Pengembangan Model Klasifikasi Gambar Kucing dan Anjing Menggunakan Convolutional Neural Network (CNN) dengan Augmentasi Gambar

Daniel Jayasutra
Teknik Informatika, Fakultas Ilmu Komputer,
Universitas Mercubuana,
Jakarta, Indonesia
danieljayasutra@gmail.com

***Abstract*— Peningkatan kemampuan komputer dalam memahami dan mengklasifikasikan gambar telah menginspirasi pengembangan berbagai aplikasi yang relevan dalam berbagai bidang, termasuk di antaranya pengenalan hewan peliharaan seperti kucing dan anjing. Dalam penelitian ini, Saya melakukan sebuah model klasifikasi gambar yang menggunakan Convolutional Neural Network (CNN) untuk membedakan antara gambar kucing dan anjing. Saya menggunakan dataset gambar kucing dan anjing yang telah disiapkan sebelumnya dan menerapkan augmentasi gambar untuk meningkatkan keragaman data. Model CNN yang dibangun terdiri dari beberapa lapisan konvolusi dan lapisan-lapisan terkait untuk mengekstrak fitur dari gambar-gambar tersebut. Saya melatih model menggunakan metode generator data dan mengamati perkembangannya selama proses pelatihan. Hasil eksperimen menunjukkan bahwa model yang diusulkan mampu membedakan antara gambar kucing dan anjing dengan akurasi yang signifikan. Selain itu, saya juga melakukan prediksi pada gambar-gambar baru untuk menguji kinerja model secara praktis. Hasil prediksi menunjukkan bahwa model cenderung berhasil dalam mengklasifikasikan gambar-gambar tersebut dengan tingkat keyakinan yang memadai.**

Kata Kunci: Convolutional Neural Network, Klasifikasi Gambar, Kucing, Anjing, Augmentasi Gambar

I. PENDAHULUAN

Salah satu tantangan yang telah lama dihadapi dalam bidang visi komputer adalah klasifikasi objek dalam citra secara umum. Kemampuan untuk menduplikasi kemampuan manusia dalam memahami informasi citra merupakan tujuan utama, sehingga komputer dapat mengenali objek dalam citra dengan cara yang serupa dengan manusia. Proses feature engineering yang umumnya digunakan dalam visi komputer memiliki keterbatasan, di mana teknik-teknik ini hanya efektif pada dataset tertentu dan tidak memiliki kemampuan untuk menggeneralisasi pada berbagai jenis citra. Perbedaan antara citra, seperti perbedaan sudut pandang, skala, kondisi pencahayaan, deformasi objek, dan lainnya, menjadi faktor utama yang menyulitkan proses klasifikasi objek secara umum.

Para peneliti dan akademisi telah lama berusaha menemukan solusi untuk tantangan ini. Salah satu pendekatan yang berhasil digunakan adalah penggunaan Jaringan Syaraf Tiruan (JST) yang terinspirasi dari jaringan syaraf pada manusia. Konsep ini kemudian berkembang menjadi bidang yang lebih luas, yaitu Deep Learning.

Pada tahun 1989, Yann LeCun dan rekan-rekannya berhasil melakukan klasifikasi citra kode zip menggunakan sebuah kasus khusus dari Feed Forward Neural Network yang dikenal dengan nama Convolutional Neural Network (CNN)[1]. Namun, pengembangan Deep Learning terhenti karena keterbatasan perangkat keras, hingga pada tahun 2009, Jurgen mengembangkan sebuah Recurrent Neural Network (RNN) yang memberikan hasil signifikan dalam pengenalan tulisan tangan. Seiring dengan perkembangan komputasi pada perangkat keras Graphical Processing Unit (GPU), pengembangan DNN (Deep Neural Network) berkembang pesat.

Pada tahun 2012, sebuah CNN berhasil melakukan pengenalan citra dengan akurasi yang menyaingi manusia pada dataset tertentu[2] . Saat ini, Deep Learning telah menjadi topik yang hangat dalam dunia Machine Learning karena kemampuannya yang signifikan dalam memodelkan berbagai data kompleks seperti citra dan suara. Metode Deep Learning yang paling signifikan dalam pengenalan citra saat ini adalah Convolutional Neural Network (CNN). CNN mencoba meniru sistem pengenalan citra pada visual cortex manusia sehingga memiliki kemampuan yang baik dalam mengolah informasi citra.

II. DASAR TEORI

A. Convolutional Neural Network

1) Convolutional Neural Network (CNN) adalah jenis arsitektur jaringan saraf tiruan yang sangat efektif dalam memproses dan menganalisis gambar. CNN menggunakan lapisan konvolusi untuk mengekstraksi fitur-fitur penting dari gambar, seperti tepi, sudut, dan tekstur, mirip dengan cara manusia mengidentifikasi fitur visual. Lapisan konvolusi ini diikuti oleh lapisan pooling untuk mengurangi dimensi spasial dan lapisan aktivasi untuk memperkenalkan non-linearitas. CNN juga memiliki lapisan fully connected untuk menghubungkan fitur-fitur tersebut ke output yang diinginkan, seperti kelas-kelas gambar dalam klasifikasi.

CNN awalnya dikembangkan dengan nama NeoCognitron oleh Kunihiko Fukushima, seorang peneliti di NHK Broadcasting Science Research Laboratories di Kinuta, Setagaya, Tokyo, Jepang [3]. Konsep tersebut kemudian dikembangkan lebih lanjut oleh Yann LeCun, seorang

peneliti di AT&T Bell Laboratories di Holmdel, New Jersey, Amerika Serikat. LeCun sukses menerapkan model CNN yang dikenal sebagai LeNet dalam studinya tentang pengenalan angka dan tulisan tangan [1]. Pada tahun 2012, Alex Krizhevsky dan Ilya Sutskever meraih kesuksesan besar dengan menerapkan CNN dalam kompetisi ImageNet Large Scale Visual Recognition Challenge 2012[4]. Prestasi ini menjadi titik balik dalam membuktikan keunggulan metode Deep Learning, terutama CNN, dalam mengatasi tantangan pengenalan objek dalam citra. Metode CNN telah terbukti mengungguli pendekatan Machine Learning lainnya, seperti SVM, dalam masalah klasifikasi objek pada citra.

B. Dasar CNN

Convolutional Neural Networks (CNNs) memiliki lapisan input, lapisan output, banyak lapisan tersembunyi, dan jutaan parameter, memungkinkan mereka untuk mempelajari objek dan pola yang rumit. Mereka menggunakan proses konvolusi dan pooling untuk mengecilkan sampel input yang diberikan sebelum menerapkan fungsi aktivasi, di mana semua itu adalah lapisan tersembunyi yang terhubung sebagian, dengan lapisan yang sepenuhnya terhubung di akhir yang menghasilkan lapisan output. Bentuk output serupa dengan ukuran gambar input.

Konvolusi adalah proses menggabungkan dua fungsi untuk menghasilkan output dari fungsi yang lain. Gambar input dikonvolusi dengan penerapan filter dalam CNN, menghasilkan peta Fitur. Filter adalah bobot dan bias yang merupakan vektor yang dihasilkan secara acak dalam jaringan. Alih-alih memiliki bobot dan bias individual untuk setiap neuron, CNN menggunakan bobot dan bias yang sama untuk semua neuron. Banyak filter dapat dibuat, masing-masing menangkap aspek yang berbeda dari input. Kernel adalah nama lain untuk filter.

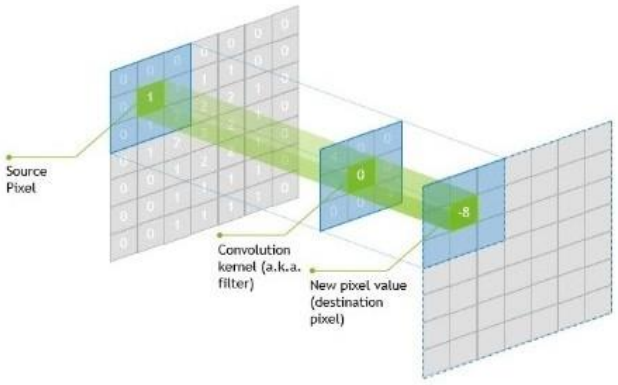
1) Convolutional Layer

Dalam jaringan saraf konvolusional (CNN), elemen utama adalah lapisan konvolusi. Lapisan ini umumnya berisi vektor masukan, seperti gambar, filter, seperti detektor fitur, dan vektor keluaran, seperti peta fitur. Gambar diabstraksi menjadi peta fitur, juga dikenal sebagai peta aktivasi, setelah melewati lapisan konvolusi.

Peta Fitur = Gambar Masukan x Detektor Fitur

Masukan dikonvolusi oleh lapisan konvolusi, yang kemudian meneruskan keluaran ke lapisan berikutnya. Ini analog dengan respons neuron terhadap stimulus tunggal di korteks visual. Setiap neuron konvolusi hanya memproses data untuk bidang reseptif yang ditugaskan kepadanya.
Konvolusi adalah fungsi pengelompokan dalam matematika. Konvolusi terjadi dalam CNN ketika dua matriks (larik berangka yang disusun dalam kolom dan baris) digabungkan untuk menghasilkan matriks ketiga.

Dalam lapisan konvolusi CNN, konvolusi ini digunakan untuk menyaring data masukan dan menemukan informasi.



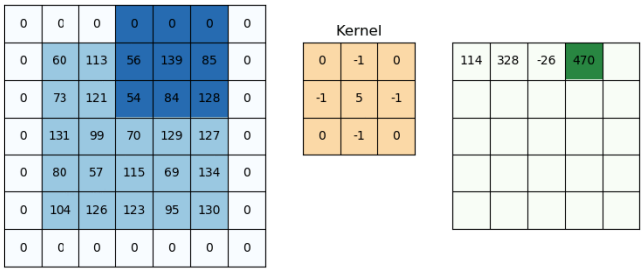
Gambar.1. Convolutional Layer

Dalam CNN, elemen pusat kernel ditempatkan di atas piksel sumber. Setelah itu, piksel sumber diganti dengan jumlah tertimbang dari dirinya sendiri dan piksel tetangga. Parameter sharing dan koneksi lokal adalah dua prinsip yang digunakan dalam CNN. Semua neuron dalam peta fitur menggunakan bobot yang sama, yang dikenal sebagai parameter sharing. Koneksi lokal merujuk pada ide setiap neuron hanya terhubung ke bagian tertentu dari gambar masukan (berbeda dengan jaringan saraf di mana semua neuron terhubung sepenuhnya). Ini mengurangi jumlah parameter dalam sistem dan mempercepat perhitungan.

2) Padding and Stride

Padding dan stride memiliki dampak pada bagaimana prosedur konvolusi dilakukan. Mereka dapat digunakan untuk meningkatkan atau mengurangi dimensi (tinggi dan lebar) vektor masukan/keluaran.

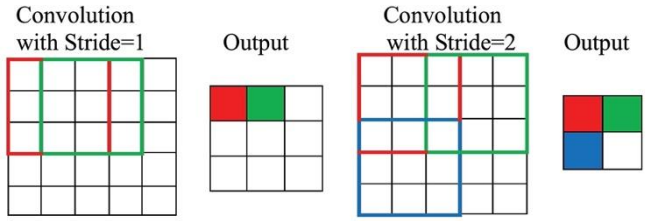
Padding adalah istilah yang digunakan dalam jaringan saraf konvolusional untuk menggambarkan berapa banyak piksel yang ditambahkan ke sebuah gambar saat diproses oleh kernel CNN. Jika padding dalam CNN diatur ke nol, misalnya, setiap nilai piksel yang ditambahkan akan memiliki nilai nol. Jika padding nol diatur menjadi satu, sebuah batas satu piksel dengan nilai piksel nol akan ditambahkan ke gambar.



Gambar.2. Padding

Padding bekerja dengan meningkatkan wilayah pemrosesan jaringan saraf konvolusional. Kernel adalah filter jaringan saraf yang bergerak melalui gambar, memindai setiap piksel dan mengubah data menjadi format yang lebih kecil atau lebih besar. Padding ditambahkan ke bingkai gambar untuk membantu kernel dalam memproses gambar dengan memberikan lebih banyak ruang bagi kernel untuk mencakup

gambar. Menambahkan padding ke gambar yang diproses CNN memberikan analisis gambar yang lebih akurat.

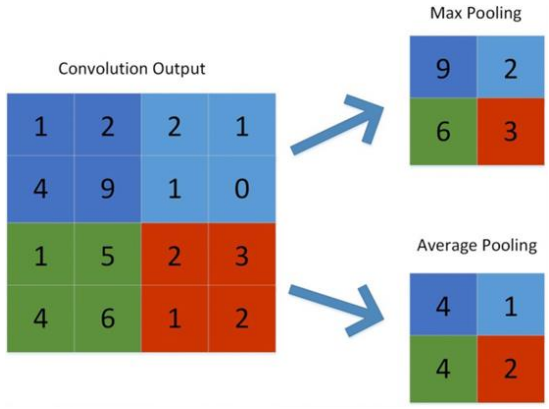


Gambar.3. Stride

Stride menentukan bagaimana filter melakukan konvolusi di atas matriks masukan, yaitu berapa banyak piksel yang bergeser. Ketika stride diatur ke 1, filter bergerak melintasi satu piksel pada satu waktu, dan ketika stride diatur ke 2, filter bergerak melintasi dua piksel pada satu waktu. Semakin kecil nilai stride, semakin kecil outputnya, dan sebaliknya.

3) Pooling

Tujuannya adalah untuk secara bertahap menyusutkan ukuran spasial representasi untuk mengurangi jumlah parameter dan komputasi dalam jaringan. Lapisan pooling memperlakukan setiap peta fitur secara terpisah.



Gambar.4. Pooling

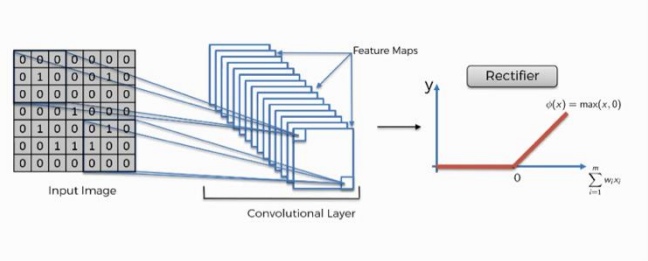
Berikut adalah beberapa metode untuk pooling:

Max-pooling: Memilih elemen paling signifikan dari peta fitur. Fitur-fitur signifikan dari peta fitur disimpan dalam lapisan max-pooling yang dihasilkan. Ini adalah metode yang paling populer karena menghasilkan hasil terbaik.
Average pooling: Melibatkan menghitung rata-rata untuk setiap wilayah dari peta fitur.

Pooling secara bertahap mengurangi dimensi spasial representasi untuk mengurangi jumlah parameter dan komputasi dalam jaringan, serta mencegah overfitting. Jika tidak ada pooling, output memiliki resolusi yang sama dengan input.

4) ReLU

Fungsi aktivasi linear terkoreksi, atau ReLU singkatnya, adalah fungsi linear berpotongan yang, jika masukan positif, menghasilkan keluaran secara langsung; jika tidak, menghasilkan nol. Karena model yang menggunakannya lebih cepat dalam pelatihan dan umumnya menghasilkan kinerja yang lebih tinggi, ReLU telah menjadi fungsi aktivasi default untuk banyak jenis jaringan saraf.



Gambar.5. ReLU

Di akhir CNN, terdapat lapisan Fully connected neuron. Seperti dalam Jaringan Saraf Konvensional, neuron dalam lapisan yang sepenuhnya terhubung memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya dan bekerja secara serupa. Setelah pelatihan, vektor fitur dari lapisan yang sepenuhnya terhubung digunakan untuk mengklasifikasikan gambar ke dalam kategori yang berbeda. Setiap unit aktivasi dalam lapisan berikutnya terhubung ke semua masukan dari lapisan ini. Overfitting terjadi karena semua parameter terdapat dalam lapisan yang sepenuhnya terhubung. Overfitting dapat dikurangi dengan menggunakan berbagai strategi, termasuk dropout.

C. Augmentasi Gambar

Jaringan yang dalam memerlukan sejumlah besar data pelatihan untuk mencapai kinerja yang baik. Untuk membangun klasifikasi gambar yang kuat menggunakan sedikit data pelatihan, augmentasi gambar biasanya diperlukan untuk meningkatkan kinerja jaringan yang dalam. Augmentasi gambar menciptakan gambar pelatihan secara artifisial melalui berbagai cara pemrosesan atau kombinasi dari beberapa pemrosesan, seperti rotasi acak, pergeseran, geseran, dan pembalikan, dll.

Sebuah generator gambar yang di-augmentasi dapat dengan mudah dibuat menggunakan API ImageDataGenerator dalam Keras. ImageDataGenerator menghasilkan batch data gambar dengan augmentasi data real-time. Kode-kode paling dasar untuk membuat dan mengonfigurasi ImageDataGenerator serta melatih jaringan saraf dalam mendalam dengan gambar yang di-augmentasi adalah sebagai berikut.

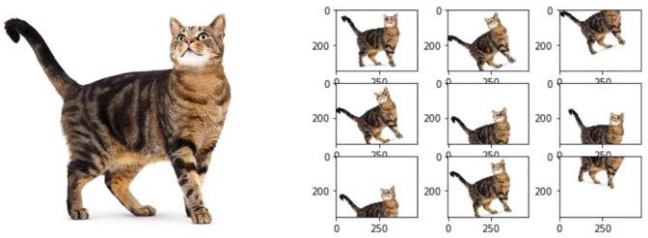
```
datagen = ImageDataGenerator()
datagen.fit(train)
X_batch, y_batch = datagen.flow(X_train, y_train,
batch_size=batch_size)
model.fit_generator(datagen, samples_per_epoch=len(train),
epochs=epochs)
```

Kita dapat bereksperimen dengan kode berikut untuk membuat gambar-gambar yang di-augmentasi dengan properti yang diinginkan. Dalam kasus kami, generator data berikut menghasilkan batch 9 gambar yang di-augmentasi

dengan rotasi sebesar 30 derajat dan pergeseran horizontal sebesar 0.5.

```
datagen = ImageDataGenerator(rotation_range=30, horizontal_flip=0.5)
datagen.fit(img)

i=0
for img_batch in datagen.flow(img, batch_size=9):
    for img in img_batch:
        plt.subplot(330 + 1 + i)
        plt.imshow(img)
        i=i+1
    if i >= batch_size:
        break
```



Gambar.6. Gambar Original dan Augmentasi

III. DESAIN PERANGKAT LUNAK

A. Praproses dan Pengolahan Data Input

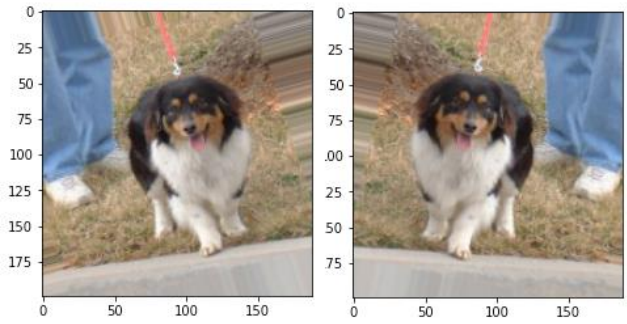
Dalam proses pengembangan model klasifikasi gambar kucing dan anjing, langkah awal yang dilakukan adalah praproses dan pengolahan data input. Pertama-tama, dataset berisi gambar-gambar kucing dan anjing dimuat menggunakan OpenCV, sebuah library komputer visi open source. Setiap gambar kemudian dikonversi ke dalam ruang warna RGB untuk memudahkan visualisasi dan pemrosesan lebih lanjut.

Mengingat dataset memiliki ukuran yang beragam, langkah selanjutnya adalah menyesuaikan ukuran setiap gambar ke ukuran yang seragam agar memastikan konsistensi dalam proses pelatihan model. Selain itu, untuk mempercepat konvergensi model selama proses pelatihan, dilakukan normalisasi data dengan membagi setiap pixel dalam gambar dengan 255. Selanjutnya, untuk meningkatkan keberagaman dataset dan mengurangi risiko overfitting, dilakukan augmentasi data menggunakan ImageDataGenerator dari Keras.

```
from keras.preprocessing.image import ImageDataGenerator

image_gen = ImageDataGenerator(rotation_range=30,
                               width_shift_range=0.1,
                               height_shift_range=0.1,
                               rescale=1/255,
                               shear_range=0.2,
                               zoom_range=0.2,
                               horizontal_flip=True,
                               fill_mode='nearest',)
```

Augmentasi data ini melibatkan berbagai teknik seperti rotasi, pergeseran, zoom, dan lainnya. Contoh dari hasil augmentasi data dapat dilihat pada gambar di bawah ini:



Gambar.7. Augmentasi Gambar

B. Proses Training

Proses training adalah tahap penting dalam pengembangan model Convolutional Neural Network (CNN) untuk klasifikasi gambar kucing dan anjing. Pada tahap ini, model CNN belajar dari data yang disiapkan sebelumnya untuk mengenali pola-pola yang mewakili gambar kucing dan anjing. Langkah pertama adalah membangun model CNN dengan arsitektur yang sesuai, yang terdiri dari beberapa layer Convolutional dan MaxPooling untuk mengekstraksi fitur dari gambar, diikuti oleh layer Dense untuk klasifikasi. Setelah model dibangun, langkah selanjutnya adalah melakukan kompilasi, di mana fungsi loss, optimizer, dan metrik evaluasi ditentukan. Saya menggunakan binary crossentropy sebagai fungsi loss, Adam optimizer, dan akurasi sebagai metrik evaluasi.

```
from keras.models import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Dropout,Activation

input_shape=(150,150,3)

model = Sequential()

model.add(Conv2D(filters=32,kernel_size=(3,3),input_shape=(150,150,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape=(150,150,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape=(150,150,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))

model.add(Dropout(0.5)) #randomly turned 50% off neurons - help in overfitting

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Setelah model dikompilasi, proses pelatihan dimulai dengan memanggil metode **fit_generator,** yang memungkinkan model memproses dataset dalam batch-batch kecil. Selama proses pelatihan, model menyesuaikan bobotnya berdasarkan gradien dari fungsi loss, dengan tujuan untuk meminimalkan loss dan meningkatkan akurasi. Evaluasi model dilakukan pada akhir setiap epoch menggunakan dataset validasi, di mana grafik yang menampilkan perubahan loss dan akurasi dapat membantu dalam memahami kemajuan model. Setelah proses pelatihan selesai, model dapat dievaluasi lebih lanjut pada data pengujian untuk mengukur akurasi prediksi pada data baru.

```
batch_size= 16

train_image_gen = image_gen.flow_from_directory('CATS_DOGS/train',
                                                target_size=input_shape[:2],
                                                batch_size=batch_size,
                                                class_mode='binary')

test_image_gen = image_gen.flow_from_directory('CATS_DOGS/test',
                                               target_size=input_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 18743 images belonging to 2 classes.
Found 6251 images belonging to 2 classes.
```

```
train_image_gen.class_indices
```

```
{'CAT': 0, 'DOG': 1}
```

```
import warnings
warnings.filterwarnings('ignore')

# Menggunakan epochs = 30 untuk mencapai akurasi > 70%
# dan dapat menjalankan 100 epochs untuk mencapai akurasi 85%.

results = model.fit_generator(train_image_gen,epochs=30,steps_per_epoch=150,
                              validation_data=test_image_gen,validation_steps=12)
#150*16 Gambar per epoch untuk menghemat waktu pelatihan
```

Proses training adalah langkah penting dalam pengembangan model machine learning, dan kesuksesan model bergantung pada kualitas data, arsitektur model, serta proses training yang efisien. Dengan melalui proses training yang baik, model CNN dapat menghasilkan prediksi yang akurat untuk membedakan gambar kucing dan anjing dengan tingkat kepercayaan yang tinggi.

IV. UJI COBA DAN HASIL

Dalam bab ini, kami melakukan uji coba serta evaluasi terhadap model Convolutional Neural Network (CNN) yang telah kami bangun untuk tujuan klasifikasi gambar kucing dan anjing. Proses uji coba ini penting untuk memastikan bahwa model yang telah dibuat mampu menggeneralisasi pola yang ditemukan dalam data pelatihan ke dalam data baru yang tidak pernah dilihat sebelumnya. Dalam konteks kode yang disediakan, kita akan menggunakan beberapa gambar dari dataset pengujian untuk mengukur akurasi prediksi model. Proses testing dimulai dengan memilih salah satu gambar dari dataset pengujian dan memuatnya ke dalam model untuk melakukan prediksi kelas.

Berikut adalah contoh kode untuk melakukan prediksi pada satu gambar dari dataset pengujian:

```
from keras.preprocessing import image
import numpy as np

dog_file = 'CATS_DOGS/test/DOG/10000.jpg'
dog_img = image.load_img(dog_file,target_size=(150,150))
dog_img = image.img_to_array(dog_img) # to array #dog_img.shape = 150,150,3

# Tetapi, kita perlu mengonversi gambar ini menjadi sebuah batch sehingga
# jaringan saraf menganggapnya sebagai sebuah batch dengan dimensi gambar ini.

dog_img = np.expand_dims(dog_img,axis=0) #now dog_img.shape = 1,150,150,3

dog_img = dog_img/255 #normalize
```

```
model.predict_classes(dog_img) # Untuk memprediksi kelas
```

```
array([[1]], dtype=int32)
```

```
# Untuk memeriksa akurasi

model.predict(dog_img)
```

```
array([[0.6940112]], dtype=float32)
```

Dalam kode di atas, kita memuat gambar anjing dari dataset pengujian, mengonversi gambar tersebut ke dalam format yang dapat diterima oleh model, dan kemudian melakukan prediksi kelas menggunakan model yang telah dilatih sebelumnya. Hasil prediksi menunjukkan bahwa gambar yang diprediksi merupakan kelas "1", yang menandakan bahwa model mengklasifikasikan gambar tersebut sebagai anjing. Selain itu, model memberikan probabilitas prediksi sebesar 0.6940112 untuk kelas tersebut. Probabilitas ini menunjukkan keyakinan model terhadap prediksi yang dilakukan, di mana semakin tinggi nilai probabilitasnya, semakin yakin model bahwa gambar tersebut termasuk dalam kelas yang diprediksi. Dengan demikian, hasil prediksi ini memberikan pemahaman tentang keputusan yang diambil oleh model CNN dan seberapa kuat keyakinan model terhadap prediksi tersebut.

Selain itu, setelah proses pelatihan selesai, kita dapat menampilkan kurva pelatihan untuk memahami perkembangan performa model selama proses pelatihan. Berikut adalah contoh kode untuk menampilkan kurva pelatihan:

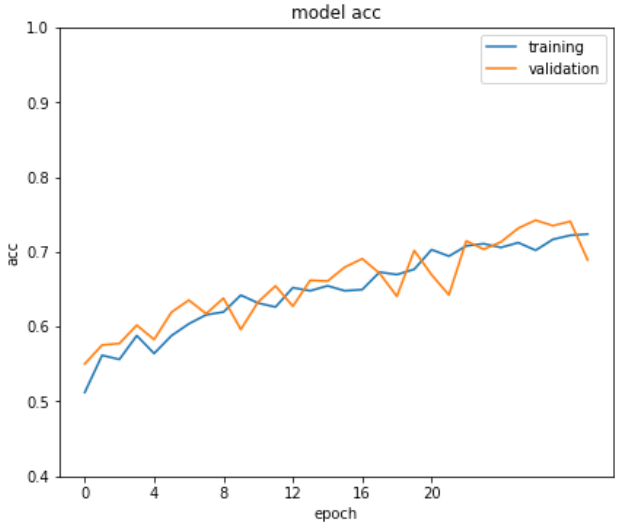
```
print(results.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

```
def display_training_curves(training, validation, title, subplot):
  ax = plt.subplot(subplot)
  ax.plot(training)
  ax.plot(validation)
  ax.set_title('model '+ title)
  ax.set_ylabel(title)
  ax.set_xlabel('epoch')
  ax.legend(['training', 'validation'])
  ax.set_ylim(0.4,1)
  ax.set_xticks([0,4,8,12,16,20])

plt.subplots(figsize=(6,10))
plt.tight_layout()
display_training_curves(results.history['acc'], results.history['val_acc'], 'acc', 211)
```

Kode di atas mendefinisikan sebuah fungsi untuk menampilkan kurva pelatihan, yang kemudian digunakan untuk menampilkan kurva akurasi pada dataset pelatihan dan validasi. Kurva ini membantu dalam memahami seberapa baik model dapat belajar dari data pelatihan dan seberapa baik dapat menggeneralisasi pada data baru.



Gambar.8. Model Akurasi

Proses testing adalah tahap penting dalam pengembangan model machine learning, yang memungkinkan kita untuk mengukur seberapa baik model dapat melakukan prediksi pada data baru yang belum pernah dilihat sebelumnya. Dengan melakukan proses testing secara cermat dan menganalisis kurva pelatihan, kita dapat memastikan bahwa model yang dikembangkan memiliki kinerja yang baik dan dapat diandalkan dalam mengklasifikasikan gambar kucing dan anjing.

V. KESIMPULAN/RINGKASAN

Dalam penelitian ini, Saya telah mengembangkan sebuah model Convolutional Neural Network (CNN) menggunakan Keras untuk mengklasifikasikan gambar-gambar kucing dan anjing. Proses pengembangan model melibatkan pemrosesan data yang cermat, termasuk augmentasi data untuk meningkatkan keragaman dataset. Hasil dari pelatihan model menunjukkan bahwa model CNN yang kami bangun mampu mempelajari fitur-fitur yang relevan dari gambar-gambar tersebut, dengan akurasi validasi mencapai sekitar 68-70%.

Evaluasi pada dataset uji juga menghasilkan prediksi yang memuaskan, menunjukkan kemampuan model untuk membedakan antara gambar kucing dan anjing dengan baik. Meskipun demikian, kami menyadari bahwa masih terdapat ruang untuk perbaikan dan eksperimen lebih lanjut. Penyesuaian hyperparameter atau eksplorasi arsitektur model yang lebih kompleks dapat menjadi langkah selanjutnya untuk meningkatkan akurasi dan generalisasi model ini. Keseluruhan, penelitian ini menunjukkan bahwa penggunaan CNN dengan augmentasi data adalah pendekatan yang menjanjikan dalam mengatasi masalah klasifikasi gambar kucing dan anjing, dan dapat menjadi landasan bagi penelitian-penelitian lebih lanjut dalam bidang ini.

DAFTAR PUSTAKA

[1] [1] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[2] [2] A.Coates, H.Lee and A.Y. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," 2011.

[3] [3] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," Biological Cybernetics,1980

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105)

[5] D. Kalita, "Basics of CNN in Deep Learning," Analytics Vidhya, Nov. 29, 2023. Available: https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning

[6] S. Lau, "Image Augmentation for Deep Learning: Histogram Equalization," Towards Data Science, Jul. 11, 2017. Available: https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2.