PROJECT BASED LEARNING
CLASSIFICATION OF HORSE OR HUMAN IMAGES USING THE CONVOLUTION NEURAL NETWORK (CNN)

Heggie Ilham[1)], Nadhea Allya Maharani[2)], Laili Putri Yanida[3)], Sandy Wijaya[4)]

Program Studi Sains Data, Jurusan Sains, Institut Teknologi Sumatera

heggie.120450005@student.itera.ac.id [1)], nadhea.120450007@student.itera.ac.id [2)], laili.120450041@student.itera.ac.id [3)], sandy.120450047@student.itera.ac.id [4)]

ABSTRAK

convolutional neural networks (CNN) adalah salah satu bagian dari neural networks yang berperan penting untuk algoritma deep learning. CNN bertugas untuk mengidentifikasi hal-hal yang sifatnya visual, misalnya saja mengenali objek pada gambar. Dalam hal ini dilakukan pemrosesan algoritma CNN untuk menetukan klasifikasi antara manusia dan kuda.

Kata Kunci : CNN, Optimizer, Aktivasi

I. PENDAHULUAN

1.1 Latar Belakang

C*onvolutional Neural Network* adalah variasi dari *Multilayer Perceptron* yang terinspirasi oleh jaringan syaraf manusia. *CNN* merupakan salah satu jenis *Neural Network* yang dapat mendeteksi dan mengenali objek pada sebuah image.[1]

Hubel dan Wiesel merupakan peneliti awal yang pertama kali dilakukan dengan melakukan penelitian visual cortex pada penglihatan kucing. Visual cortex adalah bagian dari otak yang berfungsi untuk memproses informasi visual yang berisi susunan kompleks dari sel. Hasil dari penelitian tersebut teridentifikasi 2 tipe sel dasar, yaitu sel tipe sederhana akan merespon dengan maksimal terhadap pola tertentu sedangkan sel tipe kompleks memiliki bidang reseptif yang lebih besar dan lokal invariant terhadap posisi sesuai dengan pola.

Convolutional Neural Networks merupakan susunan neuron 3D yang terdiri dari (lebar, tinggi, kedalaman). *CNN* dapat dibedakan menjadi 2 jenis layer yaitu :

1. Layer Ekstraksi Fitur Gambar : berada pada awal arsitektur yang tersusun dari beberapa layer dan setiap layer tersusun dari neuron yang terkoneksi pada daerah lokal. Pada layer ini dapat menerima input gambar secara langsung dan memprosesnya hingga menghasilkan keluaran berupa vektor untuk diolah layer berikutnya.
2. Layer Klasifikasi : merupakan layer yang tersusun dari beberapa layer dan setiap layer memiliki neuron yang terkoneksi secara penuh dengan layer lain. Hasil keluaran yang didapatkan dari layer ini berupa skoring kelas untuk klasifikasi.

Dengan Demikian *CNN* merupakan metode yang dapat digunakan untuk mentransformasikan gambar original layer per layer dari nilai piksel gambar kedalam nilai skorsing kelas untuk klasifikasi. Setiap layer ada yang memiliki hyperparameter dan tidak memiliki parameter (bobot dan bias pada neuron) [2].

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diketahui, didapatkan rumusan masalah sebagai berikut :

1. Bagaimana mengumpulkan dan merapikan dataset yang mencakup gambar - gambar manusia dan kuda dengan berbagai pose dan kondisi?
2. Bagaimana mengevaluasi kinerja model *CNN* dengan metrik yang sesuai, seperti akurasi, presisi, recall dan F1-score?

1.3 Tujuan

Berdasarkan latar belakang yang telah diketahui, didapatkan rumusan masalah sebagai berikut :

1. Mengetahui cara mengumpulkan dan merapikan dataset yang mencakup gambar - gambar manusia dan kuda dengan berbagai pose dan kondisi.
2. Mengetahui kinerja model CNN dengan metrik yang sesuai, seperti akurasi, presisi, recall dan F1-score.

II. TINJAUAN PUSTAKA

2.1 Deep Learning

Deep learning adalah salah satu metode *machine learning* yang terinspirasi otak manusia untuk memproses data dan menciptakan pola untuk pengambilan keputusan. *Deep learning* terdiri dari banyak jenis, diantaranya yang paling sering digunakan ialah *Convolution Neural Network (CNN), Multilayer Perceptrons (MLP),* dan *Recurrent Neural Network (RNN) [5].*

2.2 Convolution Neural Network (CNN)

Convolution Neural Network atau yang biasa disingkat *CNN* merupakan salah satu jenis dari *Deep Learning* dan merupakan metode pengembangan dari *Multilayer Perceptrons (MLP)*. Metode *CNN* dapat digunakan untuk mengklasifikasi, mengidentifikasi dan mengenali pola yang ada dalam citra. Metode *CNN* banyak diaplikasikan untuk klasifikasi gambar, karena kemampuan *CNN* dalam menemukan pola pada gambar [3]. *CNN* dapat memahami detail gambar karena memiliki arsitektur yang sesuai dengan cara kerja otak manusia dalam memproses informasi visual. *CNN* dapat mengolah data dua dimensi, seperti citra atau suara [4].

2.2.1 Konsep CNN

Konsep dasar Convolutional Neural Network (CNN) merupakan sebuah jenis arsitektur jaringan saraf tiruan yang dengan rancangan untuk memproses data gambar atau data yang memiliki struktur grid seperti gambar, seperti citra piksel atau data spasial. CNN memiliki komponen - komponen utama yang membuatnya efektif dalam tugas-tugas pengenalan pola dalam gambar. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi.

2.2.2 Arsitektur CNN

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara. Sebuah CNN terdiri dari beberapa layer. Arsitektur *Convolutional Neural Network (CNN)* merujuk pada desain dan struktur

jaringan saraf tiruan yang digunakan untuk tugas pengolahan gambar dan pengenalan pola dalam data berstruktur grid, seperti citra piksel. Arsitektur CNN memungkinkan model untuk secara efektif mengekstraksi fitur-fitur penting dari gambar dan digunakan secara luas dalam berbagai aplikasi pengolahan citra, termasuk pengenalan objek, klasifikasi gambar, deteksi wajah. Pada CNN terdapat 3 lapisan yaitu,

1. Convolutional layer, menjadi bagian terluar yang bertugas untuk mengidentifikasi hal-hal sederhana seputar gambar maupun suara.
2. Pooling layer, dikenal sebagai downsampling ini bertugas untuk menindaklanjuti identifikasi objek terhadap sebuah objek visual dan audio
3. Fully-connected (FC) tugasnya untuk mengenali objek dan bentuk sehingga dapat menemukan objek sebenarnya yang dimaksud.

2.4 Fungsi Aktivasi

Fungsi aktivasi umum yang digunakan termasuk:

1. ReLU (Rectified Linear Activation): ReLU adalah salah satu fungsi aktivasi yang paling umum digunakan. Ini bekerja dengan cara mengaktifkan semua nilai positif dan mematikan semua nilai negatif. Fungsi ini membantu dalam pelatihan yang efisien dan sering digunakan pada lapisan-lapisan tersembunyi.
2. Sigmoid: Fungsi sigmoid mengubah keluaran menjadi rentang antara 0 dan 1. Ini sering digunakan di lapisan output jaringan untuk masalah klasifikasi biner, di mana jika ingin menghasilkan probabilitas.

III. METODE

3.1 Dataset

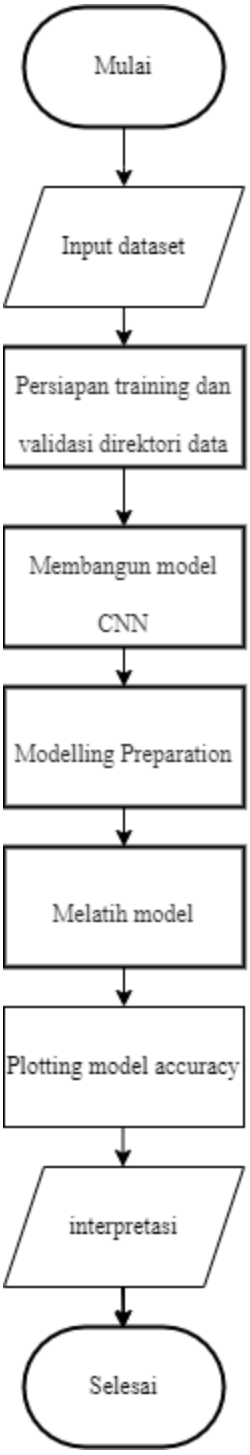
Dataset yang digunakan pada project ini berupa gambar Horse or Human yang diperoleh melalui googleapis.com dimana googleapis merupakan layanan yang terintegrasi Google Cloud Storage. Dengan rincian training data horse images 500 dan training data images 527

Gambar 1. Sample Dataset

3.2 Algoritma

Project ini memiliki beberapa tahapan dalam melakukan klasifikasi gambar horse or human ini. Tahap pertama adalah mengunduh data pada website googleapis.com, kemudian dilakukan ekstrak data dari file zip ke direktori data. Tahap kedua adalah menyusun direktori data pelatihan dan validasi. Tahap ketiga, membangun model *CNN* dengan beberapa lapisan konvolusional dan penggabungan diikuti lapisan padat. Tahap keempat adalah menyusun model yang telah dibuat dengan *loss, optimizer,* dan *metrics.* Tahap kelima adalah augmentasi data untuk kumpulan data pelatihan menggunakan *ImageDataGenerator.* Tahap keenam adalah mengalirkan data pelatihan dan validasi secara batch dengan menggunakan data generator. Tahap ketujuh adalah melatih model untuk beberapa periode dan memantau akurasi validasi. Tahap kedelapan adalah memvisualisasikan representasi perantara menggunakan beberapa contoh gambar. Tahap terakhir adalah merencanakan akurasi dan kerugian model selama pelatihan.

Dengan demikian, tahapan-tahapan ini membentuk sebuah alur yang terarah dan sistematis untuk melakukan klasifikasi image horse or human.



Gambar 2. Diagram Alir

Adapun tahap-tahap yang akan dilalui dalam melakukan klasifkasi gambar sebagai berikut :

1. Mulai, merupakan langkah awal dari proses membangun model

2. Input Dataset, dilakukan penginputan data gambar manusia dan kuda dengan menggunakan library pandas
3. Persiapan training dan validasi direktori data
4. Membangun model CNN
5. Persiapan Model, dimana model dipersiapkan untuk proses pelatihan
6. Melatih model, dilakukan pelatihan model menggunakan data training
7. *Plotting model accuracy,* langkah untuk memplot akurasi model. Akurasi model dapat digunakan untuk mengevaluasi kinerja model.

3.5 Metode Pengolahan Data

Adapun tahapan yang dilakukan dalam melakukan analisis menggunakan CNN adalah sebagai berikut :

3.5.1 Import Library

Mengimpor modul `urllib.request` untuk mengizinkan akses ke operasi jaringan, seperti mengunduh data dari internet. Kemudian, mengimpor modul `zipfile` untuk menangani file berformat zip, yang kemungkinan akan digunakan untuk mengekstrak file yang diunduh. Lalu dilakukan pengimporan pustaka TensorFlow, dimana merupakan sebuah platform open-source untuk kecerdasan buatan dan pembelajaran mesin. Selanjutnya adalah mengimpor modul `os` untuk berinteraksi dengan sistem operasi, seperti membuat dan mengelola direktori. Lalu dilakukan import kelas `ImageDataGenerator` yang berasal dari modul `tensorflow.keras.preprocessing.image`. Kelas ini digunakan untuk menghasilkan data citra tambahan secara real-time selama pelatihan model, seperti rotasi, pembalikan, dan pergeseran citra. Selanjutnya diimpor algoritma pengoptimal yang umumnya digunakan untuk melatih model dalam pembelajaran mesin.

3.5.2 Import Dataset

Pada tahap ini dilakukan proses membaca dan menampilkan data dari Data yang digunakan diperoleh melalui link googleapis dengan rincian sebagai berikut *https://storage.googleapis.com/tensorflow-1-public/course2/week3/horse-or-human.zip*. Selanjutnya akan dilakukan ekstraksi isi dari file zip tersebut ke direktori

data. Setelah selesai, akan dilakukan penutupan file zip kedua menggunakan *library import urllib.request import zipfile.*

3.5.3 Direktori Pelatihan dan Validasi

Selanjutnya dilakukan direktori pelatihan dan validasi Sehingga akan menampilkan seperti pada gambar 2. Output pada gambar 2 tersebut adalah bagian dari hasil dari kode yang telah dijelaskan sebelumnya. Output ini mencakup beberapa nama file gambar dari dataset yang digunakan untuk melatih dan menguji model klasifikasi kuda dan manusia.

```
['horse43-5.png', 'horse12-5.png', 'horse28-9.png', 'horse08-6.png',
'horse02-9.png',    'horse36-9.png',    'horse36-5.png',    'horse37-1.png',
'horse45-7.png', 'horse19-0.png']
['human03-14.png',         'human05-08.png',         'human16-02.png',
'human02-13.png',   'human17-30.png',   'human12-10.png',   'human06-05.png',
'human04-28.png', 'human14-20.png', 'human02-17.png']
VAL SET HORSES: ['horse4-345.png', 'horse5-018.png', 'horse2-201.png',
'horse5-519.png',   'horse5-504.png',   'horse4-043.png',   'horse5-275.png',
'horse1-276.png', 'horse1-455.png', 'horse6-153.png']
VAL    SET    HUMANS:    ['valhuman04-07.png',    'valhuman05-02.png',
'valhuman05-14.png',         'valhuman04-14.png',         'valhuman02-03.png',
'valhuman02-20.png',         'valhuman05-07.png',         'valhuman02-18.png',
'valhuman04-08.png', 'valhuman01-15.png']
```

Gambar 3. Melihat direktori pelatihan dan validasi kuda dan manusia dari nama file

Setiap kelompok output tersebut mencantumkan daftar file gambar yang merupakan bagian dari dataset yang digunakan untuk melatih dan menguji model klasifikasi kuda dan manusia. Gambar-gambar ini merupakan sampel data yang digunakan dalam proses pembuatan dan evaluasi model. Berikut gambar 3 jumlah total gambar kuda dan manusia pada direktori:

```
total training horse images: 500
total training human images: 527
total validation horse images: 128
total validation human images: 128
```

Gambar 4. Jumlah total gambar kuda dan manusia di direktori

Output pada gambar 3 memberikan informasi tentang jumlah gambar kuda dan manusia dalam set pelatihan dan validasi, dan jumlahnya adalah sebagai berikut:

a. Total gambar kuda dalam set pelatihan adalah 500.

b. Total gambar manusia dalam set pelatihan adalah 527.

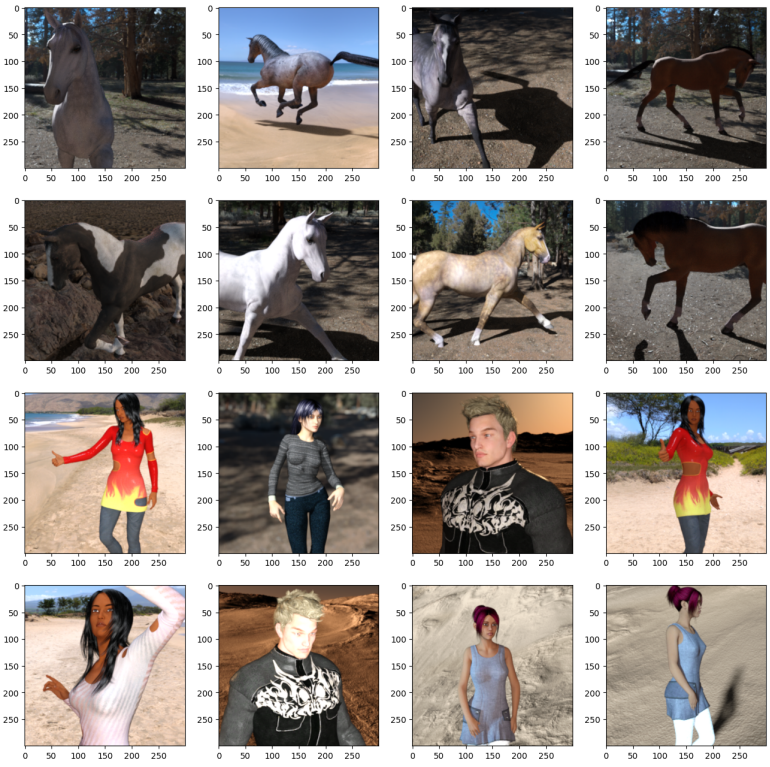
c. Total gambar kuda dalam set validasi adalah 128.

d. Total gambar manusia dalam set validasi adalah 128.

Dalam hal ini merupakan data yang digunakan dalam pelatihan dan pengujian model klasifikasi kuda dan manusia. Jumlah gambar dalam set validasi yang seimbang antara kuda dan manusia dapat membantu memastikan bahwa model yang dilatih mampu mengenali kedua kategori dengan baik.

3.5.4 Konfigurasikan parameter matplotlib

Digunakan Matplotlib untuk menampilkan gambar-gambar dalam konfigurasi 4x4. *%matplotlib inline* digunakan untuk menampilkan plot langsung di dalam Jupyter Notebook. Selanjutnya, variabel *nrows* dan *ncols* menentukan ukuran grafik 4x4, dan *pic_index* diinisialisasi sebagai 0. Selanjutnya dilakukan pemrosesan melalui kodingan untuk menampilkan 16 gambar (8 kuda dan 8 manusia) dalam konfigurasi 4x4 dengan menggunakan Matplotlib.



Gambar 4. Jumlah total gambar kuda dan manusia di direktori

Gambar-gambar ini diambil dari dataset pelatihan, dan ukuran grafik disesuaikan agar sesuai dengan konfigurasi tersebut. Sumur dan label subplot disembunyikan untuk tampilan yang bersih.

3.5.4 Membangun Model

Dilakukan pengimporan pustaka TensorFlow ke dalam skrip Python. TensorFlow adalah platform open-source yang banyak digunakan untuk pengembangan model machine learning dan kecerdasan buatan. Setelah melakukan impor dapat digunakan berbagai fitur dan fungsi TensorFlow untuk membangun, melatih, dan menerapkan model.

3.5.4.1 Model Arsitektur GoogleNet

Dilakukan pendefinisian model arsitektur GoogleNet menggunakan TensorFlow. dimana untuk setiap bagian dari model ini meliputi :

1. Conv2D Layer (Input Layer) dengan jumlah filter: 64, ukuran kernel (7,7), Stride 2, Padding: 'same', Fungsi aktivasi 'relu' , bentuk input (150, 150, 3)
2. MaxPooling2D Layer dengan ukuran pool: (3,3) dan stride: 2
3. Inception Block 1

   Conv2D Layer:

   a. Jumlah filter: 64
   b. Ukuran kernel: (1,1)
   c. Padding: 'same'
   d. Fungsi aktivasi: 'relu'

   Conv2D Layer:

   a. Jumlah filter: 128
   b. Ukuran kernel: (3,3)
   c. Padding: 'same'
   d. Fungsi aktivasi: 'relu'

   MaxPooling2D Layer

   a. Ukuran pool: (3,3)
   b. Stride: 2

4. Inception Block 2

Conv2D Layer:

a. Jumlah filter: 192
b. Ukuran kernel: (1,1)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 96
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 128
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

MaxPooling2D Layer

a. Ukuran pool: (3,3)
b. Stride: 2

5. Inception Block 3

Conv2D Layer:

a. Jumlah filter: 160
b. Ukuran kernel: (1,1)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 128
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 256
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

MaxPooling2D Layer:

a. Ukuran pool: (3,3)
b. Stride: 2

6. Inception Block 4

Conv2D Layer:

a. Jumlah filter: 256
b. Ukuran kernel: (1,1)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 128
b. ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 256
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

MaxPooling2D Layer:

a. Ukuran pool: (3,3)
b. Stride: 2

7. Inception Block 5

Conv2D Layer:

a. Jumlah filter: 384
b. Ukuran kernel: (1,1)

c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 192
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

Conv2D Layer:

a. Jumlah filter: 256
b. Ukuran kernel: (3,3)
c. Padding: 'same'
d. Fungsi aktivasi: 'relu'

8. Output Layer
   a. Flatten Layer
   b. Dense Layer dengan satu neuron dan fungsi aktivasi 'sigmoid' untuk tugas klasifikasi biner.

Model ini menggunakan beberapa blok Inception, yang dikenal karena kemampuannya menggabungkan filter dengan ukuran kernel yang berbeda untuk menangkap informasi spasial pada berbagai skala dalam gambar. Model diakhiri dengan satu lapisan Dense dengan fungsi aktivasi sigmoid untuk tugas klasifikasi biner.

Arsitektur GoogleNet, yang juga dikenal sebagai Inception, adalah salah satu arsitektur neural network yang dirancang khusus untuk tugas-tugas pengolahan citra. Berikut adalah penjelasan untuk setiap bagian dari model ini:

1. Conv2D Layer (Input Layer)
   a. Merupakan lapisan konvolusi pertama dengan 64 filter, ukuran kernel 7x7, dan fungsi aktivasi ReLU.
   b. Stride 2 digunakan untuk mengurangi dimensi gambar setelah konvolusi.
   c. Padding *same* digunakan agar dimensi gambar tidak berkurang.

2. MaxPooling2D Layer
    a. Digunakan untuk mengurangi dimensi gambar melalui operasi pooling.
    b. Pooling dilakukan dengan ukuran pool 3x3 dan stride 2.
3. Inception Blocks
    a. Model menggunakan beberapa blok Inception, yang terdiri dari beberapa jalur konvolusi dengan ukuran kernel yang berbeda.
    b. Inception Block 1, 2, 3, 4, dan 5 memiliki struktur yang berbeda untuk menangkap informasi spasial pada berbagai tingkat skala dalam gambar.
    c. Setiap blok diakhiri dengan MaxPooling2D untuk mengurangi dimensi gambar.
4. Flatten Layer
    a. Digunakan untuk mengubah output dari lapisan konvolusi menjadi vektor satu dimensi sebelum diteruskan ke lapisan Dense.
5. Flatten LayerFlatten Layer
    a. Lapisan ini memiliki satu neuron yang menggunakan fungsi aktivasi sigmoid.
    b. Cocok untuk tugas klasifikasi biner, seperti identifikasi gambar kuda atau manusia.
    c. Output model adalah probabilitas bahwa gambar termasuk dalam satu kelas.

Dengan menggunakan blok Inception, arsitektur GoogleNet dapat menangkap fitur-fitur kompleks pada berbagai tingkat hierarki, membantu meningkatkan kinerja model dalam tugas klasifikasi citra. Fungsi aktivasi sigmoid pada lapisan output menghasilkan output antara 0 dan 1, yang dapat diartikan sebagai probabilitas kelas positif dalam konteks tugas klasifikasi biner. Model ini dirancang untuk memproses gambar berukuran 150 x 150 piksel dengan tiga saluran warna (RGB).

3.5.4.2 Model Arsitektur Model EfficientNet

Dalam hal ini dilakukan pendefinisian fungsi *efficientnet* untuk membuat model arsitektur *EfficientNet. EfficientNet* adalah arsitektur neural network yang dikenal karena efisiensinya dalam hal jumlah parameter dan kinerja relatif terhadap ukuran model. Berikut adalah penjelasan singkat untuk setiap bagian dari fungsi tersebut:

1. Inisialisasi Parameter
    a. *input_shape* : Bentuk input gambar. Defaultnya adalah (224, 224, 3).
    b. *num_classes* : Jumlah kelas output. Defaultnya adalah 1000, sesuai dengan model yang umumnya digunakan untuk ImageNet.
    c. *phi* : Rasio skala yang dapat diatur untuk memilih variasi model. Defaultnya adalah 1.0.
2. Fungsi Convolutional Block (`conv_block`)
    a. Menerima input `x`, jumlah filter, ukuran kernel, strides, dan opsi untuk menggunakan batch normalization.
    b. Melakukan konvolusi 2D, *batch normalization* (opsional), dan aktivasi Swish.
3. Fungsi *MBConv Block* (`*mbconv_block*`)
    a. Menerima input `x`, jumlah filter, ukuran kernel, strides, rasio ekspansi, dan opsi untuk menggunakan batch normalization.
    b. Mengimplementasikan blok Mobile Inverted *Bottleneck Convolution (MBConv)* dengan ekspansi, konvolusi kedalaman *(depthwise convolution*), dan proyeksi.
4. Fungsi MBConv Block dengan Dropout (`*mbconv_block_dropout*`)
    a. Menerima input `x`, jumlah filter, ukuran kernel, strides, rasio ekspansi, dan dropout rate.

b. Sama dengan `mbconv_block`, tetapi dengan opsi untuk menambahkan dropout setelah blok MBConv.

5. Definisi Model EfficientNet (`efficientnet`)
    a. Menerima parameter `input_shape`, `num_classes`, dan `phi`.
    b. Mendefinisikan struktur model menggunakan blok-blok MBConv dengan variasi konfigurasi berdasarkan nilai `phi`.
    c. Blok-blok MBConv ini membentuk struktur EfficientNet yang memadukan keefisienan dan kinerja yang baik.
6. Contoh Penggunaan
    a. Membuat model dengan memanggil fungsi *`efficientnet`*.
    b. Menampilkan ringkasan model menggunakan *`summary()`*.

Fungsi ini memungkinkan pengguna untuk membuat model EfficientNet dengan mudah, dan variasi model dapat diatur melalui parameter `phi`. Model ini dapat digunakan untuk berbagai tugas klasifikasi citra dengan jumlah kelas yang berbeda.

IV. HASIL DAN PEMBAHASAN

4.1 Model Arsitektur GoogleNet

setiap lapisan (layer) yang terlibat, dan jumlah parameter yang ada di setiap lapisan.

1. Model Type:
    a. Sequential Model: Model ini dibangun sebagai urutan linear dari lapisan-lapisan. Informasi lapisan diurutkan dari awal hingga akhir.
2. Arsitektur Model:
    a. Input: Dimensi input gambar adalah (None, 150, 150, 3), yang berarti gambar berukuran 150 x 150 piksel dengan 3 saluran warna (RGB).
    b. Output: Dimensi output adalah (None, 1), menunjukkan model ini digunakan untuk tugas klasifikasi biner.
3. Lapisan-lapisan Model:

Conv2D Layers:

a. Beberapa lapisan konvolusi (`Conv2D`) dengan berbagai parameter seperti jumlah filter, ukuran kernel, dan fungsi aktivasi.

MaxPooling2D Layers:

a. Lapisan max pooling (`MaxPooling2D`) yang mengurangi dimensi spasial gambar.

Flatten Layer:

a. Lapisan flatten (`Flatten`) yang mengubah output dari lapisan konvolusi menjadi vektor satu dimensi.

Dense Layer:

a. Lapisan dense (`Dense`) dengan satu neuron dan fungsi aktivasi sigmoid di akhir, sesuai dengan tugas klasifikasi biner.

4. Total Parameters:
    a. Jumlah total parameter dalam model ini adalah 2,750,529.
    b. Jumlah parameter yang dapat dilatih (trainable params) adalah 2,750,529, yang berarti model dapat memperbarui nilai-nilai parameter ini selama pelatihan.
    c. Tidak ada parameter yang tidak dapat dilatih (non-trainable params) dalam model ini.
5. Ukuran Model:
    a. Ukuran total model (termasuk bobot, bias, dan parameter lainnya) adalah sekitar 10.49 MB.

Ringkasan ini memberikan wawasan tentang kompleksitas model, jumlah parameter yang harus dioptimalkan selama pelatihan, dan informasi lainnya yang diperlukan untuk mengerti struktur keseluruhan model.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 75, 75, 64)        9472

 max_pooling2d_5 (MaxPoolin  (None, 37, 37, 64)        0
 g2D)

 conv2d_16 (Conv2D)          (None, 37, 37, 64)        4160

 conv2d_17 (Conv2D)          (None, 37, 37, 128)       73856

 max_pooling2d_6 (MaxPoolin  (None, 18, 18, 128)       0
 g2D)

 conv2d_18 (Conv2D)          (None, 18, 18, 192)       24768

 conv2d_19 (Conv2D)          (None, 18, 18, 96)        165984

 conv2d_20 (Conv2D)          (None, 18, 18, 128)       110720

 max_pooling2d_7 (MaxPoolin  (None, 8, 8, 128)         0
 g2D)


 conv2d_23 (Conv2D)          (None, 8, 8, 256)         295168

 max_pooling2d_8 (MaxPoolin  (None, 3, 3, 256)         0
 g2D)

 conv2d_24 (Conv2D)          (None, 3, 3, 256)         65792

 conv2d_25 (Conv2D)          (None, 3, 3, 128)         295040

 conv2d_26 (Conv2D)          (None, 3, 3, 256)         295168

 max_pooling2d_9 (MaxPoolin  (None, 1, 1, 256)         0
 g2D)

 conv2d_27 (Conv2D)          (None, 1, 1, 384)         98688

 conv2d_28 (Conv2D)          (None, 1, 1, 192)         663744

 conv2d_29 (Conv2D)          (None, 1, 1, 256)         442624

 flatten_1 (Flatten)         (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 2750529 (10.49 MB)
Trainable params: 2750529 (10.49 MB)
Non-trainable params: 0 (0.00 Byte)
```

Dimana :

1. Model Type: Model sequential adalah tumpukan lapisan neural network yang disusun secara berurutan.
2. Lapisan Input : Layer (type), Output Shape, Param Ini menunjukkan header untuk bagian yang memberikan informasi tentang setiap lapisan.
3. Conv2D Layer (Lapisan Konvolusi Pertama): conv2d_15 (Conv2D); (None, 75, 75, 64) ; 9472 merupakan Lapisan konvolusi pertama dengan nama "conv2d_15". Dengan Output shape: (None, 75, 75, 64) dan parameter (Bobot dan Bias): 9,472.

4. MaxPooling2D Layer (Lapisan Max Pooling Pertama): max_pooling2d_5 ;(MaxPooling2D);(None, 37, 37, 64) ; 0 , dimana Lapisan max pooling pertama dengan output shape: (None, 37, 37, 64) dan tidak ada parameter karena max pooling tidak memiliki bobot yang dapat diubah.
5. Conv2D Layers dan MaxPooling2D Layers Berikutnya merupakan baris-baris berikutnya memberikan informasi yang serupa untuk lapisan-lapisan konvolusi dan max pooling selanjutnya.
6. Flatten Layer : flatten_1 (Flatten); (None, 256); 0, dimana Lapisan flatten yang mengubah output menjadi vektor satu dimensi, dengan output shape: (None, 256) dan Tidak ada parameter.
7. Dense Layer (Lapisan Dense Terakhir): dense_1 (Dense);(None, 1);257, dimana Lapisan dense terakhir dengan satu neuron dan fungsi aktivasi sigmoid. Output shape: (None, 1)parameter (Bobot dan Bias): 257.
8. Total Parameters dengan jumlah total params: 2,750,529
9. Trainable Parameters dan Non-trainable Parameters: Trainable params berjumlah 2,750,529, Non-trainable params: 0. Dimana Trainable params adalah jumlah parameter yang dapat dioptimalkan selama pelatihan. Dan Non-trainable params adalah parameter yang tetap konstan selama pelatihan (misalnya, Batch Normalization memiliki parameter non-trainable).
10. Ukuran Model dengan Total params: 2,750,529 (10.49 MB) Menunjukkan jumlah total parameter dan ukuran total model.

Langkah selanjutnya adalah mengatur generator data yang akan membaca gambar di folder sumber, mengubahnya menjadi tensor float32, dan memberi mereka makan (dengan labelnya) ke model. Ada generator untuk gambar pelatihan dan data validasi. Generator ini akan menghasilkan batch gambar ukuran 150x150 dan labelnya (biner).dan didapatkan

```
Found 1027 images belonging to 2 classes.
Found 256 images belonging to 2 classes.
```

Dimana :

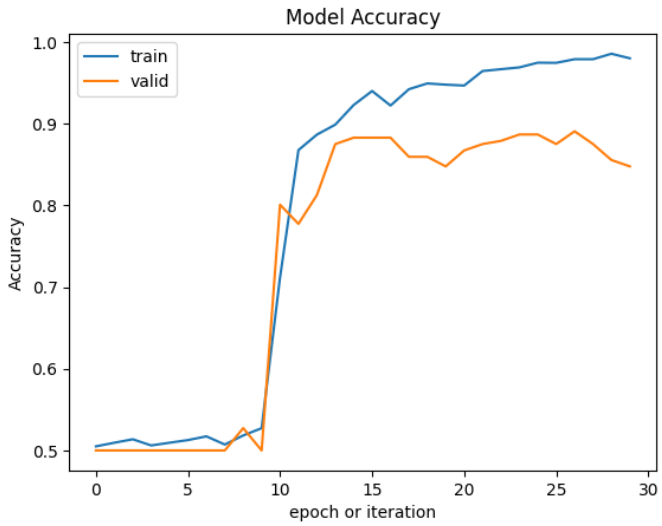
a. *Found 1027 images belonging to 2 classes.* menunjukkan bahwa generator gambar menemukan 1027 gambar dalam direktori yang dipecahkan menjadi 2 kelas.
b. *Found 256 images belonging to 2 classes.* menunjukkan bahwa generator gambar menemukan 256 gambar dalam direktori yang dipecahkan menjadi 2 kelas (biasanya untuk data validasi).

Ini adalah output informasi yang bermanfaat saat menggunakan generator gambar pada TensorFlow untuk memastikan bahwa dataset telah dimuat dengan benar. Jumlah gambar dan jumlah kelas ini dapat berguna untuk memastikan konsistensi dalam dataset dan bahwa model telah diatur dengan benar untuk melatih dan memvalidasi data tersebut.

Selanjutnya dilakukan training dimana Pada setiap epoch, model dievaluasi pada data pelatihan dan data validasi. Loss (kerugian) diukur seberapa jauh prediksi model dari label yang sebenarnya. Akurasi mengukur sejauh mana model dapat mengklasifikasikan data dengan benar. Dalam beberapa kasus, akurasi pada data pelatihan dan data validasi dapat memberikan indikasi sejauh mana model dapat generalisasi terhadap data yang belum pernah dilihat sebelumnya.

Ketika melihat output ini, penting untuk memantau perubahan loss dan akurasi selama pelatihan. Dalam contoh ini, mungkin ada pertanyaan tentang apakah model mengalami overfitting (terlalu beradaptasi dengan data pelatihan dan tidak dapat menggeneralisasi dengan baik ke data validasi) atau underfitting (tidak cukup beradaptasi dengan data pelatihan).

```
Epoch 1/30
8/8 [==============================] - 22s 707ms/step - loss: 0.2500 - accuracy: 0.5050 - val_loss: 0.2500 - val_accuracy: 0.5000
Epoch 2/30
8/8 [==============================] - 8s 967ms/step - loss: 0.2500 - accuracy: 0.5095 - val_loss: 0.2500 - val_accuracy: 0.5000
Epoch 3/30
8/8 [==============================] - 7s 952ms/step - loss: 0.2499 - accuracy: 0.5137 - val_loss: 0.2500 - val_accuracy: 0.5000
Epoch 4/30
8/8 [==============================] - 7s 801ms/step - loss: 0.2497 - accuracy: 0.5061 - val_loss: 0.2499 - val_accuracy: 0.5000
Epoch 5/30
8/8 [==============================] - 8s 973ms/step - loss: 0.2497 - accuracy: 0.5095 - val_loss: 0.2498 - val_accuracy: 0.5000
Epoch 6/30
8/8 [==============================] - 7s 841ms/step - loss: 0.2492 - accuracy: 0.5127 - val_loss: 0.2496 - val_accuracy: 0.5000
Epoch 7/30
8/8 [==============================] - 8s 1s/step - loss: 0.2482 - accuracy: 0.5172 - val_loss: 0.2490 - val_accuracy: 0.5000
Epoch 8/30
8/8 [==============================] - 6s 796ms/step - loss: 0.2472 - accuracy: 0.5072 - val_loss: 0.2569 - val_accuracy: 0.5000
Epoch 9/30
8/8 [==============================] - 9s 1s/step - loss: 0.2483 - accuracy: 0.5184 - val_loss: 0.2477 - val_accuracy: 0.5273
Epoch 10/30
8/8 [==============================] - 6s 797ms/step - loss: 0.2456 - accuracy: 0.5273 - val_loss: 0.2359 - val_accuracy: 0.5000
Epoch 11/30
8/8 [==============================] - 6s 806ms/step - loss: 0.2179 - accuracy: 0.7119 - val_loss: 0.1651 - val_accuracy: 0.8008
Epoch 12/30
8/8 [==============================] - 8s 967ms/step - loss: 0.1258 - accuracy: 0.8676 - val_loss: 0.1865 - val_accuracy: 0.7773
Epoch 13/30
8/8 [==============================] - 6s 788ms/step - loss: 0.0943 - accuracy: 0.8865 - val_loss: 0.1545 - val_accuracy: 0.8125
Epoch 14/30
8/8 [==============================] - 8s 971ms/step - loss: 0.0807 - accuracy: 0.8988 - val_loss: 0.1098 - val_accuracy: 0.8750
Epoch 15/30

8/8 [==============================] - 7s 853ms/step - loss: 0.0590 - accuracy: 0.9229 - val_loss: 0.1113 - val_accuracy: 0.8828
Epoch 16/30
8/8 [==============================] - 8s 957ms/step - loss: 0.0504 - accuracy: 0.9399 - val_loss: 0.1015 - val_accuracy: 0.8828
Epoch 17/30
8/8 [==============================] - 6s 813ms/step - loss: 0.0553 - accuracy: 0.9221 - val_loss: 0.1019 - val_accuracy: 0.8828
Epoch 18/30
8/8 [==============================] - 8s 954ms/step - loss: 0.0450 - accuracy: 0.9422 - val_loss: 0.1257 - val_accuracy: 0.8594
Epoch 19/30
8/8 [==============================] - 7s 938ms/step - loss: 0.0400 - accuracy: 0.9492 - val_loss: 0.1250 - val_accuracy: 0.8594
Epoch 20/30
8/8 [==============================] - 6s 795ms/step - loss: 0.0377 - accuracy: 0.9477 - val_loss: 0.1375 - val_accuracy: 0.8477
Epoch 21/30
8/8 [==============================] - 8s 966ms/step - loss: 0.0378 - accuracy: 0.9466 - val_loss: 0.1190 - val_accuracy: 0.8672
Epoch 22/30
8/8 [==============================] - 8s 997ms/step - loss: 0.0299 - accuracy: 0.9644 - val_loss: 0.1160 - val_accuracy: 0.8750
Epoch 23/30
8/8 [==============================] - 6s 802ms/step - loss: 0.0267 - accuracy: 0.9666 - val_loss: 0.1077 - val_accuracy: 0.8789
Epoch 24/30
8/8 [==============================] - 8s 963ms/step - loss: 0.0265 - accuracy: 0.9689 - val_loss: 0.0992 - val_accuracy: 0.8867
Epoch 25/30
8/8 [==============================] - 7s 854ms/step - loss: 0.0219 - accuracy: 0.9746 - val_loss: 0.1014 - val_accuracy: 0.8867
Epoch 26/30
8/8 [==============================] - 6s 781ms/step - loss: 0.0190 - accuracy: 0.9744 - val_loss: 0.1187 - val_accuracy: 0.8750
Epoch 27/30
8/8 [==============================] - 8s 988ms/step - loss: 0.0171 - accuracy: 0.9789 - val_loss: 0.0993 - val_accuracy: 0.8906
Epoch 28/30
8/8 [==============================] - 7s 930ms/step - loss: 0.0167 - accuracy: 0.9789 - val_loss: 0.1157 - val_accuracy: 0.8750
Epoch 29/30
8/8 [==============================] - 8s 1s/step - loss: 0.0132 - accuracy: 0.9855 - val_loss: 0.1345 - val_accuracy: 0.8555
```



Grafik tersebut menunjukkan akurasi prediksi model dari waktu ke waktu. Grafik ini memiliki dua garis, yaitu garis biru dan garis merah. Garis biru mewakili akurasi model pada data pelatihan, sedangkan garis merah mewakili akurasi model pada data validasi.Pada awal pelatihan, akurasi model pada data pelatihan dan data validasi sama-sama rendah. Hal ini menunjukkan bahwa model masih belum akurat dalam membuat prediksi. Seiring berjalannya pelatihan, akurasi model pada data pelatihan mulai meningkat. Hal ini menunjukkan bahwa model mulai belajar untuk membuat prediksi yang lebih akurat.

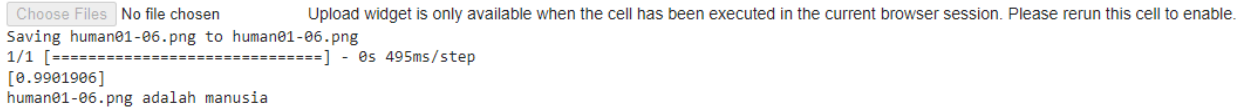
Akurasi model pada data validasi juga mulai meningkat, tetapi dengan laju yang lebih lambat. Hal ini menunjukkan bahwa model mulai overfitting, yaitu belajar terlalu banyak dari data pelatihan sehingga menjadi kurang akurat pada data baru.Pada akhirnya, akurasi model pada data validasi mencapai titik terendah. Titik ini menunjukkan bahwa model telah mencapai akurasi yang optimal.

Grafik diatas menunjukkan akurasi prediksi model dari waktu ke waktu.Pada garis biru mewakili loss model pada data pelatihan, sedangkan garis merah mewakili loss model pada data validasi. Loss adalah ukuran kesalahan prediksi model. Semakin rendah loss, semakin akurat prediksi model.Pada awal pelatihan, loss model pada data pelatihan dan data validasi sama-sama tinggi. Hal ini menunjukkan bahwa model masih belum akurat dalam membuat prediksi. Seiring berjalannya pelatihan, loss model pada data pelatihan mulai menurun. Hal ini menunjukkan bahwa model mulai belajar untuk membuat prediksi yang lebih akurat. Loss model pada data validasi juga mulai menurun, tetapi dengan laju yang lebih lambat. Hal ini menunjukkan bahwa model mulai overfitting, yaitu belajar terlalu banyak dari data pelatihan sehingga menjadi kurang akurat pada data baru.

Pada akhirnya, loss model pada data validasi mencapai titik terendah. Titik ini menunjukkan bahwa model telah mencapai akurasi yang optimal.Berdasarkan gambar tersebut, dapat disimpulkan bahwa model ini mengalami overfitting. Hal ini dapat dilihat dari perbedaan laju penurunan loss model pada data pelatihan dan data validasi.

Selanjutnya dilakukan prediksi dengan pengunggahan file gambar untuk membuktikan hasil klasifikasi manusia serta kuda. hasil prediksi dari model klasifikasi apakah gambar yang diunggah mengandung manusia atau kuda. Prosesnya dapat diuraikan sebagai berikut meminta pengguna untuk mengunggah gambar menggunakan `files.upload()` dimana dilakukan Looping

Gambar yang Diunggah untuk setiap nama file dalam file yang diunggah dilakukan pembuat path lengkap file di Google Colab, memuat gambar dengan ukuran target 150x150 piksel mengubah gambar menjadi array numerik, menormalkan nilai piksel gambar antara 0 dan menambahkan dimensi untuk batching, menjadikannya array 4 dimensi. Lalu dilakukan pembuatan prediksi gambar yaitu membuat array vertikal dari gambar yang sudah diproses menggunakan model yang sudah ditraining untuk memprediksi kelas gambar (manusia atau kuda) dengan batch size 10.



Maka didapatkan interpretasi gambar yaitu apabila nilai prediksi kelas manusia lebih dari 0.5, maka gambar diidentifikasi sebagai manusia dan akan menampilkan nama file dan label prediksi "manusia".sebaliknya jika nilai prediksi kelas manusia tidak lebih dari 0.5, maka gambar diidentifikasi sebagai kuda dan akan menampilkan nama file dan label prediksi "kuda".

Secara keseluruhan, dilakukan interaksi sederhana dengan pengguna, memungkinkan untuk mengunggah gambar dan melihat hasil prediksi dari model klasifikasi. Dengan model yang telah dilatih sebelumnya untuk mengenali manusia dan kuda berdasarkan dataset yang relevan.

4.1 Model Arsitektur EfficientNet

Model: "model_1"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 224, 224, 3)]	0
conv2d_60 (Conv2D)	(None, 112, 112, 32)	896
batch_normalization_8 (BatchNormalization)	(None, 112, 112, 32)	128
activation_22 (Activation)	(None, 112, 112, 32)	0
depthwise_conv2d_7 (DepthwiseConv2D)	(None, 112, 112, 32)	320
activation_23 (Activation)	(None, 112, 112, 32)	0
conv2d_61 (Conv2D)	(None, 112, 112, 16)	528
activation_24 (Activation)	(None, 112, 112, 16)	0
conv2d_62 (Conv2D)	(None, 112, 112, 144)	2448
activation_25 (Activation)	(None, 112, 112, 144)	0
depthwise_conv2d_8 (DepthwiseConv2D)	(None, 56, 56, 144)	1440
batch_normalization_9 (BatchNormalization)	(None, 56, 56, 144)	576
activation_26 (Activation)	(None, 56, 56, 144)	0
conv2d_63 (Conv2D)	(None, 56, 56, 24)	3480
activation_27 (Activation)	(None, 56, 56, 24)	0
conv2d_64 (Conv2D)	(None, 56, 56, 240)	6000
activation_28 (Activation)	(None, 56, 56, 240)	0
depthwise_conv2d_9 (DepthwiseConv2D)	(None, 28, 28, 240)	6240
batch_normalization_10 (BatchNormalization)	(None, 28, 28, 240)	960
activation_29 (Activation)	(None, 28, 28, 240)	0
conv2d_65 (Conv2D)	(None, 28, 28, 40)	9640
activation_30 (Activation)	(None, 28, 28, 40)	0
conv2d_66 (Conv2D)	(None, 28, 28, 480)	19680
activation_31 (Activation)	(None, 28, 28, 480)	0
depthwise_conv2d_10 (DepthwiseConv2D)	(None, 14, 14, 480)	4800
batch_normalization_11 (BatchNormalization)	(None, 14, 14, 480)	1920
activation_32 (Activation)	(None, 14, 14, 480)	0
conv2d_67 (Conv2D)	(None, 14, 14, 80)	38480
activation_33 (Activation)	(None, 14, 14, 80)	0
dropout_3 (Dropout)	(None, 14, 14, 80)	0
conv2d_68 (Conv2D)	(None, 14, 14, 672)	54432
activation_34 (Activation)	(None, 14, 14, 672)	0

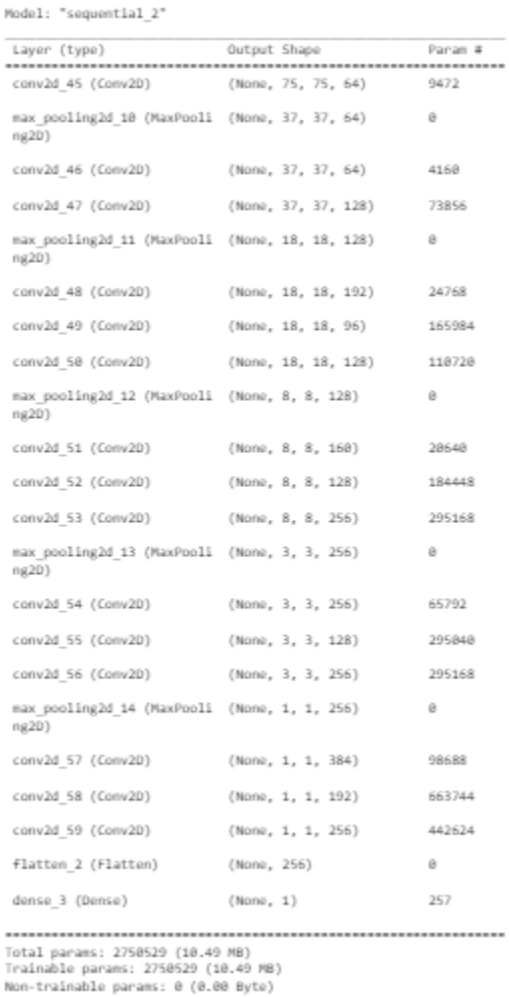
```
depthwise_conv2d_11 (Depth  (None, 14, 14, 672)       17472
wiseConv2D)

batch_normalization_12 (Ba  (None, 14, 14, 672)       2688
tchNormalization)

activation_35 (Activation)  (None, 14, 14, 672)       0

conv2d_69 (Conv2D)          (None, 14, 14, 112)       75376

activation_36 (Activation)  (None, 14, 14, 112)       0

conv2d_70 (Conv2D)          (None, 14, 14, 1152)      130176

activation_37 (Activation)  (None, 14, 14, 1152)      0

depthwise_conv2d_12 (Depth  (None, 7, 7, 1152)        29952
wiseConv2D)

batch_normalization_13 (Ba  (None, 7, 7, 1152)        4608
tchNormalization)

activation_38 (Activation)  (None, 7, 7, 1152)        0

conv2d_71 (Conv2D)          (None, 7, 7, 192)         221376

activation_39 (Activation)  (None, 7, 7, 192)         0

dropout_4 (Dropout)         (None, 7, 7, 192)         0

conv2d_72 (Conv2D)          (None, 7, 7, 1920)        370560

activation_40 (Activation)  (None, 7, 7, 1920)        0

depthwise_conv2d_13 (Depth  (None, 7, 7, 1920)        19200
wiseConv2D)

batch_normalization_14 (Ba  (None, 7, 7, 1920)        7680
tchNormalization)

activation_41 (Activation)  (None, 7, 7, 1920)        0

conv2d_73 (Conv2D)          (None, 7, 7, 320)         614720

activation_42 (Activation)  (None, 7, 7, 320)         0

dropout_5 (Dropout)         (None, 7, 7, 320)         0

conv2d_74 (Conv2D)          (None, 7, 7, 1280)        410880

batch_normalization_15 (Ba  (None, 7, 7, 1280)        5120
tchNormalization)

activation_43 (Activation)  (None, 7, 7, 1280)        0

global_average_pooling2d_1  (None, 1280)              0
 (GlobalAveragePooling2D)

dense_4 (Dense)             (None, 1000)              1281000

=================================================================
Total params: 3342776 (12.75 MB)
Trainable params: 3330936 (12.71 MB)
Non-trainable params: 11840 (46.25 KB)
_________________________________________________________________
```

Komponen-komponen yang terdapapat pada output tersebut adalah

1. Layer Input:
    a. Nama: input_3
    b. Bentuk: (None, 224, 224, 3) - Menunjukkan gambar masukan dengan dimensi 224x224 piksel dan 3 saluran warna (RGB).

2. Arsitektur:
    a. Beberapa lapisan konvolusi (Conv2D) dengan berbagai ukuran filter.
    b. Konvolusi separabel dalam kedalaman (DepthwiseConv2D) untuk beberapa lapisan.
    c. Normalisasi batch (BatchNormalization) dan fungsi aktivasi (misalnya, Activation) setelah beberapa lapisan.
    d. Lapisan Dropout (Dropout) untuk regularisasi.
    e. Layer Global Average Pooling:
3. Lapisan Global Average Pooling 2D (GlobalAveragePooling2D) diterapkan, mengurangi dimensi spasial menjadi vektor tunggal (None, 1280).
4. Layer Fully Connected: Lapisan Dense (Dense_4) dengan 1000 unit, yang umumnya digunakan untuk tugas klasifikasi dengan 1000 kelas.
5. Parameter
    a. Total Parameter: 3.342.776
    b. Parameter yang Dapat Dilatih: 3.330.936
    c. Parameter yang Tidak Dapat Dilatih: 11.840

Kemudian dilakukan peninjauan terhadap efficientNet dan dihasilkan



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_45 (Conv2D)           (None, 75, 75, 64)        9472

max_pooling2d_10 (MaxPooli   (None, 37, 37, 64)        0
ng2D)

conv2d_46 (Conv2D)           (None, 37, 37, 64)        4160

conv2d_47 (Conv2D)           (None, 37, 37, 128)       73856

max_pooling2d_11 (MaxPooli   (None, 18, 18, 128)       0
ng2D)

conv2d_48 (Conv2D)           (None, 18, 18, 192)       24768

conv2d_49 (Conv2D)           (None, 18, 18, 96)        165984

conv2d_50 (Conv2D)           (None, 18, 18, 128)       110720

max_pooling2d_12 (MaxPooli   (None, 8, 8, 128)         0
ng2D)

conv2d_51 (Conv2D)           (None, 8, 8, 160)         20640

conv2d_52 (Conv2D)           (None, 8, 8, 128)         184448

conv2d_53 (Conv2D)           (None, 8, 8, 256)         295168

max_pooling2d_13 (MaxPooli   (None, 3, 3, 256)         0
ng2D)

conv2d_54 (Conv2D)           (None, 3, 3, 256)         65792

conv2d_55 (Conv2D)           (None, 3, 3, 128)         295040

conv2d_56 (Conv2D)           (None, 3, 3, 256)         295168

max_pooling2d_14 (MaxPooli   (None, 1, 1, 256)         0
ng2D)

conv2d_57 (Conv2D)           (None, 1, 1, 384)         98688

conv2d_58 (Conv2D)           (None, 1, 1, 192)         663744

conv2d_59 (Conv2D)           (None, 1, 1, 256)         442624

flatten_2 (Flatten)          (None, 256)               0

dense_3 (Dense)              (None, 1)                 257

=================================================================
Total params: 2750529 (10.49 MB)
Trainable params: 2750529 (10.49 MB)
Non-trainable params: 0 (0.00 Byte)
```

Dari output yang didapat terdapat struktur sebagai berikut :

1. Arsitektur Model : Model ini menggunakan pendekatan Sequential, yang berarti lapisan-lapisan ditambahkan satu per satu dalam urutan tertentu.

2. Lapisan Konvolusi

   Lapisan Conv2D dengan berbagai ukuran filter dan jumlah filter:

   a. Conv2D_45: 64 filter ukuran (3x3), output (None, 75, 75, 64).
   b. Conv2D_46: 64 filter ukuran (3x3), output (None, 37, 37, 64).
   c. Conv2D_47: 128 filter ukuran (3x3), output (None, 37, 37, 128).
   d. Conv2D_48: 192 filter ukuran (3x3), output (None, 18, 18, 192).
   e. Conv2D_49: 96 filter ukuran (3x3), output (None, 18, 18, 96).

f. Conv2D_50: 128 filter ukuran (3x3), output (None, 18, 18, 128).
g. Conv2D_51: 160 filter ukuran (3x3), output (None, 8, 8, 160).
h. Conv2D_52: 128 filter ukuran (3x3), output (None, 8, 8, 128).
i. Conv2D_53: 256 filter ukuran (3x3), output (None, 8, 8, 256).
j. Conv2D_54: 256 filter ukuran (3x3), output (None, 3, 3, 256).
k. Conv2D_55: 128 filter ukuran (3x3), output (None, 3, 3, 128).
l. Conv2D_56: 256 filter ukuran (3x3), output (None, 3, 3, 256).
m. Conv2D_57: 384 filter ukuran (3x3), output (None, 1, 1, 384).
n. Conv2D_58: 192 filter ukuran (3x3), output (None, 1, 1, 192).
o. Conv2D_59: 256 filter ukuran (3x3), output (None, 1, 1, 256).
p. MaxPooling2D digunakan setelah beberapa lapisan konvolusi untuk mereduksi dimensi spasial.

3. Lapisan Flatten dan Dense
    a. Lapisan Flatten (`Flatten_2`) digunakan untuk meratakan output dari lapisan konvolusi sebelumnya.
    b. Lapisan Dense (`Dense_3`) dengan satu unit, yang menghasilkan output biner (None, 1).

4. Parameter:
    a. Total Parameter: 2,750,529 (10.49 MB)
    b. Parameter yang Dapat Dilatih: 2,750,529 (10.49 MB)
    c. Parameter yang Tidak Dapat Dilatih: 0

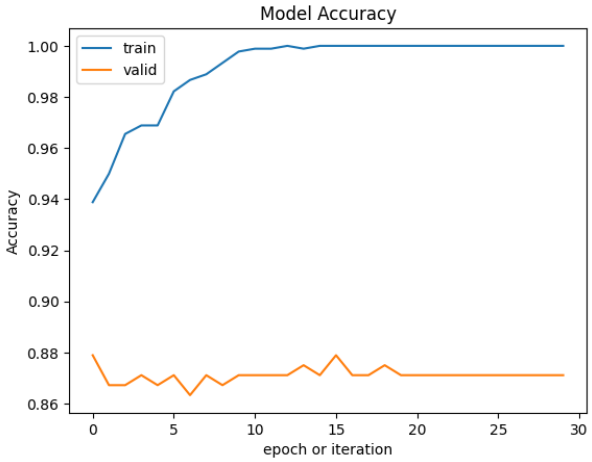
Kemudian dilakukan pembagain kelas data

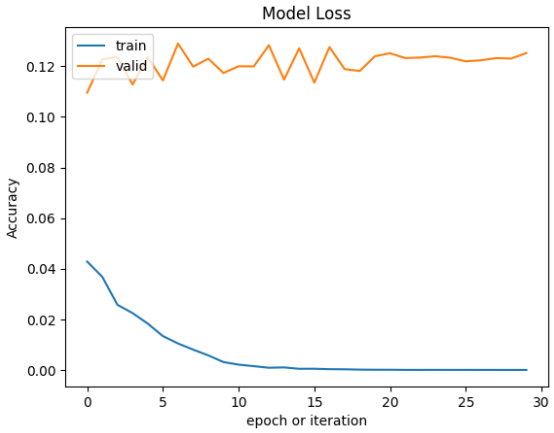
```
Found 1027 images belonging to 2 classes.
Found 256 images belonging to 2 classes.
```

Dalam kode pelatihan model, tingkat verbositas (output selama pelatihan) diatur dengan parameter `verbose`. Pada contoh, `verbose=1` memberikan informasi progres seperti loss dan akurasi pada setiap epoch, sedangkan `verbose=0` akan tidak menampilkan output selama pelatihan dan didapatkan output :

```
Epoch 1/30
8/8 [==============================] - 7s 801ms/step - loss: 0.0428 - accuracy: 0.9388 - val_loss: 0.1095 - val_accuracy: 0.8789
Epoch 2/30
8/8 [==============================] - 7s 1s/step - loss: 0.0368 - accuracy: 0.9499 - val_loss: 0.1227 - val_accuracy: 0.8672
Epoch 3/30
8/8 [==============================] - 6s 773ms/step - loss: 0.0257 - accuracy: 0.9655 - val_loss: 0.1236 - val_accuracy: 0.8672
Epoch 4/30
8/8 [==============================] - 7s 934ms/step - loss: 0.0225 - accuracy: 0.9689 - val_loss: 0.1128 - val_accuracy: 0.8711
Epoch 5/30
8/8 [==============================] - 6s 757ms/step - loss: 0.0183 - accuracy: 0.9689 - val_loss: 0.1236 - val_accuracy: 0.8672
Epoch 6/30
8/8 [==============================] - 7s 937ms/step - loss: 0.0134 - accuracy: 0.9822 - val_loss: 0.1144 - val_accuracy: 0.8711
Epoch 7/30
8/8 [==============================] - 6s 792ms/step - loss: 0.0105 - accuracy: 0.9867 - val_loss: 0.1290 - val_accuracy: 0.8633
Epoch 8/30
8/8 [==============================] - 9s 1s/step - loss: 0.0081 - accuracy: 0.9889 - val_loss: 0.1198 - val_accuracy: 0.8711
Epoch 9/30
8/8 [==============================] - 6s 863ms/step - loss: 0.0058 - accuracy: 0.9933 - val_loss: 0.1230 - val_accuracy: 0.8672
Epoch 10/30
8/8 [==============================] - 7s 937ms/step - loss: 0.0031 - accuracy: 0.9978 - val_loss: 0.1173 - val_accuracy: 0.8711
Epoch 11/30
8/8 [==============================] - 6s 765ms/step - loss: 0.0021 - accuracy: 0.9989 - val_loss: 0.1200 - val_accuracy: 0.8711
Epoch 12/30
8/8 [==============================] - 7s 804ms/step - loss: 0.0015 - accuracy: 0.9989 - val_loss: 0.1199 - val_accuracy: 0.8711
Epoch 13/30
8/8 [==============================] - 8s 983ms/step - loss: 8.9810e-04 - accuracy: 1.0000 - val_loss: 0.1283 - val_accuracy: 0.8711
Epoch 14/30
8/8 [==============================] - 6s 782ms/step - loss: 0.0010 - accuracy: 0.9989 - val_loss: 0.1147 - val_accuracy: 0.8750
Epoch 15/30
8/8 [==============================] - 7s 925ms/step - loss: 4.8471e-04 - accuracy: 1.0000 - val_loss: 0.1271 - val_accuracy: 0.8711

Epoch 16/30
8/8 [==============================] - 7s 803ms/step - loss: 5.0338e-04 - accuracy: 1.0000 - val_loss: 0.1135 - val_accuracy: 0.8789
Epoch 17/30
8/8 [==============================] - 8s 1s/step - loss: 3.3318e-04 - accuracy: 1.0000 - val_loss: 0.1276 - val_accuracy: 0.8711
Epoch 18/30
8/8 [==============================] - 6s 768ms/step - loss: 2.8902e-04 - accuracy: 1.0000 - val_loss: 0.1188 - val_accuracy: 0.8711
Epoch 19/30
8/8 [==============================] - 7s 930ms/step - loss: 1.5171e-04 - accuracy: 1.0000 - val_loss: 0.1181 - val_accuracy: 0.8750
Epoch 20/30
8/8 [==============================] - 6s 772ms/step - loss: 1.0954e-04 - accuracy: 1.0000 - val_loss: 0.1239 - val_accuracy: 0.8711
Epoch 21/30
8/8 [==============================] - 7s 935ms/step - loss: 1.0354e-04 - accuracy: 1.0000 - val_loss: 0.1251 - val_accuracy: 0.8711
Epoch 22/30
8/8 [==============================] - 6s 775ms/step - loss: 5.4249e-05 - accuracy: 1.0000 - val_loss: 0.1232 - val_accuracy: 0.8711
Epoch 23/30
8/8 [==============================] - 7s 926ms/step - loss: 4.6528e-05 - accuracy: 1.0000 - val_loss: 0.1234 - val_accuracy: 0.8711
Epoch 24/30
8/8 [==============================] - 6s 770ms/step - loss: 6.0890e-05 - accuracy: 1.0000 - val_loss: 0.1240 - val_accuracy: 0.8711
Epoch 25/30
8/8 [==============================] - 7s 952ms/step - loss: 5.3392e-05 - accuracy: 1.0000 - val_loss: 0.1233 - val_accuracy: 0.8711
Epoch 26/30
8/8 [==============================] - 6s 781ms/step - loss: 5.2413e-05 - accuracy: 1.0000 - val_loss: 0.1219 - val_accuracy: 0.8711
Epoch 27/30
8/8 [==============================] - 6s 781ms/step - loss: 4.5863e-05 - accuracy: 1.0000 - val_loss: 0.1223 - val_accuracy: 0.8711
Epoch 28/30
8/8 [==============================] - 7s 950ms/step - loss: 4.3497e-05 - accuracy: 1.0000 - val_loss: 0.1232 - val_accuracy: 0.8711
Epoch 29/30
8/8 [==============================] - 6s 792ms/step - loss: 4.0170e-05 - accuracy: 1.0000 - val_loss: 0.1230 - val_accuracy: 0.8711
Epoch 30/30
8/8 [==============================] - 6s 764ms/step - loss: 3.8013e-05 - accuracy: 1.0000 - val_loss: 0.1252 - val_accuracy: 0.8711
```

Kemudian didapatkan grafik dari nilai akurasi dan loss sebagai berikut:





Dari hasil pelatihan pada Epoch 1-5 Terjadi penurunan loss dan peningkatan akurasi pada data pelatihan dan validasi. Epoch 6-15 Model terus meningkat dengan penurunan loss dan peningkatan akurasi di kedua dataset. Epoch 16-30:Performa model tetap stabil dengan nilai akurasi yang tinggi di atas 99%.

Model mencapai akurasi tinggi (>99%) pada akhir pelatihan, menunjukkan kemampuan baik dalam mempelajari pola pada dataset.Penurunan loss pada setiap epoch menandakan model semakin baik dalam mengoptimalkan parameter.Konsistensi antara data pelatihan dan validasi menunjukkan bahwa model tidak mengalami overfitting atau underfitting. Kinerja yang baik pada data validasi (val_accuracy: 0.8711) menandakan generalisasi model yang baik pada data yang belum pernah dilihat sebelumnya.

Gw nymbi pindahin ke template paper ya nad bolehh

V. PENUTUP

5.1 Kesimpulan

Dalam penelitian ini, dilakukan implementasi Convolutional Neural Network (CNN) untuk klasifikasi gambar manusia dan kuda. CNN digunakan untuk mengenali objek pada gambar dengan fokus pada aspek visual. Dua arsitektur model yang digunakan adalah GoogleNet dan EfficientNet, dengan tujuan untuk membandingkan kinerja keduanya. digunakan dua arsitektur model, yaitu GoogleNet dan EfficientNet. GoogleNet memiliki arsitektur dengan

beberapa blok Inception, sementara EfficientNet dikenal karena efisiensinya dalam jumlah parameter dan kinerja. Proses pengolahan data melibatkan impor library, pengunduhan dataset, pembuatan generator gambar, dan training model. Tambahin perbedaan nilai akurasi gugelnet sama efficientnetnya

5.2 Saran

Dalam projek ini, lebih baik dilakukan analisis lebih mendalam terhadap kinerja model, dan melakukan eksplorasi lanjutan terkait aktivasi optimizer yang digunakan. Dapat dilakukan pula

DAFTAR PUSTAKA

[1] Lina.Qolbiyah, "Apa Itu Convolutional Neural Network?,Januari 2019.Https://Medium.Com/@16611110/Apa Itu-Convolutional-Neural-Network-836f 70b193a4

[2] Trivusi, "Pengertian Dan Cara Kerja Algoritma Convolutional Network (Cnn), Neural Juli 2022.-Cnn.Html Https://Www.Triv

[3] https://www.sciencedirect.com/science/article/pii/S2772632023000521?via%3Dihub