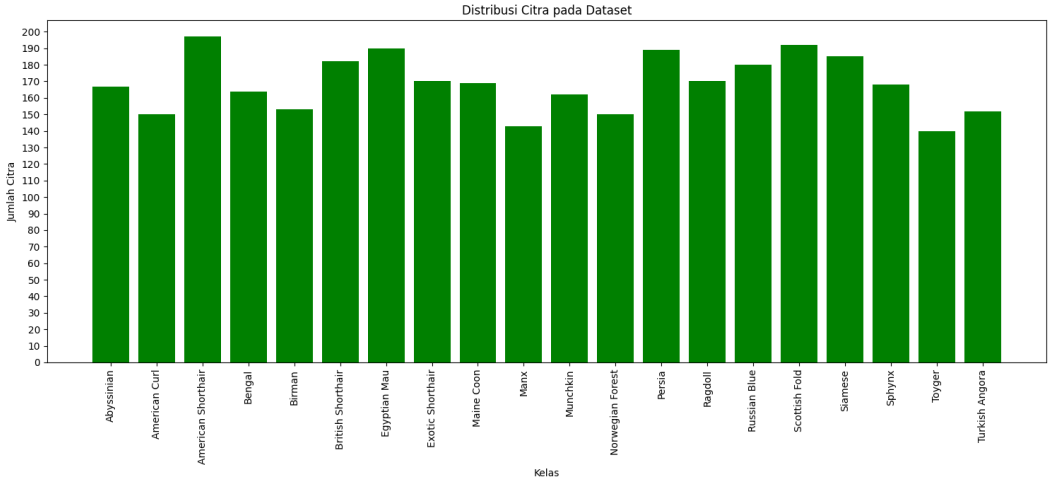
BAB IV
HASIL DAN PEMBAHASAN

4.1. Hasil *Exploratory Data Analysis* (EDA)

4.1.1 Analisis Distribusi Data

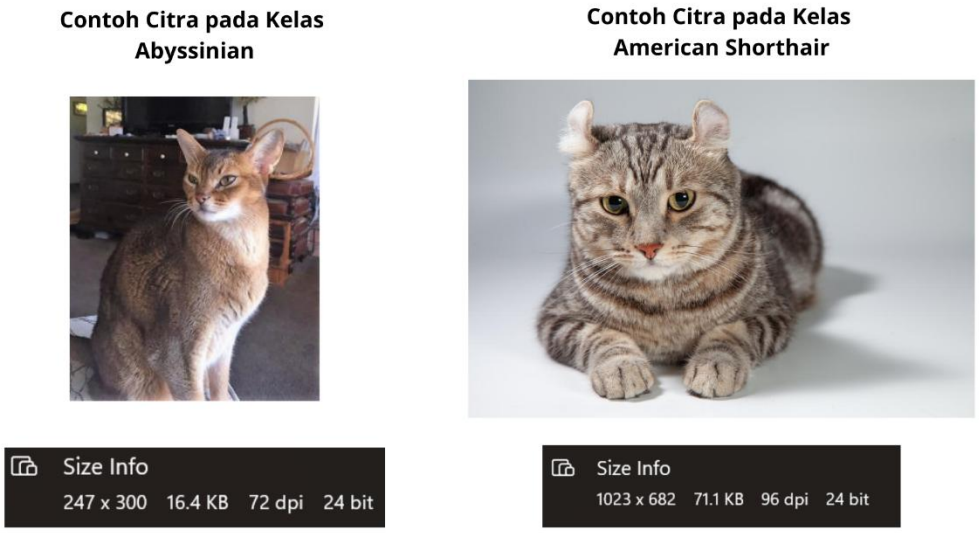
Distribusi jumlah citra pada setiap kelas ras kucing yang digunakan dalam penelitian menunjukkan adanya variasi jumlah data antar kelas yang ditunjukkan pada Gambar 4.1. Ras *American Shorthair* memiliki jumlah citra terbanyak, yaitu 197 citra, sedangkan ras *Toyger* memiliki jumlah citra paling sedikit, yaitu 140 citra. Hasil visualisasi tersebut menunjukkan bahwa distribusi data pada dataset tidak sepenuhnya seragam, sehingga terdapat perbedaan jumlah citra pada masing-masing kelas ras kucing.



Gambar 4.1 Distribusi Jumlah Citra Setiap Kelas Pada Dataset

4.1.2 Analisis Variasi Resolusi

Citra pada dataset memiliki variasi ukuran resolusi yang ditunjukkan pada Gambar 4.2. Gambar pertama yang merupakan salah satu contoh citra pada kelas *Abyssinian* memiliki resolusi sebesar 247 x 300 piksel, sedangkan gambar kedua yang merupakan salah satu contoh citra pada kelas *American Shorthair* memiliki resolusi sebesar 1023 x 682 piksel. Perbedaan resolusi tersebut menunjukkan bahwa ukuran citra pada dataset tidak seragam sehingga diperlukan proses *resizing* untuk menyesuaikan ukuran *input* model CNN.



Gambar 4.2 Variasi Resolusi pada Dataset

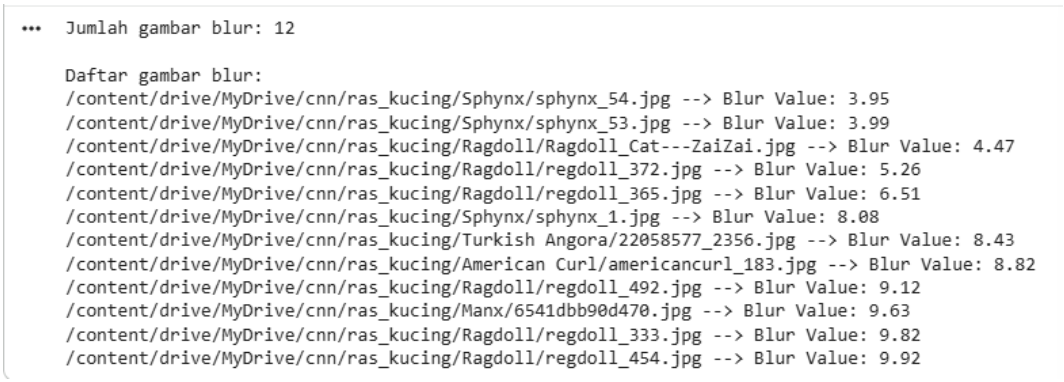
4.1.3 Pengecekan Gambar Blur dan Rusak

Pengecekan kualitas citra dilakukan menggunakan *library Pillow* (PIL) dengan memverifikasi kemampuan file gambar untuk dibuka dan dibaca dengan baik, serta memastikan kesesuaian mode warna citra. Pada penelitian ini mode warna yang digunakan adalah RGB dan RGBA sesuai dengan format warna yang digunakan pada proses pelatihan model CNN. Selain itu, pengecekan citra blur dilakukan dengan menggunakan *Laplacian Variance* dan memberikan nilai *threshold* sebesar 10 untuk mengukur tingkat ketajaman citra. Berdasarkan hasil pengecekan, ditemukan 7 citra yang memiliki mode warna P (*Paletted*), seperti yang ditunjukkan pada Gambar 4.3, sehingga tidak sesuai dengan format warna yang digunakan dalam penelitian. Selain itu, ditemukan 12 citra yang terdeteksi *blur* yang ditunjukkan pada Gambar 4.4 karena memiliki nilai ketajaman di bawah *threshold* yang telah ditentukan.

```
Jumlah gambar rusak: 7

Contoh gambar rusak:
/content/drive/MyDrive/cnn/ras_kucing/Egyptian Mau/Egyptian_Mau_191.jpg --> Mode warna tidak sesuai: P
/content/drive/MyDrive/cnn/ras_kucing/Egyptian Mau/Egyptian_Mau_167.jpg --> Mode warna tidak sesuai: P
/content/drive/MyDrive/cnn/ras_kucing/Egyptian Mau/Egyptian_Mau_145.jpg --> Mode warna tidak sesuai: P
/content/drive/MyDrive/cnn/ras_kucing/Egyptian Mau/Egyptian_Mau_129.jpg --> Mode warna tidak sesuai: L
/content/drive/MyDrive/cnn/ras_kucing/Abyssinian/Abyssinian_34.jpg --> Mode warna tidak sesuai: P
/content/drive/MyDrive/cnn/ras_kucing/Abyssinian/Image_22.png --> Mode warna tidak sesuai: P
/content/drive/MyDrive/cnn/ras_kucing/Abyssinian/Image_23.png --> Mode warna tidak sesuai: P
```

Gambar 4.3 Jumlah Gambar Rusak



```
Jumlah gambar blur: 12

Daftar gambar blur:
/content/drive/MyDrive/cnn/ras_kucing/Sphynx/sphynx_54.jpg --> Blur Value: 3.95
/content/drive/MyDrive/cnn/ras_kucing/Sphynx/sphynx_53.jpg --> Blur Value: 3.99
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/Ragdoll_Cat---ZaiZai.jpg --> Blur Value: 4.47
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/regdoll_372.jpg --> Blur Value: 5.26
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/regdoll_365.jpg --> Blur Value: 6.51
/content/drive/MyDrive/cnn/ras_kucing/Sphynx/sphynx_1.jpg --> Blur Value: 8.08
/content/drive/MyDrive/cnn/ras_kucing/Turkish Angora/22058577_2356.jpg --> Blur Value: 8.43
/content/drive/MyDrive/cnn/ras_kucing/American Curl/americancurl_183.jpg --> Blur Value: 8.82
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/regdoll_492.jpg --> Blur Value: 9.12
/content/drive/MyDrive/cnn/ras_kucing/Manx/6541dbb90d470.jpg --> Blur Value: 9.63
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/regdoll_333.jpg --> Blur Value: 9.82
/content/drive/MyDrive/cnn/ras_kucing/Ragdoll/regdoll_454.jpg --> Blur Value: 9.92
```

Gambar 4.4 Jumlah Gambar Blur

4.2 *Pre-processing* Data

Tahap *pre-processing* ini dilakukan untuk meningkatkan kualitas dataset sehingga data yang digunakan dalam proses *training* model CNN menjadi lebih bersih, seragam, dan terstruktur. Selain itu, *pre-processing* bertujuan untuk mengurangi pengaruh perbedaan karakteristik citra yang dapat mempengaruhi performa model. Pada penelitian ini, *pre-processing* meliputi data *cleaning*, *resizing* citra, normalisasi, dan augmentasi data.

4.2.1 *Data Cleaning*

Data *cleaning* dilakukan berdasarkan hasil EDA yang menunjukkan adanya citra rusak dengan mode warna yang tidak sesuai serta citra *blur*. Pada tahap ini, citra yang memiliki mode warna selain RGB dan RGBA, serta citra yang memiliki nilai ketajaman di bawah *threshold* sebesar 10 dihapus dari dataset dengan menggunakan fungsi `os.remove(img_path)`.Berdasarkan hasil data *cleaning*, sebanyak 7 citra dengan mode warna P (*Palleted*) dan 12 citra *blur* berhasil dihapus dari dataset. Dengan demikian jumlah data yang digunakan pada proses pelatihan model berkurang dari 3.373 citra menjadi 3.354 citra. Hasil pada tahap *data cleaning* ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil *Data Cleaning*

Keterangan	Jumlah Citra
Citra Rusak (Mode Warna P (*Palleted*)	7
Citra Blur	12
Total Citra Yang Dihapus	19
Jumlah Dataset Awal	3.373

Keterangan	Jumlah Citra
(Sebelum *Data Cleaning*)	
Jumlah Dataset Akhir (Setelah *Data Cleaning*)	3.354

4.2.2 *Resize*

Seluruh citra pada dataset diubah ukurannya melalui proses *resizing* untuk menyeragamkan dimensi citra yang sebelumnya memiliki ukuran yang beragam. Proses *resizing* dilakukan menggunakan fungsi `cv2.resize()` dari *library OpenCV* dengan ukuran target sebesar 224 × 224 piksel. Sebagai contoh, salah satu citra pada dataset yang semula berukuran 500 × 333 piksel, seperti ditunjukkan pada Gambar 4.5, diubah menjadi ukuran 224 × 224 piksel seperti pada Gambar 4.6. Ukuran tersebut dipilih karena sesuai dengan ukuran *input* yang digunakan pada model CNN serta dapat mengurangi kebutuhan komputasi selama proses pelatihan. Berdasarkan hasil *resizing*, seluruh citra berhasil diubah menjadi ukuran 224 × 224 piksel sehingga memiliki dimensi yang seragam. Hasil ini memudahkan proses pelatihan model CNN karena seluruh citra telah memiliki ukuran *input* yang konsisten.

Gambar 4.5 Citra Sebelum di *Resize*

Gambar 4.6 Citra Sesudah di Resize

4.2.3 *Split Data*

Pada tahap ini, dataset dibagi menjadi data *Training*, *Validation*, dan *Testing*. Pembagian data dilakukan dengan rasio 80:10:10 untuk memastikan ketersediaan data yang cukup pada proses pelatihan, validasi, dan pengujian model. Hasil pembagian dataset dapat dilihat pada Tabel 4.2, berdasarkan tabel tersebut sebagian besar data diletakkan ke data *training* untuk mendukung proses pembelajaran model CNN. Sementara itu, data *validation* digunakan untuk memantau performa model selama proses pelatihan, dan data *testing* digunakan untuk mengevaluasi performa model setelah *training* selesai dilakukan.

Tabel 4.2 Pembagian Dataset

Jenis Dataset	Jumlah Citra	Persentase
Training	2.677	80 %
Validation	333	10 %
Testing	344	10 %
Total	3.354	100 %

4.2.4 Augmentasi Data

Augmentasi data dilakukan untuk menambah variasi citra pada data *training* sekaligus mengurangi perbedaan jumlah data antar kelas. Pada penelitian ini, augmentasi diterapkan pada kelas yang jumlah data *training*nya belum mencapai 200 citra. Proses augmentasi dilakukan hingga setiap kelas memiliki 200 citra sebagai data *training.* Jumlah tersebut dipilih untuk meningkatkan ketersediaan data pelatihan tanpa menghasilkan data secara berlebihan sehingga risiko *overfitting* dapat diminimalkan.

Proses augmentasi dilakukan menggunakan *library TensorFlow*/Keras melalui *ImageDataGenerator* dengan parameter:

1. *Rotation_range* = 15
2. *Horizontal_flip* = True
3. *Width_shift_range* = 0.1
4. *Height_shift_range* = 0.1

Parameter tersebut dipilih untuk menghasilkan variasi citra yang masih menyerupai kondisi nyata tanpa mengubah karakteristik utama ras kucing yang menjadi objek klasifikasi. Hasil augmentasi disimpan secara fisik ke dalam folder dataset dan selanjutnya digunakan pada proses pelatihan model CNN. Contoh hasil augmentasi yang dihasilkan dapat dilihat pada Gambar 4.7.

Gambar 4.7 Contoh Citra Sebelum dan Sesudah Augmentasi

Jumlah data *training* sebelum augmentasi adalah sebanyak 2.677 citra. Setelah proses augmentasi dilakukan, jumlah data *training* meningkat menjadi 3.996 citra. Peningkatan jumlah data menunjukkan bahwa proses augmentasi berhasil menambah variasi citra pada data *training* sehingga dataset yang digunakan pada proses pelatihan menjadi lebih banyak dan citra antar kelas menjadi lebih seimbang.

4.3 Analisis Percobaan Konfigurasi Hyperparameter MobileNetV2

Model MobileNetV2 yang digunakan merupakan model *pre-trained* dengan bobot awal yang diperoleh dari dataset *ImageNet*. Untuk memperoleh konfigurasi yang menghasilkan performa terbaik, dilakukan beberapa optimasi hyperparameter. Setiap skenario pelatihan terdiri atas dua tahap, yaitu *feature extraction* dan *fine tuning*. Pada tahap *feature extraction*, seluruh *layer* pada

model dasar MobileNetV2 dibekukan (*freeze*) sehingga hanya lapisan klasifikasi tambahan yang dilatih. Setelah itu, proses pelatihan dilanjutkan ke tahap *fine tuning* dengan membuka kembali sebagian *layer* pada model dasar (*unfreeze layer*) sehingga model dapat menyesuaikan fitur yang telah dipelajari dengan karakteristik dataset ras kucing yang digunakan dalam penelitian.

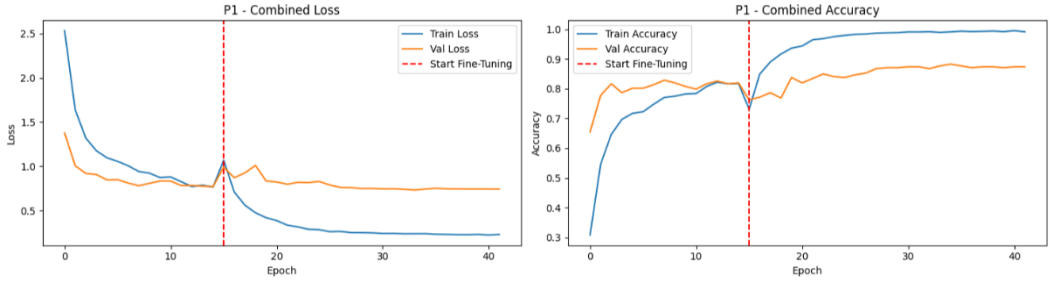
4.3.1 Konfigurasi Hyperparameter 1

Pada konfigurasi hyperparameter yang ditunjukkan pada Tabel 4.3, menggunakan *learning rate* sebesar 0,001 pada tahap *feature extraction* dan 0,0001 pada tahap *fine tuning*. Dan proses *training* memanfaatkan *callback EarlyStopping* dengan nilai *patience* 7 dan *ReduceLROnPlateau* dengan nilai *patience* 3 untuk membantu mengoptimalkan proses pelatihan.

Tabel 4.3 Konfigurasi Hyperparameter 1

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,00001 (1e-4)
Batch Size	32
Epoch FE	30
Epoch FT	30
Unfreeze Layer	30

Berdasarkan pada Gambar 4.8, grafik histori pelatihan menggunakan hyperparameter diatas, menunjukkan tahap *feature extraction* berlangsung hingga *epoch* ke-15 meskipun jumlah *epoch* yang ditentukan sebanyak 30 *epoch*. Hal ini terjadi karena mekanisme *Earlystopping* menghentikan pelatihan setelah tidak ditemukan peningkatan *validation accuracy* selama tujuh *epoch* berturut-turut. Sebelum proses *training* dihentikan, *ReduceLROnPlateau* menurunkan *learning rate* dari 0,001 menjadi 0,0002 pada *epoch* ke 11 setelah tidak terjadi perbaikan pada nilai *validation loss* selama tiga *epoch* berturut-turut. Performa terbaik pada tahap *feature extraction* tercatat pada *epoch* ke 8 dengan *validation accuracy* sebesar 82,88% dan *validation loss* sebesar 0,7798.



Gambar 4.8 Grafik *History* Pelatihan Pada Hyperparameter 1

Berdasarkan grafik *accuracy*, terlihat bahwa nilai *train accuracy* meningkat secara signifikan setelah tahap *fine tuning* dilakukan dengan membuka kembali *layer* terakhir sebanyak 30 *layer*. Nilai *validation accuracy* juga mengalami peningkatan dari 82,88% menjadi 88,29% dan *validation loss* sebesar 0,7416 pada *epoch* ke 20. Pada *fine tuning callback ReduceLROnPlateau* juga menurunkan *learning rate* dari 0,0001 menjadi 0,00002 setelah tidak terjadi perbaikan pada nilai *validation loss* selama 3 *epoch* sebelumnya. Namun, pada akhir *training* terlihat adanya jarak yang cukup besar antara kurva *train accuracy* dan *validation accuracy*. Nilai *train accuracy* mencapai 99,06%, sedangkan *validation accuracy* mencapai 88,29%, sehingga terdapat selisih sebesar 10,77%. Hasil tersebut menunjukkan bahwa nilai *validation accuracy* terbaik ada pada *epoch* ke 20 dan konfigurasi hyperparameter 1 masih menunjukkan indikasi *overfitting*.

4.3.2 Konfigurasi Hyperparameter 2

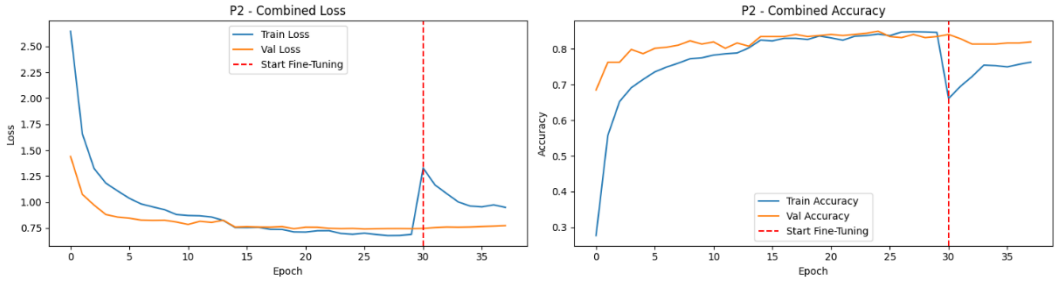
Pada konfigurasi Hyperparameter 2 yang ditunjukkan pada Tabel 4.4, digunakan *learning rate feature extraction* sebesar 0,001 dan *learning rate fine tuning* sebesar 0,000005. Penurunan *learning rate* pada *fine tuning* dan jumlah *unfreeze layer* dilakukan untuk mengevaluasi pengaruh penyesuaian bobot terhadap performa model.

Tabel 4.4 Konfigurasi Hyperparameter 2

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,000005 (5e-6)
Batch Size	32
Epoch FE	30

Parameter	Nilai
Epoch FT	30
Unfreeze Layer	25

Berdasarkan grafik pada Gambar 4.9, pada tahap *feature extraction* model menunjukkan proses pembelajaran yang cukup stabil. Nilai *validation accuracy* meningkat secara bertahap hingga mencapai 84,98% pada *epoch* ke-25 dan nilai *train accuracy* mencapai 84,10% sehingga generalisasi model masih tergolong baik. Selama proses *training, callback ReduceLROnPlateau* beberapa kali menurunkan *learning rate* karena tidak terjadi perbaikan pada *validation loss*. Penurunan *learning rate* tersebut membantu model memperoleh peningkatan performa hingga mencapai nilai *validation accuracy* terbaik sebesar 84,98%



Gambar 4.9 Grafik *History* Pelatihan Pada Hyperparameter 2

Setelah memasuki tahap *fine tuning*, sebanyak 25 *layer* terakhir MobileNetV2 dibuka kembali untuk dilatih menggunakan *learning rate* sebesar 0,000005. *Validation loss* juga tidak menunjukkan penurunan yang berarti selama tahap *fine tuning*. *Callback ReduceLROnPlateau* kembali aktif dan menurunkan *learning rate* mencapai 0,0000002. Meskipun demikian, performa model tetap tidak mengalami peningkatan sehingga proses pelatihan dihentikan pada *epoch* ke 8. Konfigurasi Hyperparameter 2 menghasilkan *validation accuracy* mencapai 84,08% dengan *validation loss* 0.7467. dan *train accuracy* mencapai 65,47%. Hal ini menunjukkan bahwa penggunaan *learning rate fine tuning* sebesar 0,000005 dan 25 *unfreeze layer* menyebabkan proses pembaruan bobot berlangsung lambat.

4.3.3 Konfigurasi Hyperparameter 3

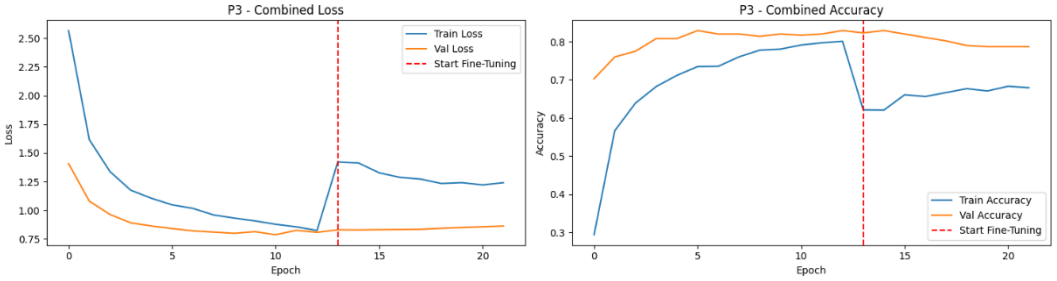
Berdasarkan grafik pada Gambar 4.10, konfigurasi hyperparameter 3 yang ditunjukkan pada Tabel 4.5 dengan menggunakan *learning rate fine tuning* sebesar 0,000002 dan 20 *unfreeze layer*. Dibandingkan konfigurasi sebelumnya, nilai *learning rate* yang digunakan lebih kecil sehingga proses pembaruan bobot

berlangsung sangat lambat. Hasil *training* menunjukkan bahwa *validation accuracy* yang diperoleh sebesar 82,88% di tahap *fine tuning* dan memiliki kesamaan hasil *validation accuracy* pada tahap *feature extraction*.

Tabel 4.5 Konfigurasi Hyperparameter 3

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,000002 (2e-6)
Batch Size	32
Epoch FE	30
Epoch FT	35
Unfreeze Layer	20

Kondisi tersebut menunjukkan bahwa proses *fine tuning* tidak memberikan peningkatan performa terhadap model. Selain itu, grafik *accuracy* menunjukkan penurunan performa setelah memasuki tahap *fine tuning*, sedangkan grafik *loss* mengalami peningkatan pada data *training* maupun *validation*. Hasil ini mengindikasikan bahwa nilai *learning rate* yang terlalu kecil dan jumlah *unfreeze layer* yang lebih sedikit menyebabkan proses adaptasi model terhadap dataset ras kucing menjadi kurang optimal.



Gambar 4.10 Grafik *History* Pelatihan Pada Parameter 3

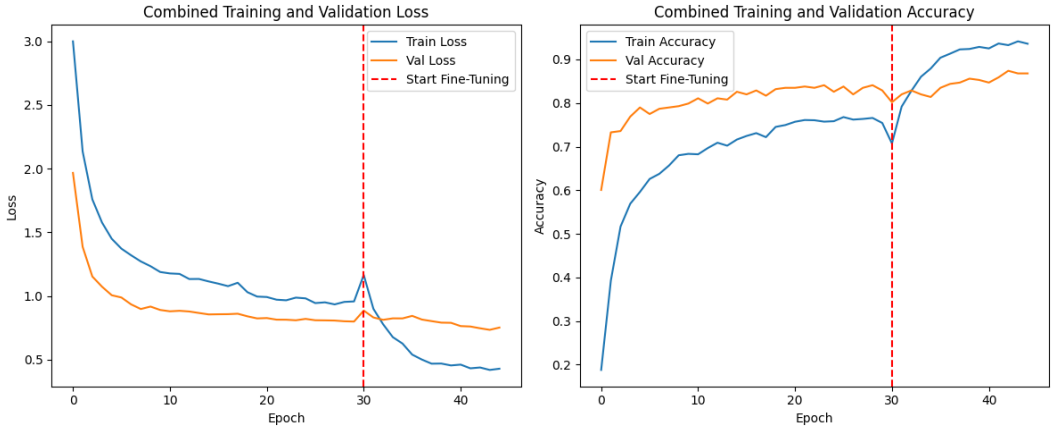
4.3.4 Konfigurasi Hyperparameter 4

Pada konfigurasi Hyperparameter 4 yang ditunjukkan pada Tabel 4.6 dengan menggunakan *learning rate feature extraction* sebesar 0,001 dan *learning rate* sebesar 0,0001. Konfigurasi ini dilakukan dengan beberapa penyesuaian, yaitu mengurangi jumlah *unfreeze layer* dari 30 menjadi 20 *layer*, mengurangi jumlah *epoch fine tuning*, serta meningkatkan nilai *dropout* dari 0,5 menjadi 0,6. Penyesuaian tersebut dilakukan untuk mengurangi kecenderungan *overfitting*.

Tabel 4.6 Konfigurasi Hyperparameter 4

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,0001 (1e-4)
Batch Size	32
Epoch FE	30
Epoch FT	15
Unfreeze Layer	20

Berdasarkan grafik *loss* pada Gambar 4.11, pada tahap *feature extraction* nilai *train loss* dan *validation loss* mengalami penurunan secara bertahap hingga mencapai kondisi yang relatif stabil. Pada epoch ke 18, *callback ReduceLROnPlateau* menurunkan *learning rate* menjadi 0,0002. Setelah penurunan *learning rate*, performa model kembali meningkat dan menghasilkan *validation accuracy* terbaik pada tahap *feature extraction* sebesar 84,08% pada *epoch* ke 24.



Gambar 4.11 Grafik History Pelatihan Pada Hyperparameter 4

Setelah memasuki tahap *fine tuning*, sebanyak 20 *layer* terakhir dibuka kembali untuk dilatih menggunakan *learning rate* sebesar 0,0001. Pada awal tahap *fine tuning* terlihat adanya peningkatan *train loss* dan penurunan *train accuracy*. Kondisi ini wajar terjadi karena model mulai menyesuaikan kembali bobot pada *layer* yang sebelumnya dibekukan. Seiring berjalannya proses *training*, nilai *train loss* kembali menurun dan *train accuracy* meningkat secara konsisten. Pada *epoch* ke 6 juga dilakukan penyesuaian *learning rate* yang

membantu meningkatkan *validation accuracy* secara bertahap menjadi 87,39% pada *epoch* ke 13 dengan *train accuracy* mencapai 93,41%.

Selisih antara *train accuracy* dan *validation accuracy* lebih kecil dibandingkan konfigurasi hyperparameter 1 disebabkan bahwa peningkatan nilai *dropout* menjadi 0,6, pengurangan jumlah *unfreeze* menjadi 20, serta batasan pada *epoch fine tuning* menjadi 15 berhasil mengurangi kecenderungan model untuk menghafal data *training* secara berlebihan.

4.3.5 Konfigurasi Hyperparameter Terbaik MobileNetV2

Hasil *training* yang telah dilakukan dengan mencoba beberapa konfigurasi hyperparameter dapat dilihat pada Tabel 4.7.

Tabel 4.7 Perbandingan Hasil Konfigurasi Hyperparameter MobileNetV2

Konfigurasi	***Validation Accuracy***	***Validation Loss***
Hyperparameter 1	88,29 %	0,7416
Hyperparameter 2	84,08 %	0,7467
Hyperparameter 3	82,88 %	0,8294
Hyperparameter 4	87,39 %	0,7469

Berdasarkan Tabel diatas, konfigurasi hyperparameter 1 menghasilkan *validation accuracy* tertinggi sebesar 88,29%. Namun, konfigurasi tersebut masih menunjukkan indikasi *overfitting* yang tinggi dengan selisih *train accuracy* dan *validation accuracy* sebesar 10,77%. Sementara itu, konfigurasi hyperparameter 4 menghasilkan *validation accuracy* sebesar 87,39% dengan slisih *train accuracy* dan *validation accuracy* yang lebih kecil yaitu 6,02%. Hasil tersebut menunjukkan bahwa konfigurasi hyperparameter 4 memiliki kemampuan generalisasi yang lebih baik sehingga dipilih sebagai konfigurasi terbaik untuk model MobileNetV2.

4.4 Analisis Hasil Konfigurasi Hyperparameter ResNet50

Model ResNet50 dilatih menggunakan beberapa konfigurasi hyperparameter yang berbeda untuk memperoleh konfigurasi yang menghasilkan performa terbaik pada tugas klasifikasi citra ras kucing. Analisis dilakukan terhadap hasil *training* setiap konfigurasi berdasarkan nilai *accuracy* dan l*oss* yang diperoleh selama proses *training* dan *validation.*

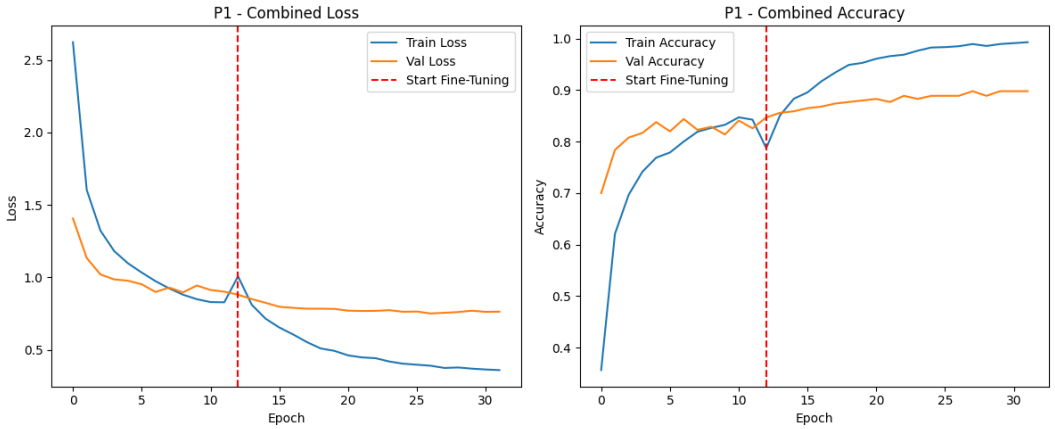
4.4.1 Konfigurasi Hyperparameter 1

Konfigurasi hyperparameter 1 yang ditunjukkan pada Tabel 4.8 menggunakan *learning rate feature extraction* sebesar 0,001 dan *learning rate fine tuning* sebesar 0,00001 dan 50 *unfreeze layer*. Jumlah *unfreeze layer* yang relatif besar dipilih karena ResNet50 merupakan arsitektur CNN yang memiliki struktur jaringan yang lebih dalam dibandingkan model CNN konvensional serta dilengkapi mekanisme *residual connection.*

Tabel 4.8 Konfigurasi Hyperparameter 1 ResNet50

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,00001 (1e-5)
Batch Size	32
Epoch FE	30
Epoch FT	20
Unfreeze Layer	50

Berdasarkan grafik pada Gambar 4.12, pada tahap *feature extraction* model menunjukkan nilai *validation accuracy* mencapai 84,38% pada *epoch* ke 7 yang merupakan nilai terbaik pada tahap ini. Setelah *epoch* ke 7, peningkatan performa mulai melambat dan tidak menunjukkan perbaikan yang signifikan. Oleh karena itu, *callback EarlyStopping* menghentikan proses *training* pada *epoch* ke 12 dan mengembalikan bobot model terbaik pada *epoch* ke 7.



Gambar 4.12 Grafik *History* Pelatihan pada Hyperparameter 1 ResNet50

Pada tahap *fine tuning* dengan membuka kembali 50 *layer* terakhir ResNet50. Berdasarkan grafik *accuracy*, tahap *fine tuning* memberikan peningkatan performa yang cukup signifikan. Nilai *validation accuracy* meningkat secara bertahap menjadi 89,79% dan *validation loss* mencapai 0,7559 pada *epoch* ke 16. Peningkatan tersebut menunjukkan bahwa pembukaan 50 *layer* terakhir memungkinkan model menyesuaikan fitur – fitur yang telah dipelajari dari *ImageNet* dengan karakteristik citra ras kucing secara lebih optimal. Di sisi lain, *train accuracy* juga mengalami peningkatan hingga mencapai 98,95%, yang menunjukkan bahwa model mampu mempelajari pola pada data *training* dengan sangat baik.

Berdasarkan grafik *accuracy* dan *loss*, terlihat bahwa proses *fine tuning* berhasil meningkatkan kemampuan model dalam mengenali pola pada dataset citra ras kucing. Hal ini ditunjukkan oleh peningkatan *validation accuracy* dari tahap *feature extraction* ke tahap *fine tuning*. Namun demikian, terdapat selisih sekitar antara *train accuracy* dan *validation accuracy* pada performa terbaik model. Selisih tersebut menunjukkan adanya indikasi *overfitting*, meskipun model masih mampu mempertahankan performa validasi yang tinggi. Secara keseluruhan, konfigurasi hyperparameter 1 menunjukkan bahwa kombinasi *learning rate fine tuning* sebesar 0,00001 dan 50 *unfreeze layer* mampu meningkatkan performa model ResNet50 secara efektif.

4.4.2 Konfigurasi Hyperparameter 2

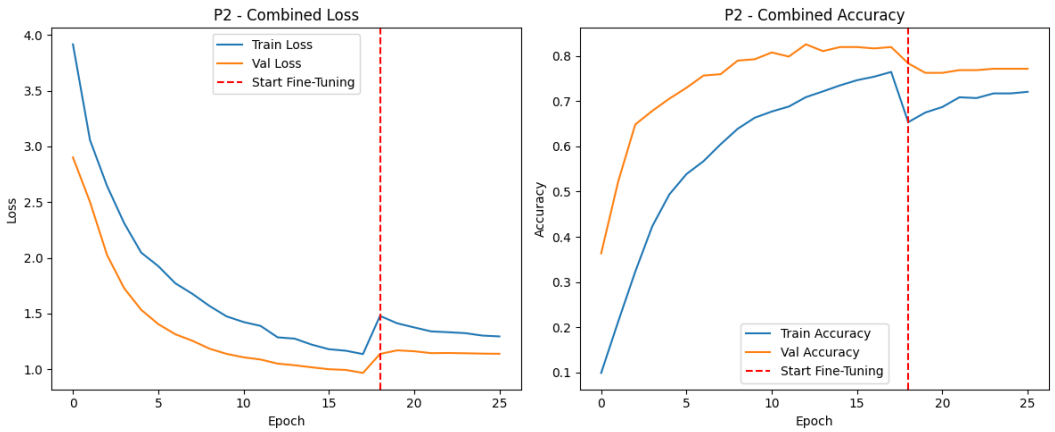
Konfigurasi hyperparameter 2 menggunakan *learning rate feature extraction* sebesar 0,0001, *learning rate fine tuning* sebesar 0,000001, *batch size* 32, *epoch feature extraction* sebanyak 30 *epoch*, *epoch fine tuning* sebanyak 20 *epoch*, dan 50 *unfreeze layer* yang ditunjukkan pada Tabel 4.9. Konfigurasi ini dirancang untuk mengevaluasi pengaruh penggunaan *learning rate* yang lebih kecil dibandingkan konfigurasi hyperparameter 1 terhadap performa model ResNet50.

Tabel 4.9 Konfigurasi Hyperparameter 2 ResNet50

Parameter	Nilai
Learning Rate FE	0,0001 (1e-4)
Learning Rate FT	0,000001 (1e-6)

Parameter	Nilai
Batch Size	32
Epoch FE	30
Epoch FT	20
Unfreeze Layer	50

Berdasarkan grafik pada Gambar 4.13, pada tahap *feature extraction* model menunjukkan peningkatan performa secara bertahap. Nilai *validation accuracy* meningkat dari 36,34% pada *epoch* pertama menjadi 82,58% pada *epoch* ke-13 yang merupakan performa terbaik pada tahap ini. Setelah *epoch* ke-13, peningkatan performa mulai melambat dan tidak menunjukkan perbaikan yang signifikan. Oleh karena itu, *callback EarlyStopping* dengan *patience* 5 menghentikan proses pelatihan pada *epoch* ke-18. Setelah memasuki tahap *fine tuning*, sebanyak 50 *layer* terakhir ResNet50 dibuka kembali untuk dilatih menggunakan *learning rate* sebesar 0,000001. Nilai *learning rate* yang sangat kecil dipilih untuk menjaga kestabilan proses penyesuaian bobot pada model *pre-trained*. Namun, berdasarkan grafik *accuracy*, peningkatan performa yang dihasilkan relatif terbatas. Nilai *validation accuracy* terbaik hanya mencapai 78,38% pada *epoch* pertama *fine tuning* dan tidak mampu melampaui performa terbaik yang telah diperoleh pada tahap *feature extraction*.



Gambar 4.13 Grafik History Pelatihan pada Hyperparameter 2 ResNet50

Berdasarkan grafik *loss*, terlihat bahwa *train loss* dan *validation loss* mengalami penurunan secara bertahap selama tahap *feature extraction*. Namun, setelah memasuki tahap *fine tuning*, *validation loss* cenderung stagnan dan tidak

menunjukkan perbaikan yang berarti. Kondisi tersebut menyebabkan *callback ReduceLROnPlateau* aktif pada *epoch* ke-4 dan menurunkan *learning rate* dari 0,000001 menjadi 0,0000002. Secara keseluruhan, konfigurasi hyperparameter 2 menghasilkan *validation accuracy* 78,38% dan *validation loss* mencapai 1,1378 pada tahap *fine tuning*. Hal ini menunjukkan bahwa penggunaan *learning rate feature extraction* dan *learning rate fine tuning* yang terlalu kecil menyebabkan proses pembelajaran model menjadi kurang optimal.

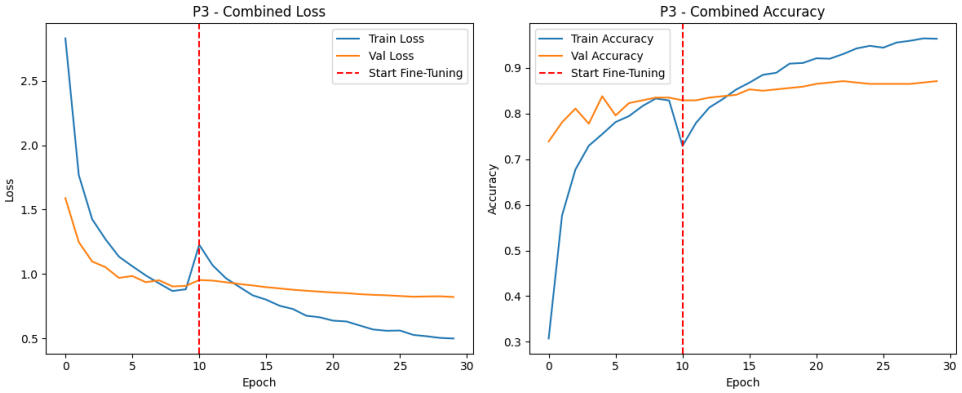
4.4.3 Konfigurasi Hyperparameter 3

Konfigurasi hyperparameter 3 menggunakan *learning rate feature extraction* sebesar 0,001, *learning rate fine tuning* sebesar 0,000005 ditunjukkan pada Tabel 4.10. Konfigurasi ini dirancang untuk mengevaluasi pengaruh pembukaan layer yang lebih banyak pada tahap *fine tuning* serta penggunaan *batch size* yang lebih besar dibandingkan konfigurasi sebelumnya.

Tabel 4.10 Konfigurasi Hyperparameter 3 ResNet50

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,000005 (5e-6)
Batch Size	64
Epoch FE	50
Epoch FT	30
Unfreeze Layer	70

Berdasarkan Gambar 4.14, pada tahap *feature extraction* model menunjukkan peningkatan performa yang cukup cepat. Nilai *validation accuracy* meningkat dari 73,87% pada *epoch* pertama menjadi 83,78% pada *epoch* ke-5. Setelah *epoch* tersebut, performa validasi cenderung berfluktuasi dan tidak menunjukkan peningkatan yang signifikan. Oleh karena itu, *callback EarlyStopping* menghentikan proses pelatihan pada *epoch* ke-10 dan mengembalikan bobot model terbaik pada epoch ke-5.



Gambar 4.14 Grafik *History* Pelatihan pada Hyperparameter 3 ResNet50

Berdasarkan grafik *accuracy*, tahap *fine tuning* berhasil meningkatkan performa model secara bertahap. Nilai *validation accuracy* meningkat dari 83,78% menjadi 87,09% pada *epoch* ke-13 yang merupakan performa terbaik model. Pada saat yang sama, *train accuracy* terus meningkat hingga mencapai sekitar 93,18% Peningkatan tersebut menunjukkan bahwa pembukaan 70 *layer* terakhir mampu membantu model mempelajari karakteristik dataset secara lebih mendalam dibandingkan tahap *feature extraction*. Secara keseluruhan, konfigurasi hyperparameter 3 menghasilkan *validation accuracy* terbaik sebesar 87,09% dengan *validation loss* sebesar 0,8447 pada *epoch* ke 13. Dengan demikian, konfigurasi ini menunjukkan bahwa peningkatan jumlah *unfreeze layer* hingga 70 *layer* dan penggunaan *batch size* yang lebih besar belum mampu menghasilkan performa terbaik pada model ResNet50 untuk klasifikasi citra ras kucing dengan dataset yang digunakan pada penelitian ini.

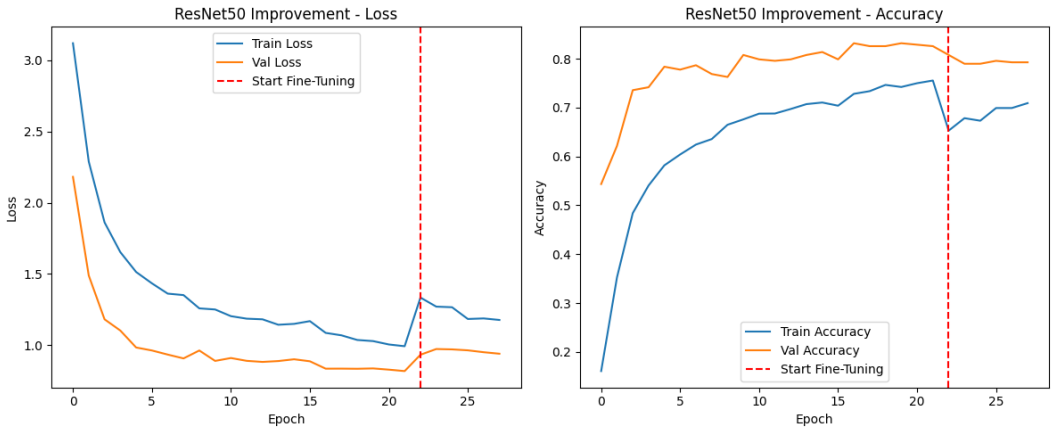
4.4.4 Konfigurasi Hyperparameter 4

Konfigurasi hyperparameter 4 yang ditunjukkan pada Tabel 4.11 merupakan pengembangan dari konfigurasi hyperparameter 1 yang sebelumnya menghasilkan performa terbaik pada model ResNet50. Perubahan dilakukan dengan mengurangi jumlah *unfreeze layer* dari 50 menjadi 35 *layer*, menurunkan jumlah *neuron* pada *dense layer* dari 256 menjadi 128 *neuron*, meningkatkan nilai *dropout* dari 0,5 menjadi 0,6. Perubahan tersebut dilakukan untuk mengurangi indikasi *overfitting* yang masih ditemukan pada konfigurasi hyperparameter 1.

Tabel 4.11 Konfigurasi Hyperparameter 4 ResNet50

Parameter	Nilai
Learning Rate FE	0,001 (1e-3)
Learning Rate FT	0,000001 (1e-6)
Batch Size	32
Epoch FE	30
Epoch FT	20
Unfreeze Layer	35

Berdasarkan grafik pada Gambar 4.15, Terlihat bahwa selama tahap *feature extraction*, nilai *train accuracy* dan *validation accuracy* meningkat secara bertahap hingga mencapai performa terbaik sebesar 83,18% pada *epoch* ke-17. Namun, setelah memasuki tahap *fine tuning*, nilai *validation accuracy* tidak mengalami peningkatan yang signifikan dan justru cenderung menurun hingga berada pada 80,78%. Berdasarkan grafik *loss*, terlihat bahwa *train loss* dan *validation loss* mengalami penurunan selama tahap *feature extraction*. Namun, setelah memasuki tahap *fine tuning*, nilai *validation loss* tetap berada pada kisaran yang relatif tinggi yaitu 0.9318.



Gambar 4.15 Grafik *History* Pelatihan pada Hyperparameter 4 ResNet50

Hasil tersebut diduga dipengaruhi oleh kombinasi beberapa perubahan hyperparameter yang dilakukan, yaitu pengurangan jumlah *unfreeze layer* dari 50 menjadi 35 *layer*, penurunan jumlah *neuron* pada *dense layer* dari 256 menjadi 128 *neuron*, peningkatan nilai *dropout* dari 0,5 menjadi 0,6, serta penggunaan *learning rate fine tuning* yang lebih kecil sebesar 0,000001. Kombinasi perubahan

tersebut menyebabkan kapasitas model untuk mempelajari pola data menjadi lebih terbatas sehingga proses *fine tuning* tidak mampu meningkatkan performa model seperti yang diharapkan.

4.4.5 Konfigurasi Hyperparameter Terbaik ResNet50

Hasil *training* yang telah dilakukan dengan mencoba beberapa konfigurasi hyperparameter dapat dilihat pada Tabel 4.12. konfigurasi hyperparameter 1 menghasilkan *validation accuracy* tertinggi sebesar 89,79%, serta *validation loss* terendah sebesar 0,7559 dibandingkan konfigurasi lainnya. Hasil tersebut menunjukkan bahwa konfigurasi hyperparameter 1 memiliki performa yang paling baik sehingga dipilih sebagai konfigurasi terbaik pada model ResNet50.

Tabel 4.12 Perbandingan Hasil Konfigurasi Hyperparameter ResNet50

Konfigurasi	***Validation Accuracy***	***Validation Loss***
Hyperparameter 1	89,79%	0,7559
Hyperparameter 2	78,38%	1.1378
Hyperparameter 3	87,09%	0,8447
Hyperparameter 4	80,78%	0.9318

4.5 Analisis Hasil Konfigurasi Hyperparameter AlexNet

Model AlexNet pada penelitian ini dibangun dan dilatih dari awal (*from scratch*) tanpa menggunakan bobot *pre-trained* dari dataset lain. Berbeda dengan MobileNetV2 dan ResNet50 yang menerapkan *transfer learning*, seluruh parameter pada model AlexNet dipelajari secara langsung menggunakan dataset ras kucing yang digunakan dalam penelitian.

4.5.1 Konfigurasi Hyperparameter 1

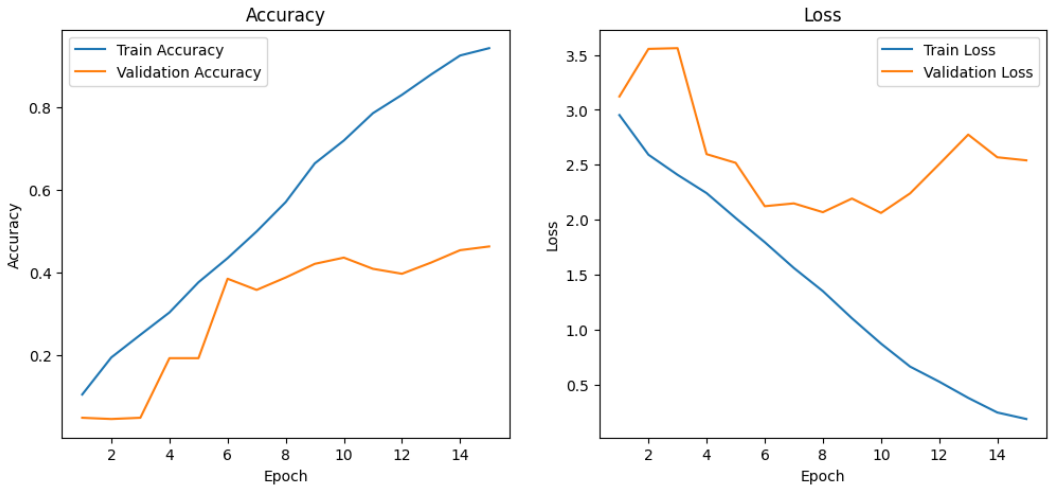
Konfigurasi hyperparameter 1 ditunjukkan pada Tabel 4.13, dan menggunakan *callback EarlyStopping* dengan nilai *patience* 5, serta *ReduceLROnPlateau* dengan *patience* 3. Serta menggunakan arsitektur yang terdiri atas lima *convolutional layer*, *max pooling layer*, dan *fully connected layer* berukuran 1024 dan 512 *neuron*, *dropout* sebesar 0,5 pada setiap *fully connected*.

Tabel 4.13 Konfigurasi Hyperparameter 1 AlexNet

Parameter	**Nilai**
Learning Rate	0,0001 (1e-4)

Parameter	Nilai
Batch Size	32
Epoch	30

Berdasarkan grafik pada Gambar 4.16, nilai *train accuracy* meningkat secara konsisten dari 10,44% pada *epoch* pertama hingga mencapai 94,22% pada *epoch* ke-15. Peningkatan tersebut menunjukkan bahwa model mampu mempelajari pola pada data *training* dengan sangat baik. Namun, peningkatan *train accuracy* tidak diikuti oleh peningkatan *validation accuracy* yang sebanding. Nilai *validation accuracy* tertinggi hanya mencapai 46,25% pada *epoch* ke-15.



Gambar 4.16 Grafik *History* Pelatihan pada Hyperparameter 1 AlexNet

Secara keseluruhan, konfigurasi hyperparameter 1 menghasilkan *validation accuracy* sebesar 43,54% dengan *validation loss* sebesar 2.0619. Meskipun model mampu mencapai *train accuracy* sebesar 71,87%, perbedaan yang cukup besar antara *train accuracy* dan *validation accuracy* menunjukkan bahwa model mengalami *overfitting*. Hasil ini mengindikasikan bahwa konfigurasi yang digunakan belum mampu menghasilkan kemampuan generalisasi yang baik pada model AlexNet untuk klasifikasi citra ras kucing.

4.5.2 Konfigurasi Hyperparameter 2

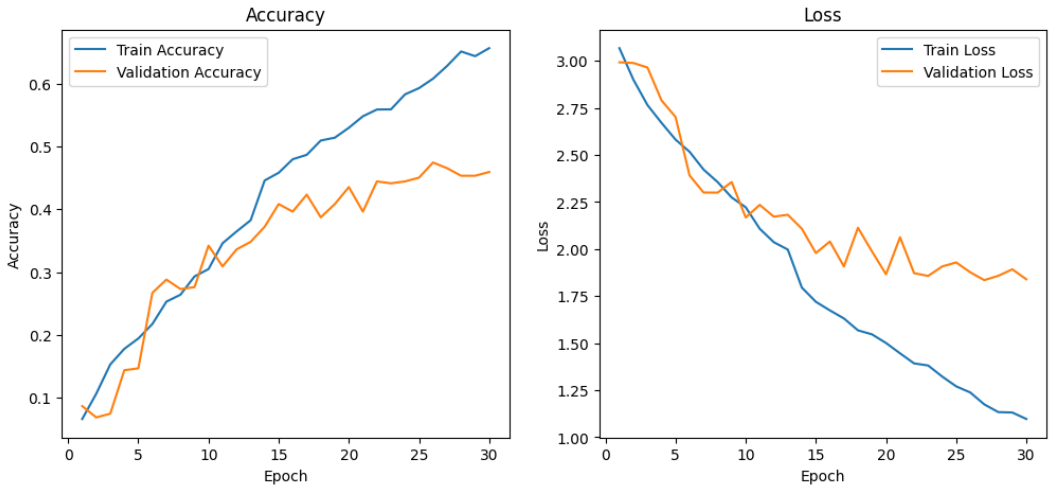
Konfigurasi hyperparameter 2 yang ditunjukkan pada Tabel 4.14, dengan mengurangi jumlah *neuron pada fully connected layer* menjadi 512-256 neuron, serta meningkatkan nilai *droupout* menjadi 0,6. Perubahan tersebut dilakukan untuk mengurangi kompleksitas model dan menekan kecenderungan *overfitting*

yang ditemukan pada konfigurasi hyperparameter 1. Selain itu ukuran *input* citra juga diubah menjadi 128 x 128 piksel.

Tabel 4.14 Konfigurasi Hyperparameter 2 AlexNet

Parameter	Nilai
Learning Rate	0,0001 (1e-4)
Batch Size	32
Epoch	30

Berdasarkan grafik pada Gambar 4.17, nilai *train accuracy* meningkat secara bertahap dari 6,66% pada *epoch* pertama menjadi 65,64% pada *epoch* ke-30. Sementara itu, *validation accuracy* meningkat dari 8,71% menjadi 47,45% pada *epoch* ke-26 yang merupakan nilai tertinggi selama proses pelatihan. Selisih antara *train accuracy* dan *validation accuracy* pada konfigurasi ini lebih kecil. Pada akhir pelatihan, *train accuracy* mencapai 65,64% sedangkan *validation accuracy* mencapai 45,95%. Kondisi tersebut menunjukkan bahwa kemampuan generalisasi model mengalami perbaikan dan indikasi *overfitting* menjadi lebih rendah dibandingkan konfigurasi sebelumnya.



Gambar 4.17 Grafik *History* Pelatihan pada Hyperparameter 2 AlexNet

Berdasarkan grafik *loss*, *train loss* mengalami penurunan secara konsisten dari 3.0675 menjadi 1.0968 hingga akhir pelatihan. *Validation loss* juga menunjukkan penurunan dari 2.9922 menjadi sekitar 1.8391. Secara keseluruhan, Konfigurasi hyperparameter 2 menghasilkan *validation accuracy* terbaik sebesar 47,45% pada *epoch* ke 26. Meskipun nilai tersebut hanya sedikit lebih tinggi

dibandingkan konfigurasi hyperparameter 1, grafik *accuracy* dan *loss* menunjukkan bahwa model memiliki kemampuan generalisasi yang lebih baik serta indikasi *overfitting* yang lebih rendah.

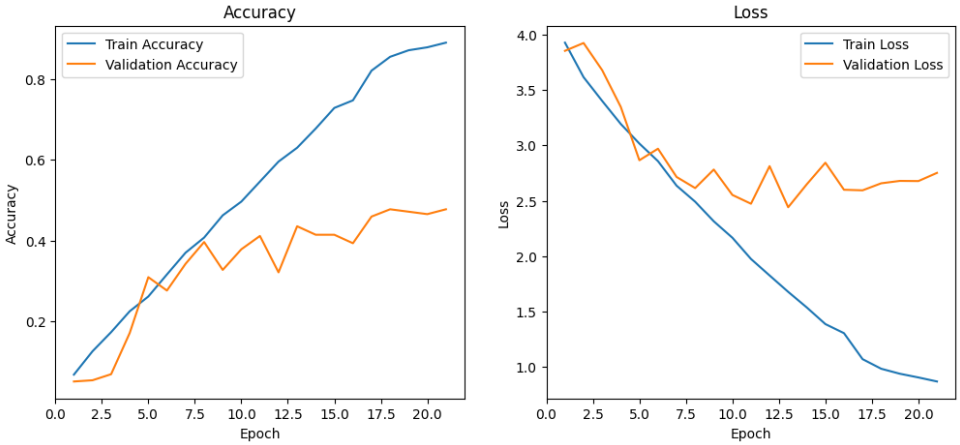
4.5.3 Konfigurasi Hyperparameter 3

Konfigurasi hyperparameter 3 yang ditunjukkan pada Tabel 4.15, melakukan penyesuaian pada bagian *fully connected layer* dengan mengurangi jumlah *neuron* menjadi 512 dan 256 neuron, menambahkan regularisasi L2 sebesar 0,001 pada *layer dense* pertama, serta menerapkan *dropout* sebesar 0,6 dan 0,5. Selain itu, nilai *patience* pada *callback EarlyStopping* ditingkatkan menjadi 8 *epoch* agar model memiliki kesempatan belajar lebih lama sebelum pelatihan dihentikan. Penyesuaian tersebut dilakukan untuk mengurangi *overfitting* yang masih ditemukan pada konfigurasi sebelumnya**.**

Tabel 4.15 Konfigurasi Hyperparameter 3 AlexNet

Parameter	Nilai
Learning Rate	0,0001 (1e-4)
Batch Size	32
Epoch	30

Berdasarkan grafik pada Gambar 4.18, nilai *train accuracy* meningkat secara bertahap dari 6,81% pada *epoch* pertama hingga mencapai 89,01% pada epoch ke-21. Sementara itu, *validation accuracy* meningkat dari 5,11% menjadi 47,75% yang merupakan nilai tertinggi selama proses pelatihan. Namun, selisih antara *train accuracy* dan *validation accuracy* masih cukup besar. Kondisi ini menunjukkan bahwa model masih mengalami *overfitting* meskipun berbagai teknik regularisasi telah diterapkan.



Gambar 4.18 Grafik *History* Pelatihan pada Hyperparameter 3 AlexNet

Berdasarkan grafik loss, *train loss* mengalami penurunan secara konsisten dari sekitar 3.9281 menjadi 0.8671. Sebaliknya, *validation loss* hanya mengalami penurunan pada awal pelatihan hingga mencapai nilai terendah sekitar 2.4411 pada *epoch* ke-13, kemudian kembali mengalami fluktuasi dan cenderung meningkat pada *epoch-epoch* berikutnya.

Secara keseluruhan, konfigurasi hyperparameter 3 menghasilkan *validation accuracy* terbaik sebesar 43,54% pada epoch ke-13. Namun, *grafik accuracy* dan *loss* menunjukkan bahwa model masih mengalami *overfitting* yang ditandai dengan selisih yang cukup besar antara *train accuracy* dan *validation accuracy*. Meskipun penambahan regularisasi L2 dan *dropout* berhasil meningkatkan performa validasi, kemampuan generalisasi model AlexNet masih belum optimal untuk klasifikasi citra ras kucing dengan dataset yang digunakan pada penelitian.

4.5.4 Konfigurasi Hyperparameter Terbaik AlexNet

Hasil *training* yang telah dilakukan dengan mencoba beberapa konfigurasi hyperparameter dapat dilihat pada Tabel 4.16. berdasarkan hasil *training*, konfigurasi hyperparameter 2 yang menghasilkan nilai *validation accuracy* tertinggi sebesar 47,45% dan *validation loss* terendah sebesar 1.8766. Hasil yang diperoleh menunjukkan bahwa performa model AlexNet masih berada di bawah MobileNetV2 dan ResNet50. Hal ini dapat dipengaruhi oleh pendekatan pelatihan yang digunakan, di mana AlexNet dibangun dan dilatih dari awal (*from scratch*) tanpa memanfaatkan bobot *pre-trained* dari dataset *ImageNet*. Akibatnya, seluruh

parameter model harus dipelajari langsung dari dataset penelitian sehingga membutuhkan data dan proses pelatihan yang lebih besar untuk mencapai performa yang optimal.

Tabel 4.16 Perbandingan Hasil Konfigurasi Hyperparameter AlexNet

Konfigurasi	***Validation Accuracy***	***Validation Loss***
Hyperparameter 1	43,54%	2.0619
Hyperparameter 2	47,45%	1.8766
Hyperparameter 3	43,54 %	2.4411

4.6 Evaluasi Kinerja Model MobileNetV2

Setelah melakukan *training* dan proses optimasi hyperparameter selesai dilakukan, konfigurasi terbaik pada model MobileNetV2 dievaluasi menggunakan data *testing* yang terdiri dari 344 citra dari 20 kelas ras kucing. Hasil evaluasi menunjukkan bahwa model memperoleh nilai *Test Accuracy* sebesar 88,95% dan *Test Loss* sebesar 0,6167 seperti yang ditunjukkan pada Gambar 4.19. Nilai tersebut diperoleh menggunakan fungsi *model.evaluate*() pada *TensorFlow*/Keras.

```
test_loss, test_acc = model.evaluate(test_generator)

print("\nTest Loss     :", test_loss)
print("Test Accuracy:", test_acc)
```

```
Found 344 images belonging to 20 classes.
11/11 ━━━━━━━━━━━━━━━━━━━━ 10s 397ms/step - accuracy: 0.8895 - loss: 0.6168

Test Loss     : 0.6167775392532349
Test Accuracy: 0.8895348906517029
```

Gambar 4.19 Hasil Test Menggunakan Data *Testing* Pada Model MobileNetV2

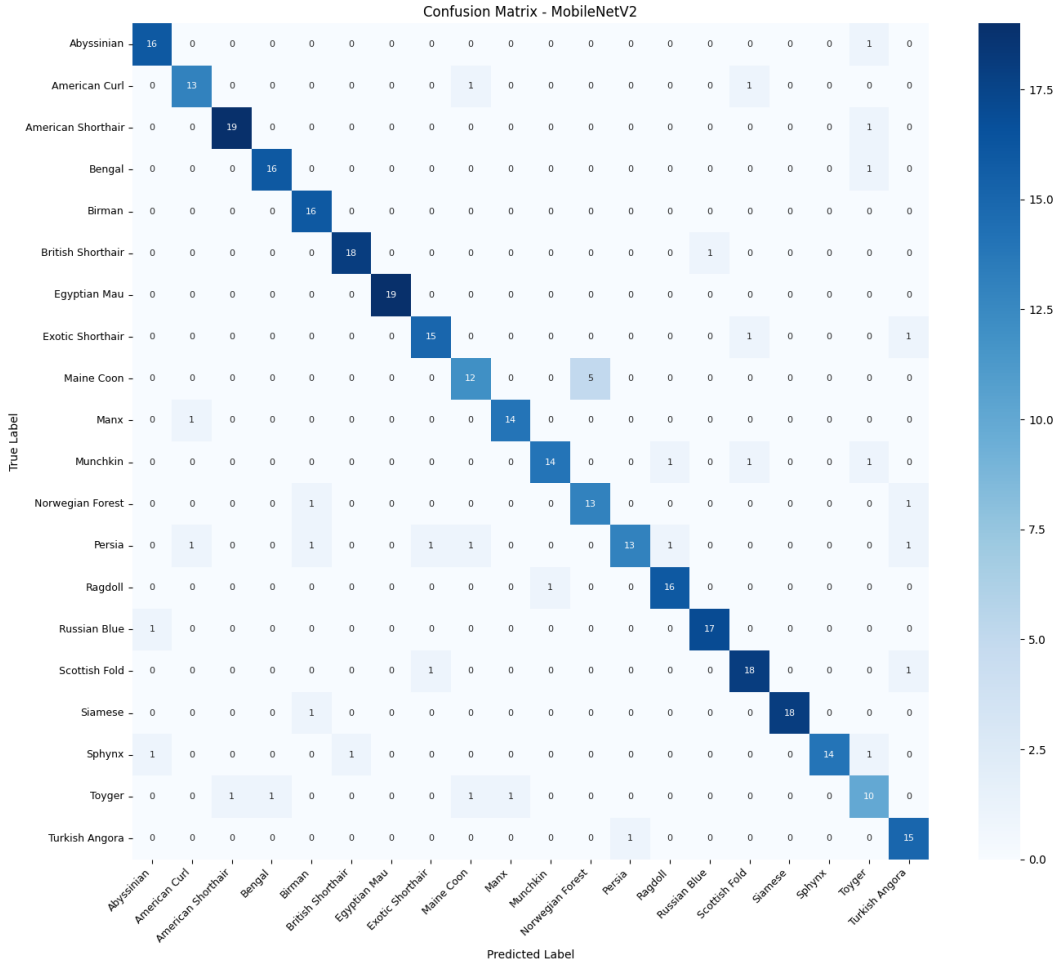
Berdasarkan *confusion matrix* pada Gambar 4.20, model berhasil mengklasifikasikan 306 citra dengan benar dari total 344 citra data testing. Oleh karena itu, nilai *accuracy* dapat dihitung menggunakan persamaan sebagai berikut:

$$Accuracy = \frac{\text{Jumlah Prediksi Benar}}{\text{Jumlah Seluruh Data Testing}} = \frac{306}{344}$$

= 0.8895

Nilai yang berada pada diagonal utama menunjukkan jumlah citra yang berhasil diklasifikasi dengan benar (*True Positive)*, sedangkan nilai yang berada di

luar diagonal menunjukkan kesalahan klasifikasi yang dilakukan oleh model. Semakin banyak nilai yang berada pada diagonal utama, semakin baik kemampuan model dalam melakukan klasifikasi.



Gambar 4.20 *Confusion Matrix* pada Model MobileNetV2

Sebagai contoh, pada kelas *Abyssinian* terdapat 16 citra yang berhasil diklasifikasikan sebagai Ras *Abyssinian* (*True Positive*), 1 citra yang diprediksi sebagai Ras *Toyger* (False Negative), dan 2 citra dari kelas lain yang diprediksi sebagai Ras *Abyssinian* (*False Positive*). Hasil tersebut menghasilkan nilai *precision* sebesar 0,89, nilai *recall* sebesar 0,94, nilai F1-*Score* sebesar 0,91, sebagaimana ditunjukkan pada *classification report* pada Gambar 4.21.

1. Hasil Implementasi *Precision*, *Recall*, dan *F1-Score* pada Kelas *Abyssinian*

a. $Precision = \frac{TP}{TP+FP} = \frac{16}{16+2}$

$= 0{,}89$

b. $Recall = \frac{TP}{TP+FN} = \frac{16}{16+1}$

= 0,94

c. $F1 - Score = 2\ x\ \frac{\text{Precision x Recall}}{\text{Precision + Recall}} = 2\ x\ \frac{0,89\ x\ 0,94}{0,89+0,94}$

$= 2\ x\ \frac{0,84}{1,83} = 0,91$

```
Found 344 images belonging to 20 classes.
11/11 ━━━━━━━━━━━━━━━━━━━━ 12s 569ms/step
Accuracy: 0.8895348837209303

Classification Report:

                    precision    recall  f1-score   support

        Abyssinian       0.89      0.94      0.91        17
     American Curl       0.87      0.87      0.87        15
American Shorthair       0.95      0.95      0.95        20
            Bengal       0.94      0.94      0.94        17
            Birman       0.84      1.00      0.91        16
 British Shorthair       0.95      0.95      0.95        19
      Egyptian Mau       1.00      1.00      1.00        19
  Exotic Shorthair       0.88      0.88      0.88        17
       Maine Coon       0.80      0.71      0.75        17
              Manx       0.93      0.93      0.93        15
          Munchkin       0.93      0.82      0.88        17
  Norwegian Forest       0.72      0.87      0.79        15
           Persia       0.93      0.68      0.79        19
          Ragdoll       0.89      0.94      0.91        17
      Russian Blue       0.94      0.94      0.94        18
     Scottish Fold       0.86      0.90      0.88        20
           Siamese       1.00      0.95      0.97        19
            Sphynx       1.00      0.82      0.90        17
            Toyger       0.67      0.71      0.69        14
    Turkish Angora       0.79      0.94      0.86        16

          accuracy                           0.89       344
         macro avg       0.89      0.89      0.89       344
      weighted avg       0.89      0.89      0.89       344
```

Gambar 4.21 *Classification Report* Model MobileNetV2

4.7 Evaluasi Kinerja Model ResNet50

Setelah melakukan *training* dan proses optimasi hyperparameter selesai dilakukan, konfigurasi terbaik pada model ResNet50 dievaluasi menggunakan data *testing* dengan jumlah 344 citra. Hasil evaluasi menunjukkan bahwa ResNet50 memperoleh nilai *Test Accuracy* sebesar 89,53% dan *Test Loss* sebesar

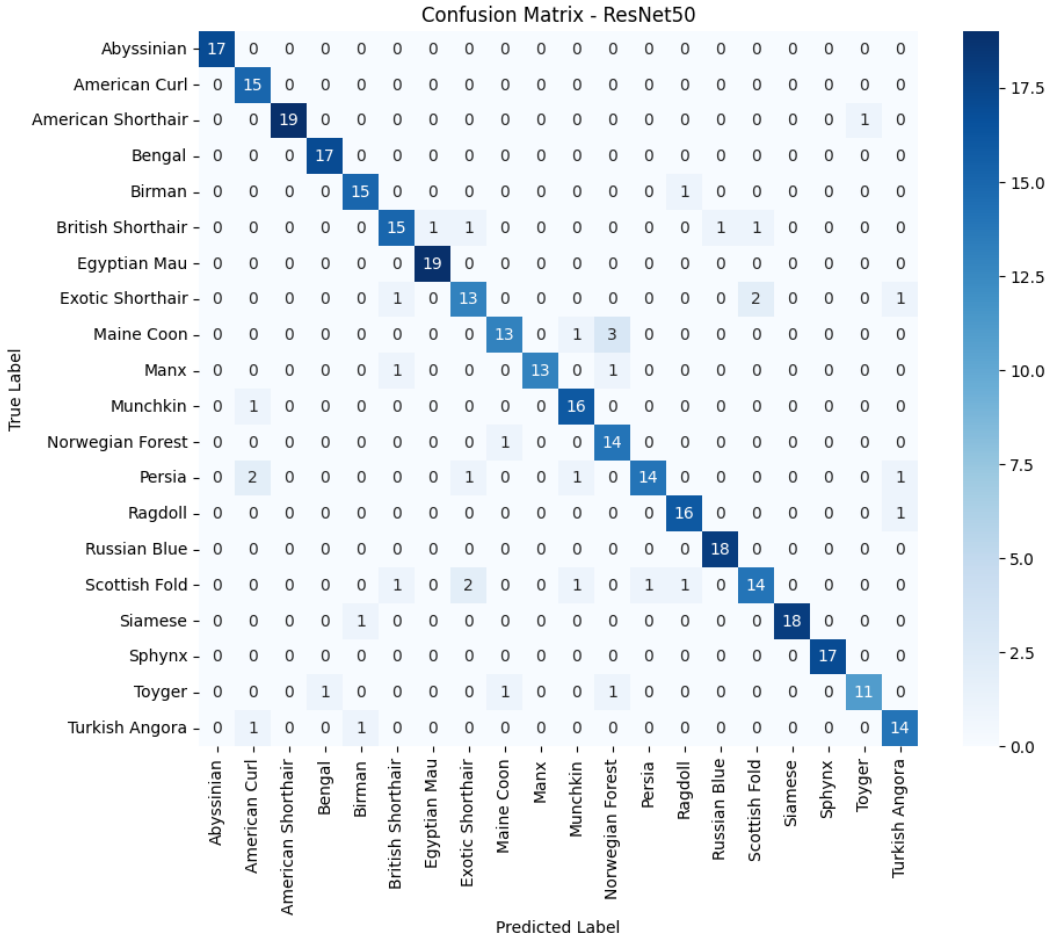
0.6586 yang ditunjukkan pada Gambar 4.22. Nilai tersebut diperoleh menggunakan fungsi model.evaluate() pada *TensorFlow*/Keras.

```
Found 344 images belonging to 20 classes.
11/11 ━━━━━━━━━━━━━━━━━━━━ 10s 352ms/step - accuracy: 0.8953 - loss: 0.6586

Test Loss     : 0.6586
Test Accuracy : 0.8953
11/11 ━━━━━━━━━━━━━━━━━━━━ 9s 503ms/step
```

Gambar 4.22 Hasil Test Menggunakan Data Testing Pada Model ResNet50

Berdasarkan *confusion matrix* pada Gambar 4.23, sebagian besar nilai berada pada diagonal utama yang menunjukkan bahwa mayoritas citra berhasil diklasifikasikan dengan benar. Pada kelas *Maine Coon*, hanya 13 dari 17 citra yang berhasil diklasifikasikan dengan benar, sementara beberapa citra lainnya diprediksi sebagai *Munchkin* dan *Norwegian Forest.*



Gambar 4.23 *Confusion Matrix* Model ResNet50

Sebagai contoh pada kelas *Abyssinian* dan *Sphynx* berhasil diprediksi dengan benar sehingga menghasilkan nilai *recall* sebesar 1,00 yang dapat dilihat pada Gambar 4.24. selain itu, kelas Persia juga menunjukkan nilai *recall* yang relatif rendah yaitu 0,74. Berdasarkan *confusion matrix*, beberapa citra Persia diprediksi sebagai *American Curl*, *Exotic Shorthair*, dan *Munchkin*. Hal ini menunjukkan bahwa model masih mengalami kesulitan dalam membedakan ras Persia dengan ras lain yang memiliki ciri wajah dan tekstur bulu yang serupa.

Classification Report:

	precision	recall	f1-score	support
Abyssinian	1.00	1.00	1.00	17
American Curl	0.79	1.00	0.88	15
American Shorthair	1.00	0.95	0.97	20
Bengal	0.94	1.00	0.97	17
Birman	0.88	0.94	0.91	16
British Shorthair	0.83	0.79	0.81	19
Egyptian Mau	0.95	1.00	0.97	19
Exotic Shorthair	0.76	0.76	0.76	17
Maine Coon	0.87	0.76	0.81	17
Manx	1.00	0.87	0.93	15
Munchkin	0.84	0.94	0.89	17
Norwegian Forest	0.74	0.93	0.82	15
Persia	0.93	0.74	0.82	19
Ragdoll	0.89	0.94	0.91	17
Russian Blue	0.95	1.00	0.97	18
Scottish Fold	0.82	0.70	0.76	20
Siamese	1.00	0.95	0.97	19
Sphynx	1.00	1.00	1.00	17
Toyger	0.92	0.79	0.85	14
Turkish Angora	0.82	0.88	0.85	16
accuracy			0.90	344
macro avg	0.90	0.90	0.89	344
weighted avg	0.90	0.90	0.89	344

Gambar 4.24 *Classification Report* Model ResNet50

Berikut salah satu nilai *classification report* dengan menggunakan contoh kelas Persia. Terdapat 14 citra yang berhasil diprediksi sebagai Persia (*True Positive*), 5 citra Persia yang salah diprediksi ke kelas lain (*False Negative*), dan 1 citra yang pada kelas lain diprediksi sebagai *Persia* (*False Positive*).

1. Hasil implementasi *Precision*, *Recall*, dan F1-*Score* pada Kelas *Persia*

a. $Precision = \frac{TP}{TP+FP} = \frac{14}{14+1}$

$= 0{,}93$

b. $Recall = \frac{TP}{TP+FN} = \frac{14}{14+5}$

$= 0{,}74$

c. $F1 - Score = 2\ x\ \frac{\text{Precision x Recall}}{\text{Precision + Recall}} = 2\ x\ \frac{0{,}93\ \text{x}\ 0{,}74}{0{,}93+0{,}74}$

$= 2\ x\ \frac{0{,}69}{1{,}67} = 0{,}82$

4.8 Evaluasi Kinerja Model AlexNet

Setelah proses *training* selesai, model AlexNet dengan konfigurasi hyperparameter terbaik dievaluasi menggunakan data *testing* untuk mengukur performa generelisasinya. Hasil pengujian menunjukkan *Test Accuracy* sebesar 49.41% dan *Test Loss* sebesar 1,7125 yang dapat dilihat pada Gambar 4.25.

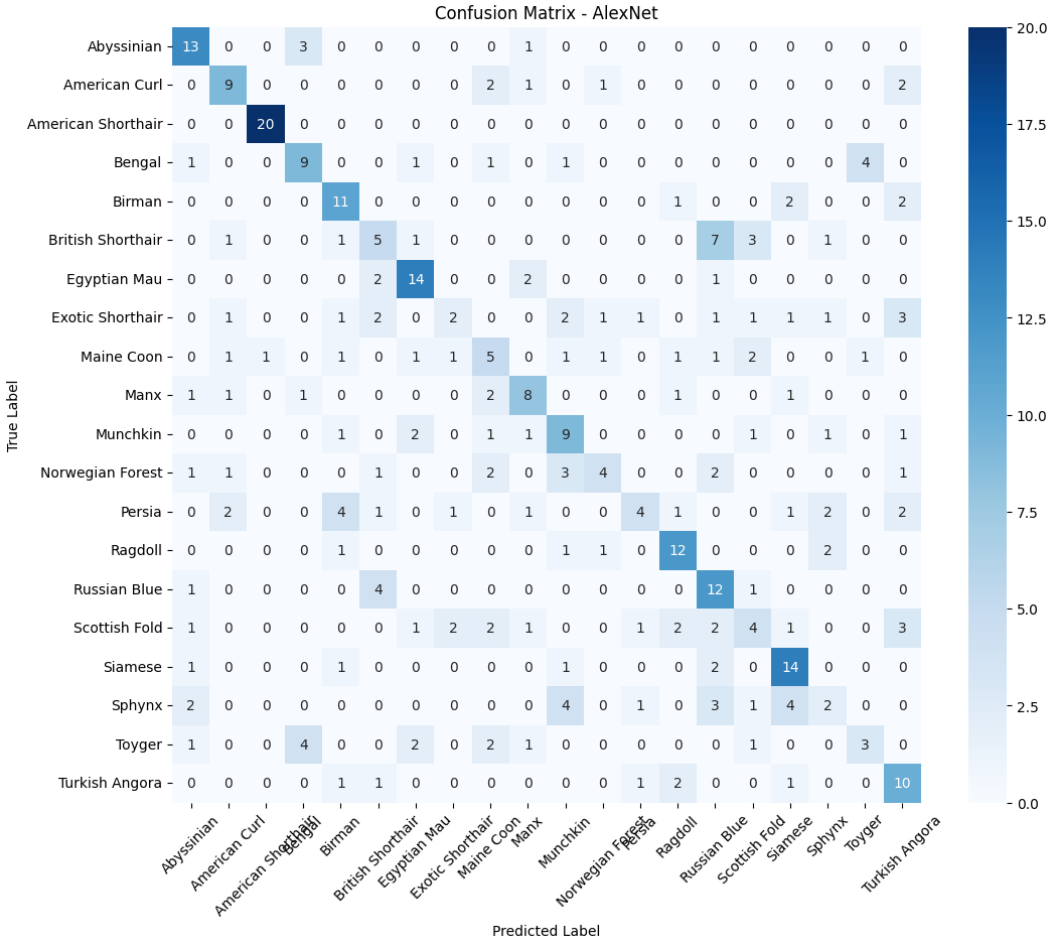
```
test_loss, test_acc = model.evaluate(test_generator)

print("Test Loss     :", test_loss)
print("Test Accuracy:", test_acc)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 4s 227ms/step - accuracy: 0.4942 - loss: 1.7125
Test Loss     : 1.712509274482727
Test Accuracy: 0.49418604373931885
```

Gambar 4.25 Hasil Test Model AlexNet Menggunakan Data *Testing*

Berdasarkan *confusion matrix* pada Gambar 4.26, dapat dilihat bahwa prediksi model AlexNet masih banyak berada di luar diagonal utama, yang menunjukkan tingginya kesalahan klasifikasi antar kelas. Beberapa kelas masih dapat dikenali dengan baik, seperti *American Shorthair* yang memperoleh *recall* sebesar 1,00 dengan seluruh citra berhasil diklasifikasikan dengan benar. Namun, sebagian besar kelas lainnya menunjukkan performa yang rendah. Sebagai contoh, kelas *Exotic Shorthair* dan *Sphynx* hanya memperoleh *recall* sebesar 0,12, sedangkan kelas Persia memperoleh *recall* sebesar 0,21 yang ditunjukkan pada Gambar 4.27.



Gambar 4.26 *Confusion Matrix* Model AlexNet

Hal ini menunjukkan bahwa model masih mengalami kesulitan dalam membedakan karakteristik visual antar ras kucing. Secara keseluruhan, hasil *confusion matrix* menunjukkan bahwa AlexNet belum mampu menghasilkan representasi fitur yang optimal untuk mengklasifikasikan 20 ras kucing. Kondisi ini menyebabkan tingkat kesalahan klasifikasi yang relatif tinggi dibandingkan model MobileNetV2 dan ResNet50.

Seperti contoh pada salah satu ras kucing yang digunakan dalam dataset yaitu *Turkish Angora*. *Turkish Angora* memiliki *True Positive* sebanyak 10 citra, *False Negative* sebanyak 6 citra, dan *False Positive* sebanyak 14 citra.

1. Implementasi *Precision*, *Recall*, dan F1-*Score* pada Kelas *Turkish Angora*

a. $Precision = \frac{TP}{TP+FP} = \frac{10}{10+14}$

$= 0{,}4166$

b. $Recall = \frac{TP}{TP+FN} = \frac{10}{10+6}$

= 0,625

c. $F1 - Score = 2\ x\ \frac{\text{Precision x Recall}}{\text{Precision + Recall}} = 2\ x\ \frac{0{,}4166\text{x } 0{,}625}{0{,}4166+0{,}625}$

$= 2\ x\ \frac{0.2602}{1.0416} = 0{,}4998$

	precision	recall	f1-score	support
Abyssinian	0.59	0.76	0.67	17
American Curl	0.56	0.60	0.58	15
American Shorthair	0.95	1.00	0.98	20
Bengal	0.53	0.53	0.53	17
Birman	0.50	0.69	0.58	16
British Shorthair	0.31	0.26	0.29	19
Egyptian Mau	0.64	0.74	0.68	19
Exotic Shorthair	0.33	0.12	0.17	17
Maine Coon	0.29	0.29	0.29	17
Manx	0.50	0.53	0.52	15
Munchkin	0.41	0.53	0.46	17
Norwegian Forest	0.50	0.27	0.35	15
Persia	0.50	0.21	0.30	19
Ragdoll	0.60	0.71	0.65	17
Russian Blue	0.39	0.67	0.49	18
Scottish Fold	0.29	0.20	0.24	20
Siamese	0.56	0.74	0.64	19
Sphynx	0.22	0.12	0.15	17
Toyger	0.38	0.21	0.27	14
Turkish Angora	0.42	0.62	0.50	16
accuracy			0.49	344
macro avg	0.47	0.49	0.47	344
weighted avg	0.48	0.49	0.47	344

Gambar 4.27 *Classification Report* Model AlexNet

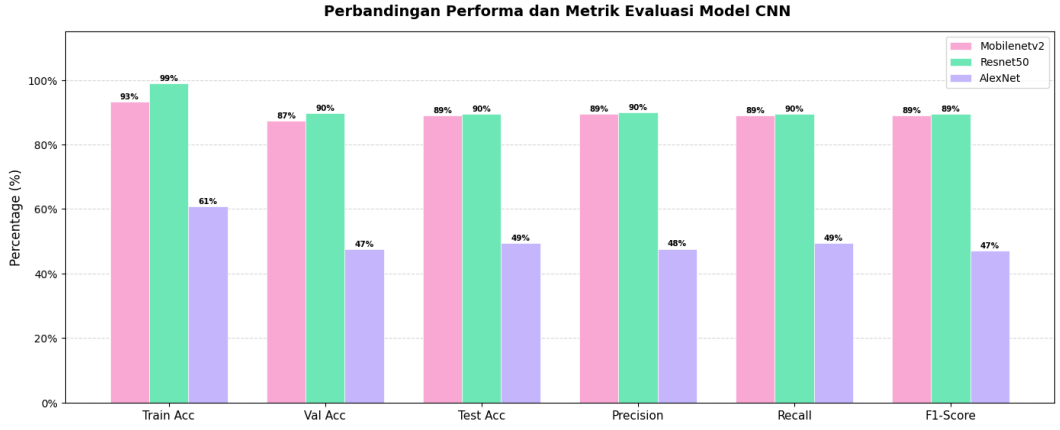
4.9 Pembahasan

Pada bagian ini dilakukan analisis perbandingan performa dari tiga arsitektur CNN yang digunakan dalam penelitian, yaitu MobileNetV2, ResNet50, dan AlexNet. Perbandingan dilakukan berdasarkan hasil performa dan evaluasi menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score* untuk mengidentifikasi model CNN yang menghasilkan kinerja terbaik pada klasifikasi

citra ras kucing. Hasil perbandingan ini selanjutnya digunakan sebagai dasar dalam menentukan model yang akan diimplementasikan pada aplikasi *mobile*.

4.9.1 Analisis Perbandingan Performa Arsitektur

Analisis ini dilakukan untuk membandingkan performa keseluruhan dari ketiga model CNN yang digunakan dalam penelitian, sehingga dapat diketahui arsitektur yang memberikan kinerja paling optimal dalam klasifikasi citra ras kucing. Rangkuman performa dan nilai metrik evaluasi dari setiap model dapat dilihat pada Gambar 4.28.



Model	Train Acc	Val Acc	Test Acc	Precision	Recall	F1-Score
Mobilenetv2	93%	87%	89%	89%	89%	89%
Resnet50	99%	90%	90%	90%	90%	89%
AlexNet	61%	47%	49%	48%	49%	47%

Gambar 4.28 Rangkuman Perbandingan Performa dan Metrik Evaluasi Model CNN

Berdasarkan grafik diatas, ResNet50 memperoleh performa terbaik dengan nilai *test accuracy* sebesar 90%**,** lebih tinggi dibandingkan MobileNetV2 yang mencapai 89% dan AlexNet yang hanya memperoleh 49%**.** Keunggulan ResNet50 juga terlihat pada metrik evaluasi lainnya, yaitu *precision* sebesar 90%**,** *recall* sebesar 90%**,** dan F1-*Score* sebesar 89%**.** Dengan demikian, ResNet50 dapat dinyatakan sebagai model yang memiliki tingkat klasifikasi paling baik di antara ketiga arsitektur CNN yang diuji. Hasil ini menunjukkan bahwa penggunaan arsitektur CNN yang didukung *transfer learning* memberikan performa yang lebih baik dibandingkan model CNN yang dibangun dari awal berdasarkan pada dataset yang digunakann pada penelitian ini. Performa tinggi pada ResNet50 dan MobileNetV2 dipengaruhi oleh penggunaan *pre-trained* dari dataset *ImageNet*

serta proses *feature extraction* dan *fine tuning* yang dilakukan melalui konfigurasi hyperparameter.

Sebaliknya AlexNet menunjukkan performa yang jauh lebih rendah dibandingkan kedua model lainnya. Hal ini diduga karena AlexNet dilatih tanpa memanfaatkan bobot *pre-trained* sehingga seluruh fitur harus dipelajari dari awal hanya berdasarkan pada dataset penelitian. Dengan jumlah data pelatihan sebanyak 3.996 citra yang terbagi ke dalam 20 kelas ras kucing, proses akan menjadi lebih sulit dibandingkan model yang telah memiliki pengetahuan awal dari *ImageNet*.

Selain itu, karakteristik dataset juga mempengaruhi hasil klasifikasi. Dataset yang digunakan merupakan dataset asli yang memiliki distribusi jumlah citra yang tidak sepenuhnya seimbang pada setiap kelas. Ketidakseimbangan tersebut dapat menyebabkan model lebih mudah mempelajari kelas tertentu dibandingkan kelas lainnya, terutama pada model dengan kemampuan ekstraksi fitur yang terbatas seperti AlexNet. Kondisi ini terlihat dari *confusion matrix* AlexNet yang menunjukkan kesalahan klasifikasi lebih banyak dibandingkan MobileNetV2 dan ResNet50.

Secara keseluruhan, hasil penelitian menunjukkan bahwa model CNN seperti MobileNetV2 dan ResNet50 yang berbasis pada *transfer learning*, lebih efektif dalam mengklasifikasikan citra ras kucing pada dataset yang digunakan pada penelitian ini. Dari ketiga model yang diuji, ResNet50 dipilih sebagai model terbaik karena memperoleh nilai *accuracy*, *precision*, *recall*, dan *F1-score* tertinggi, sehingga model tersebut ditetapkan sebagai model arsitektur yang akan diimplementasikan pada aplikasi *mobile* berbasis *android*.

4.9.2 Analisis Kesalahan (*Error Analyze*)

Berdasarkan *confusion matrix* ketiga model, sebagian besar kesalahan klasifikasi terjadi pada ras kucing yang memiliki karakteristik visual yang mirip. Salah satu kesalahan yang konsisten ditemukan pada MobileNetV2 dan ResNet50 adalah antara ras *Maine Coon* dan *Norwegian Forest*. Kedua ras tersebut memiliki kemiripan pada bentuk tubuh, ukuran badan, serta tekstur bulu yang panjang dan lebat sehingga menyulitkan model dalam membedakan keduanya secara akurat.

Kesalahan klasifikasi juga ditemukan pada ras Persia yang beberapa kali diprediksi sebagai *Turkish Angora*, *Birman*, maupun *Exotic Shorthair*. Hal ini diduga karena ras - ras tersebut memiliki karakteristik visual yang serupa, seperti bulu yang panjang, bentuk wajah yang relatif membulat, serta variasi warna bulu yang hampir sama pada beberapa citra. Selain itu, sudut pengambilan gambar yang tidak selalu menampilkan ciri khas wajah pesek pada ras Persia dapat menyebabkan model lebih fokus pada fitur lain yang juga dimiliki oleh ras lain.

Pada ras *Scottish Fold*, kesalahan klasifikasi umumnya terjadi dengan *British Shorthair* dan *Exotic Shorthair*. Kondisi ini dapat disebabkan oleh kemiripan bentuk wajah yang cenderung bulat dan mata yang besar. Ciri khas utama *Scottish Fold* berupa telinga yang melipat tidak selalu terlihat jelas pada seluruh citra sehingga model mengalami kesulitan dalam mengenali perbedaan antar ras tersebut.

Selain faktor arsitektur model, kesalahan klasifikasi juga dapat dipengaruhi oleh kemiripan antar ras, variasi pose kucing, pencahayaan, sudut pengambilan gambar, serta distribusi jumlah citra yang tidak sepenuhnya seimbang pada dataset asli. Faktor-faktor tersebut menyebabkan beberapa ciri khas ras tidak selalu terlihat secara jelas sehingga meningkatkan kemungkinan terjadinya salah klasifikasi.

4.10 Hasil Implementasi Aplikasi

Berdasarkan hasil evaluasi yang telah dilakukan, model ResNet50 dipilih sebagai model terbaik untuk diimplementasikan pada aplikasi *mobile*. Sebelum digunakan pada sistem, model hasil pelatihan yang tersimpan dalam format .keras terlebih dahulu dikonversi ke format *TensorFlow SavedModel*. Format ini dipilih karena menyimpan keseluruhan komponen model, meliputi arsitektur jaringan, bobot hasil pelatihan, serta fungsi inferensi yang diperlukan saat proses prediksi. Proses konversi dilakukan dengan memuat model hasil pelatihan menggunakan fungsi `load_model()` kemudian mengekspornya ke format *TensorFlow SavedModel* menggunakan fungsi `export()`. *Source code* yang digunakan untuk proses konversi ditunjukkan pada Tabel 4.17.

Tabel 4.17 *Source Code* Proses Konversi Model

```
from tensorflow.keras.models import load_model

model =
load_model("/content/drive/MyDrive/cnn/resnet50/ft_resnet_p1.ke
ras", compile=False)

model.export("/content/drive/MyDrive/cnn/resnet50/saved_model")
```

Model dalam format *TensorFlow SavedModel* selanjutnya digunakan pada API yang berfungsi sebagai penghubung antara aplikasi *android* dan model CNN yang berjalan pada *server*.

4.10.1 Implementasi API

Model ResNet50 diimplementasikan ke dalam API berbasis *Flask* yang berfungsi sebagai penghubung antara aplikasi *android* dan model CNN. API menerima gambar kucing yang dikirimkan dari aplikasi *android*, kemudian melakukan proses klasifikasi menggunakan model ResNet50 dan mengembalikan hasil prediksi ke aplikasi *android*. Pada implementasinya, API *Flask* dibangun menggunakan *library* yang ditunjukkan pada Tabel 4.18.

Tabel 4.18 *Library* yang digunakan pada Pembuatan API

```
from flask import Flask, request, jsonify
import tensorflow as tf
import numpy as np
from PIL import Image
import gdown
import zipfile
import os

from tensorflow.keras.applications.resnet50 import
preprocess_input
```

Selanjutnya, API menyimpan daftar 20 kelas ras kucing yang digunakan oleh model selama proses klasifikasi. Daftar kelas tersebut ditunjukkan pada Tabel 4.19. Daftar kelas tersebut digunakan untuk mengkonversi hasil prediksi model yang masih berupa indeks kelas menjadi nama ras kucing. Selain itu, API juga menyimpan informasi tambahan untuk setiap ras kucing.

Tabel 4.19 *Class Label* 20 Ras Kucing pada API

```
class_names = [
    "Abyssinian",
```

```
    "American_Curl",
    "American_Shorthair",
    "Bengal",
    "Birman",
    "British_Shorthair",
    "Egyptian_Mau",
    "Exotic_Shorthair",
    "Maine_Coon",
    "Manx",
    "Munchkin",
    "Norwegian_Forest",
    "Persia",
    "Ragdoll",
    "Russian_Blue",
    "Scottish_Fold",
    "Siamese",
    "Sphynx",
    "Toyger",
    "Turkish_Angora"
]
```

Proses utama pada API dilakukan melalui *endpoint /predict*. *Endpoint* ini menerima gambar yang dikirimkan oleh pengguna, kemudian melakukan beberapa tahapan pengolahan sebelum proses klasifikasi dilakukan. Tahapan tersebut dapat dilihat pada *source code endpoint* yang ditunjukkan pada Tabel 4.20.

Tabel 4.20 Implementasi *Endpoint /Preddict*

```
@app.route("/predict", methods=["POST"])
def predict():
    try:
        # Ambil file gambar
        file = request.files["file"]

        # Open image
        img = Image.open(file).convert("RGB")

        # Resize
        img = img.resize((224, 224))

        # Convert ke numpy
        img_array = np.array(img)

        # Preprocessing ResNet50
        img_array = preprocess_input(img_array)
```

```
        # Tambah batch dimension
        img_array = np.expand_dims(img_array, axis=0)

        # Prediksi
        prediction = infer(tf.constant(img_array))

        # Ambil output tensor
        output = list(prediction.values())[0].numpy()

        # Ambil index kelas tertinggi
        predicted_class = np.argmax(output)

        # Nama ras
        nama_ras = class_names[predicted_class]

        # Confidence
        confidence = float(np.max(output) * 100)
```

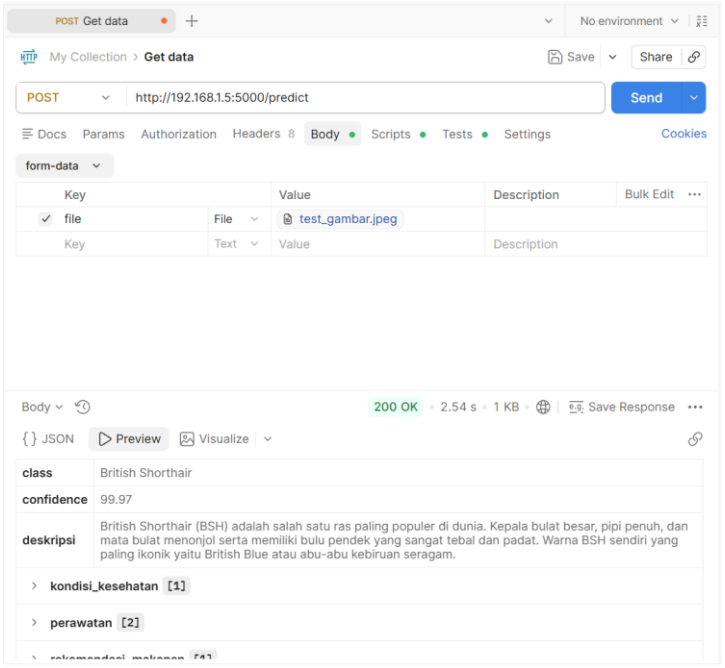
Model menghasilkan nilai probabilitas untuk setiap kelas ras kucing. Kelas dengan nilai probabilitas tertinggi dipilih sebagai hasil prediksi, sedangkan nilai probabilitas tertinggi digunakan sebagai *confidence score* yang menunjukkan tingkat keyakinan model terhadap hasil klasifikasi. API menerapkan *confidence threshold* sebesar 70% yang ditunjukkan pada *source code* Table 4.21. Apabila *confidence score* di bawah nilai tersebut, sistem tidak langsung menampilkan nama ras tertentu, melainkan memberikan hasil "Ras Tidak Dikenali".

Tabel 4.21 Implementasi *Confidence Score*

```
# CEK CONFIDENCE
        if confidence < 70:
            result = {
                "class": "Ras Tidak Dikenali",
                "confidence": round(confidence, 2),
                "deskripsi": "Ras kucing tidak terdeteksi. Pastikan
foto kucing cukup jelas dan tidak buram ya!",
                "perawatan": [],
                "rekomendasi_makanan": [],
                "kondisi_kesehatan": []
            }
            return jsonify(result)
```

Penerapan *confidence threshold* bertujuan untuk mengurangi kemungkinan munculnya hasil klasifikasi yang kurang akurat. Hasil implementasi API *Flask* ditunjukkan pada Gambar 4.29. Pada implementasi ini digunakan aplikasi

Postman untuk memastikan *endpoint/predict* dapat menerima gambar dan mengembalikan hasil klasifikasi yang dihasilkan oleh model ResNet50.



Gambar 4.29 Hasil Implementasi Flask Menggunakan *Postman*

Berdasarkan gambar diatas, API berhasil menerima file gambar yang dikirimkan melalui metode POST pada *endpoint* /predict. Pada contoh implementasi tersebut, gambar yang dikirimkan berhasil diklasifikasikan sebagai ras *British Shorthair* dengan nilai *confidence score* sebesar 99,97%**.** Selain menampilkan hasil klasifikasi, sistem juga berhasil menampilkan informasi lengkap mengenai ras *British Shorthair* sehingga pengguna tidak hanya memperoleh hasil identifikasi ras kucing, tetapi juga informasi pendukung mengenai karakteristik dan perawatannya. Hasil implementasi ini menunjukkan bahwa model ResNet50 telah berhasil diintegrasikan ke dalam API *Flask* dan mampu digunakan untuk melakukan klasifikasi citra ras kucing sesuai dengan rancangan sistem yang telah dibuat.

4.10.2 Implementasi Antarmuka Aplikasi *Android*

Aplikasi Klasifikasi Ras Kucing yang diberi nama MeonggScan dikembangkan menggunakan *Android Studio* versi Panda 2 tahun 2025.3.2 dengan bahasa pemrograman Kotlin sebagai bahasa utama.

Implementasi antarmuka dilakukan menggunakan beberapa komponen *android* seperti *ConstraintLayout***,** *ScrollView***,** *CardView***,** *RecyclerView***,**

ImageView, *TextView*, dan *Button* yang ditunjukkan pada Tabel 4.22 untuk mendukung tampilan yang responsif dan mudah digunakan.

Tabel 4.22 Implementasi Komponen *Android Studio*

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:overScrollMode="never">

</ScrollView
```

Aplikasi yang dikembangkan terdiri dari lima halaman utama, yaitu *splash screen*, *home*, *analyze*, hasil klasifikasi, info aplikasi, dan riwayat klasifikasi. Setiap halaman memiliki fungsi yang berbeda sesuai dengan kebutuhan pengguna dalam melakukan proses identifikasi ras kucing menggunakan model CNN ResNet50.

4.10.2.1 Halaman *Splash Screen*

Halaman *splash screen* merupakan halaman pertama yang ditampilkan ketika aplikasi dijalankan. Berdasarkan Gambar 4.30, halaman *splash screen* menampilkan logo aplikasi dan nama aplikasi MeonggScan dengan tujuan untuk memberikan identitas aplikasi kepada pengguna. Setelah beberapa detik, pengguna secara otomatis diarahkan menuju halaman utama tanpa memerlukan interaksi tambahan.

Dari sisi desain antarmuka, halaman *splash screen* menggunakan kombinasi warna krem dan cokelat yang dipilih untuk memberikan kesan hangat, ramah, dan identik dengan tema hewan peliharaan. Elemen jejak kaki kucing yang ditempatkan pada beberapa sudut layar digunakan sebagai elemen dekoratif sekaligus memperkuat identitas visual aplikasi. Tata letak yang sederhana dengan fokus pada logo dan nama aplikasi bertujuan agar pengguna dapat langsung mengenali fungsi aplikasi tanpa terganggu oleh elemen lain yang tidak diperlukan.

Gambar 4.30 Halaman *Splash Screen*

4.10.2.2 Halaman *Home*

Halaman *Home* merupakan halaman utama yang ditampilkan setelah pengguna melewati *splash screen*. Halaman ini berfungsi sebagai pusat aplikasi dan berfungsi untuk memberikan informasi awal mengenai fungsi aplikasi kepada pengguna. Berdasarkan Gambar 4.31, pada bagian atas halaman ditampilkan sapaan “Halo, Pawrents!” yang bertujuan memberikan kesan ramah dan interaktif kepada pengguna.

Halaman *Home* juga menampilkan daftar kucing populer dalam bentuk *card* yang dapat digeser secara horizontal dengan menerapkan Komponen *ScrollView* pada *Android Studio*. Pada bagian bawah terdapat *Bottom Navigation*

bar yang terdiri dari menu *Home*, *Analyze*, dan *History*. Menu *Home* digunakan untuk menampilkan halaman utama aplikasi, menu *Analyze* digunakan untuk memulai proses klasifikasi ras kucing, sedangkan menu *history* digunakan untuk melihat riwayat hasil klasifikasi yang telah dilakukan sebelumnya. Penggunaan *Bottom Navigation Bar* memudahkan pengguna dalam berpindah antar fitur utama aplikasi dengan cepat dan intuitif.

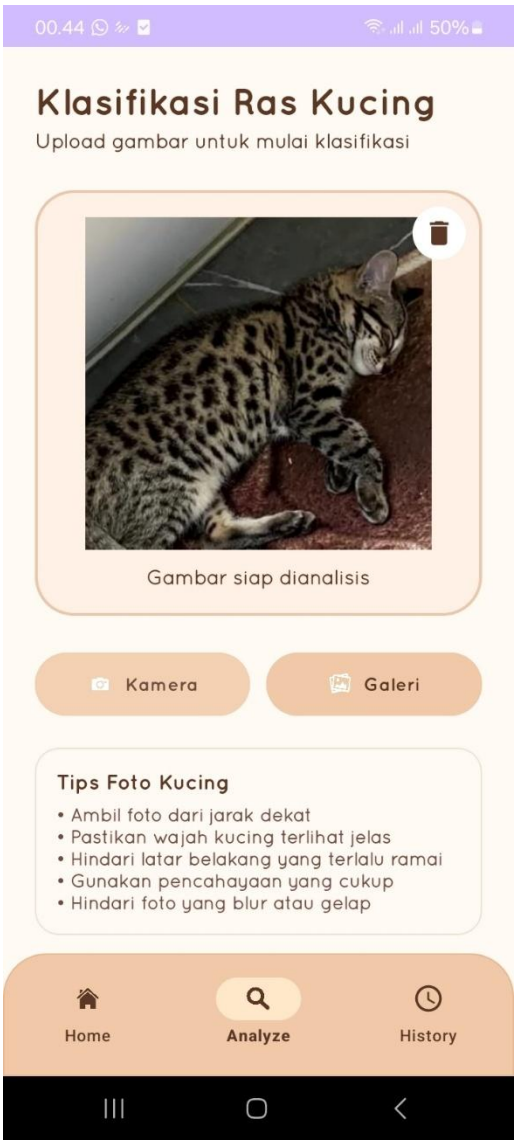
Gambar 4.31 Halaman *Home*

4.10.2.3 Halaman *Analyze*

Halaman *Analyze* merupakan halaman yang digunakan pengguna untuk melakukan proses identifikasi ras kucing. Pada halaman ini, pengguna dapat memilih gambar kucing yang akan dianalisis melalui kamera maupun galeri

perangkat. Berdasarkan Gambar 4.32, pada bagian tengah halaman terdapat area *image placeholder* yang digunakan untuk menampilkan gambar kucing yang dipilih oleh pengguna.

Setelah gambar berhasil dipilih, pengguna dapat menekan tombol mulai klasifikasi untuk mengirimkan gambar ke *server* melalui API *Flask*. Selanjutnya sistem akan melakukan proses *preprocessing* sesuai dengan kebutuhan model ResNet50, kemudian menjalankan proses klasifikasi untuk menentukan ras kucing yang terdeteksi. Hasil klasifikasi yang diperoleh dari model CNN kemudian dikembalikan oleh *server* dan ditampilkan pada halaman hasil prediksi.

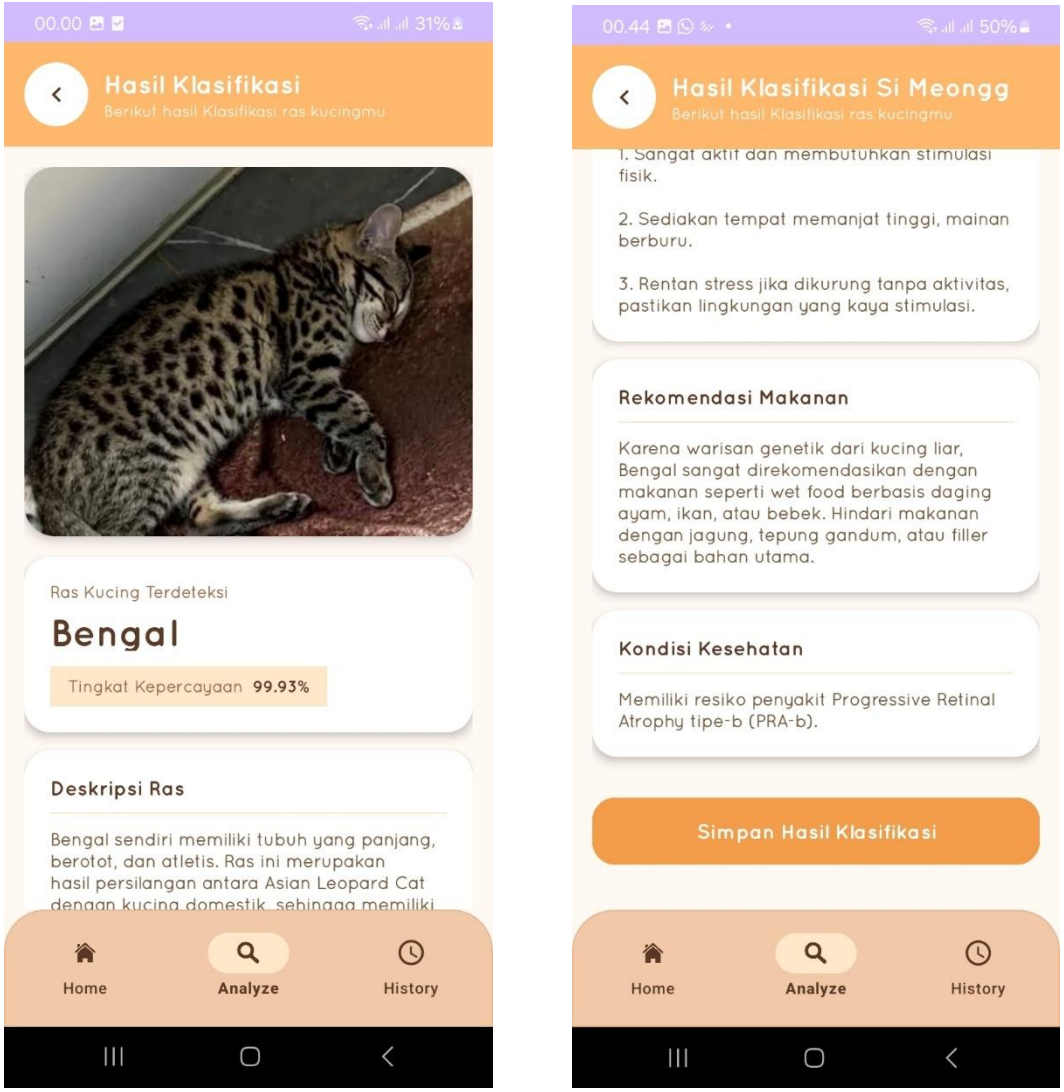


Gambar 4.32 Halaman *Analyze*

4.10.2.4 Halaman Hasil Klasifikasi

Halaman Hasil Klasifikasi merupakan halaman yang digunakan untuk menampilkan hasil identifikasi ras kucing berdasarkan citra yang telah dipilih pengguna pada halaman *Analyze*. Halaman ini akan ditampilkan setelah sistem berhasil menerima hasil prediksi dari model ResNet50 melalui API *Flask*. Berdasarkan Gambar 4.33, sistem menampilkan kembali citra kucing yang telah dipilih oleh pengguna sebagai referensi visual terhadap hasil klasifikasi yang diperoleh. Selain itu, ditampilkan nama ras kucing yang berhasil diidentifikasi beserta nilai tingkat kepercayaan (*confidence score*) yang dihasilkan oleh model ResNet50. Secara teknis, ketika pengguna menekan tombol mulai lasifikasi pada halaman *Analyze*, citra akan dikirimkan ke *server* melalui API *Flask* menggunakan metode HTTP POST.

Selanjutnya server melakukan proses *preprocessing* sesuai dengan kebutuhan model ResNet50, kemudian menjalankan proses inferensi untuk menghasilkan prediksi ras kucing. Setelah proses klasifikasi selesai, API mengirimkan hasil prediksi kembali ke aplikasi *android* dalam format JSON yang berisi nama ras, nilai *confidence score*, serta informasi tambahan mengenai ras yang terdeteksi. Informasi tersebut dikirimkan bersamaan dengan hasil prediksi sehingga pengguna tidak hanya mengetahui nama ras kucing yang terdeteksi, tetapi juga memperoleh penjelasan singkat mengenai karakteristik ras tersebut. Implementasi halaman hasil klasifikasi menunjukkan bahwa integrasi antara aplikasi *Android*, API Flask, dan model ResNet50 telah berjalan dengan baik. Sistem mampu menerima citra dari pengguna, melakukan klasifikasi ras kucing, dan menampilkan hasil prediksi beserta informasi pendukung secara otomatis sesuai dengan rancangan yang telah dibuat. Dan pada halaman ini terdapat *button* simpan hasil klasifikasi untuk mempermudah pengguna mengakses kembali hasil klasifikasi yang telah dilakukan.



Gambar 4.33 Halaman Hasil Klasifikasi

4.10.2.5 Halaman *History*

Halaman *history* klasifikasi digunakan untuk menampilkan hasil klasifikasi ras kucing yang telah dilakukan oleh pengguna. Berdasarkan Gambar 4.34, halaman riwayat dilengkapi dengan fitur *search bar* untuk mencari data riwayat serta tombol hapus *history* yang digunakan untuk menghapus data yang tersimpan. Secara teknis, setiap hasil klasifikasi yang diperoleh dari model ResNet50 akan disimpan ke dalam *database* lokal aplikasi ketika pengguna mengklik *button* simpan hasil klasifikasi pada halaman hasil klasifikasi sebelumnya. Ketika halaman *history* dibuka, sistem akan mengambil data yang tersimpan pada *database* dan menampilkannya dalam bentuk daftar (*list view*).

Informasi yang disimpan meliputi nama ras kucing, nilai *confidence score*, gambar hasil klasifikasi, serta waktu identifikasi dilakukan.

Dengan adanya mekanisme penyimpanan ini, pengguna dapat melihat kembali hasil klasifikasi yang pernah dilakukan meskipun aplikasi telah ditutup. Selain itu, pengguna dapat memilih salah satu data pada daftar riwayat untuk melihat informasi lebih lanjut. Sistem kemudian akan menampilkan detail ras kucing yang sama seperti pada halaman hasil klasifikasi, sehingga pengguna tetap dapat mengakses informasi deskripsi, perawatan, rekomendasi makanan, serta kondisi kesehatan tanpa perlu melakukan proses identifikasi ulang.

Gambar 4.34 Halaman *History*

4.10.2.6 Halaman Tentang Aplikasi

Halaman Tentang Aplikasi digunakan untuk memberikan informasi umum mengenai aplikasi MeonggScan. Berdasarkan Gambar 4.35, halaman ini menampilkan logo aplikasi, nama aplikasi, serta deskripsi singkat mengenai fungsi aplikasi sebagai sistem pengenalan ras kucing berbasis *Deep Learning* menggunakan model CNN. Selain itu, halaman ini juga menampilkan informasi mengenai dataset dan model yang digunakan dalam penelitian, yaitu dataset yang terdiri dari 20 ras kucing dan model ResNet50 sebagai model terbaik hasil evaluasi. Informasi tersebut disediakan untuk membantu pengguna memahami teknologi yang digunakan dalam proses klasifikasi ras kucing pada aplikasi MeonggScan.

Gambar 4.35 Halaman Tentang Aplikasi

4.11 Hasil Pengujian Aplikasi pada *Black Box Testing*

Aplikasi MeonggScan melakukan pengujian menggunakan *Black Box Testing*. Pengujian ini bertujuan untuk memastikan bahwa seluruh fitur yang terdapat pada aplikasi telah berjalan sesuai dengan kebutuhan fungsional yang telah dirancang. Pengujian ini dilakukan dengan menjalankan serangkaian skenario yang telah dirancang untuk mencakup semua fitur utama aplikasi. Hasil dari setiap skenario pengujjian dapat dilihat pada Tabel 4.23.

Tabel 4.23 Skenario dan Hasil Pengujian *Black Box Testing*

No	Fitur yang Diuji	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Halaman *Home*	Menampilkan halaman utama aplikasi	Membuka aplikasi	Menampilkan halaman *home* beserta seluruh elemen yang tersedia	Sesuai
2	Pemilihan Gambar dari Galeri	Memilih gambar melalui galeri	Klik tombol Galeri dan pilih gambar dari galeri	Menampilkan gambar yang dipilih pada area *preview*	Sesuai
3	Pengambilan Gambar melalui Kamera	Mengambil gambar menggunakan kamera	Klik tombol Kamera dan ambil gambar	Menampilkan gambar hasil pengambilan kamera pada area *preview*	Sesuai
4	Klasifikasi Citra	Melakukan klasifikasi terhadap citra kucing	Klik tombol Mulai Klasifikasi	Menampilkan hasil klasifikasi ras kucing beserta nilai *confidence score*,	Sesuai

No	Fitur yang Diuji	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Pengujian
				deskripsi, informasi perawatan, rekomendasi makanan, dan kondisi kesehatan yang perlu diperhatikan	
5	Riwayat Klasifikasi	Menampilkan data riwayat klasifikasi	Membuka halaman riwayat	Menampilkan seluruh data hasil klasifikasi yang telah dilakukan oleh pengguna	Sesuai
6	Validasi Input Gambar	Melakukan klasifikasi tanpa memilih gambar	Klik tombol Mulai Klasifikasi tanpa memilih gambar	Menampilkan pesan peringatan untuk memilih gambar terlebih dahulu	Sesuai
7	Pencarian Riwayat Klasifikasi	Mencari riwayat berdasarkan nama ras kucing	Memasukkan nama ras kucing pada kolom pencarian	Menampilkan data riwayat yang sesuai dengan nama ras kucing	Sesuai

No	Fitur yang Diuji	Skenario Pengujian	Langkah Pengujian	Hasil yang Diharapkan	Hasil Pengujian
				yang dicari	
8	Hapus Riwayat Klasifikasi	Menghapus salah satu data riwayat klasifikasi	Klik tombol Hapus pada salah satu data riwayat	Data riwayat yang dipilih berhasil dihapus	Sesuai
9	Hapus Semua Riwayat	Menghapus seluruh data riwayat klasifikasi	Klik tombol hapus semua pada halaman riwayat	Seluruh data riwayat klasifikasi berhasil dihapus	Sesuai
10	Navigasi Menu	Berpindah antar halaman menggunakan navigation bar	Klik menu *Home*, analyze, dan *history* pada *navigation bar*	Sistem menampilkan halaman yang sesuai dengan menu yang dipilih	Sesuai

4.12 Hasil Pengujian Klasifikasi pada Aplikasi

Pengujian dilakukan menggunakan beberapa sampel ras kucing yang tersedia dalam dataset penelitian. Setiap ras kucing diuji menggunakan 5 citra melalui fitur kamera dan 5 citra melalui fitur galeri pada aplikasi MeonggScan. Hasil klasifikasi yang diperoleh kemudian dicatat secara manual berdasarkan keluaran prediksi aplikasi dan disajikan pada Tabel 4.24.

Tabel 4.24 Hasil Pengujian Klasifikasi Pada Aplikasi

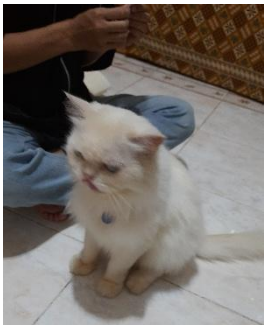
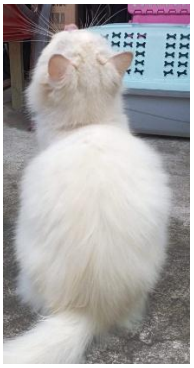
No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
Citra yang di ambil melalui galeri					
1		*British Shorthair*	*British Shorthair*	86.02%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
2		*British Shorthair*	*British Shorthair*	80.77%	Benar
3		*British Shorthair*	*British Shorthair*	99.09%	Benar
4		*British Shorthair*	*British Shorthair*	88.98%	Benar
5		*British Shorthair*	*British Shorthair*	98.26%	Benar
	Citra yang di ambil melalui kamera				
6		*British Shorthair*	*British Shorthair*	96.21%	Benar
7		*British Shorthair*	*British Shorthair*	99.79%	Benar

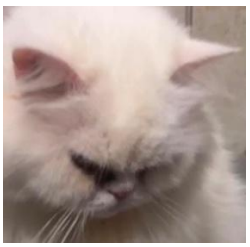
No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
8		*British Shorthair*	*British Shorthair*	98.53%	Benar
9		*British Shorthair*	*British Shorthair*	92.4%	Benar
10		*British Shorthair*	*British Shorthair*	91.42%	Benar
	Citra yang di ambil melalui galeri				
11		Persia	Persia	99.04%	Benar
12		Persia	Persia	99.71%	Benar

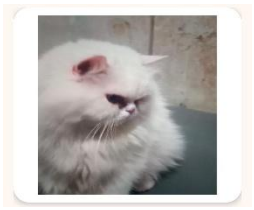
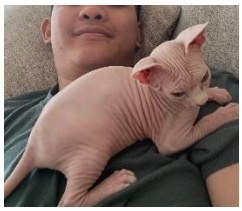
No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
13		Persia	Persia	96.34%	Benar
14		Persia	Persia	90.84%	Benar
15		Persia	Persia	97.4%	Benar
	Citra yang di ambil melalui kamera				
16		Persia	Persia	98.67%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
17		Persia	Ras Tidak Dikenali	31.06%	Salah
18		Persia	Persia	89.6%	Benar
19		Persia	Persia	97.16%	Benar
20		Persia	Ras Tidak Dikenali	47.47%	Salah
	Citra yang di ambil melalui galeri				
21		*Turkish Angora*	*Turkish Angora*	79.69%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
22		*Turkish Angora*	*Turkish Angora*	78.51%	Benar
23		*Turkish Angora*	*Turkish Angora*	90.86%	Benar
24		*Turkish Angora*	*Turkish Angora*	84.61%	Benar
25		*Turkish Angora*	*Turkish Angora*	73.96%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
	Citra yang di ambil melalui kamera				
26		*Turkish Angora*	*Turkish Angora*	97.91%	Benar
27		*Turkish Angora*	*Turkish Angora*	79.8%	Benar
28		*Turkish Angora*	*Turkish Angora*	73.81%	Benar
29		*Turkish Angora*	*Turkish Angora*	71.73%	Benar
30		*Turkish Angora*	*Turkish Angora*	91.35%	Benar
	Citra yang di ambil melalui galeri				
31		*Exotic Shorthair*	*Exotic Shorthair*	74.55%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
32		*Exotic Shorthair*	*Exotic Shorthair*	83.05%	Benar
33		*Exotic Shorthair*	*Exotic Shorthair*	85.58%	Benar
34		*Exotic Shorthair*	*Exotic Shorthair*	77.53%	Benar
35		*Exotic Shorthair*	*Exotic Shorthair*	75.79%	Benar
	Citra yang di ambil melalui kamera				
36		*Exotic Shorthair*	Persia	83.94%	Salah

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
37		*Exotic Shorthair*	*Exotic Shorthair*	79.98%	Benar
38		*Exotic Shorthair*	Persia	74.84%	Salah
39		*Exotic Shorthair*	*Exotic Shorthair*	98.37%	Benar
40		*Exotic Shorthair*	*Exotic Shorthair*	89.26%	Benar
	Citra yang diambil melalui galeri				
41		*Sphynx*	*Sphynx*	99.97%	Benar
42		*Sphynx*	*Sphynx*	100%	Benar

No	Citra yang Diuji	Kelas Sebenarnya	Hasil Prediksi	*Confidence Score*	Keterangan
43		*Sphynx*	*Sphynx*	99.99%	Benar
44		*Sphynx*	*Sphynx*	100%	Benar
45		*Sphynx*	*Sphynx*	100%	Benar

Berdasarkan hasil pengujian yang ditunjukkan diatas, aplikasi MeonggScan berhasil mengklasifikasikan 41 dari 45 gambar uji dengan benar, dan 4 citra lainnya mengalami kesalahan klasifikasi. Tingkat akurasi aplikasi dihitung berdasarkan perbandingan antara jumlah prediksi yang benar dengan jumlah seluruh data uji, yaitu:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{41}{45}$$

= 0.9111

Hasil tersebut menunjukkan bahwa aplikasi MeonggScan memiliki performa yang baik dalam mengidentifikasi ras kucing berdasarkan gambar yang diberikan. Adapun kesalahan klasifikasi yang terjadi pada beberapa gambar kemungkinan disebabkan oleh kemiripan karakteristik visual antar ras kucing, kurangnya pencahayaan, dan sudut pengambilan gambar yang berbeda.