nilai recall kelas Bombay

$$Recall = \frac{47}{0+0+0+47+0+0+0+0+0+0+0+0} = \frac{47}{47} = 1.00$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{33}{0+0+0+1+33+0+0+2+0+2+0+0} = \frac{33}{38} = 0.87$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{41}{0+2+0+1+0+41+0+0+0+0+0+0} = \frac{41}{44} = 0.93$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{39}{0+1+0+0+0+1+39+1+2+0+0+0} = \frac{39}{44} = 0.89$$

h. Nilai recall kelas Persian

$$Recall = \frac{37}{0+0+0+0+0+0+0+37+0+0+0+0} = \frac{37}{37} = 1.00$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{24}{0+0+3+0+0+0+0+2+24+0+0+0} = \frac{24}{29} = 0.83$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{36}{0+0+0+1+0+0+0+0+0+36+0+0} = \frac{36}{37} = 0.97$$

k. Nilai recall kelas Siamese

$$Recall = \frac{36}{0+0+4+0+0+0+0+0+0+0+40+0} = \frac{36}{40} = 0.90$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario kelima adalah:

$$\Sigma\, Recall = \frac{0.98+0.80+0.95+1.00+0.87+0.93+0.89+1.00+0.83+0.97+0.90+1.00}{12} =$$

$$\frac{11.12}{12} = 0.9267$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario kelima untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2 \times 0.98 \times 0.98}{0.98+0.98} = 0.98$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.92\times0.80}{0.92+0.80} = 0.85$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.84\times0.95}{0.84+0.95} = 0.89$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.94\times1.00}{0.94+1.00} = 0.97$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times1.00\times0.87}{1.00+0.87} = 0.93$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times0.84\times0.93}{0.84+0.93} = 0.88$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times0.95\times0.89}{0.95+0.89} = 0.92$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.88\times1.00}{0.88+1.00} = 0.94$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.86\times0.83}{0.86+0.83} = 0.84$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.95\times0.97}{0.95+0.97} = 0.96$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times1.00\times0.90}{1.00+0.90} = 0.95$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times1.00\times1.00}{1.00+1.00} = 1.00$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario kelima adalah:

$$\Sigma\, F1score = \frac{0.98+0.85+0.89+0.97+0.93+0.88+0.92+0.94+0.84+0.96+0.95+1.00}{12} = 0.9258$$

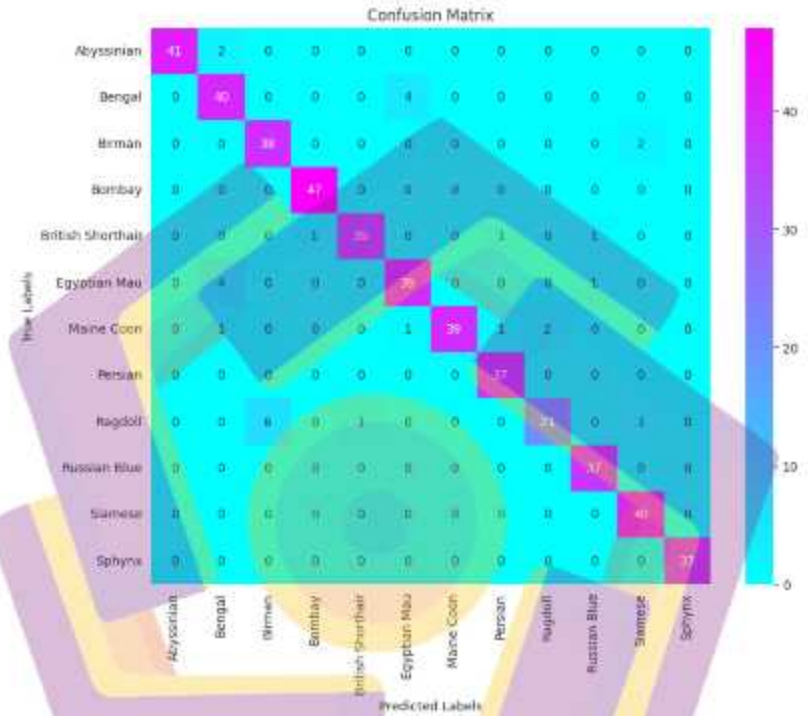
Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan classification report agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.22.

Tabel 4. 22. Classification Report pada Skenario Kelima

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	0.98	0.98	0.98
Bengal	0.92	0.80	0.85
Birman	0.84	0.95	0.89
Bombay	0.94	1.00	0.97
British Shorthair	1.00	0.87	0.93
Egyptian Mau	0.84	0.93	0.88
Maine Coon	0.95	0.89	0.92
Persian	0.88	1.00	0.94
Ragdoll	0.86	0.83	0.84
Russian Blue	0.95	0.97	0.96
Siamese	1.00	0.90	0.95
Sphynx	1.00	1.00	1.00
Accuracy			**0.93**

Classification report yang ditampilkan pada tabel 4.23 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan diatas, pengujian pada skenario kelima diketahui menghasilkan nilai accuracy 0.9271, precision 0.9300, recall 0.9267, dan f1-score 0.9258.

4.5.3.6.Skenario keenam



Gambar 4. 9. Confusion Matrix skenario keenam

Gambar 4.9 adalah hasil pengujian dari confusion matrix yang dimana dari 403 data terbagi dalam kelas, misalnya saja untuk kelas Abyssinian terdapat 41 data di prediksi benar dan 2 data di prediksi salah. Begitu juga dengan kelas yang lainnya.

Dalam menentukan nilai akurasim precision, recall, dan f1-score diperlukan istilah yaitu TP (True Positive), TN (True Negative), FP (False Positive), dan FN

(False Negative). Maka hasil pengujian confusion matrix yang dapat dihitung sebagai berikut:

1. Akurasi

   Nilai akurasi didapatkan dari keseluruhan jumlah TP dari masing-masng kelas. Jumlah nilai TP adalah 41 + 40 + 38 + 47 + 35 + 39 + 39 + 37 + 21 + 37 + 40 + 37 = 451, kemudian jumlah data secara keseluruhan adalah 480 data. Nilai akurasi adalah total TP dibagi dengan total data, maka:

   $$Akurasi = \frac{451}{480} = 0.9396$$

2. Precision

   Precision didapatkan dengan menghitung nilai TP dibagi dengan TP + FP, seperti pada skenario keenam untuk kasus pada confusion matrix multi class dalam menentukan nilai precision dihitung pada masing-masing kelas.

   a. Nilai precision kelas Abyssinian

      $$Precision = \frac{41}{41+0+0+0+0+0+0+0+0+0+0} = \frac{41}{41} = 1.00$$

   b. Nilai precision kelas Bengal

      $$Precision = \frac{40}{2+40+0+0+0+4+1+0+0+0+0+0} = \frac{40}{47} = 0.85$$

   c. Nilai precision kelas Birman

      $$Precision = \frac{38}{0+0+38+0+0+0+0+0+6+0+0+0} = \frac{38}{44} = 0.86$$

   d. Nilai precision kelas Bombay

      $$Precision = \frac{47}{0+0+0+47+1+0+0+0+0+0+0+0} = \frac{47}{48} = 0.98$$

   e. Nilai precision kelas British Shorthair

$$Precision = \frac{35}{0+0+0+0+35+0+0+0+1+0+0+0} = \frac{35}{36} = 0.97$$

f. Nilai precision kelas Egyptian Mau

$$Precision = \frac{39}{0+4+0+0+0+39+1+0+0+0+0+0} = \frac{39}{44} = 0.89$$

g. Nilai precision kelas Maine Coon

$$Precision = \frac{39}{0+0+0+0+0+0+39+0+0+0+0+0} = \frac{39}{39} = 1.00$$

h. Nilai precision kelas Persian

$$Precision = \frac{37}{0+0+0+0+1+0+1+37+0+0+0+0} = \frac{37}{39} = 0.95$$

i. Nilai precision kelas Ragdoll

$$Precision = \frac{21}{0+0+0+0+0+0+2+0+21+0+0+0} = \frac{21}{23} = 0.91$$

j. Nilai precision kelas Russian Blue

$$Precision = \frac{37}{0+0+0+1+1+0+0+0+0+37+0+0} = \frac{37}{39} = 0.95$$

k. Nilai precision kelas Siamese

$$Precision = \frac{40}{0+0+2+0+0+0+0+0+1+0+40+0} = \frac{40}{43} = 0.93$$

l. Nilai precision kelas Sphynx

$$Precision = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil precision secara keseluruhan didapatkan dengan menjumlahkan semua nilai precision dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga precision dari skenario kelima adalah:

$$\Sigma\, Precision = \frac{1.00+0.85+0.86+0.98+0.97+0.89+1.00+0.95+0.91+0.95+0.93+1.00}{12} =$$

$$\frac{11.29}{12} = 0.9408$$

3. Recall

Recall didapatkan dengan menghitung nilai TP dibagi dengan TP + FN, namun untuk kasus pada confusion matrix multi class dalam menentukan nilai recall dihitung pada masing-masing kelas.

a. Nilai recall kelas Abyssinian

$$Recall = \frac{41}{41+2+0+0+0+0+1+0+0+0+0+0} = \frac{41}{43} = 0.95$$

b. Nilai recall kelas Bengal

$$Recall = \frac{40}{0+40+0+0+0+1+0+0+0+0+0+0} = \frac{40}{44} = 0.91$$

c. Nilai recall kelas Birman

$$Recall = \frac{38}{0+0+38+0+0+0+0+0+0+0+2+0} = \frac{38}{40} = 0.95$$

d. Nilai recall kelas Bombay

$$Recall = \frac{47}{0+0+0+47+0+0+0+0+0+0+0+0} = \frac{47}{48} = 1.00$$

e. Nilai recall kelas British Shorthair

$$Recall = \frac{35}{0+0+0+1+35+0+0+1+0+1+0+0} = \frac{35}{38} = 0.92$$

f. Nilai recall kelas Egyptian Mau

$$Recall = \frac{39}{0+4+0+0+0+39+0+0+0+1+0+0} = \frac{39}{44} = 0.89$$

g. Nilai recall kelas Maine Coon

$$Recall = \frac{39}{0+1+0+0+0+1+39+1+2+0+0+0} = \frac{39}{44} = 0.89$$

h. Nilai recall kelas Persian

$$Recall = \frac{37}{0+0+0+0+0+0+0+37+0+0+0+0} = \frac{37}{37} = 1.00$$

i. Nilai recall kelas Ragdoll

$$Recall = \frac{21}{0+0+6+0+1+0+0+0+21+0+1+0} = \frac{21}{29} = 0.72$$

j. Nilai recall kelas Russian Blue

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+37+0+0} = \frac{37}{37} = 1.00$$

k. Nilai recall kelas Siamese

$$Recall = \frac{40}{0+0+4+0+0+0+0+0+0+0+40+0} = \frac{40}{40} = 1.00$$

l. Nilai recall kelas Sphynx

$$Recall = \frac{37}{0+0+0+0+0+0+0+0+0+0+0+37} = \frac{37}{37} = 1.00$$

Hasil recall secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga recall dari skenario kelima adalah:

$$\Sigma\, Recall = \frac{0.95+0.91+0.95+1.00+0.92+0.89+0.89+1.00+0.72+1.00+1.00+1.00}{12} =$$

$$\frac{11.23}{12} = 0.9358$$

4. F1-Score

F1-score didapatkan dengan menghitung nilai 2 kali total precision dikalikan dengan total recall lalu dibagi dengan total precision ditambahkan dengan total recall, seperti pada skenario kelima untuk kasus pada confusion matrix multi class dalam menentukan nilai f1-score dihitung pada masing-masing kelas.

a. Nilai f1-score kelas Abyssinian

$$F1score = \frac{2\times1.00\times0.95}{1.00+0.95} = 0.98$$

b. Nilai f1-score kelas Bengal

$$F1score = \frac{2\times0.85\times0.91}{0.85+0.91} = 0.88$$

c. Nilai f1-score kelas Birman

$$F1score = \frac{2\times0.86\times0.95}{0.86+0.95} = 0.90$$

d. Nilai f1-score kelas Bombay

$$F1score = \frac{2\times0.98\times1.00}{0.98+1.00} = 0.99$$

e. Nilai f1-score kelas British Shorthair

$$F1score = \frac{2\times0.97\times0.92}{0.97+0.92} = 0.95$$

f. Nilai f1-score kelas Egyptian Mau

$$F1score = \frac{2\times0.89\times0.89}{0.89+0.89} = 0.89$$

g. Nilai f1-score kelas Maine Coon

$$F1score = \frac{2\times1.00\times0.89}{1.00+0.89} = 0.94$$

h. Nilai f1-score kelas Persian

$$F1score = \frac{2\times0.95\times1.00}{0.95+1.00} = 0.97$$

i. Nilai f1-score kelas Ragdoll

$$F1score = \frac{2\times0.91\times0.72}{0.91+0.72} = 0.81$$

j. Nilai f1-score kelas Russian Blue

$$F1score = \frac{2\times0.95\times1.00}{0.95+1.00} = 0.97$$

k. Nilai f1-score kelas Siamese

$$F1score = \frac{2\times0.93\times1.00}{0.93+1.00} = 0.96$$

l. Nilai f1-score kelas Sphynx

$$F1score = \frac{2\times1.00\times1.00}{1.00+1.00} = 1.00$$

Hasil f1-score secara keseluruhan didapatkan dengan menjumlahkan semua nilai recall dari masing-masing kelas yang kemudian dibagi dengan jumlah kelas. Sehingga f1-score dari skenario kelima adalah:

$$\Sigma\, F1score = \frac{0.98+0.88+0.90+0.99+0.95+0.89+0.94+0.97+0.81+0.97+0.96+1.00}{12} = 0.9367$$

Maka dari hasil perhitungan pada confusion matrix ini dapat disimpulkan menggunakan classification report agar mudah untuk menganalisanya yang bisa dilihat pada tabel 4.23.

Tabel 4. 23. Classification Report pada Skenario Keenam

Nama Kelas	Precision	Recall	F1-Score
Abyssinian	1.00	0.95	0.98
Bengal	0.85	0.91	0.88
Birman	0.90	0.95	0.93
Bombay	0.98	1.00	0.99
British Shorthair	0.97	0.92	0.95
Egyptian Mau	0.89	0.89	0.89
Maine Coon	1.00	0.89	0.94
Persian	0.95	1.00	0.97
Ragdoll	0.92	0.83	0.87
Russian Blue	0.95	1.00	0.97
Siamese	0.93	0.97	0.95
Sphynx	1.00	1.00	1.00
Accuracy			**0.94**

Classification report yang ditampilkan pada tabel 4.24 merupakan ringkasan dari performa evaluasi model klasifikasi. Pada precision, recall, dam f1-score masing-masing kelas sudah terhitung pada confusion matrix. Dari perhitungan diatas, pengujian pada skenario keenam diketahui menghasilkan nilai accuracy 0.9396, precision 0.9408, recall 0.9358, dan f1-score 0.9367.

4.5.4. Hasil Analisis Identifikasi per kelasnya

Di Skenario 1, meski model mencapai akurasi yang cukup tinggi yaitu 87.5%, beberapa kelas seperti Bengal dan Ragdoll terlihat mengalami kesulitan yang dimana dengan F1-score yang lebih rendah yaitu 0.80 dan 0.77. Hal ini menunjukkan bahwa ciri-ciri khusus dari kucing-kucing ini tidak diidentifikasi dengan baik oleh model dalam skenario ini.

Pada skenario 2 mengalami sedikit penurunan dalam akurasi secara keseluruhan yaitu 86.04% yang menunjukkan bahwa perubahan dalam data atau metode mungkin tidak selalu mengarah pada hasil yang lebih baik. Penurunan kinerja terutama terlihat pada kelas Birman dan Ragdoll yaitu 0.77 dan 0.71 (nilai F1-score), yang menandakan perlunya penyesuaian lebih lanjut dalam model untuk mengatasi keterbatasan ini.

Pada skenario 3 mengalami sedikit peningkatan dalam akurasi yaitu 88.12%) namun masih menunjukkan bahwa model tersebut mengalami kesulitan dengan kelas Ragdoll yaitu 0.77 (nilai F1-score). Hal ini menunjukkan adanya karakteristik unik pada kelas ini yang belum sepenuhnya ditangkap oleh model.

Pada skenario 4 mengalami perubahan signifikan dengan peningkatan besar dalam akurasi yaitu 94.58%. Peningkatan ini menunjukkan bahwa metode yang diterapkan dalam skenario ini sangat efektif dalam mengatasi kekurangan yang terlihat dalam skenario sebelumnya. Setiap kelas menunjukkan peningkatan yang signifikan, menunjukkan bahwa model ini lebih holistik dan bisa menangkap variasi dalam data lebih efektif.

Terakhir, pada skenario 5 dan 6 menunjukkan penurunan kecil dalam akurasi dibandingkan dengan skenario 4, namun masih mempertahankan performa yang sangat baik. Dalam hal ini menunjukkan bahwa model tersebut stabil dan mampu memberikan hasil yang konsisten meski terjadi perubahan dalam parameter atau data.

4.5.5. Hasil Perbandingan pada Pengujian dari Model

Pada hasil pengujian dari model tiap skenarionya dilakukan dengan membandingkan hasil pengujian berdasarkan dari nilai accuracy, precision, recall, dan f1-score dari masing-masing skenario. Pada gambar 4.4 menunjukkan hasil perbandingan pada pengujiannya.

Gambar 4. 10 Hasil perbandingan pada pengujian tiap skenario

Pada gambar 4.10 menunjukkan bahwa pada skenario keempat yaitu CNN Transfer Learning dengan arsitektur Xception mendapatkan hasil yang tertinggi dibandingkan dengan skenario yang lainnya dengan akurasi 0.9458 atau 94.58%.

Pada skenario keempat memiliki peningkatan akurasi, precision, dan recall yang signifikan dibandingkan dengan skenario lainnya. Hal ini menandakan bahwa model ini secara keseluruhan lebih baik dalam memprediksi label yang benar dan memiliki keseimbangan yang baik antara precision dan recall.

Selanjutnya pada skenario kedua mendapatkan hasil yang terendah dibandingkan dengan skenario yang lainnya dengan akurasi 0.8604 atau 86.04%. Pada skenario kedua yaitu CNN Transfer Learning dan Fine-Tuning dengan arsitektur Inception-V3 memiliki peningkatan dalam precision dibandingkan dengan skenario pertama, yang bisa menunjukkan bahwa model lebih selektif dalam memprediksi positif dan menghasilkan lebih sedikit false positives. Lalu, pada recall dan f1-score relatif lebih rendah daripada precision, yang bisa menunjukkan bahwa prediksi positif yang dibuat biasanya benar, model ini mungkin melewatkan beberapa true positives yaitu memiliki lebih banyak false negative.

Namun, pada skenario 3 dan 6 yang dimana model kombinasi CNN dan SVM baik arsitektur Inception-V3 maupun Xception tidak menunjukkan nilai akurasi yang tinggi dibandingkan dengan yang tidak menggunakan kombinasi. Dalam hal ini dapat disimpulkan bahwa arsitektur CNN yang digunakan sudah sangat efektif dalam mengekstraksi ciri dan melakuk an klasifikasi sehingga tambahan SVM tidak memberikan nilai lebih. Hal ini menunjukkan bahwa representasi fitur yang dipelajari oleh CNN sudah cukup kuat dan mampu mengklasifikasi data dengan baik. Penggabungan dua model yang kompleks tidak selalu menghasilkan peningkatan performa.

4.5.6. Hasil Analisis dengan Penelitian Sebelumnya

Berikut adalah perbandingan dengan penelitian sebelumnya yang terbaik yang berkaitan dengan klasifikasi ras kucing yang ditujukan pada tabel 4.24.

Tabel 4. 24. Perbandingan Penelitian Sebelumnya

Peneliti	Dataset	Jumlah Kelas	Model yang terbaik	Akurasi
(Afif et al., 2020)	Oxford-IIIT Pet	12	CNN Transfer Learning dan Fine-Tuning Arsitektur Xception	93.75%
(Imanuel & Setiabudi, 2022)	Oxford-IIIT Pet	12	CNN Pre-Trained arsitektur Xception	89.59%
(X. Zhang et al., 2019)	The Cat Fanciers' Association (CFA)	14	SSD Mobilenet_v1 FPN	81.74%.
(Qatrunnada et al., 2022)	Cat Breeds Dataset (Kaggle)	5	CNN multi-object	94%
(Dai et al., 2021)	Oxford-IIIT Pet	12	Mask R-CNN	87,54%
(Mahardi et al., 2020)	CDC (Cat and Dog Classifier)	21	CNN VGG19	84,07%
Penelitian yang diusulkan	Oxford-IIIT Pet	12	CNN Xception	94.58%
			CNN Xception dan fine-tuning	92.71%
			Kombinasi Xception-SVM	93.94%
			CNN Inception-V3	87.50%
			CNN Inception-V3 dan fine-tuning	86.04%
			Kombinasi Inception-V3 SVM	88.12%

Pada penelitian sebelumnya memang belum ada yang menggunakan metode kombinasi dengan SVM pada klasifikasi ras kucing. Namun, pada penelitian sebelumnya menggunakan deep learning khususnya CNN dengan menggunakan arsitektur yang sudah pernah dilatih sebelumnya yang dimana dapat mencapai akurasi yang cukup tinggi dibandingkan dengan metode lainnya seperti machine learning.

Pada penelitian yang dilakukan oleh (Afif et al., 2020) terlihat adanya penggunaan dataset Oxford-IIIT Pet dengan kategori yang mencakup 12 kelas berbeda. Dalam studi ini, teknik transfer learning diterapkan pada arsitektur CNN Xception, yang berujung pada pencapaian akurasi sebesar 93.75%. Sementara itu, (Imanuel & Setiabudi, 2022) mengadopsi pendekatan serupa dengan dataset yang sama, namun dengan penggunaan model CNN yang telah dilatih sebelumnya, dan mencatatkan akurasi sebesar 89.59%.

Di penelitian lainnya, (X. Zhang et al., 2019) mengambil langkah yang berbeda dengan mengeksplorasi dataset The Cat Fanciers' Association yang memuat 14 ras berbeda, dan menggunakan model SSD Mobilenet_v1 FPN yang menghasilkan akurasi sebesar 81.74%.

Penelitian oleh (Qatrunnada et al., 2022) meneliti penggunaan dataset Cat Breeds dari Kaggle yang hanya terdiri dari 5 kelas, dengan model CNN multi-objek yang berhasil mencapai akurasi yang mengesankan yaitu 94%.

(Dai et al., 2021) kembali ke dataset Oxford-IIIT Pet namun menggunakan model yang lebih kompleks, yaitu Mask R-CNN, dan berhasil meraih akurasi 87.54%. (Mahardi et al., 2020) menggunakan dataset yang lebih luas, yaitu CDC yang menggabungkan data dari 21 jenis anjing dan kucing, dengan pendekatan transfer learning pada model VGG19, yang menghasilkan akurasi sebesar 84.07%.

Terakhir, penelitian yang sedang dibahas mengambil metode yang berbeda dengan menggunakan kombinasi CNN-SVM dan menggunakan arsitektur Inception-V3 dan Xception dalam proses transfer learning. Pada karakteristik Arsitektur Inception-V3 dan Xception memiliki pendekatan arsitektural yang

berbeda dalam mengekstraksi fitur. Arsitektur Inception-V3 mungkin tidak seefisien Xception dalam konteks tertentu, sehingga penambahan SVM memberikan sedikit peningkatan dalam klasifikasi. Sedangkan pada arsitektur Xception dengan efisiensinya yang lebih tinggi, mungkin sudah mencapai batas akurasi yang dapat dicapai untuk dataset tertentu, sehingga penambahan SVM tidak memberikan manfaat yang signifikan. Pada kompleksitas model dan dataset, arsitektur Xception adalah model yang lebih kompleks dibandingkan dengan Inception-V3. Ketika dataset Oxford-IIIT Pet tidak terlalu kompleks atau variatif, model yang lebih sederhana dari arsiktetur Inception-V3 dengan penambahan SVM mungkin cukup untuk mencapai peningkatan akurasi. Sebaliknya, model yang lebih kompleks yaitu arsitektur Xception sudah cukup efektif sehingga SVM tidak memberikan perbedaan yang signifikan.

Hasil yang diperoleh menarik untuk diperhatikan, ketika menggunakan CNN dengan transfer learning pada Xception, model mencapai akurasi sebesar 94.58%. Namun, ketika CNN Xception dikombinasikan dengan SVM, sedikit penurunan terlihat dengan akurasi menjadi 93.96%. Sedangkan, ketika menggunakan CNN dengan transfer learning pada Inception-V3, model mencapai akurasi sebesar 87.50%. Namun, ketika CNN dikombinasikan dengan SVM, sedikit mengalami kenaikan dengan akurasi menjadi 88.12%. Hal ini menunjukkan bahwa efektivitas kombinasi CNN dan SVM bervariasi tergantung pada arsitektur CNN yang digunakan dan sifat dataset tertentu.

Perbedaan tersebut meski relatif kecil menandakan bahwa untuk penggunaan CNN yang dimodifikasi dengan teknik transfer learning pada arsitektur

Xception sudah sangat efektif. Dalam kasus ini, tidak mengalami peningkatan yang signifikan dalam performa. Hal ini dapat menunjukkan bahwa representasi fitur yang dihasilkan oleh CNN sudah cukup baik untuk klasifikasi ras kucing, sehingga penambahan SVM tidak banyak memberikan kontribusi tambahan.

Ada terdapat pada penelitian sebelumnya (Agarap, 2017) menganalisis bahwa model CNN-SVM memiliki performa yang baik dalam klasifikasi gambar, tetapi akurasinya sedikit lebih rendah yaitu 99.04% dibandingkan dengan model CNN-Softmax yaitu 99.23%. Pengujian menggunakan dataset MNIST dan Fashion-MNIST mengungkapkan bahwa sementara CNN-SVM berhasil mencapai akurasi tinggi, terdapat ruang untuk peningkatan. Penelitian ini menyarankan bahwa penggunaan model CNN yang lebih canggih dan teknik pra-pemrosesan data bisa meningkatkan hasil ini. Studi ini juga menyoroti potensi penggabungan SVM ke dalam CNN untuk klasifikasi gambar dan membuka jalur untuk penelitian lebih lanjut dalam area ini.

Kesimpulan ini dapat memberi implikasi penting dalam pemilihan arsitektur model untuk klasifikasi gambar, menunjukkan bahwa terkadang model yang lebih sederhana bisa memberikan hasil yang sebanding, atau bahkan lebih baik, dibandingkan dengan model yang lebih kompleks. Hal ini juga menggarisbawahi pentingnya validasi model yang ekstensif untuk menemukan kombinasi algoritma yang optimal, tergantung pada kasus penggunaan spesifik dan karakteristik datase

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan serangkaian tahapan skenario, analisis dan pembahasan yang telah dilakukan dalam penelitian ini dapat disimpulkan bahwa:

a. Metode kombinasi CNN dan SVM memberikan hasil akurasi yang cukup baik adalah 94.37% pada arsitektur Xception. Pada kombinasi CNN dan SVM arsitektur Inception-V3 dapat memberikan nilai akurasi yang lebih baik dari metode CNN Inception-V3 adalah 88.12% walaupun hasil akurasinya tidak lebih tinggi dari arsitektur Xception.
b. Metode CNN Transfer Learning dapat memberikan nilai akurasi yang tinggi menggunakan arsitektur Xception dengan nilai akurasi adalah 94.58%.
c. Dari hasil confusion matrix pada model CNN Transfer Learning arsitektur Xception menghasilkan nilai akurasi, precision, recall, dan f1-score adalah sebesar 94.58%, 94.80%, 94.50%, dan 94.50%.
d. Preprocessing dan augmentasi data yang digunakan memiliki dampak signifikan pada hasil. Penggunaan augmentasi yang tepat dapat meningkatkan kemampuan model dan mempengaruhi hasil akhir.
e. Model kombinasi CNN dan SVM baik arsitektur Inception-V3 maupun Xception tidak menunjukkan nilai akurasi yang tinggi dibandingkan dengan yang tidak menggunakan kombinasi. Dalam hal ini dapat disimpulkan

bahwa arsitektur CNN yang digunakan sudah sangat efektif dalam mengekstraksi ciri dan melakukan klasifikasi sehingga tambahan SVM tidak memberikan nilai lebih.

f. Parameter dan hyperparameter baik pada CNN maupun SVM dapat memainkan peran kritis dalam menentukan hasil. Parameter yang tepat dapat memaksimalkan potensi dari kombinasi dari CNN dan SVM.

g. Tantangan yang disajikan oleh tugas klasifikasi juga mempengaruhi pilihan arsitektur. Misalnya, Xception lebih cocok untuk klasifikasi yang lebih kompleks, sedangkan Inception-V3 lebih baik untuk yang lebih sederhana.

5.2. Saran

Untuk mendapatkan hasil pengujian yang benar-benar baik ada beberapa saran yang peneliti sampaikan adalah:

1. Pada dataset menggunakan selain dari *Oxford-IIIT Pet* agar bisa mengeksplorasi metode yang ingin dimplementasikan.
2. Jika menggunakan dataset dari Oxford-IIIT Pet direkomendasikan untuk meneliti lebih lanjut mengenai augmentasi data pada kelas yang memiliki nilai kecil seperti Bengal, Birman, dan Ragdoll dengan nilai dibawah 0.80.
3. Tambahkan kelas kumpulan data untuk ras kucing dengan karakteristik serupa.
4. Menggunakan metode preprocessing dan/atau augmentasi lainnya untuk mendapatkan citra yang lebih baik agar mendapatkan hasil akurasi yang maksimal.

5. Menggunakan metode selain CNN dan/atau kombinasi CNN-SVM agar mendapatkan hasil yang beragam.
6. Merekomendasikan penelitian lebih lanjut untuk mengeksplorasi kombinasi berbagai arsitektur CNN dengan SVM pada dataset yang berbeda dan dalam konteks tugas klasifikasi yang bervariasi, untuk lebih memahami kondisi-kondisi yang mempengaruhi efektivitas kombinasi tersebut.

DAFTAR PUSTAKA

Afif, M., Fawwaz, A., Ramadhani, K. N., & Sthevanie, F. (2020). Klasifikasi Ras pada Kucing menggunakan Algoritma Convolutional Neural Network(CNN). *Jurnal Tugas Akhir Fakultas Informatika*, *8*(1), 715–730.

Agarap, A. F. (2017). *An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification*. 5–8. http://arxiv.org/abs/1712.03541

Agustin, T., Purwidiantoro, M. H., Utami, E., & ... (2022). CNN and SVM Combination for Multi-Class Classification of Diabetic Retinopathy Based on Fundus Imaging. *…*, *15*(2), 108–119. http://ejournal.amikompurwokerto.ac.id/index.php/telematika/article/view/2086

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Aloysius, N., & Geetha, M. (2018). A review on deep convolutional neural networks. *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, *2018-Janua*, 588–592. https://doi.org/10.1109/ICCSP.2017.8286426

Audebert, N., Le Saux, B., & Lefevre, S. (2019). Deep learning for classification of hyperspectral data: A comparative review. *IEEE Geoscience and Remote Sensing Magazine*, *7*(2), 159–173. https://doi.org/10.1109/MGRS.2019.2912563

Awang Pona, K. Z., & K K, K. P. (2021). Hyperparameter Tuning of Deep learning Models in Keras. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing*, *01*(01), 36–40. https://doi.org/10.55011/staiqc.2021.1104

Azahro Choirunisa, N., Karlita, T., & Asmara, R. (2021). Deteksi Ras Kucing Menggunakan Compound Model Scaling Convolutional Neural Network. *Technomedia Journal*, *6*(2), 236–251. https://doi.org/10.33050/tmj.v6i2.1704

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A. L., Deng, D., & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *13*(2), 1–43. https://doi.org/10.1002/widm.1484

Bradley C. Love. (2022). Comparing supervised and unsupervised category learning. *Psychonomic Bulletin & Review*, *9*(4), 829–835.

Cervantes, J., Garcia-Lamont, F., Rodriguez-Mazahua, L., & Lopez, A. (2020). A

comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*(xxxx), 189–215. https://doi.org/10.1016/j.neucom.2019.10.118

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Dai, Y., Liu, Y., & Zhang, S. (2021). Mask R-CNN-based Cat Class Recognition and Segmentation. *Journal of Physics: Conference Series*, *1966*(1). https://doi.org/10.1088/1742-6596/1966/1/012010

Dong, N., Zhao, L., Wu, C. H., & Chang, J. F. (2020). Inception v3 based cervical cell classification combined with artificially extracted features. *Applied Soft Computing*, *93*, 106311. https://doi.org/10.1016/j.asoc.2020.106311

Ebrahimi, P., Salamzadeh, A., Soleimani, M., Khansari, S. M., Zarea, H., & Fekete-Farkas, M. (2022). Startups and Consumer Purchase Behavior: Application of Support Vector Machine Algorithm. *Big Data and Cognitive Computing*, *6*(2). https://doi.org/10.3390/bdcc6020034

Elhassouny, A., & Smarandache, F. (2019). Trends in deep convolutional neural Networks architectures: A review. *Proceedings of 2019 International Conference of Computer Science and Renewable Energies, ICCSRE 2019*, *1*, 1–8. https://doi.org/10.1109/ICCSRE.2019.8807741

Fossati, P., & Giancarlo, R. (2021). Purebred dogs and cats: A proposal for a better protection. *Journal of Veterinary Behavior*, *45*. *10.101*.

Imanuel, A., & Setiabudi, D. H. (2022). Penerapan Convolutional Neural Network dengan Pre-Trained Model Xception untuk Meningkatkan Akurasi dalam Mengidentifikasi Jenis Ras Kucing. *Jurnal Infra*, *10(2)*. www.kaggle.com

Ismail, A., Ahmad, S. A., Soh, A. C., Hassan, K., & Harith, H. H. (2019). Improving Convolutional Neural Network (CNN) architecture (miniVGGNet) with batch normalization and learning rate decay factor for image classification. *International Journal of Integrated Engineering*, *11*(4), 51–59. https://doi.org/10.30880/ijie.2019.11.04.006

Kading, C., Rodner, E., Freytaf, A., & Denzler, J. (2017). Fine-Tuning Deep Neural Networks in Continuous Learning Scenarios. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10116 LNCS*, 88–605. https://doi.org/10.1007/978-3-319-54526-4

Karlita, T., Choirunisa, N. A., Asmara, R., & Setyorini, F. (2022). Cat Breeds Classification Using Compound Model Scaling Convolutional Neural Networks. *Proceedings of the International Conference on Applied Science and Technology on Social Science 2021 (ICAST-SS 2021)*, *647*, 909–914.

https://doi.org/10.2991/assehr.k.220301.150

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. In *Artificial Intelligence Review* (Vol. 53, Issue 8). https://doi.org/10.1007/s10462-020-09825-6

Koklu, M., Unlersen, M. F., Ozkan, I. A., Aslan, M. F., & Sabanci, K. (2022). A CNN-SVM study based on selected deep features for grapevine leaves classification. *Measurement: Journal of the International Measurement Confederation*, *188*(January). https://doi.org/10.1016/j.measurement.2021.110425

Kong, B., Supančič, J., Ramanan, D., & Fowlkes, C. C. (2019). Cross-Domain Image Matching with Deep Feature Maps. *International Journal of Computer Vision*, *127*(11–12), 1738–1750. https://doi.org/10.1007/s11263-018-01143-3

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Mahardi, Wang, I. H., Lee, K. C., & Chang, S. L. (2020). Images Classification of Dogs and Cats using Fine-Tuned VGG Models. *2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020, ECICE 2020*, 230–233. https://doi.org/10.1109/ECICE50847.2020.9301918

Mahesh, B. (2020). Machine learning Algorithms-A Review. *International Journal of Science and Research (IJSR).[Internet]*, *9*(1), 381–386. https://doi.org/10.21275/ART20203995

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., & Hodjat, B. (2018). Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-815480-9.00015-3

Narkhede, S. (2018). *Understanding Confusion Matrix*. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Nath, S. S., Mishra, G., Kar, J., Chakraborty, S., & Dey, N. (2014). A survey of image classification methods and techniques. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014*, 554–557. https://doi.org/10.1109/ICCICCT.2014.6993023

Pan, Y., Jin, H., Gao, J., & Rauf, H. T. (2022). Identification of Buffalo Breeds Using Self-Activated-Based Improved Convolutional Neural Networks. *Agriculture*, *12*(9), 1386. https://doi.org/10.3390/agriculture12091386

Purnama, I. N. (2020). Herbal Plant Detection Based on Leaves Image Using Convolutional Neural Network With Mobile Net Architecture. *JITK (Jurnal Ilmu Pengetahuan Dan Teknologi Komputer)*, *6*(1), 27–32.

https://doi.org/10.33480/jitk.v6i1.1400

Qatrunnada, N., Fachrurrozi, M., & Syahrini, A. (2022). Cat Breeds Classification using Convolutional Neural Network for Multi-Object Image. *Sriwijaya Journal of Informatics and Applications*, *3*(2), 26–35.

Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. 1–14. http://arxiv.org/abs/1609.04747

Samsugi, S., & Naufal Falikh Suprapto, G. (2021). Otomatisasi Pakan Kucing Berbasis Mikrokontroller Intel Galileo Dengan Interface Android. *Jurnal Sains Komputer & Informatika (J-SAKTI*, *5*(1), 143–152.

Shaha, M., & Pawar, M. (2018). Transfer Learning for Image Classification. *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, *Iceca*, 656–660. https://doi.org/10.1109/ICECA.2018.8474802

Shinde, P. P., & Shah, D. S. (2018). A Review of Machine Learning and Deep Learning Applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–6. http://arxiv.org/abs/2104.12722

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Smith, L. N. (2017). Cyclical learning rates for training neural networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, *April 2015*, 464–472. https://doi.org/10.1109/WACV.2017.58

Tao, T., & Wei, X. (2022). A hybrid CNN–SVM classifier for weed recognition in winter rape field. *Plant Methods*, *18*(1), 1–12. https://doi.org/10.1186/s13007-022-00869-z

Xu, J., Zhang, Y., & Miao, D. (2020). Three-way confusion matrix for classification: A measure driven view. *Information Sciences*, *507*. https://doi.org/10.1016/j.ins.2019.06.064

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed Pooling for Convolutional Neural Networks. *The 9th International Conference on Rough Sets and Knowledge Technology (RSKT'14)*, *September*. https://doi.org/10.1007/978-3-319-11740-9

Yu, T., & Zhu, H. (2020). *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. 1–56. http://arxiv.org/abs/2003.05689

Zhang, M., Zhang, F., Lane, N. D., Shu, Y., Zeng, X., Fang, B., Yan, S., & Xu, H. (2020). Deep learning in the era of edge computing: Challenges and opportunities. *Fog Computing: Theory and Practice*, 67–78. https://doi.org/10.1002/9781119551713.ch3

Zhang, Q., Zhang, M., Chen, T., Sun, Z., Ma, Y., & Yu, B. (2019). Recent advances in convolutional neural network acceleration. *Neurocomputing*, *323*, 37–51. https://doi.org/10.1016/j.neucom.2018.09.038

Zhang, X., Yang, L., & Sinnott, R. (2019). A Mobile Application for Cat Detection and Breed Recognition Based on Deep Learning. *AI4Mobile 2019 - 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile*, 7–12. https://doi.org/10.1109/AI4Mobile.2019.8672684