Neptunus : Jurnal Ilmu Komputer Dan Teknologi Informasi
Volume. 2 No. 3 Agustus 2024
e-ISSN : 3031-898X, dan p-ISSN 3031-8998, Hal. 228-257
DOI: https://doi.org/10.61132/neptunus.v2i3.251
Available online at: https://journal.arteii.or.id/index.php/Neptunus

Analisa Komparasi Algoritma *Machine Learning* dan *Deep Learning* Dalam Klasifikasi Citra Ras Kucing

Royan Fajar Sultoni [1], Achmad Junaidi [2], Eva Yulia Puspaningrum [3]
[1,2,3] UPN "Veteran" Jawa Timur, Indonesia

Abstract, *Cats (Felis catus) are a type of carnivorous mammal from the Felidae family that was domesticated and has been one of the animals that has mingled with humans since time immemorial. Domestic cats are broadly divided into 2 types, namely village cats and purebred cats. Purebred cats have quite a varied number of types. Therefore, confusion often occurs in determining the type or breed of cat. Meanwhile, in practice, each race does not have the same treatment (especially in the aspect of care). In digital image processing, Machine Learning and Deep Learning are the main aspects in the process of applying technology that can overcome this problem, so research related to this problem was designed. This research was conducted to add insight for further research in a more sophisticated and effective image recognition process. In the experiments carried out in this research, the SVM, KNN, and CNN methods were tested with the Xception and EfficientNet-B1 architectures. Based on the final results obtained from this test, the CNN method with the Xception architecture is the best model. By using fine-tuning and a learning-rate of 1e-5, this method produces a micro average value of 0.974, on a cat breed image dataset of 13 classes and 7800 images. Meanwhile, the method that produces the fastest ETA Training and Testing is obtained by the KNN method, with an ETA Training time of 0.194 seconds, and an ETA Testing time of 1.782 seconds.*

Keywords*: Algorithm Comparison, Machine Learning, Deep Learning, SVM, KNN, CNN*

Abstrak, Kucing (*felis catus*) merupakan salah satu jenis mamalia karnivora dari *family Felidae* yang terdomestikasi dan menjadi salah satu hewan yang berbaur dengan manusia sejak dahulu kala. Kucing domestik secara garis besar dibedakan menjadi 2 jenis, yaitu kucing kampung dan kucing ras. Kucing ras memiliki kuantitas jenis yang tergolong cukup bervariasi. Oleh karenanya, seringkali terjadi kerancuan dalam menentukan jenis atau ras kucing. Sedangkan, pada prakteknya, setiap ras tidak memiliki perlakuan yang sama (terutama pada aspek perawatannya). Dalam pengolahan citra digital, *Machine Learning* dan *Deep Learning* merupakan aspek utama dalam proses pengaplikasian teknologi yang kiranya dapat mengatasi hal tersebut, sehingga dirancanglah penelitian terkait permasalahan tersebut. Penelitian ini dilakukan untuk menambah wawasan bagi penelitian yang lebih lanjut dalam proses pengenalan citra yang lebih mutakhir dan efektif. Pada percobaan yang dilakukan pada penelitian ini, diujikan metode SVM, KNN, dan CNN dengan arsitektur *Xception* dan *EfficientNet*-B1. Berdasakan hasil akhir yang diperoleh dari pengujian ini adalah, metode CNN dengan arsitektur *Xception* merupakan model terbaik. Dengan menggunakan *fine-tuning* serta *learning-rate* 1e-5, metode ini menghasilkan nilai *micro average* sebesar 0.974, pada dataset citra ras kucing sebanyak 13 kelas dan 7800 citra. Sedangkan, untuk metode yang menghasilkan ETA *Training* dan *Testing* yang paling cepat didapatkan oleh metode KNN, dengan waktu ETA *Training* sebesar 0.194 detik, dan ETA *Testing* sebesar 1.782 detik.

Kata kunci: Komparasi Algoritma*, Machine Learning, Deep Learning,* SVM, KNN, CNN

1. LATAR BELAKANG

Kucing (*felis catus*) merupakan salah satu jenis mamalia karnivora dari *family Felidae* yang terdomestikasi dan menjadi salah satu hewan yang berbaur dengan manusia sejak 6000 tahun SM (Choirunisa, Karlita and Asmara 2022). Kucing domestik secara garis besar dibedakan menjadi 2 jenis, yaitu kucing kampung dan kucing ras. Kucing campuran atau kucing kampung memiliki persentase jumlah yang lebih besar daripada kucing ras. Tetapi, kucing ras memiliki kuantitas jenis yang tergolong cukup bervariasi. Oleh karenanya, seringkali terjadi kerancuan dalam menentukan jenis atau ras kucing. Dilandasi fakta ini, belum tentu semua dari pemilik kucing dapat mengidentifikasi ras dari kucingnya. Sedangkan,

Received: Mei 28, 2024; Revised: Juni 19, 2024; Accepted: Juli 17, 2024; Online Available: Juli 19 2024;
* Royan Fajar Sultoni, bagas.adil10@gmail.com

setiap ras kucing memiliki karakteristik yang berbeda. Begitu pula dengan cara perawatan dari masing-masing ras kucing tersebut. Sehingga timbul sebuah pemikiran untuk melakukan penelitian yang memanfaatkan citra kucing terkait dengan fakta bahwa kebanyakan pemilik kucing yang tergolong awam dalam menentukan jenis ras kucingnya, dengan tujuan untuk menambah wawasan baru.

Pemanfaatan teknologi selalu linear dengan kebutuhan manusia. Dengan masifnya proses perkembangan teknologi, permasalahan penentuan ras kucing merupakan salah satu permasalahan yang dapat dipecahkan dengan pengaplikasian teknologi yang dimaksud. Dalam pengolahan citra digital, *Machine Learning* dan *Deep Learning* merupakan aspek utama dalam proses pengaplikasian teknologi tersebut, yang memiliki keunggulan masing-masing dalam peranannya untuk komputasi visual. Algoritma klasifikasi *Deep Learning* cenderung memiliki tingkat akurasi yang lebih tinggi jika dikomparasikan dengan *Machine Learning*. Tetapi, algoritma *Machine Learning* cenderung memiliki durasi pemrosesan komputasi yang lebih inferior dibandingkan durasi komputasi *Deep Learning* (Naufal and Kusuma 2023).

Dengan menerapkan pemanfaatan teknologi di bidang pengolahan citra digital, telah dilakukan beberapa penilitian terkait proses klasifikasi ras kucing menggunakan metode klasifikasi *Machine Learning* maupun *Deep Learning*. Pada penelitian sebelumya oleh (Kusuma, et al. 2022) menggunakan metode SVM dan *Naive Bayes* untuk proses klasifikasi ras kucing dengan citra sejumlah 910 buah yang terdiri dari 6 ras kucing, mendapatkan hasil bahwa metode SVM lebih baik jika dibandingkan dengan *Naive Bayes*, di mana SVM memiliki akurasi sebesar 88.4%, nilai precision sebesar 88.5%, dan juga nilai recall sebesar 88.4%. Terdapat juga penelitian lain yang dilakukan oleh (Suwarno and Mahastama 2023) dengan menggunakan *dataset* citra sidik jari pria dan wanita, yang juga menggunakan SVM sebagai metode klasifikasinya. Penelitian tersebut mendapatkan hasil akurasi hingga 70.3% (untuk tingkat *True Positive Rate*-nya sebesar 80.6% untuk sidik jari wanita, dan 60% untuk sidik jari pria). Penelitian selanjutnya yang dilakukan oleh (Choirunisa, Karlita and Asmara 2022) menggunakan metode klasifikasi CNN dengan arsitektur *EfficientNet-B0* dengan optimizer *RMSprop pada* proses klasifikasi citra kucing dengan jumlah *dataset* sebanyak 2700 citra yang terbagi menjadi 9 jenis ras kucing memiliki hasil paling maksimal sebesar 95% tingkat akurasi terhadap 180 citra kucing. Yang terakhir, pada penelitian yang dilakukan oleh (Naufal & Kusuma, 2023) pada klasifikasi citra SIBI (Sistem Isyarat Bahasa Indonesia) dengan menggunakan *Deep Learning* dengan beberapa arsitektur (*VGG-16*, *ResNet50*, *MobileNetV2*, dan *Xception*), mendapatkan hasil bahwa *Xception* merupakan arsitektur

dengan kemampuan klasifikasi yang memiliki akurasi hingga 99,5 persen. Yang terkahir, merupakan penelitian yang menggunakan metode KNN (dengan perbandingan terhadap metode WKNN) yang dilakukan oleh (Prasath et al., 2017), dan (Tarakci & Ozkan, 2021), yang menunujukkan pada beberapa dataset yang diujikan, KNN masih unggul dalam aspek akurasi dan juga ETA-nya.

2. TINJAUAN PUSTAKA

2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan salah satu algoritma *machine learning* yang mempunyai sifat terawasi (*supervised*), yang pada penggunaanya secara umum dipergunakan dalam permasalahan proses klasfikasi (Kusuma et al., 2022). Metode ini digagas oleh Vladimir N. Vapnik dan Aleksei Ya. Chervonenkis di tahun 1963 (Wijaya, 2023). Pada metode ini, item-item yang terangkum di dalam *dataset* akan diubah menjadi titik-titik koordinat tertentu pada sebuah ruang dengan dimensi sebanyak n. Kemudian model akan melakukan komputasi untuk menemukan sebuah *hyperplane* terbaik dan margin yang maksimal. Pada metode SVM terdapat sebuah fungsi yang bertujuan untuk pemetaan data yang disebut *kernel*. SVM memiliki beberapa jenis *kernel*, diantaranya adalah *linear*, *polynomial*, *sigmoid*, dan *Radial Basis Function* (RBF) *kernel*.

2.2 K-Nearest Neighbour (KNN)

K-Nearest Neighbour merupakan sebuah metode klasifikasi yang memanfaatkan nilai K yang digunakan sebagai sebuah objek yang didapati pada *dataset training*, tetapi dengan status yang terdekat dengan *dataset testing* (Tarisa Akbar et al., 2023). Pada metode KNN, tujuan utamanya adalah untuk mengidentifikasi *neighbours* atau "tetangga" terdekat dari sebuah *query point*, sehingga kita dapat memasukkan sebuah label *class* pada *point* tersebut. KNN memiliki kondisi yang harus dipenuhi, salah satunya adalah menentukan *distance* atau jarak matriks. Terdapat 4 jenis *distance*, yaitu *Euclidean*, *Manhattan*, *Minkowski*, dan *Hamming*. Untuk jenis jarak yang secara mayoritas digunakan, adalah *Euclidean*.

2.3 Arsitektur *Xception*

Xception merupakan salah satu arsitektur dari metode *Deep Learning* CNN. Xception tercipta dengan tujuan untuk membuat versi dari arsitektur *Inception* yang lebih mutakhir. Dengan menggunakan 1x1 konvolusi untuk memetakan korelasi antar saluran, dan kemudian akan memetakan korelasi spasial dari setiap saluran *output* secara terpisah. Arsitektur ini merupakan arsitektur yang berbasis pada *Depthwise Separable Convolution Layers* (arsitektur

Inception dimutakhirkan dengan memanfaatkan *depthwise separable convolutions* untuk menggantikan modul *Inceptions*) dan memiliki 36 *layer* konvolusi yang tergabung dalam proses ekstraksi fitur berdasarkan jaringan (Chollet, 2017). Arsitektur ini memiliki jumlah parameter yang cukup identik dengan *InceptionV3*.

2.4 Arsitektur *EfficientNet*

EfficientNet merupakan salah satu arsitektur yang dikembangkan oleh *Google* dengan memanfaatkan skala-*width-depth compound scaling* dengan tujuan mencapai kinerja yang lebih efektif dengan kuantitas parameter yang lebih rendah (Tan & Le, 2019). *EfficientNet* mendapatkan hasil akhir akurasi *top*-1 *state-of-the-art* sebanyak 84,3% pada pengujian dataset *ImageNet* dengan jumlah parameter sebanyak 66 juta, berbanding dengan arsitektur *GPipe* yang memiliki jumlah parameter sebanyak 556 juta (tetapi dengan hasil akurasi yang sama). Hal ini membuktikan bahwa *EfficientNet-B7* berukuran 8,4x lebih kecil dan 6,1x lebih cepat dibandingkan *Gpipe* .

2.5 Evaluasi Performa

Evaluasi Performa merupakan langkah penentuan atau pengkomparasian data hasil akhir pengujian antara masing-masing model klasifikasi (Solihin et al., 2023). Salah satu cara pengimplementasian proses evaluasi ini dapat menggunakan *Confusion Matrix* di mana terdapat 4 poin utama dalam penentuannya, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

2.6 Google Colaboratory

Google Colaboratory merupakan sebuah *cloud computing environment* yang disediakan oleh *Google* dan dapat diakses secara gratis. IDE ini berbasis *Python* dan R yang memungkinkan penggunanya untuk memanfaatkan semua *library* yang sudah tersedia dan dapat langsung dipanggil (berbasis *Jupyter Notebook*). Dikembangkan untuk keperluan penelitian terkait dengan *machine learning*, *deep learning*, maupun *data scientific*. *Google Colaboratory* sangat memudahkan penggunanya dalam kolaborasi antar tim, dikarenakan dapat dibagikan dalam lingkup yang diinginkan pengguna.

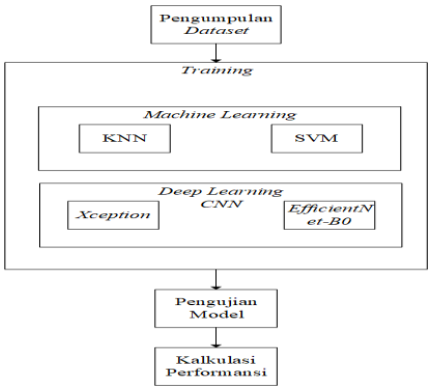
2.7 *Optimizer Adaptive Momentum*

Optimizer merupakan sebuah algoritma yang memiliki fungsi untuk penyesuaian bobot dan parameter lainnya dari model-model CNN selama proses pelatihan berlangsung (Kingma & Ba, 2014). Optimizer menekan angka fungsi *loss*, sehingga akan membuat kinerja dan tingkat akurasi dari model menjadi lebih tinggi. *Optimizer Adaptive Moment Estimation* atau disingkat sebagai Adam, merupakan *optimizer* yang menggabungkan fitur-fitur terbaik dari algoritma *Root Mean Square Propagation* (RMSProp) dan AdaGrad yang mempunyai

kapabilitas untuk penanganan gradien yang jarang pada masalah yang *noisy* (Danendra, 2023). Penelitian yang dilakukan oleh (Yaqub et al., 2020) menyatakan bahwa *optimizer* ini mengurangi biaya komputasi, membutuhkan sedikit memori dalam implementasinya, dan terbukti memiliki rerata *error* paling minim dan akurasi paling tinggi ketika mencapai nilai minimum pada *epoch* tertentu jika dibandingkan dengan *optimizer* lain. Berikut merupakan notasinya.

3. METODOLOGI PENELITIAN

Pada penelitian ini tentunya terdapat beberapa tahapan-tahapan dalam melakukan penelitian secara terstruktur. Dalam penelitian ini, terdapat 3 metode untuk diimplementasikan dalam proses klasifikasi citra ras kucing, yaitu SVM, KNN, dan CNN. Tahapan untuk melakukan proses pengujian metode-metode di atas akan dimulai dengan proses pengumpulan (akuisisi) *dataset*, proses pelatihan model, proses pengujian model, dan yang terakhir merupakan perhitungan performa akurasi menggunakan *confusion matrix*. Perbandingan performa antara *deep learning* dan *machine learning* dilakukan berdasarkan hasil dari penghitungan *confusion matrix* dari masing-masing model. Perbandingan performa perlu dilakukan untuk membuktikan metode mana yang menjadi opsi terbaik dengan mempertimbangkan akurasi dan juga waktu komputasi.



Gambar 3.1. Tahapan Penelitian

4. HASIL DAN PEMBAHASAN

4.1 Hasil Skenario Pengujian Metode Klasifikasi SVM

4.1.1 Proses Pelatihan dan Pengujian Data pada Kernel RBF

Hasil yang didapatkan pada proses pelatihan dan pengujian dengan menggunakan kernel RBF meliputi pengujian pada 3 macam kelas, pada 2 macam skala. Seperti pada yang telah dijelaskan pada bagian skenario pengujian, yaitu akan terdapat pengujian dengan skala

kecil (20 citra per kelasnya), dan skala besar (200 citra per kelasnya) . Pengujian tersebut meliputi 3 kelas, 6 kelas, dan 13 kelas pada skala kecil, dan 3 kelas, 6 kelas, dan 13 kelas pada skala besar. Berikut adalah tabel rangkuman dari dokumentasi dari hasil pengujian masing-masing kelas pada kernel RBF.

Rata-Rata Waktu Training Skala Kecil (detik)		Rata-Rata Waktu Testing Skala Kecil (detik)	
3 Kelas	0.082	3 Kelas	0.048285714
6 Kelas	0.366857143	6 Kelas	0.178857143
13 Kelas	1.688571429	13 Kelas	1.066285714

Gambar 4.1. Rata-Rata Waktu *Training* dan *Testing* RBF Skala Kecil

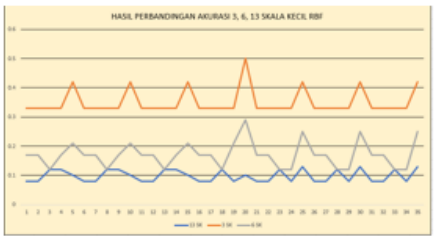
Rata-Rata Waktu Training Skala Besar (detik)		Rata-Rata Waktu Training Skala Besar (detik)	
3 Kelas	10.37657143	3 Kelas	5.779142857
6 Kelas	54.39571429	6 Kelas	23.21714286
13 Kelas	253.4691429	13 Kelas	104.9014286

Gambar 4.2. Rata-Rata Waktu *Training* dan *Testing* RBF Skala Besar

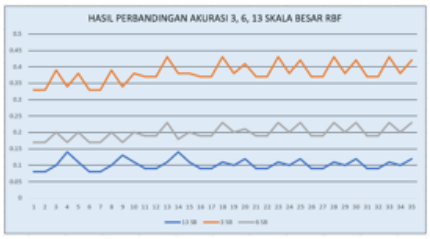
Rata-Rata Akurasi Skala Kecil		Rata-Rata Akurasi Skala Besar	
3 Kelas	0.350285714	3 Kelas	0.381142857
6 Kelas	0.170571429	6 Kelas	0.198571429
13 Kelas	0.098	13 Kelas	0.102571429

Gambar 4.3. Rata-Rata Akurasi RBF Skala Kecil dan Besar

3 gambar diatas, menunjukkan hasil dari rata-rata pengujian yang telah dilakukan pada kernel RBF. Setiap percobaan memiliki sebanyak 35 *output* seperti yang telah dijelaskan sebelumnya, dikarenakan oleh kombinasi *C* dan *Gamma* yang diujikan. Berdasarkan ketiga tabel tersebut, hasil yang didapatkan dengan kernel RBF pada setiap pengujian antara skala kecil hingga skala besar, rata-rata akurasi yang didapatkan termasuk sangat rendah. Rata-rata akurasi terbesar didapatkan oleh pengujian dengan menggunakan 3 kelas skala besar, yang hasil dari rata-ratanya hanyalah sebesar 38% (0.38).



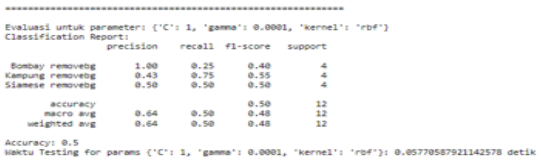
Gambar 4.4. Diagram Garis *Output* Akurasi Skala Kecil RBF



Gambar 4.5. Diagram Garis *Output* Akurasi Skala Besar RBF

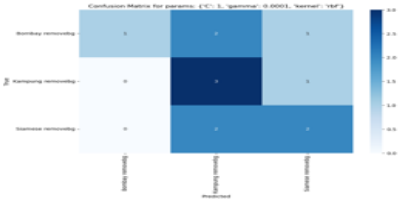
Berdasarkan 2 gambar di atas, hasil perbandingan performa akurasi yang didapatkan sangat fluktuatif pada setiap skenario pengujian dan juga jumlah skalanya. Dari hasil tersebut, terlihat bahwa model SVM menghasilkan akurasi yang menurun seiring dengan bertambahnya

jumlah kelas (pada percobaan untuk kedua skala). Pada skala kecil maupun skala besar, hasil akurasi tertinggi yang didapat, berada pada pengujian dengan jumlah 3 kelas. Hasil akurasi terus menurun pada pengujian 6 kelas dan 13 kelas. Hasil akurasi paling tinggi bernilai 50% (0.5) yang didapatkan pada percobaan ke 20 dengan 3 kelas pada skala kecil. Berikut merupakan hasil dari *classification report* dan *confusion matrix*-nya



```
============================================================

Evaluasi untuk parameter: {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
Classification Report:
                    precision    recall  f1-score   support

  Bombay removebg       1.00      0.25      0.40         4
Kampung removebg        0.43      0.75      0.55         4
 Siamese removebg       0.50      0.50      0.50         4

         accuracy                           0.50        12
        macro avg       0.64      0.50      0.48        12
     weighted avg       0.64      0.50      0.48        12

Accuracy: 0.5
Waktu Testing for params {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}: 0.05770587921142578 detik
```

Gambar 4.6. *Classification Report* Hasil Akurasi Terbaik RBF



Gambar 4.7. *Confusion Matrix* Hasil Akurasi Terbaik RBF

Pada hasil dari *classification report* di atas, dapat terlihat bahwa dengan kombinasi C senilai 1 dan Gamma senilai 0.0001, dan masing-masing kelas yang memiliki jumlah *support* sebanyak 4 per kelasnya, didapatkan skor F-1 maksimum sebesar 0.55 (55%) pada kelas Kampung, dan skor F-1 minimum sebesar 0.4 (40%) pada kelas *Bombay*. Kesimpulan terkait dengan hasil pengujian akurasi model SVM menggunakan kernel RBF adalah, hasil akurasi yang dihasilkan termasuk rendah, dan setiap bertambahnya kelas, hasil akurasinya akan menurun.

4.1.2 Proses Pelatihan dan Pengujian Data pada Kernel *Poly*

Hasil yang didapatkan pada proses pelatihan dan pengujian dengan menggunakan kernel *Poly* dapat dilihat pada tabel-tabel berikut ini.

Rata-Rata Waktu Training Skala Kecil (detik)		Rata-Rata Waktu Testing Skala Kecil (detik)	
3 Kelas	0.105	3 Kelas	0.015085714
6 Kelas	0.306571429	6 Kelas	0.046571429
13 Kelas	1.567428571	13 Kelas	0.249714286

Gambar 4.9. Rata-Rata Waktu *Training* dan *Testing Poly* Skala Kecil

Rata-Rata Waktu Training Skala Besar (detik)		Rata-Rata Waktu Training Skala Besar (detik)	
3 Kelas	7.799714286	3 Kelas	1.437142857
6 Kelas	41.06628571	6 Kelas	7.146285714
13 Kelas	237.5748571	13 Kelas	30.17

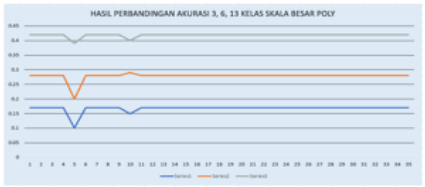
Gambar 4.10. Rata-Rata Waktu *Training* dan *Testing Poly* Skala Besar

Rata-Rata Akurasi Skala Kecil		Rata-Rata Akurasi Skala Besar	
3 Kelas	0.332571429	3 Kelas	0.418571429
6 Kelas	0.17	6 Kelas	0.278
13 Kelas	0.08	13 Kelas	0.167428571

Gambar 4.11. Rata-Rata Akurasi *Poly* Skala Kecil dan Besar

Berdasarkan ketiga tabel tersebut, hasil yang didapatkan dengan kernel *Polynomial* pada setiap pengujian antara skala kecil hingga skala besar, rata-rata akurasi yang didapatkan juga termasuk sangat rendah. Rata-rata akurasi terbesar didapatkan oleh pengujian dengan menggunakan 3 kelas skala besar, yang hasil dari rata-ratanya hanyalah sebesar 41% (0.41). Kemudian, untuk rata-rata akurasi terendah didapatkan oleh pengujian yang menggunakan data 13 kelas skala kecil di mana hasil rata-rata akurasinya adalah 8% (0.08).

Gambar 4.12. Diagram Garis *Output* Akurasi Skala Kecil *Poly*

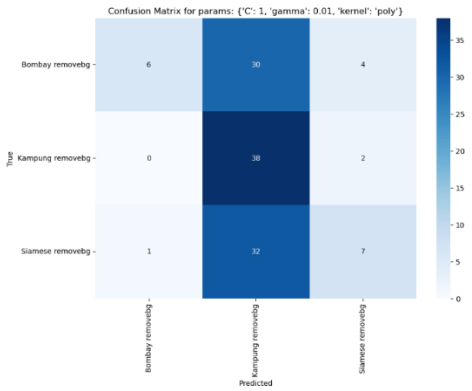


Gambar 4.13. Diagram Garis *Output* Akurasi Skala Besar *Poly*

Berdasarkan 2 gambar di atas, hasil perbandingan performa akurasi yang didapatkan dengan menggunakan kernel *Polynomial*, bersifat sangat stagnan (*flatline*) pada setiap skenario pengujian dan juga jumlah skalanya. Berbeda dengan kernel RBF yang pada setiap skenario memiliki hasil yang bervariasi, pada kernel *Poly* justru sebagian besar hasil akurasinya bersifat *uniform* (seragam), terutama pada pengujian dengan 6 dan 13 kelas pada skala kecil. Dari hasil klasifikasi kernel *Poly* hasil akurasi yang terlihat juga menurun seiring dengan bertambahnya jumlah kelas (pada percobaan untuk kedua skala). Pada skala kecil maupun skala besar, hasil akurasi tertinggi yang didapat, berada pada pengujian dengan jumlah 3 kelas.

```
============================================================

Evaluasi untuk parameter: {'C': 1, 'gamma': 0.01, 'kernel': 'poly'}
Classification Report:
                precision    recall  f1-score   support

 Bombay removebg       0.86      0.15      0.26        40
Kampung removebg       0.38      0.95      0.54        40
Siamese removebg       0.54      0.17      0.26        40

        accuracy                           0.42       120
       macro avg       0.59      0.42      0.35       120
    weighted avg       0.59      0.42      0.35       120

Accuracy: 0.425
Waktu Testing for params {'C': 1, 'gamma': 0.01, 'kernel': 'poly'}: 1.4071571826934814 det:
```

Gambar 4.14. *Classification Report* Salah Hasil Satu Akurasi Terbaik *Poly*



Gambar 4.15. *Confusion Matrix* Salah Satu Hasil Akurasi Terbaik *Poly*

Pada hasil dari *classification report* di atas, dapat terlihat bahwa dengan kombinasi C senilai 1 dan Gamma senilai 0.01, dan masing-masing kelas yang memiliki jumlah *support* sebanyak 40 per kelasnya, didapatkan skor F-1 maksimum sebesar 0.54 (54%) pada kelas Kampung, dan skor F-1 minimum sebesar 0.26 (26%) pada kelas *Bombay* dan *Siamese*. Kesimpulan terkait dengan hasil pengujian akurasi model SVM menggunakan kernel *Polynomial* adalah, hasil akurasi yang dihasilkan pada percobaan setiap kelasnya bersifat sangat stagnan (*flatline*).

4.1.3 Proses Pelatihan & Pengujian Data pada Kernel *Sigmoid*

Hasil yang didapatkan selama proses pelatihan dan pengujian model dengan menggunakan kernel *Sigmoid*, dapat dilihat pada beberapa tabel di bawah ini.

Rata-Rata Waktu Training Skala Kecil (detik)		Rata-Rata Waktu Testing Skala Kecil (detik)	
3 Kelas	0.073142857	3 Kelas	0.012885714
6 Kelas	0.262	6 Kelas	0.040857143
13 Kelas	1.350857143	13 Kelas	0.222571429

Gambar 4.16. Rata-Rata Waktu *Training* dan *Testing Sigmoid* Skala Kecil

Rata-Rata Waktu Training Skala Besar (detik)		Rata-Rata Waktu Training Skala Besar (detik)	
3 Kelas	6.478285714	3 Kelas	1.511714286
6 Kelas	30.89371429	6 Kelas	7.636857143
13 Kelas	182.7614286	13 Kelas	37.78485714

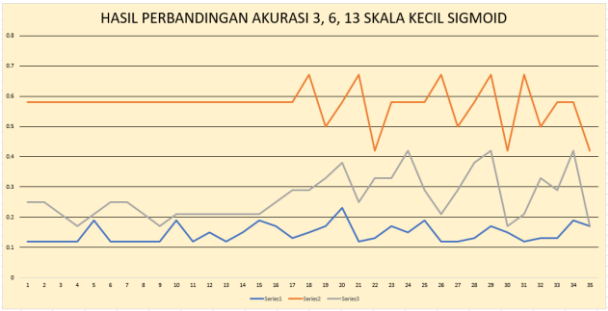
Gambar 4.16 Rata-Rata Waktu *Training* dan *Testing Sigmoid* Skala Besar

Rata-Rata Akurasi Skala Kecil		Rata-Rata Akurasi Skala Besar	
3 Kelas	0.572285714	3 Kelas	0.426571429
6 Kelas	0.265142857	6 Kelas	0.246
13 Kelas	0.146	13 Kelas	0.116857143

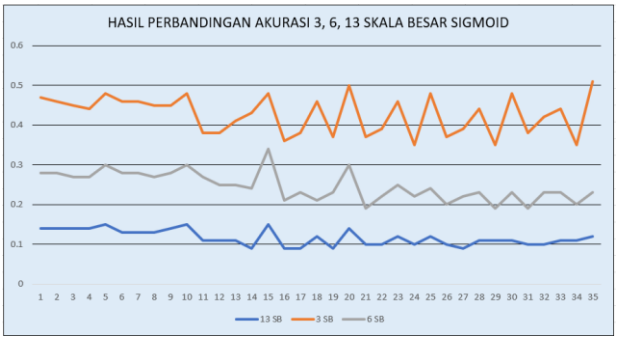
Gambar 4.17. Rata-Rata Akurasi *Sigmoid* Skala Kecil dan Besar

Berdasarkan ketiga tabel tersebut, hasil yang didapatkan oleh percobaan dengan kernel *Sigmoid* pada setiap pengujian antara skala kecil hingga skala besar, rata-rata akurasi yang didapatkan juga termasuk rendah (tetapi lebih tinggi jika dibandingkan dengan hasil pengujian 2 kernel sebelumnya). Rata-rata akurasi terbesar didapatkan oleh pengujian dengan menggunakan 3 kelas skala kecil, yang hasil dari rata-ratanya adalah sebesar 57% (0.57). Kemudian, untuk rata-rata akurasi terendah didapatkan oleh pengujian yang menggunakan

data 13 kelas skala besar di mana hasil rata-rata akurasinya adalah 11% (0.116). Walaupun hasil percobaan dengan menggunakan kernel *Sigmoid* cukup mendapatkan perbedaan yang signifikan jika dibandingkan dengan menggunakan kernel RBF dan *Poly*, yang di mana kernel *Sigmoid* mendapatkan rata-rata sebesar 0.57 sebagai hasil rata-rata terbaiknya pada data 3 kelas skala kecil, hasil tersebut tetap belum bisa dikategorikan ke dalam model yang efektif dan efisien dalam melakukan pengklasifikasian data.



Gambar 4.18. Diagram Garis *Output* Akurasi Skala Kecil *Sigmoid*

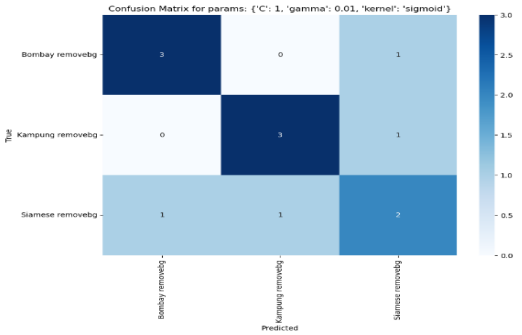


Gambar 4 19 Diagram Garis *Output* Akurasi Skala Besar *Sigmoid*

Berdasarkan 2 gambar di atas, hasil perbandingan performa akurasi yang didapatkan dengan menggunakan kernel *Sigmoid*, bersifat sangat fluktuatif pada setiap skenario pengujian dan juga jumlah skalanya. Berbeda dengan kernel *Polynomial* yang pada setiap skenario memiliki hasil yang tergolong stagnan, pada kernel *Sigmoid* justru sebagian besar hasil akurasinya sangat bervariasi, terkecuali untuk beberapa kombinasi pengujian awal pada data 3 kelas skala kecil, yang terlihat sangat datar. Hasil yang didapatkan oleh percobaan dengan penggunaan kernel *Sigmoid* menunjukkan bahwa pada beberapa titik percobaannya pada 3 kelas skala kecil mendapatkan nilai yang paling tinggi diantara kernel lainnya. Nilai akurasi sebesar 0.67 didapatkan sebanyak 5 kali pada pengujian tersebut. Tetapi seperti kernel lainnya, hasil akurasi yang terlihat juga menurun seiring dengan bertambahnya jumlah kelas (pada percobaan untuk kedua skala). Pada skala kecil maupun skala besar, hasil akurasi tertinggi yang didapat, berada pada pengujian dengan jumlah 3 kelas. Hasil akurasi terus menurun pada pengujian 6 kelas dan 13 kelas

```
============================================================

Evaluasi untuk parameter: {'C': 1, 'gamma': 0.01, 'kernel': 'sigmoid'}
Classification Report:
                    precision    recall  f1-score   support

 Bombay removebg       0.75      0.75      0.75         4
Kampung removebg       0.75      0.75      0.75         4
Siamese removebg       0.50      0.50      0.50         4

        accuracy                           0.67        12
       macro avg       0.67      0.67      0.67        12
    weighted avg       0.67      0.67      0.67        12

Accuracy: 0.6666666666666666
Waktu Testing for params {'C': 1, 'gamma': 0.01, 'kernel': 'sigmoid'}: 0.01360321044921875 detik
```

Gambar 4.20. *Classification Report* Salah Satu Hasil Akurasi Terbaik *Sigmoid*



Gambar 4.21. *Confusion Matrix* Salah Satu Hasil Akurasi Terbaik *Sigmoid*

Pada hasil dari *classification report* di atas, dapat terlihat bahwa dengan kombinasi C senilai 1 dan Gamma senilai 0.01, dan masing-masing kelas yang memiliki jumlah *support* sebanyak 4 per kelasnya didapatkan skor F-1 maksimum sebesar 0.75 (75%) pada kelas Kampung dan *Bombay*, dan skor F-1 minimum sebesar 0.5 (50%) pada kelas *Siamese*. Kesimpulan terkait dengan hasil pengujian akurasi model SVM menggunakan kernel *Sigmoid* adalah, hasil akurasi yang dihasilkan pada percobaan setiap kelasnya bersifat cukup bervariasi (walaupun tidak terlalu se-fluktuatif jika dibandingkan dengan pengujian kernel RBF), dan untuk tingkat akurasi keseluruhan terdapat beberapa kali percobaan yang bisa digolongkan memiliki tingkat akurasi tertinggi diantara kernel lainnya.

4.1.4 Proses Pelatihan dan Pengujian Data pada Kernel *Linear*

Hasil pengujian terakhir, yang didapatkan pada proses pelatihan dan pengujian dengan menggunakan kernel *Linear* akan berbeda dengan 3 kernel yang lain, dikarenakan kernel Linear tidak menerima parameter *Gamma* pada prosesnya, maka hanya parameter C saja yang diujikan. Pada parameter C yang diujikan terdapat 7 nilai, sehingga *output* yang dihasilkan juga sebanyak 7 macam. Berikut adalah beberapa tabel yang terkait dengan hasil yang didapatkan dari pengujian dengan kernel *Linear*.

Rata-Rata Waktu Training Skala Kecil (detik)		Rata-Rata Waktu Testing Skala Kecil (detik)	
3 Kelas	0.107143	3 Kelas	0.015714
6 Kelas	0.554286	6 Kelas	0.078571
13 Kelas	1.922857	13 Kelas	0.301429

Gambar 4.22. Rata-Rata Waktu *Training* dan *Testing Linear* Skala Kecil

Rata-Rata Waktu Training Skala Besar (detik)		Rata-Rata Waktu Training Skala Besar (detik)	
3 Kelas	11.25	3 Kelas	1.644286
6 Kelas	40.54286	6 Kelas	7.168571
13 Kelas	238.5543	13 Kelas	33.53857

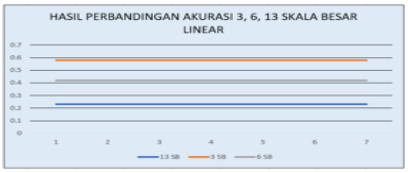
Gambar 4.23. Rata-Rata Waktu *Training* dan *Testing Linear* Skala Besar

Rata-Rata Akurasi Skala Kecil		Rata-Rata Akurasi Skala Besar	
3 Kelas	0.5	3 Kelas	0.58
6 Kelas	0.25	6 Kelas	0.42
13 Kelas	0.17	13 Kelas	0.23

Gambar 4.24. Rata-Rata Akurasi *Linear* Skala Kecil dan Besar

Berdasarkan ketiga tabel tersebut, hasil yang didapatkan oleh percobaan dengan kernel *Linear* pada setiap pengujian antara skala kecil hingga skala besar, rata-rata akurasi yang didapatkan merupakan yang tertinggi jika dibandingkan 3 kernel lainnya. Rata-rata akurasi pada skala besar untuk 13 kelas menyentuh angka 0.23, di mana hasil rata-rata dari 3 kernel yang lain bahkan tidak menyentuh 0.2. Hal ini disebabkan oleh hasil output dari kernel *Linear* yang sangat stagnan (*flatline*), tidak ada perubahan nilai akurasi yang dihasilkan walaupun menggunakan nilai parameter C yang berbeda. Rata-rata akurasi terbesar didapatkan oleh pengujian dengan menggunakan data 3 kelas skala besar, yang mendapatkan angka rata-rata akurasinya sebesar 58% (0.58). Kemudian, untuk rata-rata akurasi terendah pada kernel ini, didapatkan oleh pengujian yang menggunakan data 13 kelas skala kecil, di mana hasil rata-ratanya sebesar 17% (0.17). Walaupun hasil percobaan dengan menggunakan kernel *Linear* mendapatkan hasil pengujian dengan rata-rata akurasi sebesar 0.58 sebagai hasil rata-rata terbaiknya pada data 3 kelas skala kecil, hasil tersebut juga tetap belum bisa dikategorikan ke dalam model yang efektif dan efisien dalam melakukan pengklasifikasian data. Kemudian, untuk perbandingan dari semua hasil akurasi pada masing-masing kombinasi yang didapatkan pada masing-masing skala dapat dilihat pada 2 diagram garis berikut ini.

Gambar 4.25. Diagram Garis *Output* Akurasi Skala Kecil *Linear*



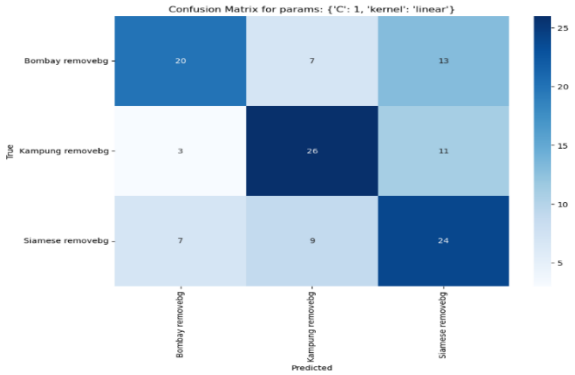
Gambar 4.26. Diagram Garis *Output* Akurasi Skala Besar *Linear*

Berdasarkan 2 gambar di atas, hasil perbandingan performa akurasi yang didapatkan dengan menggunakan kernel *Linear* bersifat sangat stagnan, atau tidak ada perubahan sama sekali pada setiap skenario pengujian dan juga jumlah skalanya (bahkan pada kernel

Polynomial yang juga stagnan, masih terdapat beberapa perubahan nilai akurasi pada beberapa titik). Pada kernel *Linear*, hasil yang dihasilkan sangat berbeda dengan kernel RBF dan Sigmoid, yang sangat fluktuatif. Secara tidak langsung, hal ini mengindikasikan bahwa parameter C (sebanyak 7 variasi nilai) yang diujikan pada kernel *Linear* tidak memiliki pengaruh sama sekali dalam penentuan nilai akurasi, tetapi hanya pada ETA pada proses *training* dan *testing*-nya. Berdasarkan hasil diatas, semua pengujian memiliki garis *flatline*. Hal ini juga yang menyebabkan rata-rata akurasi yang didapatkan merupakan yang tertinggi diantara 3 kernel lainnya (rerata tertinggi bukan berarti memiliki akurasi tertinggi). Untuk tingkat akurasi pada percobaannya, masih tetap ditemukan fenomena yang sama, yaitu semakin banyak kelas diujikan, hasil akurasinya semakin menurun. Tetapi pengujian dengan data yang lebih banyak pada kernel *Linear* alih-alih mendapatkan nilai akurasi yang lebih kecil malah mendapatkan akurasi yang lebih tinggi dibandingkan menggunakan data yang sedikit (terbukti pada perbandingan antara 3 pengujian skala kecil dan besar). Berikut merupakan salah satu contoh dari *classification_report* dan *confusion_matrix* pada percobaan yang mendapatkan hasil akurasi sebesar 0.58 (pengujian menggunakan data 3 kelas skala besar).

```
============================================================

Evaluasi untuk parameter: {'C': 1, 'kernel': 'linear'}
Classification Report:
                precision    recall  f1-score   support

 Bombay removebg       0.67      0.50      0.57        40
Kampung removebg       0.62      0.65      0.63        40
Siamese removebg       0.50      0.60      0.55        40

        accuracy                           0.58       120
       macro avg       0.60      0.58      0.58       120
    weighted avg       0.60      0.58      0.58       120

Accuracy: 0.5833333333333334
Waktu Testing for params {'C': 1, 'kernel': 'linear'}: 1.6815595626831055 detik
```

Gambar 4.27. *Classification Report* Salah Satu Hasil Akurasi Terbaik *Linear*



Gambar 4.28. *Confusion Matrix* Salah Satu Hasil Akurasi Terbaik *Linear*

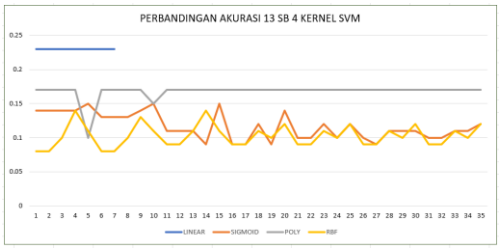
Kesimpulan terkait dengan hasil pengujian akurasi model SVM menggunakan kernel *Linear* adalah, hasil akurasi yang dihasilkan pada percobaan setiap kelasnya bersifat sangat stagnan, jauh lebih stagnan dari kernel *Polynomial*. Kemudian, untuk tingkat akurasi

keseluruhan dapat disimpulkan memiliki tingkat rerata tertinggi dibandingkan 3 kernel lain, tetapi untuk hasil akurasi yang paling tinggi masih terdapat pada kernel *Sigmoid*. Hal lain yang dapat disimpulkan adalah setiap bertambahnya jumlah kelas yang diujikan hasil akurasi yang didapatkan akan turun. Fenomena yang sama didapati pada kernel yang lain. Tetapi pada kernel *Linear*, jumlah data memiliki andil yang cukup besar untuk menentukan jumlah akurasi, dikarenakan hasil akurasi yang didapatkan oleh pengujian dengan skala besar memiliki hasil akurasi yang lebih tinggi daripada pengujian dengan skala kecil. Kernel terakhir yang diujikan ini juga menandakan bahwa metode SVM bukan metode yang cocok untuk diterapkan pada kasus klasifikasi citra ras kucing yang memiliki data dan label yang cukup banyak atau bervariasi.

4.1.5 Kesimpulan Akhir Pengujian dengan Metode SVM

Setelah melalui proses pengujian dengan menggunakan metode SVM dan juga parameter-parameter yang telah ditentukan, telah didapatkan beberapa analisa terkait kesimpulan akhir untuk metode pengujian SVM dalam permasalahan pengklasifikasian citra ras kucing. Berikut diantaranya.

1. Metode klasifikasi SVM bukanlah metode yang paling optimal dalam proses pengklasifikasian citra ras kucing. Dibuktikan dengan hasil pengujian dari keempat kernel untuk dataset keseluruhan (13 kelas skala besar), semua hasil pengujian dari masing-masing kernel tidak menghasilkan *output* dengan akurasi tinggi (dikatakan tinggi apabila memiliki nilai akurasi 0.8 ke atas). Bahkan dari keempat pengujian kernel dengan semua kombinasi parameter pengujiannya, tidak ada output yang menyentuh nilai 0.3. Berikut adalah diagram garis perbandingan akurasi 4 kernel untuk data 13 kelas dengan skala besar.



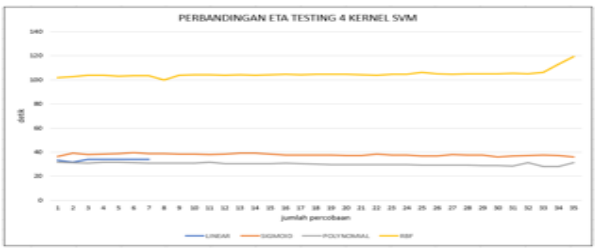
Gambar 4.29. Diagram Garis Perbandingan Akurasi 4 Kernel

Hasil angka akurasi yang divisualisasikan pada diagram garis diatas, membuktikan bahwa hasil yang didapatkan dengan menggunakan 4 kernel sama sekali tidak memiliki nilai akurasi yang baik. Hasil dengan nilai akurasi tertinggi dimiliki oleh kernel *Linear* dengan angka sebesar 0.23 yang dicapai 7x berturut-turut. Hal ini juga membuktikan bahwa pada kernel *Linear*, penggunaan parameter C tidak terlalu berpengaruh untuk hasil akurasi yang dihasilkan oleh model. Oleh karena itu, bisa disimpulkan bahwa penggunaan metode SVM

(meskipun menggunakan 4 kernel yang berbeda) tetap belum bisa dikatakan sebagai metode yang tepat untuk proses ujicoba pengklasifikasian citra ras kucing sebanyak 13 kelas dengan 2600 citra per kelasnya.

2. Dari percobaan keempat kernel, kernel *Sigmoid* merupakan kernel tercepat dalam proses *training* pada data fitur ekstraksi citra pada dataset 13 kelas ras kucing skala besar. Hal tersebut akan dibuktikan pada diagram garis berikut.

Gambar 4.30. Diagram Garis Perbandingan Waktu *Training* 4 *Kernel*



Gambar 4.31. Diagram Garis Perbandingan Waktu *Testing* 4 Kernel

Pada diagram garis diatas hasil ETA *training* dan *testing* yang diperoleh oleh 4 kernel telah divisualisasikan, dan dapat dilihat bahwa garis oranye yang melambangkan kernel *Sigmoid* memiliki waktu ETA *training* yang paling rendah. Tetapi, pada ETA *testing*, hasil yang paling rendah didapat oleh percobaan dengan kernel *Polynomial*, dan yang tertinggi adalah kernel RBF (mencapai lebih dari 100 detik dalam proses testingnya). Pada kernel *Sigmoid* ETA *training* bertahap turun mulai dari percobaan ke 15 (tetapi tetap fluktiatif), sedangkan untuk kernel *Polynomial* ETA *testing*-nya memiliki hasil yang paling rendah secara stagnan mulai dari awal percobaan hingga akhir percobaan. Oleh karenanya, kesimpulan yang dapat diambil dengan hasil yang telah didapatkan, kernel dengan ETA *training* yang paling rendah, belum tentu memiliki ETA *testing* yang paling rendah juga.

4.2 Hasil Skenario Pengujian Metode Klasifikasi KNN

4.2.1 Proses Pelatihan dan Pengujian Data

Seteleh melewati prorses pengujian sesuai dengan skenario yang telah dijabarkan, Hasil pengujian dengan metode KNN dengan menggunakan variasi nilai K, jumlah kelas dan juga skala data dapat dilihat pada gambar berikut ini.

13 SB		13 SK	
RATA-RATA AKURASI	0.12615	RATA-RATA AKURASI	0.1
RATA-RATA TRAINING	0.19462	RATA-RATA TRAINING	0.01615
RATA-RATA TESTING	1.78231	RATA-RATA TESTING	0.42231

6 SB		6 SK	
RATA-RATA AKURASI	0.24667	RATA-RATA AKURASI	0.16833
RATA-RATA TRAINING	0.095	RATA-RATA TRAINING	0.005
RATA-RATA TESTING	0.48833	RATA-RATA TESTING	0.195

3 SB		3 SK	
RATA-RATA AKURASI	0.43333	RATA-RATA AKURASI	0.30333
RATA-RATA TRAINING	0.04	RATA-RATA TRAINING	0
RATA-RATA TESTING	0.24667	RATA-RATA TESTING	0.14333

Gambar 4.32. Tabel Hasil Pengujian KNN

Pada gambar diatas, dapat diamati hasil berupa rata-rata akurasi, rata-rata training, dan rata-rata testing dari masing-masing jumlah kelas dan juga skala data yang telah diujicobakan. Rata-rata akurasi terbesar terdapat pada pengujian dengan jumlah 3 kelas dengan skala besar, yaitu sebanyak 0.43. Kemudian untuk hasil terendah terdapat pada dpengujian dengan jumlah 13 kelas dengan skala kecil. Pada tabel pengujian di atas, didapati fenomena seperti pada proses pengklasifikasian dengan metode SVM (pada kernel *Linear*, *Poly*, dan RBF). Di mana data dengan jumlah sampel yang lebih tinggi pada jumlah kelas yang sama mempunyai tingkat akurasi lebih besar dibandingkan dengan data dengan sampel yang rendah. Berikut merupakan visualisasi diagram garis hasil akurasi dari skenario pengujian KNN.

Gambar 4.33. Diagram Garis Akurasi 3 Kelas KNN

Gambar 4.34. Diagram Garis Akurasi 6 Kelas KNN

Gambar 4.36. Diagram Garis Akurasi 13 Kelas KNN

Pada ketiga gambar diatas, dapat diamati bahwa pada setiap gambar memiliki jumlah 'k' (yang terdapat pada sumbu y diagram garis tersebut) yang berbeda-beda. Hal itu dikarenakan, pada percobaan diujikan jumlah 'k' berdasarkan banyaknya kelas pada skenario pengujian yang telah dirancang. Berdasarkan ketiga gambar di atas, juga dapat diamati

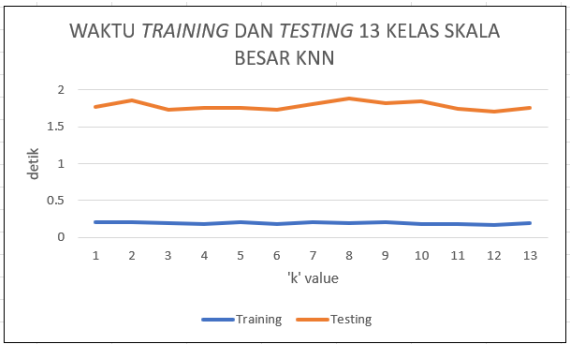
fenomena di mana akurasi yang semakin menurun dengan bertambahnya jumlah kelas yang terdapat pada data yang diujikan (sama seperti dengan metode SVM). Diagram dengan garis yang paling fluktuatif terdapat pada diagram pengujian 13 kelas (karena jumlah nilai 'k' yang diujikan lebih banyak). Dan pada diagram pengujian 13 kelas juga dapat disimpulkan bahwa hasil nilai akurasi tertinggi terdapat pada pengujian dengan menggunakan nilai 'k' = 1 (sebesar 0.15 atau 15%). Berikut adalah hasil perhitungan confusion matrix dan classification report untuk pengujian dengan nilai 'k' = 1.

Gambar 4.37. *Confusion Matrix* k =1

	precision	recall	f1-score	support
0	0.25	0.12	0.17	40
1	0.15	0.07	0.10	40
2	0.00	0.00	0.00	40
3	0.80	0.20	0.32	40
4	0.11	0.17	0.14	40
5	0.12	0.05	0.07	40
6	0.11	0.88	0.20	40
7	0.26	0.15	0.19	40
8	0.33	0.15	0.21	40
9	0.50	0.03	0.05	40
10	0.26	0.15	0.19	40
11	0.00	0.00	0.00	40
12	0.00	0.00	0.00	40
accuracy			0.15	520
macro avg	0.22	0.15	0.13	520
weighted avg	0.22	0.15	0.13	520

Gambar 4.38. *Classification Report* k =1

Pada kedua gambar diatas, dapat diamati bahwa hasil akurasi tertinggi pada klasifikasi 13 kelas dalam skala besar hanya memiliki tingkat *micro average* sebesar 0.15, di mana hasil ini merupakan hasil yang sangat rendah dan bisa dikatakan gagal. Pun berlaku pada pengujian dengan jumlah kelas lainnya, yang bahkan tidak menyentuh nilai *accuracy* sebesar 0.5 atau 50%. Hasil yang didapatkan metode KNN hampir sama rendahnya dengan metode SVM dalam proses pengklasifikasian citra ras kucing dengan 13 kelas dan 2600 sampel. Berarti dalam kata lain, kedua metode *machine learning* yang diujikan dapat dikatakan gagal dalam pegimplementasian proses klasifikasi 13 kelas citra ras kucing dengan ekstraksi fitur HOG dengan 2600 sampel.



Gambar 4.39. Diagram Garis ETA *Training* & *Testing* 13 SB

Berdasarkan diagram garis diatas, dapat diketahui bahwa baik waktu *testing* dan *training* dari metode KNN untuk data dengan 13 kelas berskala besar, tidak melebihi dari 2 detik pada masing-masing nilai 'k' yang diujikan. Berdasarkan hal ini, metode KNN merupakan metode *machine learning* dengan *running time* tercepat yang diujikan dalam percobaan ini, jika dibandingkan SVM (4 *kernel*) yang memiliki rerata ratusan detik dalam proses *training* data, dan puluhan detik dalam proses *testing* data.

4.2.2 Kesimpulan Akhir Pengujian dengan Metode KNN

Setelah melalui proses pengujian dengan menggunakan metode KNN dan juga parameter-parameter yang telah ditentukan, telah didapatkan beberapa analisa terkait kesimpulan akhir untuk metode pengujian KNN dalam permasalahan pengklasifikasian citra ras kucing. Berikut diantaranya.

1. Metode klasifikasi KNN juga bukanlah metode yang paling optimal dalam proses pengklasifikasian citra ras kucing. Dibuktikan dengan hasil pengujian dari semua skenario variasi kelas dan skala, serta dataset keseluruhan (13 kelas skala besar), semua hasil pengujiannya memiliki hasil akurasi yang sangat rendah. Proses klasifikasi bisa dikatakan berhasil jika mendapatkan nilai *micro average* sebesar 0.8 ke-atas. Sedangkan, pada metode KNN bahkan tidak dapat menyentuh 0.2 pada pengujiannya, seperti yang tertera pada gambar 4.39. Hasil akurasi tersebut lebih rendah jika dibandingkan metode SVM yang sempat menyentuh angka 0.23 pada hasil akurasinya.
2. Metode klasifikasi KNN merupakan *metode machine learning* dengan *running time* tercepat yang diujikan pada percobaan ini. Dapat dilihat dalam tabel berikut ini.

13 SB	MEAN ACCURACY	MEAN ETA TRAINING	MEAN ETA TESTING
KNN	0.12	0.194s	1.782s
SVM LINEAR	0.23	238.55s	33.53s
SVM RBF	0.1	253.46s	104.90s
SVM POLY	0.16	237.57s	30.17s
SVM SIGMOID	0.11	182.76s	37.78s

Gambar 4.40. Tabel Perbandingan Hasil SVM & KNN

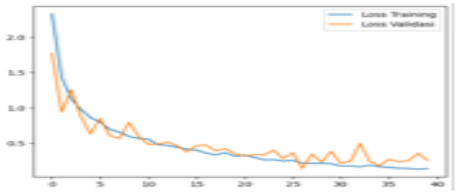
Pada tabel perbandingan diatas, dapat dilihat bahwa KNN merupakan metode pemenang dalam aspek kecepatan waktu dalam proses *training* maupun *testing*-nya, karena bahkan pada rerata dari seluruh skenario pengujian pada data dengan 13 kelas berskala besar, metode KNN memiliki rata-rata waktu yang luar biasa rendah.

4.3 Hasil Pengujian Klasifikasi CNN Arsitektur *Xception*

4.3.1 Proses Pelatihan dan Pengujian Data

Setelah melalui proses skenario pengujian dengan menggunakan metode CNN dengan arsitektur *Xception* didapatkan hasil pengamatan *output* sebagai berikut.

Gambar 4.41. Grafik Garis Akurasi 40 *Epoch Xception*



Gambar 4.42. Grafik Garis Loss 40 *Epoch Xception*

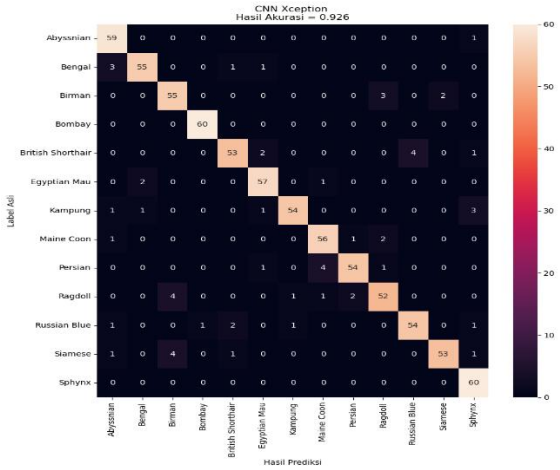
40 epoch	
ETA Training & Val (s)	537.075
Accuracy	0.8494975
Loss	0.4733475
Val_Accuracy	0.8340475
Val_Loss	0.4782825

Gambar 4.43. Grafik Rata-Rata 40 *Epoch Xception*

Pada gambar 4.41 hingga 4.43 di atas merupakan hasil pengujian arsitektur *Xception* dengan *epoch* sebanyak 40 menggunakan *fine-tuning* dengan *optimizer* Adam serta *learning rate* sebesar 1e-5. Pada diagram garis diatas terlihat bahwa seiringnya dengan bertambahnya *epoch*, nilai *loss* (baik validasi maupun *training*) akan berkurang. Sedangkan, nilai akurasi (baik validasi maupun *training*) akan bertambah. Hasil yang didapatkan ketika proses ini sangat memuaskan, hasil akurasi validasi beberapa kali mendapatkan nilai diatas 0.9. Kemudian, untuk nilai rata-rata ETA yang didapatkan sekitar 537 detik. Hal ini membuktikan bahwa CNN memiliki waktu eksekusi *training* yang lebih lama daripada KNN dan SVM. Sedangkan, untuk output pengujian divisualisasikan melalui gambar berikut.

	precision	recall	f1-score	support
Abyssnian	0.894	0.983	0.937	60.0
Bengal	0.948	0.917	0.932	60.0
Birman	0.873	0.917	0.894	60.0
Bombay	0.984	1.0	0.992	60.0
British Shorthair	0.93	0.883	0.906	60.0
Egyptian Mau	0.919	0.95	0.934	60.0
Kampung	0.964	0.9	0.931	60.0
Maine Coon	0.903	0.933	0.918	60.0
Persian	0.947	0.9	0.923	60.0
Ragdoll	0.897	0.867	0.881	60.0
Russian Blue	0.931	0.9	0.915	60.0
Siamese	0.964	0.883	0.922	60.0
Sphynx	0.896	1.0	0.945	60.0
accuracy	0.926	0.926	0.926	0.926
macro avg	0.927	0.926	0.925	780.0
weighted avg	0.927	0.926	0.925	780.0

Gambar 4.44. *Classification Report* 40 *Epoch Xception*



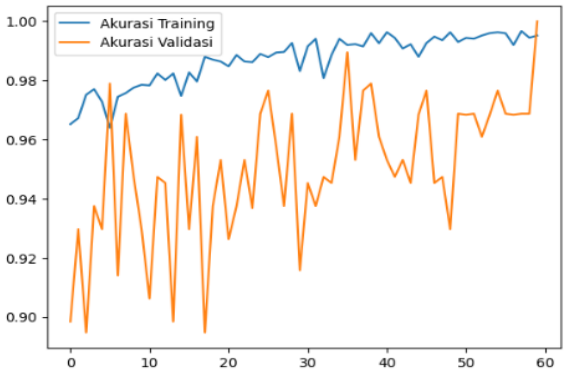
Gambar 4.45. *Confusion Matrix* 40 *Epoch Xception*

```
#predict = model.predict(test_generator, steps=np.ceil(nb_samples/32))
predict = model.predict(test_generator, steps=int(np.ceil(nb_samples / 32)))

C:\Anaconda\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_ad
  self._warn_if_super_not_called()
25/25 ━━━━━━━━━━━━━━━━━━━━ 33s 1s/step
```

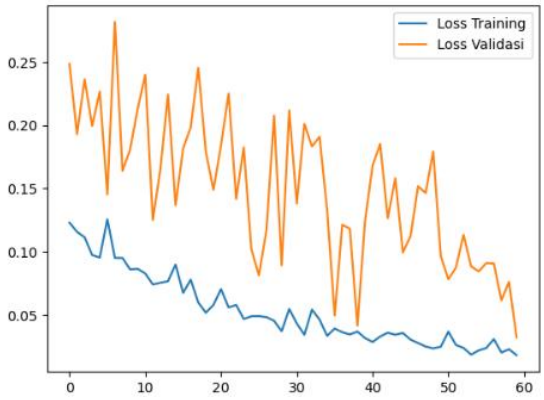
Gambar 4.46. *ETA Testing* 40 *Epoch Xception*

Berdasarkan gambar-gambar diatas, dapat diambil kesimpulan bahwa hasil pengujian dari 40 *epoch* yang dieksekusi dengan total waktu sebanyak 33 detik menghasilkan *output* akurasi atau *micro average* sebesar 0.926. Data testing terdiri atas 780 citra dengan rasio 1:1 antara masing-masing kelasnya (data terdistribusi dengan jumlah yang sama). Angka nilai akurasi sebesar 0.926 merupakan jumlah yang cukup memuaskan untuk permasalahan klasifikasi citra ras kucing ini

Kemudian, untuk hasil percobaan dengan jumlah *epoch* sebesar 60 dapat dilihat pada beberapa gambar berikut ini.



Gambar 4.47. Grafik Garis Akurasi 60 *Epoch Xception*



Gambar 4.48. Grafik Garis *Loss* 60 *Epoch Xception*

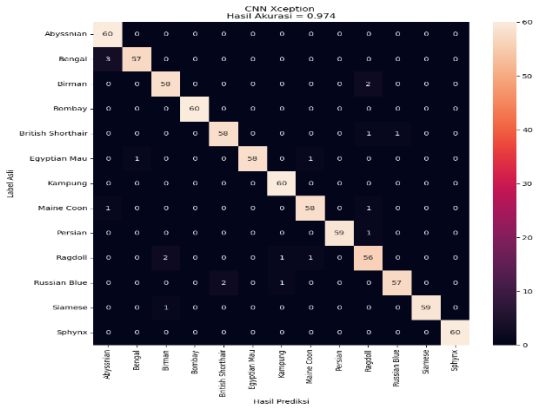
60 epoch	
ETA Training & Val (s)	523.7666667
Accuracy	0.98708
Loss	0.053318333
Val_Accuracy	0.949966667
Val_Loss	0.150101667

Gambar 4.49. Grafik Rata-Rata 60 *Epoch Xception*

Pada gambar 4.47 hingga 4.49 di atas merupakan hasil pengujian arsitektur *Xception* dengan *epoch* sebanyak 60 kali, dan juga dengan menggunakan *fine-tuning. Optimizer* tetap menggunakan Adam serta *learning rate* sebesar 1e-5. Diagram garis juga menunjukkan pola yang sama dengan ketika menggunakan 40 *epoch*. Semakin bertambah *epoch*, nilai loss akan berkurang, tetapi nilai akurasi akan bertambah (walau di sini terlihat lebih fluktutatif dikarenakan hasil per *epoch* tidak terlalu jauh perbedaannya). Hasil yang didapatkan dapat dikatakan hampir sempurna, dikarenakan hampir semua hasil akurasi validasi selalu diatas 0.9. Bahkan val_accuracy mendapatkan nilai sebesar 0.94 atau 94 persen. Tentu angka tersebut menunjukkan bahwa model bisa melakukan tugas klasifikasi citra dengan sangat baik. Kemudian, untuk rata-rata ETA yang didapatkan sebesar 523 detik, yang berarti lebih sedikit dibandingkan dengan 40 *epoch*. Memperkuat argumen bahwa metode CNN memang memiliki waktu running yang cukup lama. Beralih ke *output* pengujian 60 *epoch*, akan divisualisasikan pada gambar berikut.

	precision	recall	f1-score	support
Abyssnian	0.938	1.0	0.968	60.0
Bengal	0.983	0.95	0.966	60.0
Birman	0.951	0.967	0.959	60.0
Bombay	1.0	1.0	1.0	60.0
British Shorthair	0.967	0.967	0.967	60.0
Egyptian Mau	1.0	0.967	0.983	60.0
Kampung	0.968	1.0	0.984	60.0
Maine Coon	0.967	0.967	0.967	60.0
Persian	1.0	0.983	0.992	60.0
Ragdoll	0.918	0.933	0.926	60.0
Russian Blue	0.983	0.95	0.966	60.0
Siamese	1.0	0.983	0.992	60.0
Sphynx	1.0	1.0	1.0	60.0
accuracy	0.974	0.974	0.974	0.974
macro avg	0.975	0.974	0.974	780.0
weighted avg	0.975	0.974	0.974	780.0

Gambar 4.50. *Classification Report* 60 *Epoch Xception*



Gambar 4.51. *Confusion Matrix* 60 *Epoch Xception*

```
#predict = model.predict(test_generator, steps=np.ceil(nb_samples/32))
predict = model.predict(test_generator, steps=int(np.ceil(nb_samples / 32)))

C:\Anaconda\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapt
  self._warn_if_super_not_called()
25/25 ━━━━━━━━━━━━━━━━━━━━ 33s 1s/step
```

Gambar 4.52. *ETA Testing* 60 *Epoch Xception*

Berdasarkan gambar-gambar diatas, dapat diambil kesimpulan bahwa hasil pengujian dari 60 *epoch* yang dieksekusi dengan total waktu sebanyak 33 detik menghasilkan *output* akurasi atau *micro average* sebesar 0.974. Akurasi sebesar itu merupakan hasil yang dapat dikategorikan mendekati sempurna.

4.3.2 Kesimpulan Akhir Pengujian Arsitektur *Xception*

Setelah melalui proses pengujian dengan menggunakan metode CNN dengan arsitektur *Xception* dengan jumlah *epoch* yang berbeda, telah didapatkan beberapa analisa terkait kesimpulan akhir untuk metode pengujian CNN dalam permasalahan pengklasifikasian citra ras kucing. Berikut diantaranya. Berikut diantaranya.

1. Metode klasifikasi CNN dengan arsitektur *Xception* merupakan metode yang memiliki hasil akurasi atau *micro average* yang sangat tinggi jika dibandingkan dengan metode SVM dan KNN dalam proses klasifikasi citra ras kucing sebanyak 13 label. Bahkan, dengan jumlah dataset yang berbeda (pada *machine learning, dataset* yang digunakan dibatasi dengan tanpa melalui proses augmentasi dan dengan penghilangan *background* untuk mengurangi jumlah *noise*), metode ini mampu meraih nilai akurasi diatas 92 persen pada kedua percobaannya. Dibandingkan dengan hasil *micro average* yang dihasilkan oleh metode *machine learning* yang bahkan tidak bisa menembus angka 30 persen, tentunya hal ini merupakan perbedaan yang sangat mencolok.
2. Jumlah *epoch* yang dipergunakan saat pelatihan, mempengaruhi nilai akurasi yang akan didapatkan pada proses *testing* model. Dapat dilihat pada gambar di bawah ini.

Dokumentasi CNN Xception		
	60 epoch	40 epoch
Mean ETA Training & Val	523.7666667	537.075
ETA *Testing*	33	33
Accuracy *Testing*	0.974	0.926

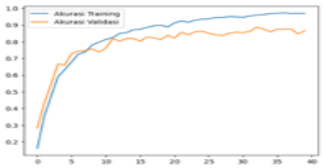
Gambar 4.53. Tabel Perbandingan *Output* Perbedaan *Epoch*

Nilai akurasi yang didapatkan melonjak jauh, di mana ketika menggunakan jumlah *epoch* sebesar 40 nilai akurasinya sebesar 0.92, dan ketika menggunakan jumlah *epoch* sebesar 60 mendapatkan nilai 0.97. Mengindikasikan bahwa jumlah epoch yang dipergunakan ketika proses pelatihan model sangat berguna dalam meraih nilai akurasi yang lebih tinggi. Walaupun pada kedua percobaan jumlah epoch terdapat kondisi sedikit overfitting, tetapi ternyata hal tersebut tidak memiliki pengaruh terhadap proses pengujian. Tetapi pada aspek yang lain, seperti kecepatan waktu ketika proses *testing* model, tidak dipengaruhi oleh besarnya jumlah *epoch* pada model tersebut.

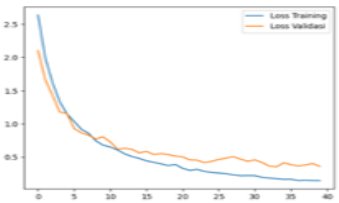
4.4 Hasil Pengujian Klasifikasi CNN Arsitektur *EfficientNet-B1*

4.4.1 Proses Pelatihan dan Pengujian Data

Setelah melalui proses skenario pengujian dengan menggunakan metode CNN dengan arsitektur *EfficientNet-B1* didapatkan hasil pengamatan *output* sebagai berikut.



Gambar 4.54. Grafik Garis Akurasi 40 *Epoch EfficeintNet-B1*

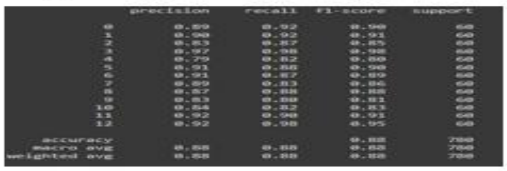


Gambar 4.55. Grafik Garis *Loss* 40 *Epoch EfficeintNet-B1*

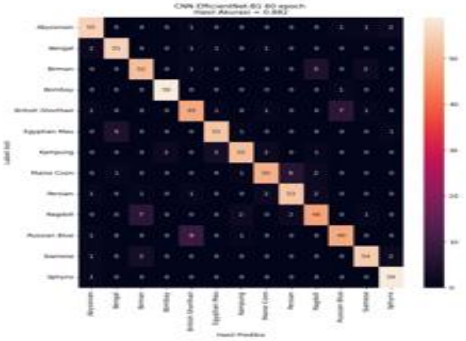
40 epoch	
ETA Training & Val (s)	317.6
Accuracy	0.838925
Loss	0.5527075
Val_Accuracy	0.7891375
Val_Loss	0.652225

Gambar 4.56. Grafik Rata-Rata 40 *Epoch EfficientNet-B1*

Pada gambar 4.54 hingga 4.56 di atas merupakan hasil pengujian arsitektur *EfficientNet* dengan *epoch* sebanyak 40 menggunakan *fine-tuning* dengan *optimizer* Adam serta *learning rate* sebesar 1e-5. Pada diagram garis diatas terlihat bahwa seperti pada percobaan pada arsitektur *Xception*, yaitu seiringnya dengan bertambahnya *epoch*, nilai *loss* dan *val_loss* akan berkurang. Sedangkan, nilai akurasi dan *val_accuracy* akan bertambah. Hasil yang didapatkan ketika proses ini kurang memuaskan apabila dibandingkan dengan percobaan pada arsitektur *Xception*. Hasil akurasi validasi sama sekali belum menyentuh atau mendapatkan nilai diatas 0.9. Sehingga, membuat nilai akurasi validasi hanya memiliki rata-rata sebesar 0.78. Kemudian, untuk nilai rata-rata ETA yang didapatkan sekitar 317.6 detik. Sehingga bisa dikatakan bahwa arsitektur *EfficientNet*-B1 memiliki running time yang lebih cepat per-epochnya jika dibandingkan dengan arsitektur *Xception*.



Gambar 4.57. *Classification Report* 40 *Epoch EfficientNet-B1*



Gambar 4.58. *Confusion Matrix* 40 *Epoch EfficentNet-B1*

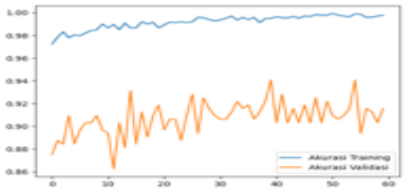
```
[ ]  predict = model.predict(test_generator, steps=int(np.cei
25/25 [==============================] - 17s 586ms/step
```

Gambar 4.59. *ETA Testing* 40 *Epoch EfficientNet-B1*

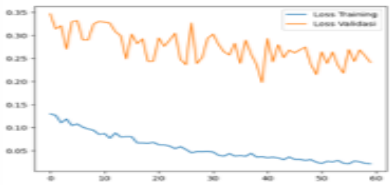
Berdasarkan gambar-gambar diatas, dapat diambil kesimpulan bahwa hasil pengujian dari 40 *epoch* yang dieksekusi dengan total waktu sebanyak 17 detik menghasilkan *output* akurasi atau *micro average* sebesar 0.882. Data pengujian yang sama terdiri atas 780 citra dengan rasio 1:1 antara masing-masing kelasnya (data terdistribusi dengan jumlah yang terjaga). Angka nilai akurasi sebesar 0.882 merupakan penurunan jumlah yang cukup signifikan untuk permasalahan klasifikasi citra ras kucing ini. Dari kedua percobaan dengan arsitektur Xception yang mendapatkan nilai akurasi diatas 0.9, menandakan bahwa EfficientNet-B1 memiliki performa yang kurang baik. Dengan kata lain, hasil pengujian

dengan 40 *epoch* pada arsitektur *EfficientNet-B1* yang menggunakan *fine-tuning* sudah dapat dikategorikan berhasil, namun kurang memuaskan dalam mengklasifikasikan citra ras kucing sebanyak 7800 citra dengan 13 kelas, jika dibandingkan dengan metode KNN dan SVM.

Kemudian, untuk hasil percobaan dengan jumlah *epoch* sebesar 60 dapat dilihat pada beberapa gambar berikut ini.



Gambar 4.60. Grafik Garis Akurasi 60 *Epoch Xception*

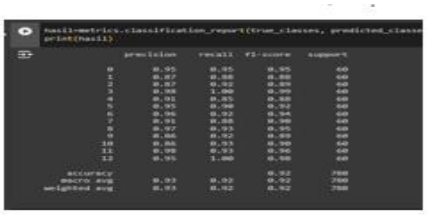


Gambar 4.61. Grafik Garis *Loss* 60 *Epoch EfficeintNet-B1*

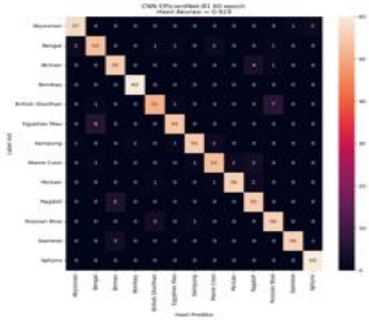
60 epoch	
ETA Training & Val (s)	322.7833333
Accuracy	0.991728333
Loss	0.055666667
Val_Accuracy	0.90661
Val_Loss	0.274275

Gambar 4.62. Grafik Rata-Rata 60 *Epoch EfficeintNet-B1*

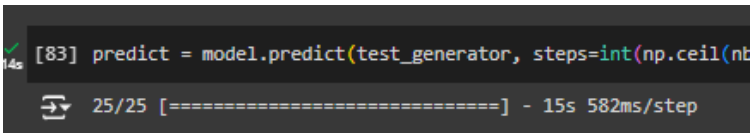
Pada gambar 4.60 hingga 4.62 di atas merupakan hasil pengujian arsitektur *Xception* dengan *epoch* sebanyak 60 kali, dan juga dengan menggunakan *fine-tuning. Optimizer* yang dipergunakan juga merupakan *optimizer* Adam, serta *learning rate* juga sebesar 1e-5. Diagram garis juga menunjukkan pola yang sama dengan ketika menggunakan 40 *epoch*. Semakin bertambah *epoch*, nilai loss akan berkurang, tetapi nilai akurasi akan bertambah (di sini hasilnya terlihat lebih fluktutatif seperti pada percobaan *Xception*, dikarenakan hasil per *epoch* tidak terlalu jauh perbedaannya). Hasil yang didapatkan termasuk sangat akurat, dikarenakan hampir semua hasil akurasi validasi selalu diatas 0.9. Hasil dari *val_accuracy* juga dapat digolongkan berhasil, karena bernilai sebesar 90 persen. Angka tersebut menunjukkan bahwa model termasuk memiliki kapabilitas dalam melakukan tugas klasifikasi citra dengan sangat baik. Kemudian, untuk rata-rata ETA yang didapatkan sebesar 322 detik, yang berarti lebih lama jika dibandingkan dengan 40 *epoch*. Temuan ini juga memperkuat argumen bahwa metode CNN memang memiliki waktu *running* yang cukup lama. Beralih ke *output* pengujian 60 *epoch*, akan divisualisasikan pada gambar berikut.



Gambar 4.63. *Classification Report 60 Epoch EfficientNet-B1*



Gambar 4.64. *Confusion Matrix 60 Epoch EfficientNet-B1*



Gambar 4.65. *ETA Testing* 60 *Epoch EfficientNet-B1*

Berdasarkan gambar-gambar diatas, dapat diambil kesimpulan bahwa hasil pengujian dari 60 *epoch* yang dieksekusi dengan total waktu sebanyak 15 detik menghasilkan *output* akurasi atau *micro average* sebesar 0.924. Akurasi tersebut tergolong besar, dan bisa menghasilkan model yang eligible dalam permasalahan klasifikasi citra. Tetapi, hasil tersebut belum dapat melampaui hasil dari percobaan menggunakan arsitektur *Xception* dengan jumlah epoch sebanyak 60, yang mencapai 0.974 pada nilai akurasinya.

4.4.4 Kesimpulan Akhir Pengujian Arsitektur *EfficientNet-B1*

Setelah melalui proses pengujian dengan menggunakan metode CNN dengan arsitektur *EfficientNet-B1* dengan jumlah *epoch* yang berbeda, telah didapatkan beberapa analisa terkait kesimpulan akhir untuk metode pengujian CNN dalam permasalahan pengklasifikasian citra ras kucing. Berikut diantaranya. Berikut diantaranya.

1. Metode klasifikasi CNN dengan arsitektur *EfficientNet-B1* merupakan metode yang memiliki hasil akurasi atau *micro average* yang sangat tinggi jika dibandingkan dengan metode SVM dan KNN dalam proses klasifikasi citra ras kucing sebanyak 13 label. Bahkan, dengan jumlah dataset yang berbeda (pada *machine learning, dataset* yang digunakan dibatasi dengan tanpa melalui proses augmentasi dan dengan penghilangan *background* untuk mengurangi jumlah *noise*), metode ini dapat mendapatkan nilai akurasi sebesar 92 persen pada percobaan keduanya (dengan 60 *epoch*). Apabila dibandingkan dengan hasil *micro average* yang dihasilkan oleh metode *machine learning* yang bahkan

tidak bisa menembus angka 30 persen, tentunya hal ini merupakan perbedaan yang sangat drastis.

2. Jumlah *epoch* yang dipergunakan saat pelatihan, mempengaruhi nilai akurasi yang akan didapatkan pada proses *testing* model. Nilai akurasi yang didapatkan bisa dikatakan cukup besar, di mana ketika menggunakan jumlah *epoch* sebesar 40 nilai akurasinya sebesar 0.88, dan ketika menggunakan jumlah *epoch* sebesar 60 mendapatkan nilai 0.92. Mengindikasikan bahwa jumlah *epoch* yang dipergunakan ketika proses pelatihan model sangat berguna dalam meraih nilai akurasi yang lebih tinggi. Kemudian, jumlah *epoch* juga mempengaruhi kecepatan waktu ketika melakukan proses *testing*. Ketika menggunakan *epoch* sebanyak 40, model menyelesaikan pelatihan untuk data sebesar 780 data citra dalam waktu 17 detik. Sedangkan, ketika menggunakan *epoch* sebanyak 60, model menyelesaikan pelatihan untuk data dengan jumlah yang sama dalam waktu 15 detik. Hal ini mengindikasikan bahwa pada percobaan dengan menggunakan arsitektur *EfficientNet*-B1, jumlah *epoch* yang dipergunakan memiliki pengaruh terhadap waktu *testing* dari model yang telah dihasilkan.
3. Setelah melakukan percobaan antara arsitektur *Xception* dan *EfficientNet*, dapat diambil kesimpulan bahwa arsitektur *Xception* memiliki nilai akurasi yang lebih tinggi, tetapi dengan ETA *training* yang lebih lama jika dibandingkan dengan *EfficientNet-B1*. Perbandingan antara keduanya dapat dilihat pada tabel berikut.

Perbandingan	Dokumentasi CNN *EfficientNet* -B1		Dokumentasi CNN *Xception*	
	60 epoch	40 epoch	40 epoch	60 epoch
Mean ETA Training	322.78s	317.6	537.075	523.7666667
ETA *Testing*	15s	17s	33	33
Accuracy *Testing*	0.924	0.882	0.926	0.974

Gambar 4.66. Tabel Perbandingan *Output Xception* dan *EfficientNet*-B1

Dari tabel tersebut, dapat diobservasi bahwa rata-rata ETA *training* yang lebih kecil dimiliki oleh arsitektur *EfficientNet*-B1, di mana selisihnya berjumlah sekitar 200 detik lebih cepat dibandingkan arsitektur *Xception*. Tetapi, setimpal dengan hasil yang didapatkan oleh arsitektur *Xception*, di mana pada kedua percoban mendapatkan hasil *micro average* atau akurasi yang lebih tinggi daripada arsitektur *EfficientNet*-B1. Sedangkan untuk waktu ETA testing, *EfficientNet*-B1 juga memiliki waktu eksekusi sekitar 55% lebih cepat pada percobaan dengan 60 *epoch*, dan 49% lebih cepat pada percobaan dengan 40 *epoch*. Jadi, kesimpulan yang dapat diambil dari percobaan kedua arsitektur

tersebut, adalah *Xception* memiliki hasil akurasi yang lebih tinggi, tetapi dengan ETA yang lebih lama jika dibandingkan dengan arsitektur *EfficientNet*-B1.

5 KESIMPULAN DAN SARAN

5.1 Kesimpulan dari Semua Pengujian

Setelah melewati proses semua skenario pengujian dari semua metode yang diujikan (SVM, KNN, CNN *Xception*, dan CNN *EfficientNet*-B1) didapatkan hasil final yang mencakup semua *output* yang diujikan. Sebagai pengingat, untuk metode *machine learning* menggunakan total data sebanyak 2600 citra (tanpa augmentasi) dengan rasio *training* dan *testing* sebesar 80:20. Sedangkan untuk *deep learning*, menggunakan total data sebanyak 7800 citra (dengan augmentasi) dengan rasio 80:10:10 untuk *training*, *validation*, dan *testing*. Berdasarkan rangkuman hasil tersebut, berikut adalah kesimpulan final yang dapat diambil.

1. Model *Xception* dengan jumlah *epoch* sebesar 60 kali memiliki hasil akurasi paling tinggi dengan nilai 0.974.
2. Metode KNN memiliki jumlah ETA *training* dan *testing* yang paling rendah, dengan rata-rata sebesar 0.194 dan 1.782 detik dari semua skenario pengujiannya.
3. Diantara kedua metode *machine learning* yang diujikan, akurasi terbaik bernilai 0.23, yang dihasilkan oleh metode SVM dengan kernel *Linear*. Memiliki arti bahwa metode SVM dan KNN dapat dikategorikan gagal dalam permaslaahan klasifikasi citra ras kucing dengan banyak kelas.
4. Diantara kedua arsitektur *deep learning* (CNN) yang diujikan, semuanya mendapatkan hasil di atas 0.85, sehingga dapat disimpulkan bahwa metode CNN merupakan metode yang cocok diujikan dalam permasalahan klasifikasi citra dengan banyak kelas.
5. Diantara kedua arsitektur CNN yang diujikan, arsitektur *EfficientNet*-B1 memiliki ETA *training* dan *testing* yang lebih cepat jika dibandingkan dengan *Xception*. Tetapi, arsitektur *Xception* memiliki nilai akurasi yang lebih baik, walau dengan ETA yang lebih lama.
6. Metode SVM dengan kernel RBF memiliki rerata ETA *testing* yang paling lama (mencapai hasil di atas 100 detik). Hal ini mengindikasikan bahwa metode *machine learning* meskipun dengan jumlah data yang lebih sedikit belum tentu memiliki waktu eksekusi yang lebih cepat dibandingkan dengan metode *deep learning*.

5.2 Saran

Setelah melihat hasil pengujian yang telah disimpulkan, terdapat beberapa saran yang dapat dilakukan untuk pengembangan skenario pengujian proses pengklasifikasian citra ras kucing dengan banyak kelas. Hal yang dapat dilakukan adalah sebagai berikut.

1. Berdasarkan pengujian yang telah dilakukan, telah disimpulkan bahwa metode CNN merupakan metode yang paling kompatibel untuk permasalahan klasifikasi citra ras kucing dengan banyak kelas, sehingga kedepannya dapat dibuat pengujian dengan memodifikasi atau menambahkan parameter pengujian pada metode CNN dengan arsitetktur *Xception* dan *EfficientNet*-B1 untuk mencari pengaturan parameter yang terbaik demi mendapatkan hasil akurasi yang lebih baik lagi.
2. Dapat dilakukan pengujian dengan aristektur lain yang belum diujikan, seperti arsitektur *ConvNeXt*, atau *MobileNetV3* untuk melihat kemampuan arsitektur tersebut dalam permasalahan klasifikasi citra ras kucing dengan banyak kelas.
3. Implementasi model terbaik (*Xception* dengan 60 *epoch*) ke dalam bentuk aplikasi berbasis *mobile* untuk menguji fungsionalitas model yang telah dibuktikan mendapat nilai akurasi yang tinggi dalam permasalahan klasifikasi citra ras kucing dengan banyak kelas, sehingga dapat membantu permasalahan utama untuk proses identifikasi ras kucing.

6.DAFTAR REFERENSI

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

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1251–1258.*

Danendra, F. A., Anggraeny, F. T., & Maulana, H. (2023). Klasifikasi Citra Penyakit Daun Cabai Rawit Dengan Menggunakan CNN Arsitektur AlexNet dan SqueezeNet. *Jurnal Informatika, 12*(01), 50–61.

Kusuma, J., Jinan, A., Zulkarnain Lubis, M., & Rosnelly, R. (2022). Komparasi Algoritma Support Vector Machine Dan Naive Bayes Pada Klasifikasi Ras Kucing. *Generic, 14*(1), 8–12.

Naufal, M. F., & Kusuma, S. F. (2023). ANALISIS PERBANDINGAN ALGORITMA MACHINE LEARNING DAN DEEP LEARNING UNTUK KLASIFIKASI CITRA SISTEM ISYARAT BAHASA INDONESIA (SIBI). *Jurnal Teknologi Informasi Dan Ilmu Komputer , 10*(4), 873–882. https://doi.org/10.25126/jtiik.2023106828

Prasath, V. B. S., Alfeilat, H. A. A., Hassanat, A. B. A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., & Salman, H. S. E. (2017). Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier -- A Review. https://doi.org/10.1089/big.2018.0175

Solihin, F., Syarief, M., Rochman, E. M. S., & Rachmad, A. (2023). Comparison of Support Vector Machine (SVM), K-Nearest Neighbor (K-NN), and Stochastic Gradient Descent (SGD) for Classifying Corn Leaf Disease based on Histogram of Oriented Gradients (HOG) Feature Extraction. *Elinvo (Electronics, Informatics, and Vocational Education), 8*(1), 121–129. https://doi.org/10.21831/elinvo.v8i1.55759

Suwarno, S., & Mahastama, A. W. (2023). ESTIMASI GENDER BERBASIS SIDIK JARI DENGAN WAVELET DAN SUPPORT VECTOR MACHINES. *Jurnal Teknologi Informasi Dan Ilmu Komputer, 10*(7), 1431–1436. https://doi.org/10.25126/jtiik.2023107972

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the 36th International Conference on Machine Learning, PMLR, 6105–6114.* http://arxiv.org/abs/1905.11946

Tarakci, F., & Ozkan, A. (2021). Comparison of classification performance of kNN and WKNN algorithms. *Selcuk University Journal of Engineering Sciences, 20*(02), 32–37. http://sujes.selcuk.edu.tr/sujes

Tarisa Akbar, A., Yudistira, N., & Ridok, A. (2023). IDENTIFIKASI GAGAL GINJAL KRONIS DENGAN MENGIMPLEMENTASIKAN METODE SUPPORT VECTOR MACHINE BESERTA K-NEAREST NEIGHBOUR (SVM-KNN). *Jurnal Teknologi Informasi Dan Ilmu Komputer, 10*(2), 301–308. https://doi.org/10.25126/jtiik.2023106059

Wijaya, E. (2023). KLASIFIKASI JENIS IKAN CUPANG MENGGUNAKAN METODE GLCM DAN SVM. *UPN Veteran Jawa Timur.*

Yaqub, M., Jinchao, F., Zia, M. S., Arshid, K., Jia, K., Rehman, Z. U., & Mehmood, A. (2020). State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images. *Brain Sciences, 10*(7), 1–19. https://doi.org/10.3390/brainsci10070427