LAPORAN

UAS TUBES ML

Computer Vision Hugging Face Course
Task 4

Oleh :

M.Jamil Al Munawar/1103213175

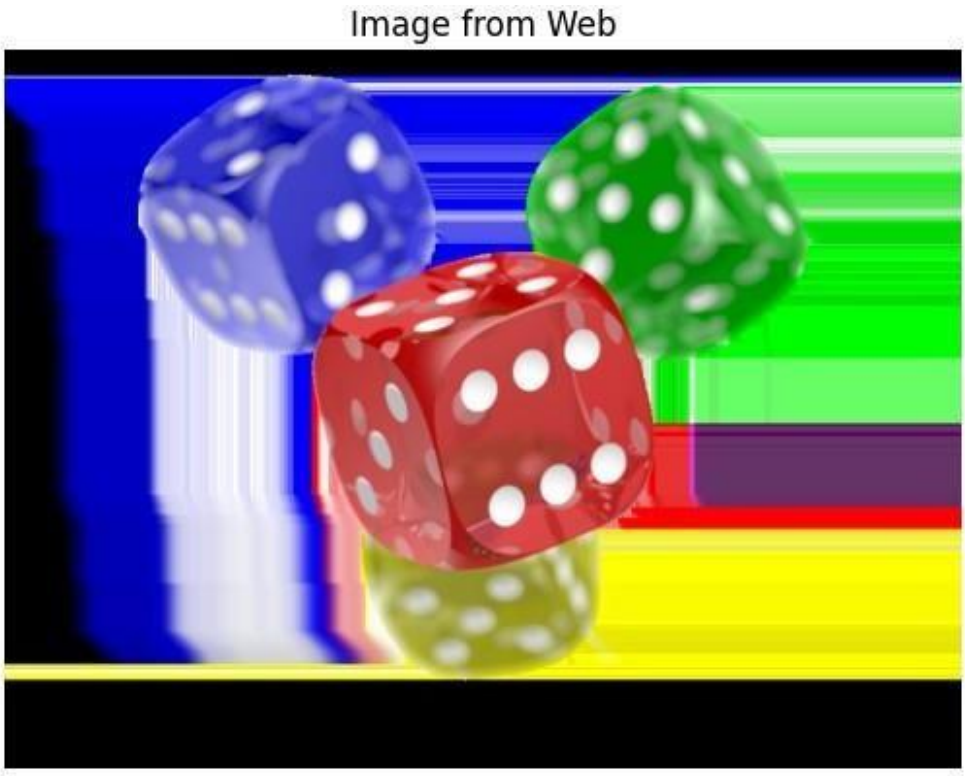
PRODI S1 TEKNIK KOMPUTER
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM
BANDUNG
2024

UNIT 1 **** Computer Vision adalah cabang dari kecerdasan buatan (AI) yang memungkinkan komputer untuk memahami, menganalisis, dan menafsirkan informasi visual dari dunia nyata, seperti gambar dan video.

1. Vision Ini mencakup konsep dasar penglihatan manusia dan bagaimana penglihatan komputer mencoba untuk meniru cara manusia memproses informasi visual. Fokus utamanya adalah memahami cara komputer "melihat" menggunakan piksel.

2. Image Pada subtopik ini, gambar dijelaskan sebagai representasi data berbasis piksel dalam berbagai format seperti grayscale, RGB, atau lainnya.

```
import cv2
import numpy as np
import requests
import matplotlib.pyplot as plt

# URL gambar dari web (ganti ini dengan URL gambar langsung yang
valid)
url =
"https://upload.wikimedia.org/wikipedia/commons/4/47/PNG_transparency_
demonstration_1.png"     # Contoh URL gambar langsung

# Mengunduh gambar dari web
response = requests.get(url)
if response.status_code == 200:
    # Mengubah data biner menjadi array numpy
    image_array = np.asarray(bytearray(response.content),
dtype=np.uint8)

    # Membaca gambar dari array numpy
    image = cv2.imdecode(image_array, cv2.IMREAD_COLOR)

    # Konversi BGR ke RGB untuk ditampilkan dengan benar
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Menampilkan gambar
    plt.imshow(image_rgb)
    plt.title("Image from Web")
    plt.axis('off')
    plt.show()
else:
    print("Gagal mengunduh gambar. Periksa URL atau koneksi
internet.")
```

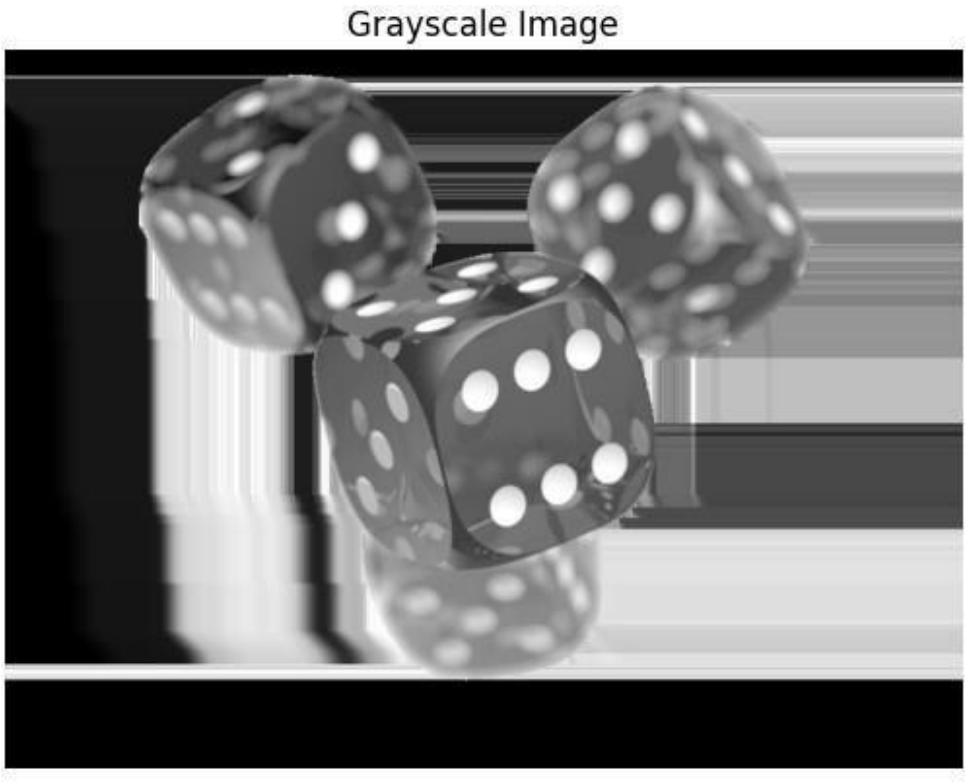


- **cv2**(OpenCV):
  - Digunakan untuk operasi pengolahan gambar seperti membaca, mengubah format warna, dan decoding gambar.
- **numpy**:
  - Membantu mengelola array numerik, termasuk representasi gambar dalam bentuk array.
- **requests**:
  - Memungkinkan pengunduhan data atau file dari URL melalui protokol HTTP.
- **matplotlib.pyplot**:
  - Digunakan untuk menampilkan gambar dalam grafik interaktif atau statis.
- **url**: Alamat gambar di internet yang akan diunduh. URL ini mengarah langsung ke file gambar berformat .png.
- **requests.get(url)**:
  - Mengirim permintaan HTTP GET untuk mengunduh konten dari URL.
  - Hasilnya disimpan dalam variabel response, yang berisi data seperti status HTTP dan konten.
- **response.status_code**:
  - Status HTTP dari permintaan.
    - **200**: Permintaan berhasil, konten gambar tersedia.
    - Kode lainnya: Gagal, seperti 404 (file tidak ditemukan). image_array = np.asarray(bytearray(response.content), dtype=np.uint8)
- **response.content**:
  - Konten biner dari gambar yang diunduh.

- **bytearray(response.content)**:
  - Mengubah data biner menjadi array byte.
- **np.asarray(..., dtype=np.uint8)**:
  - Mengonversi array byte menjadi array NumPy dengan tipe data 8-bit unsigned integer (sesuai format piksel gambar). image = cv2.imdecode(image_array, cv2.IMREAD_COLOR) image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
- **cv2.cvtColor**:
  - Mengonversi format warna gambar dari BGR (Blue-Green-Red) ke RGB (Red-Green-Blue).
- Mengapa?
  - OpenCV menggunakan format warna BGR secara default.
  - Matplotlib membutuhkan format warna RGB untuk menampilkan gambar dengan benar.
- **plt.imshow(image_rgb)**:
  - Menampilkan gambar dengan format warna RGB menggunakan Matplotlib.
- **plt.title("Image from Web")**:
  - Menambahkan judul pada gambar.
- **plt.axis('off')**:
  - Menghapus sumbu (axis) agar tampilan gambar lebih bersih.
- **plt.show()**:
  - Menampilkan gambar dalam jendela Matplotlib.

1. Imaging Subtopik ini menjelaskan cara gambar ditangkap menggunakan kamera atau sensor. Proses ini melibatkan konversi sinyal analog menjadi digital.

```
# Konversi gambar menjadi grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Menampilkan gambar grayscale
plt.imshow(gray_image, cmap='gray')
plt.title("Grayscale Image")
plt.axis('off')
plt.show()
```



- Fungsi **cv2.cvtColor**:
  - Fungsi dari OpenCV untuk mengonversi format warna gambar.
  - Parameter **image**: Input gambar yang akan dikonversi (dalam format BGR).
  - Parameter **cv2.COLOR_BGR2GRAY**:
    - Mengonversi gambar dari format BGR (Blue-Green-Red) menjadi grayscale.
    - Gambar grayscale hanya memiliki satu saluran (channel) intensitas cahaya.

Penjelasan Grayscale:

- Format Grayscale:
  - Setiap piksel memiliki nilai intensitas cahaya antara 0 (hitam) hingga 255 (putih).
- Keuntungan Grayscale:
  - Sederhana:
    - Grayscale mengurangi dimensi data dari 3 saluran (RGB) menjadi 1 saluran.
  - Efisiensi:
    - Operasi komputasi menjadi lebih cepat dan hemat memori.
  - Fokus pada Fitur:
    - Menghilangkan informasi warna yang sering kali tidak relevan untuk tugas seperti deteksi tepi atau pengenalan pola. (gray_image, cmap='gray')
- Fungsi **plt.imshow**:
  - Menampilkan gambar menggunakan pustaka Matplotlib.

  - Parameter **gray_image**: Gambar yang akan ditampilkan (dalam format grayscale).
  - Parameter **cmap='gray'**:
    - Menentukan skema warna untuk menampilkan gambar.
    - 'gray' memastikan gambar ditampilkan dalam skala abu-abu. rayscale Image")
- Fungsi **plt.title**:
  - Menambahkan judul pada gambar.
  - Judulnya adalah "Grayscale Image" untuk menunjukkan bahwa gambar telah dikonversi ke format grayscale.
- Fungsi **plt.axis('off')**:
  - Menghapus tampilan sumbu pada gambar (sumbu X dan Y).
  - Tujuannya adalah untuk memberikan tampilan gambar yang lebih bersih dan fokus.
- Fungsi **plt.show**:
  - Menampilkan gambar dalam jendela grafik interaktif atau statis Matplotlib.
  - Gambar grayscale akan ditampilkan sesuai dengan skema warna dan judul yang diberikan.

1. Imaging in Real-life Membahas aplikasi nyata, seperti fotografi atau pengawasan, di mana teknik pengolahan gambar diterapkan.

2. What is Computer Vision? Penjelasan tentang penglihatan komputer sebagai cabang kecerdasan buatan yang memungkinkan komputer untuk memahami dan menafsirkan gambar atau video.

3. Applications of Computer Vision Subtopik ini membahas berbagai aplikasi penglihatan komputer, seperti deteksi wajah, pengenalan objek, dan klasifikasi gambar.

```
#face_cascade.detectMultiScale: Mendeteksi wajah pada gambar.
#cv2.rectangle: Menggambar persegi panjang pada area wajah.


# Muat model Haar Cascade untuk pendeteksian wajah
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 
'haarcascade_frontalface_default.xml')

# Deteksi wajah
faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, 
minNeighbors=5, minSize=(30, 30))

# Gambar persegi panjang di sekitar wajah yang terdeteksi
for (x, y, w, h) in faces:
    cv2.rectangle(image_rgb, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Tampilkan hasilnya
plt.imshow(image_rgb)
```

```
plt.title("Wajah Terdeteksi")
plt.axis('off')
plt.show()
```

- **cv2.CascadeClassifier**:
  - Digunakan untuk memuat model Haar Cascade, yang merupakan model deteksi objek berbasis fitur.
  - Model Haar Cascade tersedia dalam bentuk file XML yang telah dilatih sebelumnya.
- **cv2.data.haarcascades**:
  - Direktori bawaan OpenCV yang berisi file Haar Cascade untuk berbagai objek, termasuk wajah.
- **haarcascade_frontalface_default.xml**:
  - File XML yang berisi model deteksi wajah frontal.
- **detectMultiScale**:
  - Fungsi untuk mendeteksi wajah pada gambar.
- Parameter:
  a. **gray_image**:
     - Gambar input dalam format grayscale. Deteksi wajah bekerja lebih baik pada gambar grayscale karena memproses intensitas piksel saja.
  b. **scaleFactor=1.1**:
     - Menentukan seberapa besar gambar diperkecil dalam setiap iterasi.
     - Nilai 1.1 berarti gambar diperkecil sebesar 10% di setiap langkah.

- Ukuran ini membantu mendeteksi wajah dengan berbagai ukuran.

c. **minNeighbors=5**:
   - Menentukan jumlah minimum tetangga untuk setiap kandidat deteksi wajah.
   - Nilai lebih tinggi menghasilkan deteksi yang lebih ketat dan mengurangi false positives.

d. **minSize=(30, 30)**:
   - Ukuran minimum wajah yang akan dideteksi. Wajah lebih kecil dari 30x30 piksel akan diabaikan.

Hasil:

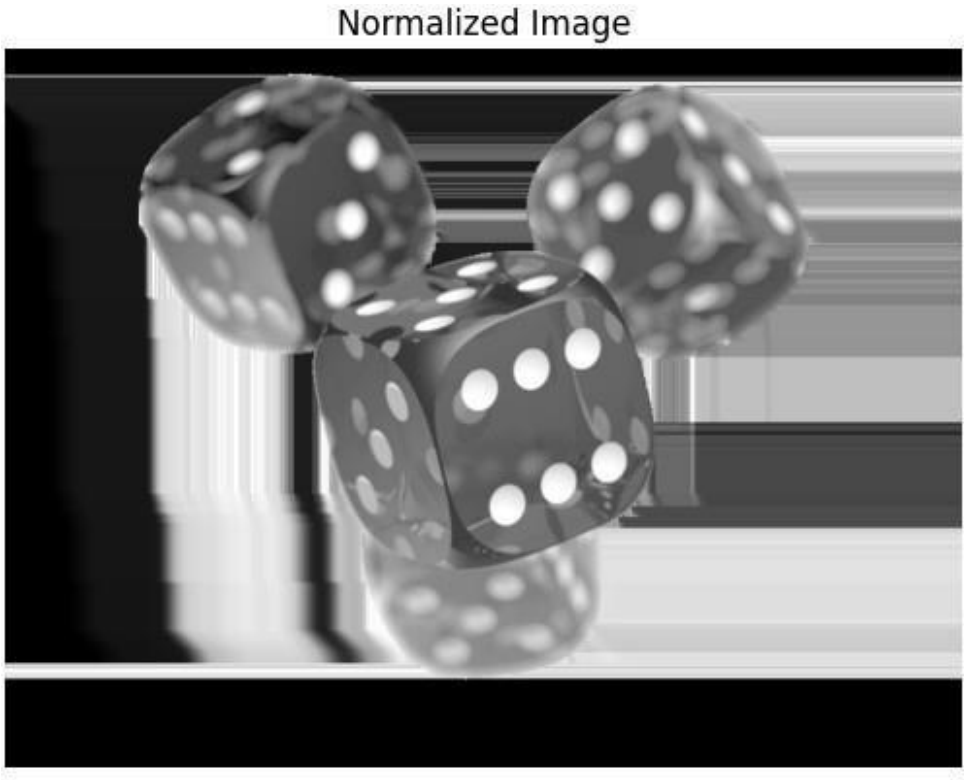
- Mengembalikan daftar wajah yang terdeteksi dalam format koordinat (`x, y, w, h`):
  - **x, y**: Koordinat sudut kiri atas wajah.
  - **w, h**: Lebar dan tinggi area wajah.
- Fungsi **cv2.rectangle**:
  - Digunakan untuk menggambar persegi panjang pada area wajah yang terdeteksi.
- Parameter:
  a. **image_rgb**:
     - Gambar berwarna di mana persegi panjang akan digambar.
  b. **(x, y)**:
     - Koordinat sudut kiri atas wajah.
  c. **(x+w, y+h)**:
     - Koordinat sudut kanan bawah wajah.
  d. **(255, 0, 0)**:
     - Warna persegi panjang dalam format RGB (biru pada OpenCV).
  e. **2**:
     - Ketebalan garis persegi panjang (2 piksel).
- **plt.imshow(image_rgb)**:
  - Menampilkan gambar berwarna (RGB) yang telah diberi persegi panjang pada wajah yang terdeteksi.
- **plt.title("Wajah Terdeteksi")**:
  - Menambahkan judul pada gambar.
- **plt.axis('off')**:
  - Menghilangkan sumbu agar tampilan lebih bersih.
- **plt.show()**:
  - Menampilkan gambar dengan persegi panjang di sekitar wajah.

1. Pre-processing for Computer Vision Tasks Langkah awal seperti pengubahan ukuran,

```
# Normalisasi gambar grayscale
normalized_image = gray_image / 255.0

# Menampilkan hasil normalisasi
plt.imshow(normalized_image, cmap='gray')
```

normalisasi, atau perbaikan kontras gambar untuk mempersiapkan data.

```
plt.title("Normalized Image")
plt.axis('off')
plt.show()
```



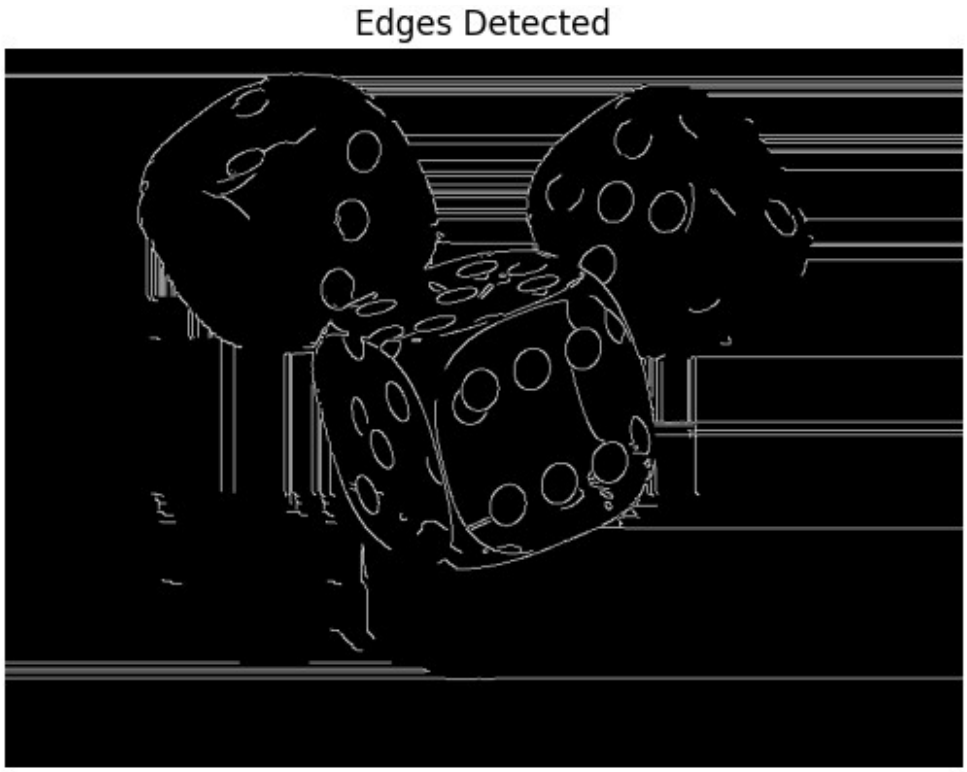
- **gray_image**:
  - Merupakan gambar dalam format grayscale, yaitu matriks piksel 2D dengan nilai intensitas antara 0 (hitam) hingga 255 (putih).
- **plt.imshow**:
  - Fungsi dari pustaka Matplotlib untuk menampilkan gambar.
  - Parameter:
    - **normalized_image**:
      - Gambar hasil normalisasi yang akan ditampilkan.
      - Memiliki nilai piksel dalam rentang 0 hingga 1.
    - **cmap='gray'**:
      - Menentukan skema warna untuk menampilkan gambar.
      - 'gray' memastikan gambar ditampilkan dalam skala abu-abu (grayscale), di mana:
        - Nilai 0 ditampilkan sebagai hitam.
        - Nilai 1 ditampilkan sebagai putih.
- **plt.title**:
  - Menambahkan judul di atas gambar.
  - Judulnya adalah "Normalized Image", untuk menunjukkan bahwa gambar telah melalui proses normalisasi.

- **plt.axis('off')**:
  - Menghapus tampilan sumbu (axis) pada gambar.
  - Hal ini bertujuan untuk memberikan fokus sepenuhnya pada gambar tanpa gangguan sumbu X dan Y.

1. Feature Description Proses mengekstraksi fitur penting dari gambar, seperti sudut atau

```
#cv2.Canny: Menggunakan algoritma Canny untuk mendeteksi tepi
# Deteksi tepi menggunakan algoritma Canny
edges = cv2.Canny(gray_image, 100, 200)

# Menampilkan tepi
plt.imshow(edges, cmap='gray')
plt.title("Edges Detected")
plt.axis('off')
plt.show()
```

tepi, untuk digunakan dalam analisis lebih lanjut.



edges = cv2.Canny(gray_image, 100, 200)

- **cv2.Canny**adalah fungsi dari OpenCV untuk mendeteksi tepi (edges) dalam gambar.
- Algoritma Canny merupakan metode populer untuk deteksi tepi karena bekerja dengan baik dalam kondisi cahaya bervariasi dan menghasilkan hasil yang tajam.

1. **gray_image**:
   - Input gambar dalam format grayscale.

- Algoritma Canny membutuhkan gambar grayscale untuk menghitung intensitas gradien piksel.

2. **100**(Threshold Bawah):
   - Nilai ambang bawah untuk mendeteksi gradien.
   - Piksel dengan intensitas gradien di bawah nilai ini akan dianggap bukan tepi.
3. **200**(Threshold Atas):
   - Nilai ambang atas untuk mendeteksi gradien.
   - Piksel dengan intensitas gradien di atas nilai ini langsung dianggap tepi.
4. Hasil **edges**:
   - Array biner (nilai piksel 0 atau 255):
     - 0 (hitam): Bukan tepi.
     - 255 (putih): Deteksi sebagai tepi.

- **plt.imshow**digunakan untuk menampilkan array gambar hasil deteksi tepi.
- Parameter:
  - **edges**: Array biner hasil dari fungsi cv2.Canny.
  - **cmap='gray'**:
    - Skema warna abu-abu (grayscale) digunakan untuk menampilkan gambar.
    - Nilai piksel 0 (hitam) menunjukkan area tanpa tepi, sedangkan 255 (putih) menunjukkan tepi yang terdeteksi.

Fungsi **plt.title**

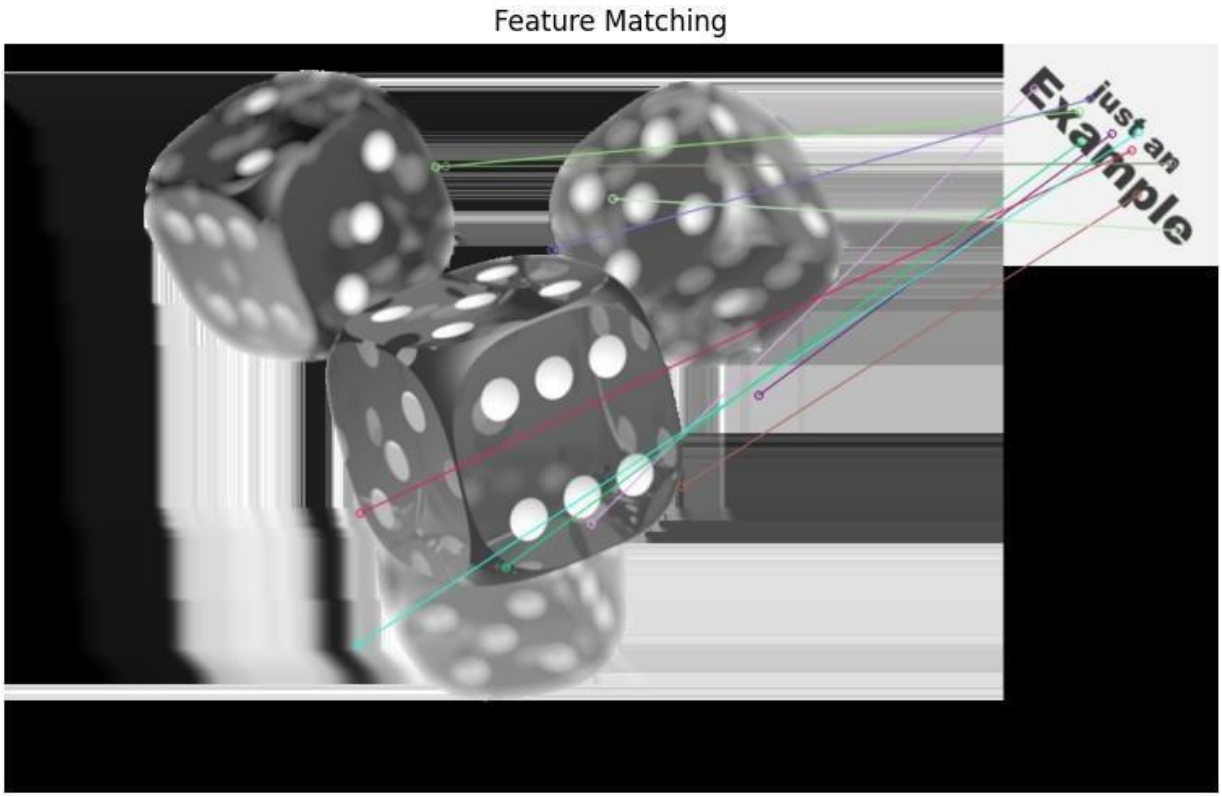
- Menambahkan judul "Edges Detected" di atas gambar untuk menjelaskan bahwa gambar menunjukkan hasil deteksi tepi.

1. Feature Matching Mencocokkan fitur dari dua gambar yang berbeda, seperti dalam

```
import cv2
import numpy as np
import requests
import matplotlib.pyplot as plt

# Fungsi untuk mengunduh gambar dari web
def download_image(url):
    response = requests.get(url)
    if response.status_code == 200:
        # Mengubah data biner menjadi array numpy
        image_array = np.asarray(bytearray(response.content),
dtype=np.uint8)
        # Membaca gambar dari array numpy
        return cv2.imdecode(image_array, cv2.IMREAD_GRAYSCALE) #
Grayscale
    else:
        print(f"Gagal mengunduh gambar dari URL: {url}")
        return None

# URL gambar
url1 =
```

pencocokan gambar atau pengenalan objek.

```
"https://upload.wikimedia.org/wikipedia/commons/4/47/PNG_transparency_
demonstration_1.png"          # Ganti dengan URL gambar pertama
url2     =  "https://upload.wikimedia.org/wikipedia/commons/a/a9/Example.jpg"            #
Ganti dengan URL gambar kedua

# Mengunduh kedua gambar image1 =
download_image(url1) image2 =
download_image(url2)

# Pastikan kedua gambar berhasil diunduh
if image1 is not None and image2 is not None:
      # Detektor SIFT
      sift = cv2.SIFT_create()

      # Deteksi dan deskripsi fitur
      keypoints1, descriptors1 = sift.detectAndCompute(image1, None) keypoints2,
      descriptors2   =   sift.detectAndCompute(image2,    None)

      # Mencocokkan fitur menggunakan BFMatcher
      bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
      matches = bf.match(descriptors1, descriptors2) matches =
      sorted(matches, key=lambda x: x.distance)

      # Gambar hasil pencocokan
      matched_image = cv2.drawMatches(
            image1, keypoints1, image2, keypoints2, matches[:10], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS
      )

      #Menampilkanhasil plt.figure(figsize=(12,
      6)) plt.imshow(matched_image, cmap='gray')
      plt.title("Feature Matching") plt.axis('off')
      plt.show() else:
      print("Gagal memproses gambar. Periksa URL gambar atau koneksi
internet.")
```



- **cv2(OpenCV)**:
  - Digunakan untuk pengolahan gambar, seperti deteksi fitur dan pencocokan.
- **numpy**:
  - Membantu dalam representasi data gambar sebagai array numerik.
- **requests**:
  - Digunakan untuk mengunduh gambar dari URL.
- **matplotlib.pyplot**:
  - Untuk menampilkan gambar hasil pengolahan.
- **requests.get(url)**:
  - Mengirim permintaan HTTP GET ke URL untuk mengunduh gambar.
- **response.status_code**:
  - Memeriksa apakah permintaan berhasil. **200**berarti unduhan berhasil.
- **np.asarray(bytearray(response.content), dtype=np.uint8)**:
  - Mengubah data biner dari gambar menjadi array NumPy.
- **cv2.imdecode**:
  - Membaca gambar dari array NumPy dalam format grayscale (**cv2.IMREAD_GRAYSCALE**).

    - Grayscale digunakan karena algoritma SIFT bekerja lebih efisien pada gambar grayscale.
- Jika gagal mengunduh, fungsi mencetak pesan kesalahan dan mengembalikan
- **cv2.SIFT_create**:
    - Membuat detektor dan deskriptor fitur SIFT (Scale-Invariant Feature Transform).
- **sift.detectAndCompute**:
    - Deteksi Fitur:
        - Menemukan titik-titik penting (keypoints) dalam gambar.
    - Deskripsi Fitur:
        - Membuat representasi numerik (deskriptor) untuk setiap keypoint.
- **Hasil**:
    - **keypoints**: Daftar objek keypoint (posisi dan orientasi fitur).
    - **descriptors**: Array NumPy yang merepresentasikan deskriptor setiap keypoint.

bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True) matches = bf.match(descriptors1, descriptors2) matches = sorted(matches, key=lambda x: x.distance)

- **cv2.BFMatcher**:
    - BFMatcher (Brute-Force Matcher) mencocokkan deskriptor dari dua gambar.
- Parameter:
    - **cv2.NORM_L2**:
        - Menggunakan norma L2 (jarak Euclidean) untuk membandingkan deskriptor.
    - **crossCheck=True**:
        - Memastikan pencocokan dua arah (descriptor A cocok dengan descriptor B, dan sebaliknya).
- **bf.match**:
    - Mencocokkan deskriptor dari dua gambar.
- **sorted(matches, key=lambda x: x.distance)**:
    - Mengurutkan hasil pencocokan berdasarkan jarak (distance).
    - Jarak lebih kecil menunjukkan kecocokan lebih baik.
- **plt.figure(figsize=(12, 6))**:
    - Mengatur ukuran jendela Matplotlib.
- **plt.imshow(matched_image, cmap='gray')**:
    - Menampilkan gambar hasil pencocokan dalam skema warna grayscale.
- **plt.title("Feature Matching")**:
    - Menambahkan judul gambar.
- **plt.axis('off')**:
    - Menghapus sumbu untuk tampilan yang lebih bersih.
- **plt.show()**:

- Menampilkan gambar pada jendela Matplotlib.

1. Real-world Applications of Feature Extraction in Computer Vision Subtopik ini mengeksplorasi bagaimana fitur yang diekstraksi digunakan dalam aplikasi dunia nyata, seperti augmented reality atau pemetaan 3D.

UNIT 2

Convolutional Neural Networks (CNNs) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk memproses data dengan struktur grid, seperti gambar. CNN telah menjadi tulang punggung dari banyak aplikasi computer vision modern, termasuk klasifikasi gambar, deteksi objek, segmentasi, dan banyak lagi.

1. Introduction to Convolutional Neural Networks CNN adalah jenis jaringan saraf tiruan yang dirancang untuk menganalisis data dengan struktur grid seperti gambar. CNN menggunakan lapisan konvolusi untuk mengekstrak fitur dari data.

```
#Conv2D: Lapisan konvolusi untuk mengekstraksi fitur gambar. #MaxPooling2D: Mengurangi dimensi data untuk mempercepat proses. #Flatten: Mengubah data 2D menjadi 1D sebelum masuk ke lapisan dense. #Dense: Lapisan fully connected untuk klasifikasi.

import tensorflow as tf
from tensorflow.keras import layers, models

# Definisi model CNN
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),  # Lapisankonvolusi
    layers.MaxPooling2D((2, 2)),        #Lapisanpooling
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),           # Flatten untuk input ke lapisan dense layers.Dense(64,
    activation='relu'),         # Lapisan fully connected layers.Dense(10,
    activation='softmax')  # Outputuntuk 10 kelas
])

# Kompilasi model
model.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Ringkasan model
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 26, 26, 32)          │             320 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 13, 13, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 11, 11, 64)          │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 5, 5, 64)            │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 1600)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 64)                  │         102,464 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 10)                  │             650 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 121,930 (476.29 KB)
 Trainable params: 121,930 (476.29 KB)
 Non-trainable params: 0 (0.00 B)
```

- **tensorflow**:
  - Pustaka machine learning untuk membangun dan melatih model neural network.
- **layers**dan **models**:
  - **layers**: Digunakan untuk mendefinisikan lapisan (layers) dalam model.
  - **models**: Membantu dalam membuat struktur model.
- **Conv2D(32, (3, 3))**:

  - Lapisan konvolusi dengan 32 filter, masing-masing memiliki ukuran kernel 3x3.
  - Fungsi utama:
    - Mengekstrak fitur lokal dari gambar input.
- **activation='relu'**:
  - Fungsi aktivasi ReLU (Rectified Linear Unit) diterapkan untuk membuat model non-linear.
  - Fungsi ReLU: $f(x) = \max(0, x)$
- **input_shape=(28, 28, 1)**:
  - Bentuk input ke model:
    - 28x28: Dimensi gambar.
    - 1: Saluran (channel) karena input berupa grayscale (1 channel).
- **MaxPooling2D((2, 2))**:
  - Lapisan pooling dengan ukuran 2x2.
  - Fungsi utama:
    - Mengurangi dimensi data (downsampling) untuk mengurangi kompleksitas.
    - Hanya mempertahankan nilai maksimum dari area 2x2.
- **Flatten**:
  - Mengubah data multi-dimensi (2D atau 3D) menjadi vektor 1D.
  - Diperlukan untuk menghubungkan lapisan konvolusi/pooling ke lapisan fully connected.
- **Dense(64)**:
  - Lapisan fully connected dengan 64 neuron.
  - Setiap neuron terhubung ke semua neuron di lapisan sebelumnya.
- **activation='relu'**:
  - Fungsi aktivasi ReLU untuk menambah non-linearitas.
- **Dense(10)**:
  - Lapisan output dengan 10 neuron (sesuai dengan jumlah kelas pada tugas klasifikasi).
- **activation='softmax'**:
  - Fungsi aktivasi Softmax:
    - Mengubah output menjadi distribusi probabilitas untuk setiap kelas.
    - Probabilitas total akan bernilai 1.

1. **optimizer='adam'**:
   - Optimizer Adam digunakan untuk memperbarui bobot selama pelatihan.
   - Keunggulan:
     - Kombinasi dari algoritma Momentum dan RMSProp.
     - Cepat dan efisien dalam mengonvergensi.
2. **loss='sparse_categorical_crossentropy'**:
   - Fungsi loss untuk klasifikasi multi-kelas.
   - Digunakan saat label target berupa integer (bukan one-hot encoded).
3. **metrics=['accuracy']**:
   - Metode evaluasi model.

- Akurasi digunakan untuk mengukur performa model pada data pelatihan/validasi. mmary()

- **model.summary**:
  - Menampilkan detail arsitektur model:
    - Nama lapisan.
    - Bentuk output dari setiap lapisan.
    - Jumlah parameter yang dapat dilatih (trainable parameters).

1. Very Deep Convolutional Networks for Large Scale Image Recognition Ini merujuk pada arsitektur jaringan dalam seperti VGGNet, yang terdiri dari banyak lapisan konvolusi.

```
#VGG16: Arsitektur CNN yang populer dengan 16 lapisan. #weights='imagenet': Memuat bobot yang telah dilatih pada dataset ImageNet.
from tensorflow.keras.applications import VGG16

# Memuat model VGG16 dengan bobot ImageNet
vgg_model = VGG16(weights='imagenet', input_shape=(224, 224, 3))

# Menampilkan arsitektur model
vgg_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5 553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 24s 0us/step
```

```
Model: "vgg16"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ input_layer_1 (InputLayer)      │ (None, 224, 224, 3)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ block1_conv1 (Conv2D)           │ (None, 224, 224, 64)   │         1,792 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ block1_conv2 (Conv2D)           │ (None, 224, 224, 64)   │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ block1_pool (MaxPooling2D)      │ (None, 112, 112, 64)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ block2_conv1 (Conv2D)           │ (None, 112, 112, 128)  │
```

```
73,856 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block2_conv2 (Conv2D)           | (None, 112, 112, 128)  |
147,584 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block2_pool (MaxPooling2D)      | (None, 56, 56, 128)    |
0 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block3_conv1 (Conv2D)           | (None, 56, 56, 256)    |
295,168 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block3_conv2 (Conv2D)           | (None, 56, 56, 256)    |
590,080 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block3_conv3 (Conv2D)           | (None, 56, 56, 256)    |
590,080 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block3_pool (MaxPooling2D)      | (None, 28, 28, 256)    |
0 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block4_conv1 (Conv2D)           | (None, 28, 28, 512)    |
1,180,160 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block4_conv2 (Conv2D)           | (None, 28, 28, 512)    |
2,359,808 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block4_conv3 (Conv2D)           | (None, 28, 28, 512)    |
2,359,808 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block4_pool (MaxPooling2D)      | (None, 14, 14, 512)    |
0 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block5_conv1 (Conv2D)           | (None, 14, 14, 512)    |
2,359,808 |
├─────────────────────────────────┼────────────────────────┼──
──────────┤
| block5_conv2 (Conv2D)           | (None, 14, 14, 512)    |
2,359,808 |
```

```
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ block5_conv3 (Conv2D)           ┃ (None, 14, 14, 512)    ┃
2,359,808 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ block5_pool (MaxPooling2D)      ┃ (None, 7, 7, 512)      ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ flatten (Flatten)               ┃ (None, 25088)          ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ fc1 (Dense)                     ┃ (None, 4096)           ┃
102,764,544 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ fc2 (Dense)                     ┃ (None, 4096)           ┃
16,781,312 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━┫
━━━━━━━━━━━━━━━┫
┃ predictions (Dense)             ┃ (None, 1000)           ┃
4,097,000 ┃
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━┻
━━━━━━━━━━━━━━━┛

 Total params: 138,357,544 (527.79 MB)

 Trainable params: 138,357,544 (527.79 MB)

 Non-trainable params: 0 (0.00 B)
```

- **tensorflow.keras.applications**:
  - Modul ini menyediakan berbagai model deep learning yang telah dilatih sebelumnya pada dataset ImageNet, termasuk VGG16.
- **VGG16**:
  - Model arsitektur convolutional neural network (CNN) yang terkenal, dirancang oleh Visual Geometry Group.
  - Digunakan untuk tugas klasifikasi gambar, dengan input gambar berukuran 224x224x3 (warna/RGB).

1. **weights='imagenet'**:
   - Bobot model telah dilatih sebelumnya menggunakan dataset ImageNet.
   - ImageNet adalah dataset besar dengan lebih dari 1 juta gambar, mencakup 1.000 kelas objek.
   - Dengan bobot pretrained:
     - Model dapat digunakan langsung untuk klasifikasi pada kelas ImageNet.

- Dapat digunakan untuk transfer learning pada dataset baru dengan kelas yang berbeda.

2. **input_shape=(224, 224, 3)**:
   - Menentukan bentuk input gambar:
     - 224x224: Dimensi spatial gambar.
     - 3: Saluran warna (RGB).
   - Input ini sesuai dengan persyaratan model VGG16.

- Menampilkan detail arsitektur model dalam format tabel.
- Informasi yang ditampilkan meliputi:
  a. Nama Lapisan:
     - Setiap lapisan dalam arsitektur model.
  b. Output Shape:
     - Dimensi output setelah melalui setiap lapisan.
  c. Param #:
     - Jumlah parameter yang dapat dilatih (trainable parameters) dan parameter tetap (non-trainable parameters).
  d. Total Params:
     - Jumlah total parameter dalam model.

1. GoogLeNet Arsitektur ini menggunakan Inception Module untuk mengurangi jumlah parameter sambil mempertahankan akurasi tinggi.

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, concatenate,
Input
from tensorflow.keras.models import Model

# Fungsi untuk Inception Module
def inception_module(x):
    conv1 = Conv2D(64, (1, 1), activation='relu')(x)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(x) conv5
    = Conv2D(32, (5, 5), activation='relu', padding='same')(x) pool =
    MaxPooling2D((3, 3), strides=(1, 1), padding='same')(x) return
    concatenate([conv1, conv3, conv5, pool], axis=-1)

# Input data
input_layer = Input(shape=(224, 224, 3)) output_layer
= inception_module(input_layer)

# Model
model = Model(inputs=input_layer, outputs=output_layer) model.summary()
```

```
Model: "functional_1"
┌─────────────────────────────┬──────────────────────────┬──────────────────
─
┐
│ Layer (type)                │ Output Shape             │          Param #
│ Connected to                │
├─────────────────────────────┼──────────────────────────┼──────────────────
─
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ input_layer_2 (InputLayer)      ┃ (None, 224, 224, 3)    ┃             0 ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d_2 (Conv2D)               │ (None, 224, 224, 64)   │           256 │
│ input_layer_2[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 224, 224, 128)  │         3,584 │
│ input_layer_2[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 224, 224, 32)   │         2,432 │
│ input_layer_2[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 224, 224, 3)    │             0 │
│ input_layer_2[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ concatenate (Concatenate)       │ (None, 224, 224, 227)  │             0 │
│ conv2d_2[0][0],                 │                        │               │
│ conv2d_3[0][0],                 │                        │               │
│ conv2d_4[0][0],                 │                        │               │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

Total params: 6,272 (24.50 KB)

Trainable params: 6,272 (24.50 KB)

Non-trainable params: 0 (0.00 B)

- **Conv2D**:
  - Lapisan konvolusi 2D untuk mengekstraksi fitur lokal dari gambar.
- **MaxPooling2D**:
  - Lapisan pooling untuk downsampling, mengurangi dimensi data sambil mempertahankan informasi penting.
- **concatenate**:

  - Digunakan untuk menggabungkan output dari berbagai jalur (branches) pada Inception Module.
- **Input**:
  - Menentukan bentuk input ke model.
- **Model**:
  - Membuat model dengan menentukan input dan output.
  - Menggunakan **kernel 1x1**:
    - Meningkatkan kedalaman (depth) jaringan.
    - Mengurangi dimensi channel sebelumnya (dimensionality reduction).
  - Menggunakan **kernel 3x3**:
    - Mengekstraksi fitur lokal.
    - Padding 'same' memastikan dimensi output tetap sama dengan input.
  - Menggunakan **kernel 5x5**:
    - Menangkap fitur dengan area reseptif lebih besar.
    - Padding 'same' memastikan dimensi output tetap sama.
  - Pooling dengan **kernel 3x3** dan stride **1x1**:
    - Menjaga dimensi output tetap sama dengan input (karena padding 'same').
    - Meningkatkan keberagaman fitur yang diekstraksi.
  - Menggabungkan output dari keempat branch di sepanjang **channel axis (-1)**.
  - Tujuannya adalah menggabungkan fitur dari berbagai skala ke dalam satu tensor.

- **Input(shape=(224, 224, 3))**:
  - Menentukan bentuk input:
    - 224x224: Dimensi spatial gambar.
    - 3: Saluran warna (RGB).
  - Tensor input ini digunakan sebagai input awal ke model.
- **inception_module(input_layer)**:
  - Fungsi inception_module dipanggil dengan **input_layer**sebagai argumen.
  - Output dari Inception Module menjadi output dari lapisan berikutnya.
- **Model(inputs=input_layer, outputs=output_layer)**:
  - Membuat model dengan:
    - **input_layer**sebagai input.
    - **output_layer**(output dari Inception Module) sebagai output.
- **model.summary()**:
  - Menampilkan detail arsitektur model, termasuk:
    - Nama setiap lapisan.
    - Dimensi output dari setiap lapisan.

- Jumlah parameter yang dapat dilatih pada setiap lapisan.

1. MobileNet MobileNet adalah arsitektur CNN ringan yang dirancang untuk perangkat dengan sumber daya terbatas.

```
from tensorflow.keras.applications import MobileNet

# Memuat MobileNet dengan bobot ImageNet
mobilenet_model = MobileNet(weights='imagenet', input_shape=(224, 224, 3))

# Menampilkan arsitektur model
mobilenet_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf.h5
17225924/17225924 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Model: "mobilenet_1.00_224"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ input_layer_3 (InputLayer)           │ (None, 224, 224, 3)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv1 (Conv2D)                       │ (None, 112, 112, 32)        │             864 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv1_bn (BatchNormalization)        │ (None, 112, 112, 32)        │             128 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv1_relu (ReLU)                    │ (None, 112, 112, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv_dw_1 (DepthwiseConv2D)          │ (None, 112, 112, 32)        │             288 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv_dw_1_bn (BatchNormalization)    │ (None, 112, 112, 32)        │             128 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv_dw_1_relu (ReLU)                │ (None, 112, 112, 32)        │               0 │
```

```
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_1 (Conv2D)                   | (None, 112, 112, 64)         |
2,048 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_1_bn (BatchNormalization)    | (None, 112, 112, 64)         |
256 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_1_relu (ReLU)                | (None, 112, 112, 64)         |
0 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pad_2 (ZeroPadding2D)           | (None, 113, 113, 64)         |
0 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_dw_2 (DepthwiseConv2D)          | (None, 56, 56, 64)           |
576 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_dw_2_bn (BatchNormalization)    | (None, 56, 56, 64)           |
256 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_dw_2_relu (ReLU)                | (None, 56, 56, 64)           |
0 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_2 (Conv2D)                   | (None, 56, 56, 128)          |
8,192 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_2_bn (BatchNormalization)    | (None, 56, 56, 128)          |
512 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_pw_2_relu (ReLU)                | (None, 56, 56, 128)          |
0 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_dw_3 (DepthwiseConv2D)          | (None, 56, 56, 128)          |
1,152 |
├──────────────────────────────────────┼──────────────────────────────┼─
──────────────┤
| conv_dw_3_bn (BatchNormalization)    | (None, 56, 56, 128)          |
512 |
├──────────────────────────────────────┼──────────────────────────────┼─
```

```
──────────────────┤
 conv_dw_3_relu (ReLU)                  (None, 56, 56, 128)          
0 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_3 (Conv2D)                     (None, 56, 56, 128)          
16,384 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_3_bn (BatchNormalization)      (None, 56, 56, 128)          
512 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_3_relu (ReLU)                  (None, 56, 56, 128)          
0 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pad_4 (ZeroPadding2D)             (None, 57, 57, 128)          
0 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_dw_4 (DepthwiseConv2D)            (None, 28, 28, 128)          
1,152 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_dw_4_bn (BatchNormalization)      (None, 28, 28, 128)          
512 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_dw_4_relu (ReLU)                  (None, 28, 28, 128)          
0 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_4 (Conv2D)                     (None, 28, 28, 256)          
32,768 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_4_bn (BatchNormalization)      (None, 28, 28, 256)          
1,024 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_pw_4_relu (ReLU)                  (None, 28, 28, 256)          
0 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
 conv_dw_5 (DepthwiseConv2D)            (None, 28, 28, 256)          
2,304 
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
```

```
| conv_dw_5_bn (BatchNormalization)          | (None, 28, 28, 256)       |
1,024 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_dw_5_relu (ReLU)                      | (None, 28, 28, 256)       |
0 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_5 (Conv2D)                         | (None, 28, 28, 256)       |
65,536 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_5_bn (BatchNormalization)          | (None, 28, 28, 256)       |
1,024 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_5_relu (ReLU)                      | (None, 28, 28, 256)       |
0 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pad_6 (ZeroPadding2D)                 | (None, 29, 29, 256)       |
0 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_dw_6 (DepthwiseConv2D)                | (None, 14, 14, 256)       |
2,304 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_dw_6_bn (BatchNormalization)          | (None, 14, 14, 256)       |
1,024 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_dw_6_relu (ReLU)                      | (None, 14, 14, 256)       |
0 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_6 (Conv2D)                         | (None, 14, 14, 512)       |
131,072 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_6_bn (BatchNormalization)          | (None, 14, 14, 512)       |
2,048 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_pw_6_relu (ReLU)                      | (None, 14, 14, 512)       |
0 |
├────────────────────────────────────────────┼──────────────────────────┼
──────────────┤
| conv_dw_7 (DepthwiseConv2D)                | (None, 14, 14, 512)       |
```

```
4,608 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_dw_7_bn (BatchNormalization)      | (None, 14, 14, 512)          |
2,048 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
|  conv_dw_7_relu (ReLU)                 |  (None, 14, 14, 512)         |
0 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_pw_7 (Conv2D)                     |  (None, 14, 14, 512)         |
262,144 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_pw_7_bn (BatchNormalization)      | (None, 14, 14, 512)          |
2,048 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
|  conv_pw_7_relu (ReLU)                 |  (None, 14, 14, 512)         |
0 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_dw_8 (DepthwiseConv2D)            |  (None, 14, 14, 512)         |
4,608 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_dw_8_bn (BatchNormalization)      | (None, 14, 14, 512)          |
2,048 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
|  conv_dw_8_relu (ReLU)                 |  (None, 14, 14, 512)         |
0 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_pw_8 (Conv2D)                     |  (None, 14, 14, 512)         |
262,144 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_pw_8_bn (BatchNormalization)      | (None, 14, 14, 512)          |
2,048 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
|  conv_pw_8_relu (ReLU)                 |  (None, 14, 14, 512)         |
0 |
├──────────────────────────────────┼──────────────────────────────┼
──────────────┤
| conv_dw_9 (DepthwiseConv2D)            |  (None, 14, 14, 512)         |
4,608 |
```

conv_dw_9_bn (BatchNormalization)	(None, 14, 14, 512)	2,048
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262,144
conv_pw_9_bn (BatchNormalization)	(None, 14, 14, 512)	2,048
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4,608
conv_dw_10_bn (BatchNormalization)	(None, 14, 14, 512)	2,048
conv_dw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_10 (Conv2D)	(None, 14, 14, 512)	262,144
conv_pw_10_bn (BatchNormalization)	(None, 14, 14, 512)	2,048
conv_pw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 14, 14, 512)	4,608

```
┃ conv_dw_11_bn (BatchNormalization)  ┃ (None, 14, 14, 512)  ┃ 2,048 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_dw_11_relu (ReLU)              ┃ (None, 14, 14, 512)  ┃ 0     ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_11 (Conv2D)                 ┃ (None, 14, 14, 512)  ┃ 262,144 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_11_bn (BatchNormalization)  ┃ (None, 14, 14, 512)  ┃ 2,048 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_11_relu (ReLU)              ┃ (None, 14, 14, 512)  ┃ 0     ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pad_12 (ZeroPadding2D)         ┃ (None, 15, 15, 512)  ┃ 0     ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_dw_12 (DepthwiseConv2D)        ┃ (None, 7, 7, 512)    ┃ 4,608 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_dw_12_bn (BatchNormalization)  ┃ (None, 7, 7, 512)    ┃ 2,048 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_dw_12_relu (ReLU)              ┃ (None, 7, 7, 512)    ┃ 0     ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_12 (Conv2D)                 ┃ (None, 7, 7, 1024)   ┃ 524,288 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_12_bn (BatchNormalization)  ┃ (None, 7, 7, 1024)   ┃ 4,096 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
┃ conv_pw_12_relu (ReLU)              ┃ (None, 7, 7, 1024)   ┃ 0     ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━┫
```

conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9,216
conv_dw_13_bn (BatchNormalization)	(None, 7, 7, 1024)	4,096
conv_dw_13_relu (ReLU)	(None, 7, 7, 1024)	0
conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1,048,576
conv_pw_13_bn (BatchNormalization)	(None, 7, 7, 1024)	4,096
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0
global_averag e_pooling2d (GlobalAverageJePooling2D)	(None, 1, 1, 1024)	0
dropout (Dropout)	(None, 1, 1, 1024)	0
conv_preds (Conv2D)	(None, 1, 1, 1000)	1,025,000
reshape_2 (Reshape)	(None, 1000)	0
predictions (Activation)	(None, 1000)	0

Total params: 4,253,864 (16.23 MB)

Trainable params: 4,231,976 (16.14 MB)

```
Non-trainable params: 21,888 (85.50 KB)
```

- **MobileNet**:
  - Salah satu arsitektur deep learning yang dioptimalkan untuk perangkat dengan sumber daya terbatas (seperti ponsel).
  - Menggunakan depthwise separable convolutions untuk mengurangi jumlah parameter sambil mempertahankan akurasi tinggi.
  - Cocok untuk tugas klasifikasi gambar dengan input gambar 224x224x3.

1. **weights='imagenet'**:
   - Model menggunakan bobot pretrained yang telah dilatih pada dataset ImageNet.
   - ImageNet adalah dataset besar yang berisi lebih dari 1 juta gambar dengan 1.000 kelas objek.
   - Dengan pretrained weights:
     - Model dapat digunakan langsung untuk klasifikasi gambar pada kelas-kelas ImageNet.
     - Dapat digunakan untuk transfer learning pada dataset dengan kelas berbeda.
2. **input_shape=(224, 224, 3)**:
   - Menentukan ukuran input gambar:
     - 224x224: Dimensi spatial gambar.
     - 3: Saluran warna (RGB).
   - Input ini sesuai dengan arsitektur MobileNet.
   - Jika tidak ditentukan, ukuran default MobileNet adalah 224x224x3.

- Menampilkan struktur arsitektur model dalam format tabel.
- Informasi yang ditampilkan:
  a. Nama Lapisan:
     - Nama setiap lapisan dalam model.
  b. Output Shape:
     - Dimensi output dari setiap lapisan.
  c. Param #:
     - Jumlah parameter yang dapat dilatih (trainable parameters) dan parameter tetap (non-trainable parameters).
  d. Total Params:
     - Total parameter yang digunakan dalam model.
- Mengganti konvolusi standar dengan dua operasi:
  a. Depthwise Convolution:
     - Operasi konvolusi diterapkan secara terpisah pada setiap saluran (channel).
     - Mengurangi jumlah perhitungan.
  b. Pointwise Convolution:
     - Operasi konvolusi menggunakan kernel 1x1 untuk menggabungkan informasi antar saluran.
     - Memastikan hasil tetap kompatibel dengan lapisan berikutnya.
- ReLU6 (Rectified Linear Unit hingga 6) digunakan sebagai fungsi aktivasi:

- Fungsi ini membatasi output hingga maksimum 6.
- Membantu menjaga stabilitas numerik pada perangkat dengan presisi rendah (seperti ponsel).

1. ConvNext - A ConvNet for the 2020s (2022) ConvNext adalah arsitektur CNN modern yang dioptimalkan untuk performa tinggi dengan lebih sedikit parameter.

```
from tensorflow.keras.applications import ConvNeXtTiny

# Memuat model ConvNextTiny dengan bobot ImageNet
convnext_model = ConvNeXtTiny(weights='imagenet', input_shape=(224, 224, 3))

# Menampilkan arsitektur model
convnext_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/convnext/convnext_tiny.h5
114735104/114735104 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
Model: "convnext_tiny"
```

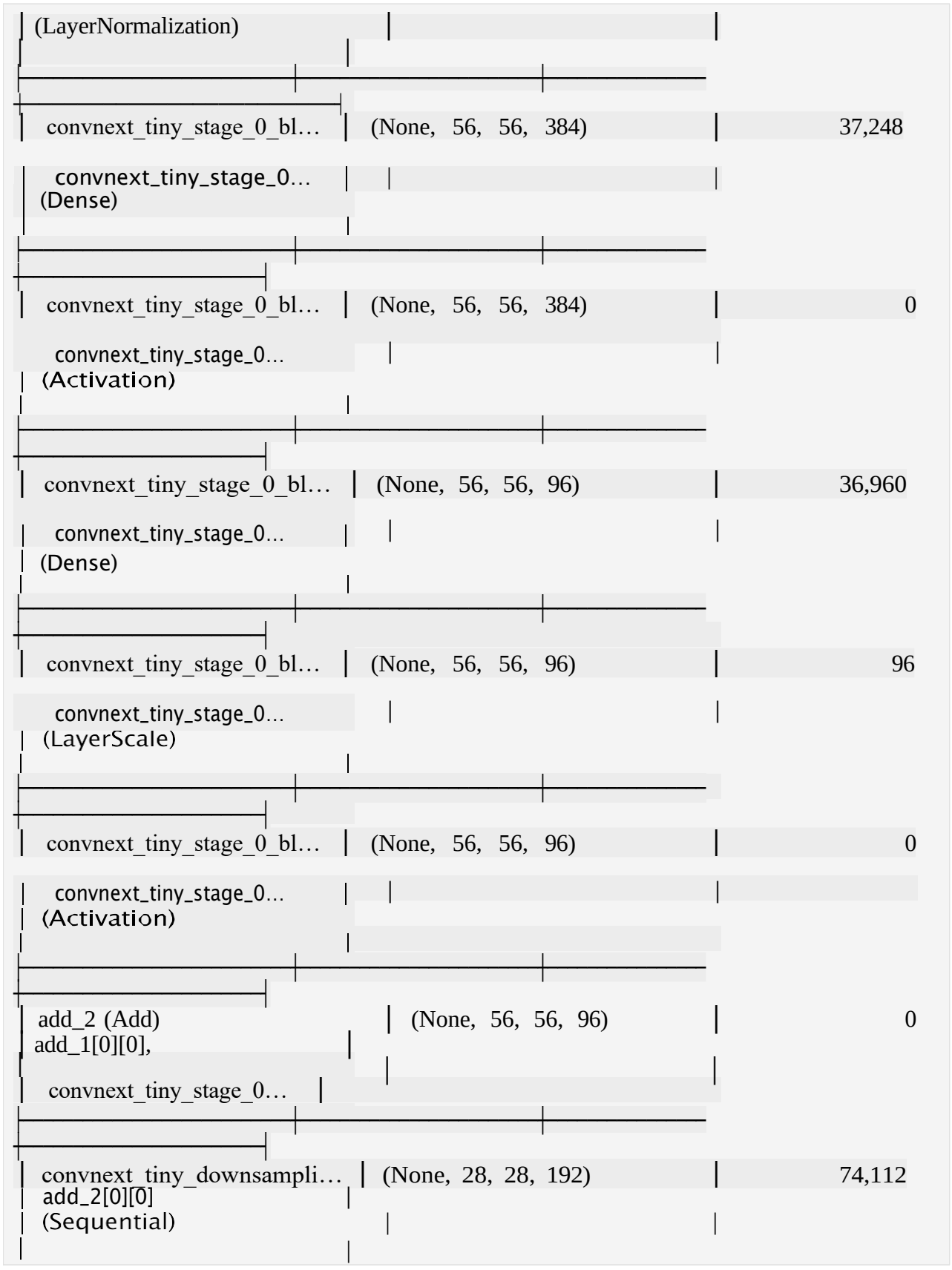
Layer (type)	Output Shape	Param #	Connected to
input_layer_4 (InputLayer)	(None, 224, 224, 3)	0	-
convnext_tiny_prestem_no… (Normalization)	(None, 224, 224, 3)	0	input_layer_4[0][0]
convnext_tiny_stem (Sequential)	(None, 56, 56, 96)	4,896	convnext_tiny_prestem…
convnext_tiny_stage_0_bl… (Conv2D)	(None, 56, 56, 96)	4,800	convnext_tiny_stem[0]…

convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	192
convnext_tiny_stage_0… (LayerNormalization)		
convnext_tiny_stage_0_bl…	(None, 56, 56, 384)	37,248
convnext_tiny_stage_0… (Dense)		
convnext_tiny_stage_0_bl…	(None, 56, 56, 384)	0
convnext_tiny_stage_0… (Activation)		
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	36,960
convnext_tiny_stage_0… (Dense)		
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	96
convnext_tiny_stage_0… (LayerScale)		
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	0
convnext_tiny_stage_0… (Activation)		
add (Add) convnext_tiny_ stem[0]… convnext_tiny_	(None, 56, 56, 96)	0
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	4,800
add[0][0] (Conv2D)		

convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	192

convnext_tiny_stage_0…			
(LayerNormalization)			
convnext_tiny_stage_0_bl…	(None, 56, 56, 384)	37,248	
convnext_tiny_stage_0… (Dense)			
convnext_tiny_stage_0_bl…	(None, 56, 56, 384)	0	
convnext_tiny_stage_0… (Activation)			
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	36,960	
convnext_tiny_stage_0… (Dense)			
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	96	
convnext_tiny_stage_0… (LayerScale)			
convnext_tiny_stage_0_bl…	(None, 56, 56, 96)	0	
convnext_tiny_stage_0… (Activation)			
add_1 (Add)	(None, 56, 56, 96)	0	add[0][0],
convnext_tiny_stage_0…			
convnext_tiny_stage_0_bl… (Conv2D)	(None, 56, 56, 96)	4,800	add_1[0][0]

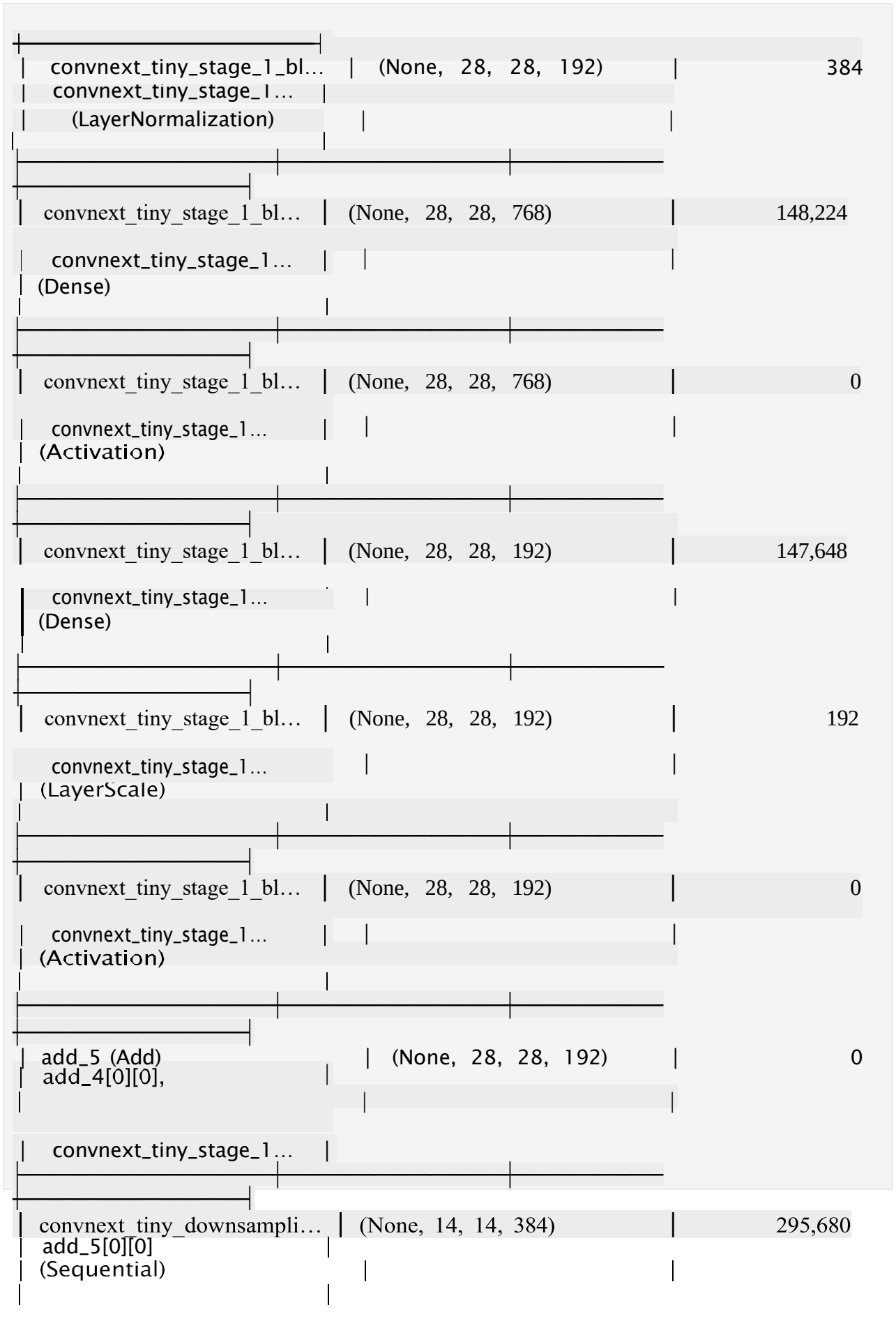
convnext_tiny_stage_0_bl… convnext_tiny_stage_0…	(None, 56, 56, 96)	192

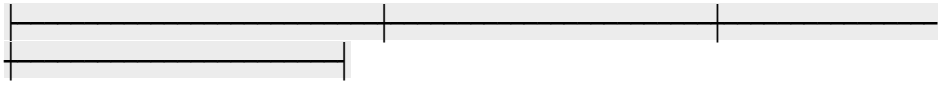

(LayerNormalization)
convnext_tiny_stage_0_bl… (None, 56, 56, 384) 37,248
convnext_tiny_stage_0… (Dense)
convnext_tiny_stage_0_bl… (None, 56, 56, 384) 0
convnext_tiny_stage_0… (Activation)
convnext_tiny_stage_0_bl… (None, 56, 56, 96) 36,960
convnext_tiny_stage_0… (Dense)
convnext_tiny_stage_0_bl… (None, 56, 56, 96) 96
convnext_tiny_stage_0… (LayerScale)
convnext_tiny_stage_0_bl… (None, 56, 56, 96) 0
convnext_tiny_stage_0… (Activation)
add_2 (Add) (None, 56, 56, 96) 0
add_1[0][0], convnext_tiny_stage_0…
convnext_tiny_downsampli… (None, 28, 28, 192) 74,112
add_2[0][0] (Sequential)
convnext_tiny_stage_1_bl… (None, 28, 28, 192) 9,600
convnext_tiny_downsam…

| (Conv2D) | |

convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (LayerNormalization)	(None, 28, 28, 192)	384	
convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (Dense)	(None, 28, 28, 768)	148,224	
convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (Activation)	(None, 28, 28, 768)	0	
convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (Dense)	(None, 28, 28, 192)	147,648	
convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (LayerScale)	(None, 28, 28, 192)	192	
convnext_tiny_stage_1_bl… convnext_tiny_stage_1… (Activation)	(None, 28, 28, 192)	0	
add_3 (Add) convnext_tiny_downsam… convnext_tiny_stage_1…	(None, 28, 28, 192)	0	
convnext_tiny_stage_1_bl… add_3[0][0] (Conv2D)	(None, 28, 28, 192)	9,600	

convnext_tiny_stage_1_bl… (LayerNormalization)	(None, 28, 28, 192)	384	convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (Dense)	(None, 28, 28, 768)	148,224	convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (Activation)	(None, 28, 28, 768)	0	convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (Dense)	(None, 28, 28, 192)	147,648	convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (LayerScale)	(None, 28, 28, 192)	192	convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (Activation)	(None, 28, 28, 192)	0	convnext_tiny_stage_1…
add_4 (Add)	(None, 28, 28, 192)	0	add_3[0][0], convnext_tiny_stage_1…
convnext_tiny_stage_1_bl… (Conv2D)	(None, 28, 28, 192)	9,600	add_4[0][0]





convnext_tiny_stage_2_bl… (Conv2D)	(None, 14, 14, 384)	19,200	convnext_tiny_downsam…
convnext_tiny_stage_2_bl… (LayerNormalization)	(None, 14, 14, 384)	768	convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (Dense)	(None, 14, 14, 1536)	591,360	convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (Activation)	(None, 14, 14, 1536)	0	convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (Dense)	(None, 14, 14, 384)	590,208	convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (LayerScale)	(None, 14, 14, 384)	384	convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (Activation)	(None, 14, 14, 384)	0	convnext_tiny_stage_2…
add_6 (Add)	(None, 14, 14, 384)	0	convnext_tiny_d ownsam…
convnext_tiny_stage_2…			

convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200

add_6[0][0] (Conv2D)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)		768
convnext_tiny_stage_2… (LayerNormalization)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)		591,360
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)		0
convnext_tiny_stage_2… (Activation)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)		590,208
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)		384
convnext_tiny_stage_2… (LayerScale)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)		0
convnext_tiny_stage_2… (Activation)			
add_7 (Add)	(None, 14, 14, 384)		0
add_6[0][0], convnext_tiny_stage_2…			

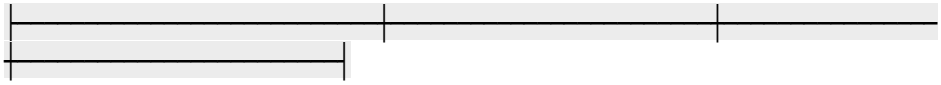
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200	add_7[0][0]

(Conv2D)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	768	
convnext_tiny_stage_2… (LayerNormalization)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	591,360	
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	0	
convnext_tiny_stage_2… (Activation)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	590,208	
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	384	
convnext_tiny_stage_2… (LayerScale)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	0	
convnext_tiny_stage_2… (Activation)			
add_8 (Add)	(None, 14, 14, 384)	0	add_7[0][0],
convnext_tiny_stage_2…			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200	add_8[0][0]

| (Conv2D) | |

convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	768	
convnext_tiny_stage_2… (LayerNormalization)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	591,360	
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	0	
convnext_tiny_stage_2… (Activation)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	590,208	
convnext_tiny_stage_2… (Dense)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	384	
convnext_tiny_stage_2… (LayerScale)			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	0	
convnext_tiny_stage_2… (Activation)			
add_9 (Add)	(None, 14, 14, 384)	0	add_8[0][0],
convnext_tiny_stage_2…			
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200	add_9[0][0]
(Conv2D)			

convnext_tiny_stage_2_bl… (LayerNormalization)	(None, 14, 14, 384)	768
convnext_tiny_stage_2_bl… (Dense)	(None, 14, 14, 1536)	591,360
convnext_tiny_stage_2_bl… (Activation)	(None, 14, 14, 1536)	0
convnext_tiny_stage_2_bl… (Dense)	(None, 14, 14, 384)	590,208
convnext_tiny_stage_2_bl… (LayerScale)	(None, 14, 14, 384)	384
convnext_tiny_stage_2_bl… (Activation)	(None, 14, 14, 384)	0
add_10 (Add)	(None, 14, 14, 384)	0
convnext_tiny_stage_2_bl… (Conv2D)	(None, 14, 14, 384)	19,200

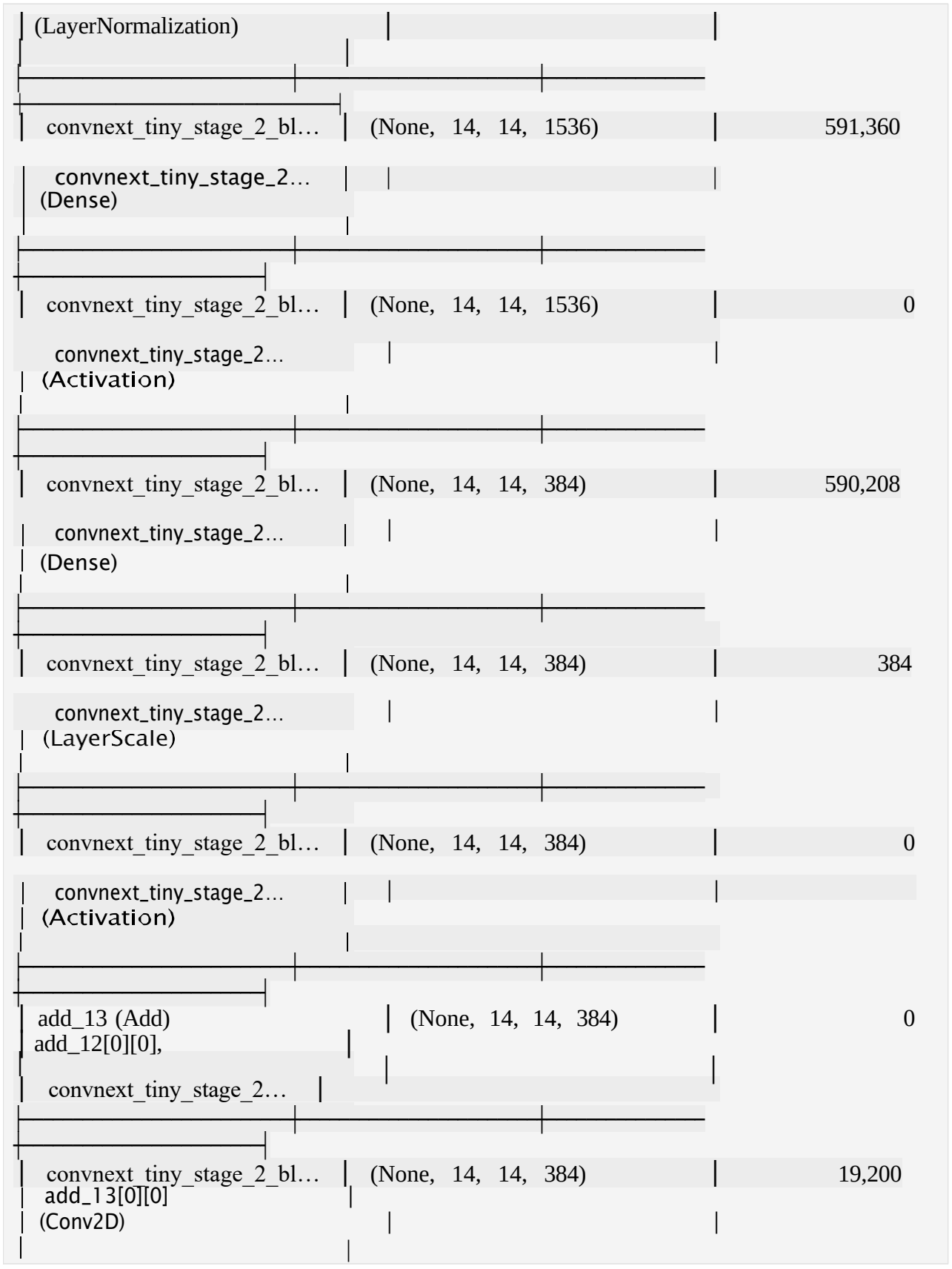


convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	768	convnext_tiny_stage_2… (LayerNormalization)
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	591,360	convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	0	convnext_tiny_stage_2… (Activation)
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	590,208	convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	384	convnext_tiny_stage_2… (LayerScale)
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	0	convnext_tiny_stage_2… (Activation)
add_11 (Add)	(None, 14, 14, 384)	0	add_10[0][0], convnext_tiny_stage_2…
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200	add_11[0][0] (Conv2D)

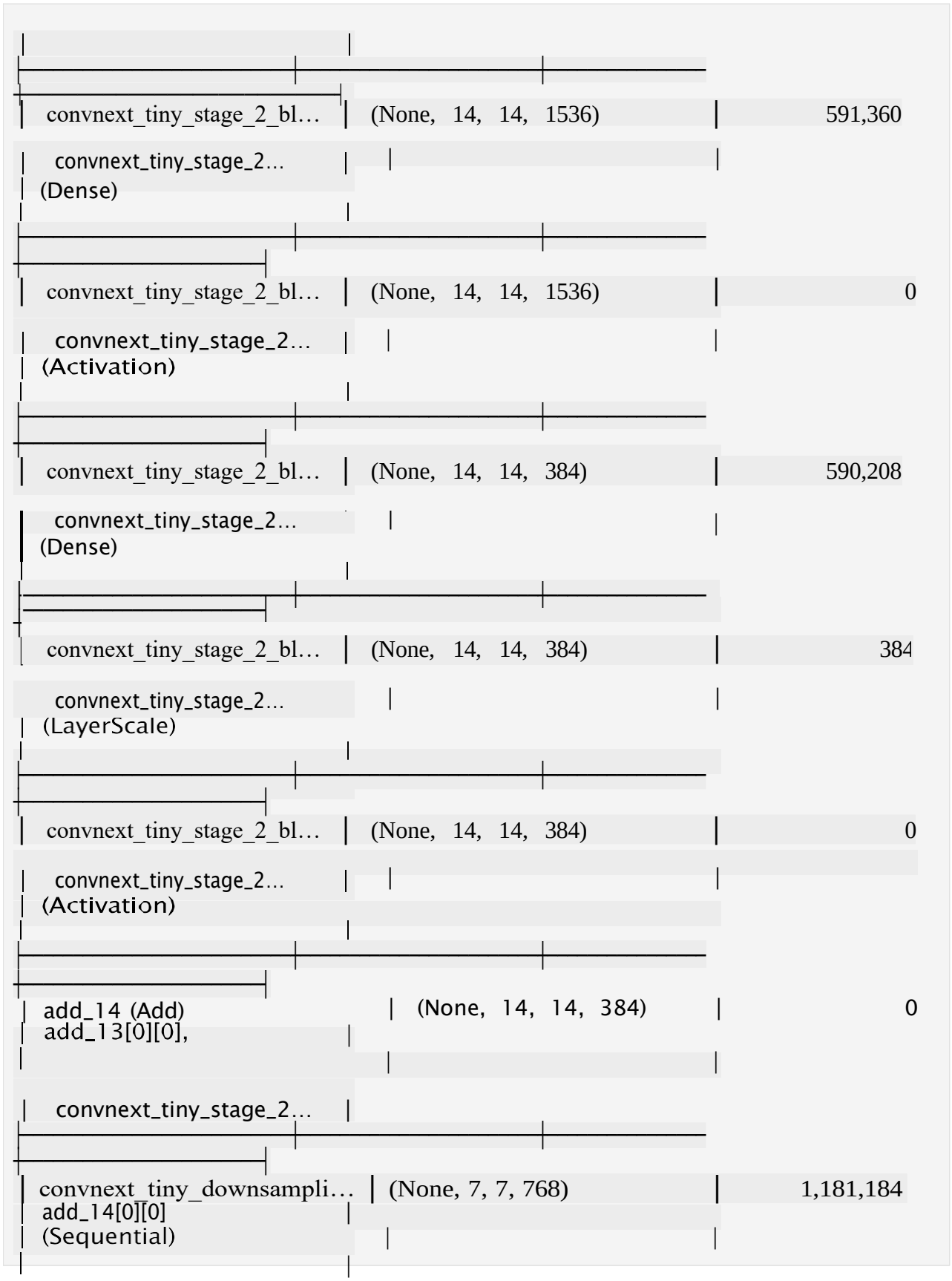
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	768

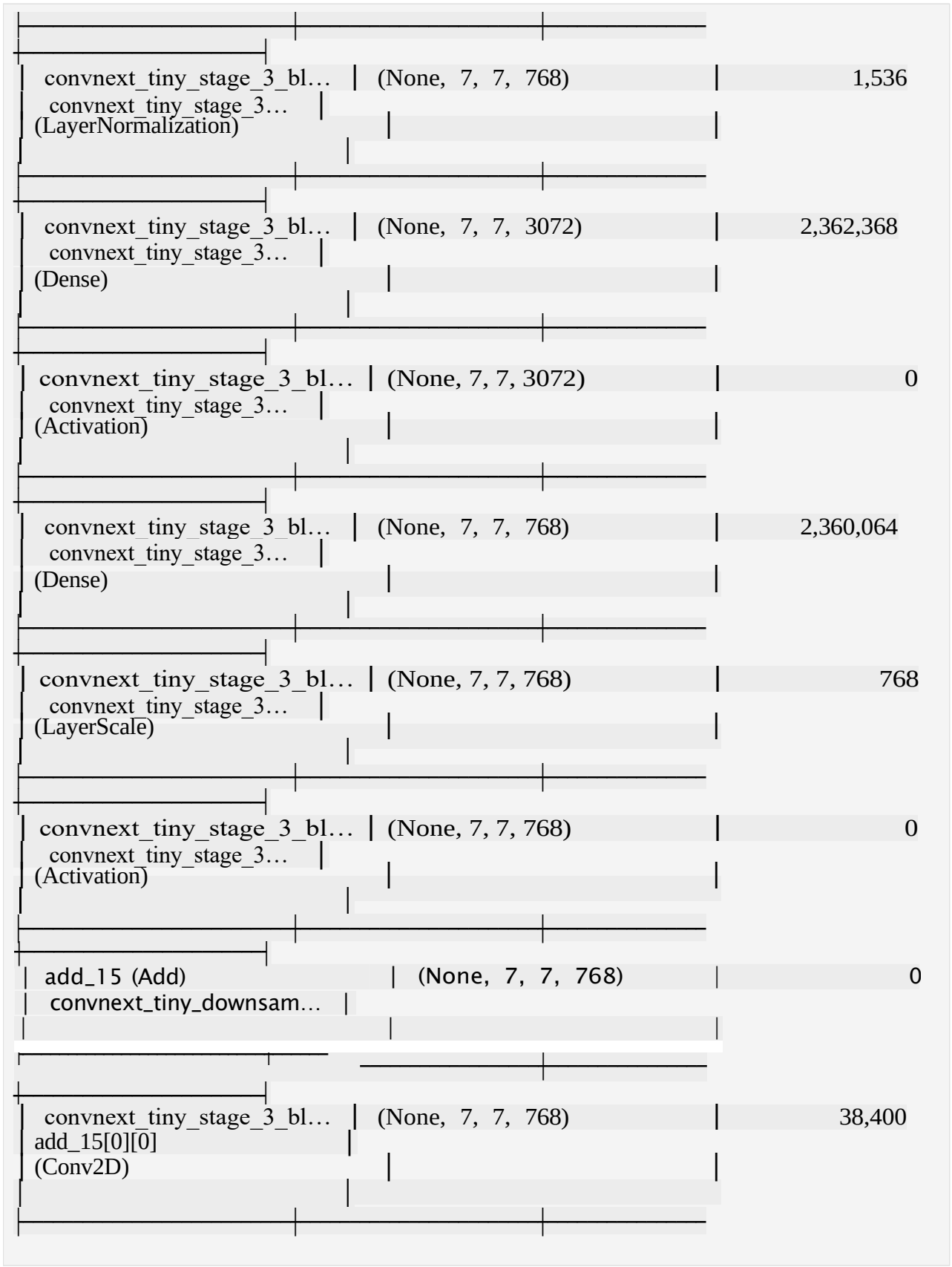
convnext_tiny_stage_2… (LayerNormalization)		
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	591,360
convnext_tiny_stage_2… (Dense)		
convnext_tiny_stage_2_bl…	(None, 14, 14, 1536)	0
convnext_tiny_stage_2… (Activation)		
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	590,208
convnext_tiny_stage_2… (Dense)		
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	384
convnext_tiny_stage_2… (LayerScale)		
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	0
convnext_tiny_stage_2… (Activation)		
add_12 (Add)	(None, 14, 14, 384)	0
add_11[0][0], convnext_tiny_stage_2…		
convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	19,200
add_12[0][0] (Conv2D)		

convnext_tiny_stage_2_bl…	(None, 14, 14, 384)	768
convnext_tiny_stage_2…		

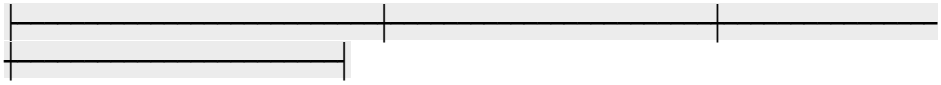

(LayerNormalization)
convnext_tiny_stage_2_bl… (None, 14, 14, 1536) 591,360
convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl… (None, 14, 14, 1536) 0
convnext_tiny_stage_2… (Activation)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 590,208
convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 384
convnext_tiny_stage_2… (LayerScale)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 0
convnext_tiny_stage_2… (Activation)
add_13 (Add) (None, 14, 14, 384) 0
add_12[0][0], convnext_tiny_stage_2…
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 19,200
add_13[0][0] (Conv2D)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 768
convnext_tiny_stage_2…

| (LayerNormalization) | | |


convnext_tiny_stage_2_bl… (None, 14, 14, 1536) 591,360
convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl… (None, 14, 14, 1536) 0
convnext_tiny_stage_2… (Activation)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 590,208
convnext_tiny_stage_2… (Dense)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 384
convnext_tiny_stage_2… (LayerScale)
convnext_tiny_stage_2_bl… (None, 14, 14, 384) 0
convnext_tiny_stage_2… (Activation)
add_14 (Add) (None, 14, 14, 384) 0
add_13[0][0], convnext_tiny_stage_2…
convnext_tiny_downsampli… (None, 7, 7, 768) 1,181,184
add_14[0][0] (Sequential)
convnext_tiny_stage_3_bl… (None, 7, 7, 768) 38,400
convnext_tiny_downsam… (Conv2D)


convnext_tiny_stage_3_bl… (LayerNormalization) (None, 7, 7, 768) 1,536
convnext_tiny_stage_3_bl… (Dense) (None, 7, 7, 3072) 2,362,368
convnext_tiny_stage_3_bl… (Activation) (None, 7, 7, 3072) 0
convnext_tiny_stage_3_bl… (Dense) (None, 7, 7, 768) 2,360,064
convnext_tiny_stage_3_bl… (LayerScale) (None, 7, 7, 768) 768
convnext_tiny_stage_3_bl… (Activation) (None, 7, 7, 768) 0
add_15 (Add) convnext_tiny_downsam… (None, 7, 7, 768) 0
convnext_tiny_stage_3_bl… add_15[0][0] (Conv2D) (None, 7, 7, 768) 38,400

convnext_tiny_stage_3_bl… (LayerNormalization)	(None, 7, 7, 768)	1,536	convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (Dense)	(None, 7, 7, 3072)	2,362,368	convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (Activation)	(None, 7, 7, 3072)	0	convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (Dense)	(None, 7, 7, 768)	2,360,064	convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (LayerScale)	(None, 7, 7, 768)	768	convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (Activation)	(None, 7, 7, 768)	0	convnext_tiny_stage_3…
add_16 (Add)	(None, 7, 7, 768)	0	add_15[0][0], convnext_tiny_stage_3…
convnext_tiny_stage_3_bl… (Conv2D)	(None, 7, 7, 768)	38,400	add_16[0][0]



convnext_tiny_stage_3_bl… convnext_tiny_stage_3… (LayerNormalization)	(None, 7, 7, 768)	1,536
convnext_tiny_stage_3_bl… (Dense)	(None, 7, 7, 3072)	2,362,368
convnext_tiny_stage_3_bl… (Activation)	(None, 7, 7, 3072)	0
convnext_tiny_stage_3_bl… (Dense)	(None, 7, 7, 768)	2,360,064
convnext_tiny_stage_3_bl… (LayerScale)	(None, 7, 7, 768)	768
convnext_tiny_stage_3_bl… (Activation)	(None, 7, 7, 768)	0
add_17 (Add) add_16[0][0], convnext_tiny_stage_3…	(None, 7, 7, 768)	0
convnext_tiny_head_gap add_17[0][0] (GlobalAveragePooling2D)	(None, 768)	0
convnext_tiny_head_layer…	(None, 768)	1,536

```
| convnext_tiny_head_ga... |
| (LayerNormalization)     |                          |                        |
|                          |                          |                        |
├──────────────────────────┼──────────────────────────┼────────────────────────┤
| convnext_tiny_head_dense | (None, 1000)             |                769,000 |
| convnext_tiny_head_la... |                          |                        |
| (Dense)                  |                          |                        |
|                          |                          |                        |
└──────────────────────────┴──────────────────────────┴────────────────────────┘

Total params: 28,589,128 (109.06 MB)

Trainable params: 28,589,128 (109.06 MB)

Non-trainable params: 0 (0.00 B)
```

1. Introduction to Transfer Learning Transfer learning memungkinkan penggunaan model yang telah dilatih sebelumnya untuk tugas baru, sehingga mempercepat pelatihan.

```
from tensorflow.keras.applications import MobileNet from
tensorflow.keras import layers, models

# Memuat MobileNet tanpa lapisan atas
base_model = MobileNet(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Menambahkan lapisan khusus untuk dataset baru
model = models.Sequential([ base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'), layers.Dense(10,
    activation='softmax')                          #10kelas
])

# Membekukan lapisan dasar
base_model.trainable = False

# Kompilasi model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5          17225924/17225924 ━━
━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ mobilenet_1.00_224 (Functional)      │ (None, 7, 7, 1024)          │       3,228,864 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ global_average_pooling2d_1           │ (None, 1024)                │               0 │
│ (GlobalAveragePooling2D)             │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 128)                 │         131,200 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 10)                  │           1,290 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 3,361,354 (12.82 MB)
 Trainable params: 132,490 (517.54 KB)
 Non-trainable params: 3,228,864 (12.32 MB)
```

1. Let's Dive Further with MobileNet Membahas adaptasi MobileNet untuk aplikasi tertentu, seperti klasifikasi objek khusus.

2. ResNet ResNet memperkenalkan Residual Blocks, yang memungkinkan pelatihan jaringan yang sangat dalam dengan mengatasi masalah vanishing gradient.

```
from tensorflow.keras.layers import Add, Conv2D, BatchNormalization,
ReLU

def residual_block(x, filters):
    shortcut = x
    x = Conv2D(filters, (3, 3), padding='same', activation=None)(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Conv2D(filters, (3, 3), padding='same', activation=None)(x)
    x = BatchNormalization()(x)
    x = Add()([shortcut, x])
```

```
    x = ReLU()(x)
    return x
```

UNIT 3

Vision Transformers (ViT) adalah pendekatan baru dalam computer vision yang menggunakan Transformers, arsitektur terkenal dalam pemrosesan bahasa alami (NLP), untuk memproses data visual. Alih-alih menggunakan Convolutional Neural Networks (CNNs), ViT memanfaatkan self-attention untuk menangkap hubungan global dalam gambar.

1. Vision Transformers for Image Classification Vision Transformers (ViTs) menggunakan mekanisme self-attention untuk menangani data gambar, menggantikan arsitektur konvolusi tradisional.

Penjelasan:

Patch Embedding: Membagi gambar menjadi "patches" kecil, lalu mengubahnya menjadi representasi vektor. Transformer Block: Menggunakan multi-head attention dan feed-forward layers untuk mempelajari hubungan antara patches. Vision Transformer Model: Model yang mencakup beberapa blok transformer dan klasifikasi menggunakan dense layer.

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Fungsi untuk membuat patch dari gambar
class PatchEmbed(layers.Layer):
    def __init__(self, patch_size, embed_dim):
        super(PatchEmbed, self).__init__()
        self.patch_size = patch_size self.projection =
        layers.Conv2D(embed_dim,
kernel_size=patch_size, strides=patch_size, padding='valid')

    def call(self, images):
        # Membagi gambar menjadi patch
        patches = self.projection(images)                # Hasil Conv2D #
        Mendapatkan dimensi patches
        batch_size = tf.shape(patches)[0]
        height = tf.shape(patches)[1] width =
        tf.shape(patches)[2] channels =
        tf.shape(patches)[3]
        # Mengubah dimensi menjadi (batch_size, num_patches, embed_dim)
        patches = tf.reshape(patches, [batch_size, height * width, channels])
        return patches

# Vision Transformer Encoder Layer
def transformer_encoder(inputs, num_heads, projection_dim, ff_dim):
    # Multi-Head Attention
    attention_output = layers.MultiHeadAttention(num_heads=num_heads,
```

```
key_dim=projection_dim)(inputs, inputs)
        attention_output = layers.Add()([attention_output, inputs]) attention_output
        = layers.LayerNormalization()(attention_output)

        # Feed Forward Network
        ffn = models.Sequential([ layers.Dense(ff_dim,
              activation="relu"),
              layers.Dense(projection_dim),
        ])
        ffn_output = ffn(attention_output)
        ffn_output = layers.Add()([ffn_output, attention_output]) return
        layers.LayerNormalization()(ffn_output)

# Vision Transformer Model
def create_vit_model(input_shape, patch_size, num_patches, projection_dim,
num_heads, ff_dim, num_layers, num_classes):
        inputs = layers.Input(shape=input_shape)

        # Patch Embedding
        patches = PatchEmbed(patch_size=patch_size, embed_dim=projection_dim)(inputs)

        # Positional Embedding
        positions = tf.range(start=0, limit=num_patches, delta=1)
        positional_embedding = layers.Embedding(input_dim=num_patches,
output_dim=projection_dim)(positions)          encoded_patches =
        patches + positional_embedding

        # Transformer Encoder Layers
        for _ in range(num_layers):
              encoded_patches = transformer_encoder(encoded_patches, num_heads,
projection_dim, ff_dim)

        # Classification Head
        representation = layers.GlobalAveragePooling1D()(encoded_patches) outputs
        = layers.Dense(num_classes, activation="softmax")
(representation)

        return models.Model(inputs=inputs, outputs=outputs)

# Parameter model
input_shape = (32, 32, 3)
patch_size = 4
num_patches = (input_shape[0] // patch_size) * (input_shape[1] // patch_size)
projection_dim = 64
num_heads = 4
ff_dim = 128
num_layers = 4
num_classes = 10
```

```
# Membuat mod
```

1. Swin Transformer Swin Transformer adalah versi transformer yang menggunakan shifting windows untuk menangkap hubungan lokal dan global dalam gambar.

```
from transformers import SwinForImageClassification, SwinConfig

# Konfigurasi model Swin
config = SwinConfig()
model = SwinForImageClassification(config)

# Menampilkan arsitektur model
print(model)
```

```
SwinForImageClassification(
  (swin): SwinModel(
    (embeddings): SwinEmbeddings(
      (patch_embeddings): SwinPatchEmbeddings(
        (projection): Conv2d(3, 96, kernel_size=(4, 4), stride=(4, 4))
      )
      (norm): LayerNorm((96,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): SwinEncoder(
      (layers): ModuleList(
        (0): SwinStage(
          (blocks): ModuleList(
            (0): SwinLayer(
              (layernorm_before): LayerNorm((96,), eps=1e-05,
elementwise_affine=True)
              (attention): SwinAttention(
                (self): SwinSelfAttention(
                  (query): Linear(in_features=96, out_features=96,
bias=True)
                  (key): Linear(in_features=96, out_features=96,
bias=True)
                  (value): Linear(in_features=96, out_features=96,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
                (output): SwinSelfOutput(
                  (dense): Linear(in_features=96, out_features=96,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
              (drop_path): Identity()
              (layernorm_after): LayerNorm((96,), eps=1e-05,
elementwise_affine=True)
```

```
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=96, out_features=384,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=384, out_features=96,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
              (1): SwinLayer(
                (layernorm_before): LayerNorm((96,), eps=1e-05,
elementwise_affine=True)
                (attention): SwinAttention(
                  (self): SwinSelfAttention(
                    (query): Linear(in_features=96, out_features=96,
bias=True)
                    (key): Linear(in_features=96, out_features=96,
bias=True)
                    (value): Linear(in_features=96, out_features=96,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=96, out_features=96,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
                (drop_path): SwinDropPath(p=0.00909090880304575)
                (layernorm_after): LayerNorm((96,), eps=1e-05,
elementwise_affine=True)
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=96, out_features=384,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=384, out_features=96,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
            )
            (downsample): SwinPatchMerging(
              (reduction): Linear(in_features=384, out_features=192,
bias=False)
              (norm): LayerNorm((384,), eps=1e-05,
```

```
elementwise_affine=True)
            )
          )
          (1): SwinStage(
            (blocks): ModuleList(
              (0): SwinLayer(
                (layernorm_before): LayerNorm((192,), eps=1e-05,
elementwise_affine=True)
                (attention): SwinAttention(
                  (self): SwinSelfAttention(
                    (query): Linear(in_features=192, out_features=192,
bias=True)
                    (key): Linear(in_features=192, out_features=192,
bias=True)
                    (value): Linear(in_features=192, out_features=192,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=192, out_features=192,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
                (drop_path): SwinDropPath(p=0.0181818176060915)
                (layernorm_after): LayerNorm((192,), eps=1e-05,
elementwise_affine=True)
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=192, out_features=768,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=768, out_features=192,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
              (1): SwinLayer(
                (layernorm_before): LayerNorm((192,), eps=1e-05,
elementwise_affine=True)
                (attention): SwinAttention(
                  (self): SwinSelfAttention(
                    (query): Linear(in_features=192, out_features=192,
bias=True)
                    (key): Linear(in_features=192, out_features=192,
bias=True)
                    (value): Linear(in_features=192, out_features=192,
bias=True)
```

```
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=192, out_features=192,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
                (drop_path): SwinDropPath(p=0.027272727340459824)
                (layernorm_after): LayerNorm((192,), eps=1e-05,
elementwise_affine=True)
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=192, out_features=768,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=768, out_features=192,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
            )
            (downsample): SwinPatchMerging(
              (reduction): Linear(in_features=768, out_features=384,
bias=False)
              (norm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
            )
          )
          (2): SwinStage(
            (blocks): ModuleList(
              (0): SwinLayer(
                (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
                (attention): SwinAttention(
                  (self): SwinSelfAttention(
                    (query): Linear(in_features=384, out_features=384,
bias=True)
                    (key): Linear(in_features=384, out_features=384,
bias=True)
                    (value): Linear(in_features=384, out_features=384,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=384, out_features=384,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
```

```
                )
            )
            (drop_path): SwinDropPath(p=0.036363635212183)
            (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (intermediate): SwinIntermediate(
              (dense): Linear(in_features=384, out_features=1536,
bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): SwinOutput(
              (dense): Linear(in_features=1536, out_features=384,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
          (1): SwinLayer(
            (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (attention): SwinAttention(
              (self): SwinSelfAttention(
                (query): Linear(in_features=384, out_features=384,
bias=True)
                (key): Linear(in_features=384, out_features=384,
bias=True)
                (value): Linear(in_features=384, out_features=384,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
              (output): SwinSelfOutput(
                (dense): Linear(in_features=384, out_features=384,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (drop_path): SwinDropPath(p=0.045454543083906174)
            (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (intermediate): SwinIntermediate(
              (dense): Linear(in_features=384, out_features=1536,
bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): SwinOutput(
              (dense): Linear(in_features=1536, out_features=384,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
```

```
            (2): SwinLayer(
              (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
              (attention): SwinAttention(
                (self): SwinSelfAttention(
                  (query): Linear(in_features=384, out_features=384,
bias=True)
                  (key): Linear(in_features=384, out_features=384,
bias=True)
                  (value): Linear(in_features=384, out_features=384,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
                (output): SwinSelfOutput(
                  (dense): Linear(in_features=384, out_features=384,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
              (drop_path): SwinDropPath(p=0.054545458406209946)
              (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
              (intermediate): SwinIntermediate(
                (dense): Linear(in_features=384, out_features=1536,
bias=True)
                (intermediate_act_fn): GELUActivation()
              )
              (output): SwinOutput(
                (dense): Linear(in_features=1536, out_features=384,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (3): SwinLayer(
              (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
              (attention): SwinAttention(
                (self): SwinSelfAttention(
                  (query): Linear(in_features=384, out_features=384,
bias=True)
                  (key): Linear(in_features=384, out_features=384,
bias=True)
                  (value): Linear(in_features=384, out_features=384,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
                (output): SwinSelfOutput(
                  (dense): Linear(in_features=384, out_features=384,
bias=True)
```

```
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (drop_path): SwinDropPath(p=0.06363636255264282)
            (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (intermediate): SwinIntermediate(
              (dense): Linear(in_features=384, out_features=1536,
bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): SwinOutput(
              (dense): Linear(in_features=1536, out_features=384,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
          (4): SwinLayer(
            (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (attention): SwinAttention(
              (self): SwinSelfAttention(
                (query): Linear(in_features=384, out_features=384,
bias=True)
                (key): Linear(in_features=384, out_features=384,
bias=True)
                (value): Linear(in_features=384, out_features=384,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
              (output): SwinSelfOutput(
                (dense): Linear(in_features=384, out_features=384,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (drop_path): SwinDropPath(p=0.0727272778749466)
            (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
            (intermediate): SwinIntermediate(
              (dense): Linear(in_features=384, out_features=1536,
bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): SwinOutput(
              (dense): Linear(in_features=1536, out_features=384,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
```

```
                )
                (5): SwinLayer(
                  (layernorm_before): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
                  (attention): SwinAttention(
                    (self): SwinSelfAttention(
                      (query): Linear(in_features=384, out_features=384,
bias=True)
                      (key): Linear(in_features=384, out_features=384,
bias=True)
                      (value): Linear(in_features=384, out_features=384,
bias=True)
                      (dropout): Dropout(p=0.0, inplace=False)
                    )
                    (output): SwinSelfOutput(
                      (dense): Linear(in_features=384, out_features=384,
bias=True)
                      (dropout): Dropout(p=0.0, inplace=False)
                    )
                  )
                  (drop_path): SwinDropPath(p=0.08181818574666977)
                  (layernorm_after): LayerNorm((384,), eps=1e-05,
elementwise_affine=True)
                  (intermediate): SwinIntermediate(
                    (dense): Linear(in_features=384, out_features=1536,
bias=True)
                    (intermediate_act_fn): GELUActivation()
                  )
                  (output): SwinOutput(
                    (dense): Linear(in_features=1536, out_features=384,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
              )
              (downsample): SwinPatchMerging(
                (reduction): Linear(in_features=1536, out_features=768,
bias=False)
                (norm): LayerNorm((1536,), eps=1e-05,
elementwise_affine=True)
              )
            )
            (3): SwinStage(
              (blocks): ModuleList(
                (0): SwinLayer(
                  (layernorm_before): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
                  (attention): SwinAttention(
                    (self): SwinSelfAttention(
```

```
                    (query): Linear(in_features=768, out_features=768,
bias=True)
                    (key): Linear(in_features=768, out_features=768,
bias=True)
                    (value): Linear(in_features=768, out_features=768,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=768, out_features=768,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
                (drop_path): SwinDropPath(p=0.09090909361839294)
                (layernorm_after): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=768, out_features=3072,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=3072, out_features=768,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
              (1): SwinLayer(
                (layernorm_before): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
                (attention): SwinAttention(
                  (self): SwinSelfAttention(
                    (query): Linear(in_features=768, out_features=768,
bias=True)
                    (key): Linear(in_features=768, out_features=768,
bias=True)
                    (value): Linear(in_features=768, out_features=768,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                  (output): SwinSelfOutput(
                    (dense): Linear(in_features=768, out_features=768,
bias=True)
                    (dropout): Dropout(p=0.0, inplace=False)
                  )
                )
                (drop_path): SwinDropPath(p=0.10000000149011612)
                (layernorm_after): LayerNorm((768,), eps=1e-05,
```

```
elementwise_affine=True)
                (intermediate): SwinIntermediate(
                  (dense): Linear(in_features=768, out_features=3072,
bias=True)
                  (intermediate_act_fn): GELUActivation()
                )
                (output): SwinOutput(
                  (dense): Linear(in_features=3072, out_features=768,
bias=True)
                  (dropout): Dropout(p=0.0, inplace=False)
                )
              )
            )
          )
        )
      )
      (layernorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (pooler): AdaptiveAvgPool1d(output_size=1)
    )
    (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

1. Convolutional Vision Transformer (CvT) CvT menggabungkan lapisan konvolusi

```
# Tidak ada library langsung untuk CvT; implementasi dapat dibuat
dengan menambahkan Conv2D pada input Vision Transformer
# Konsepnya mirip dengan menggunakan convolutional layers untuk
embedding patches
```

dengan transformer untuk memperbaiki performa dan efisiensi.

1. Dilated Neighborhood Attention Transformer (DINAT) DINAT meningkatkan perhatian lokal dengan menggunakan dilated attention. Konsep ini membutuhkan implementasi khusus yang kompleks, sering kali diimplementasikan pada framework custom seperti PyTorch.

2. MobileViT v2 MobileViT adalah arsitektur ringan berbasis Vision Transformer untuk perangkat mobile.

```
from timm import create_model

# Menggunakan MobileViT v2 dari TIMM
model = create_model('mobilevitv2_050', pretrained=True)
print(model)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
```

```
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks. Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

{"model_id":"45a6661779554e278c9c5abc575786c9","version_major":2,"version_minor":0}

```
ByobNet(
  (stem): ConvNormAct(
    (conv): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn): BatchNormAct2d(
      16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
      (drop): Identity()
      (act): SiLU(inplace=True)
    )
  )
  (stages): Sequential(
    (0): Sequential(
      (0): BottleneckBlock(
        (conv1_1x1): ConvNormAct(
          (conv): Conv2d(16, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn): BatchNormAct2d(
            32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
        )
        (conv2_kxk): ConvNormAct(
          (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
          (bn): BatchNormAct2d(
            32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
        )
        (conv2b_kxk): Identity()
        (attn): Identity()
        (conv3_1x1): ConvNormAct(
          (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn): BatchNormAct2d(
            32, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True
                (drop): Identity()
                (act): Identity()
              )
            )
            (attn_last): Identity()
            (drop_path): Identity()
            (act): Identity()
          )
        )
        (1): Sequential(
          (0): BottleneckBlock(
            (conv1_1x1): ConvNormAct(
              (conv): Conv2d(32, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
              (bn): BatchNormAct2d(
                64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): SiLU(inplace=True)
              )
            )
            (conv2_kxk): ConvNormAct(
              (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=64, bias=False)
              (bn): BatchNormAct2d(
                64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): SiLU(inplace=True)
              )
            )
            (conv2b_kxk): Identity()
            (attn): Identity()
            (conv3_1x1): ConvNormAct(
              (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
              (bn): BatchNormAct2d(
                64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): Identity()
              )
            )
            (attn_last): Identity()
            (drop_path): Identity()
            (act): Identity()
          )
          (1): BottleneckBlock(
```

```
          (shortcut): Identity()
          (conv1_1x1): ConvNormAct(
            (conv): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
          )
          (conv2_kxk): ConvNormAct(
            (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=128, bias=False)
            (bn): BatchNormAct2d(
              128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
          )
          (conv2b_kxk): Identity()
          (attn): Identity()
          (conv3_1x1): ConvNormAct(
            (conv): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
          )
          (attn_last): Identity()
          (drop_path): Identity()
          (act): Identity()
        )
      )
      (2): Sequential(
        (0): BottleneckBlock(
          (conv1_1x1): ConvNormAct(
            (conv): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
```

```
            )
            (conv2_kxk): ConvNormAct(
              (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=128, bias=False)
              (bn): BatchNormAct2d(
                128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): SiLU(inplace=True)
              )
            )
            (conv2b_kxk): Identity()
            (attn): Identity()
            (conv3_1x1): ConvNormAct(
              (conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
              (bn): BatchNormAct2d(
                128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): Identity()
              )
            )
            (attn_last): Identity()
            (drop_path): Identity()
            (act): Identity()
          )
          (1): MobileVitV2Block(
            (conv_kxk): ConvNormAct(
              (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=128, bias=False)
              (bn): BatchNormAct2d(
                128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
                (drop): Identity()
                (act): SiLU(inplace=True)
              )
            )
            (conv_1x1): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (transformer): Sequential(
              (0): LinearTransformerBlock(
                (norm1): GroupNorm1(1, 64, eps=1e-05, affine=True)
                (attn): LinearSelfAttention(
                  (qkv_proj): Conv2d(64, 129, kernel_size=(1, 1),
stride=(1, 1))
                  (attn_drop): Dropout(p=0.0, inplace=False)
                  (out_proj): Conv2d(64, 64, kernel_size=(1, 1),
stride=(1, 1))
```

```
                (out_drop): Dropout(p=0.0, inplace=False)
              )
              (drop_path1): DropPath(drop_prob=0.000)
              (norm2): GroupNorm1(1, 64, eps=1e-05, affine=True)
              (mlp): ConvMlp(
                (fc1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1,
1))
                (norm): Identity()
                (act): SiLU()
                (drop): Dropout(p=0.0, inplace=False)
                (fc2): Conv2d(128, 64, kernel_size=(1, 1), stride=(1,
1))
              )
              (drop_path2): DropPath(drop_prob=0.000)
            )
            (1): LinearTransformerBlock(
              (norm1): GroupNorm1(1, 64, eps=1e-05, affine=True)
              (attn): LinearSelfAttention(
                (qkv_proj): Conv2d(64, 129, kernel_size=(1, 1),
stride=(1, 1))
                (attn_drop): Dropout(p=0.0, inplace=False)
                (out_proj): Conv2d(64, 64, kernel_size=(1, 1),
stride=(1, 1))
                (out_drop): Dropout(p=0.0, inplace=False)
              )
              (drop_path1): DropPath(drop_prob=0.000)
              (norm2): GroupNorm1(1, 64, eps=1e-05, affine=True)
              (mlp): ConvMlp(
                (fc1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1,
1))
                (norm): Identity()
                (act): SiLU()
                (drop): Dropout(p=0.0, inplace=False)
                (fc2): Conv2d(128, 64, kernel_size=(1, 1), stride=(1,
1))
              )
              (drop_path2): DropPath(drop_prob=0.000)
            )
          )
          (norm): GroupNorm1(1, 64, eps=1e-05, affine=True)
          (conv_proj): ConvNormAct(
            (conv): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
```

```
          )
        )
      )
      (3): Sequential(
        (0): BottleneckBlock(
          (conv1_1x1): ConvNormAct(
            (conv): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
          )
          (conv2_kxk): ConvNormAct(
            (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=256, bias=False)
            (bn): BatchNormAct2d(
              256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
          )
          (conv2b_kxk): Identity()
          (attn): Identity()
          (conv3_1x1): ConvNormAct(
            (conv): Conv2d(256, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
          )
          (attn_last): Identity()
          (drop_path): Identity()
          (act): Identity()
        )
        (1): MobileVitV2Block(
          (conv_kxk): ConvNormAct(
            (conv): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=192, bias=False)
            (bn): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
```

```
            (act): SiLU(inplace=True)
          )
        )
        (conv_1x1): Conv2d(192, 96, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (transformer): Sequential(
          (0): LinearTransformerBlock(
            (norm1): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (attn): LinearSelfAttention(
              (qkv_proj): Conv2d(96, 193, kernel_size=(1, 1),
stride=(1, 1))
              (attn_drop): Dropout(p=0.0, inplace=False)
              (out_proj): Conv2d(96, 96, kernel_size=(1, 1),
stride=(1, 1))
              (out_drop): Dropout(p=0.0, inplace=False)
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(96, 192, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(192, 96, kernel_size=(1, 1), stride=(1,
1))
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
          (1): LinearTransformerBlock(
            (norm1): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (attn): LinearSelfAttention(
              (qkv_proj): Conv2d(96, 193, kernel_size=(1, 1),
stride=(1, 1))
              (attn_drop): Dropout(p=0.0, inplace=False)
              (out_proj): Conv2d(96, 96, kernel_size=(1, 1),
stride=(1, 1))
              (out_drop): Dropout(p=0.0, inplace=False)
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(96, 192, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(192, 96, kernel_size=(1, 1), stride=(1,
1))
```

```
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
          (2): LinearTransformerBlock(
            (norm1): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (attn): LinearSelfAttention(
              (qkv_proj): Conv2d(96, 193, kernel_size=(1, 1),
stride=(1, 1))
              (attn_drop): Dropout(p=0.0, inplace=False)
              (out_proj): Conv2d(96, 96, kernel_size=(1, 1),
stride=(1, 1))
              (out_drop): Dropout(p=0.0, inplace=False)
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(96, 192, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(192, 96, kernel_size=(1, 1), stride=(1,
1))
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
          (3): LinearTransformerBlock(
            (norm1): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (attn): LinearSelfAttention(
              (qkv_proj): Conv2d(96, 193, kernel_size=(1, 1),
stride=(1, 1))
              (attn_drop): Dropout(p=0.0, inplace=False)
              (out_proj): Conv2d(96, 96, kernel_size=(1, 1),
stride=(1, 1))
              (out_drop): Dropout(p=0.0, inplace=False)
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 96, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(96, 192, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(192, 96, kernel_size=(1, 1), stride=(1,
1))
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
```

```
          )
          (norm): GroupNorm1(1, 96, eps=1e-05, affine=True)
          (conv_proj):  ConvNormAct(
            (conv): Conv2d(96, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn):  BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act):  Identity()
            )
          )
        )
      )
      (4):  Sequential(
        (0):  BottleneckBlock(
          (conv1_1x1):  ConvNormAct(
            (conv): Conv2d(192, 384, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn):  BatchNormAct2d(
              384, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act):  SiLU(inplace=True)
            )
          )
          (conv2_kxk):  ConvNormAct(
            (conv): Conv2d(384, 384, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=384, bias=False)
            (bn):  BatchNormAct2d(
              384, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act):  SiLU(inplace=True)
            )
          )
          (conv2b_kxk): Identity()
          (attn):  Identity()
          (conv3_1x1):  ConvNormAct(
            (conv): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn):  BatchNormAct2d(
              256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act):  Identity()
            )
          )
          (attn_last):  Identity()
```

```
          (drop_path): Identity()
          (act): Identity()
        )
        (1): MobileVitV2Block(
          (conv_kxk): ConvNormAct(
            (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=256, bias=False)
            (bn): BatchNormAct2d(
              256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): SiLU(inplace=True)
            )
          )
          (conv_1x1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (transformer): Sequential(
            (0): LinearTransformerBlock(
              (norm1): GroupNorm1(1, 128, eps=1e-05, affine=True)
              (attn): LinearSelfAttention(
                (qkv_proj): Conv2d(128, 257, kernel_size=(1, 1),
stride=(1, 1))
                (attn_drop): Dropout(p=0.0, inplace=False)
                (out_proj): Conv2d(128, 128, kernel_size=(1, 1),
stride=(1, 1))
                (out_drop): Dropout(p=0.0, inplace=False)
              )
              (drop_path1): DropPath(drop_prob=0.000)
              (norm2): GroupNorm1(1, 128, eps=1e-05, affine=True)
              (mlp): ConvMlp(
                (fc1): Conv2d(128, 256, kernel_size=(1, 1), stride=(1,
1))
                (norm): Identity()
                (act): SiLU()
                (drop): Dropout(p=0.0, inplace=False)
                (fc2): Conv2d(256, 128, kernel_size=(1, 1), stride=(1,
1))
              )
              (drop_path2): DropPath(drop_prob=0.000)
            )
            (1): LinearTransformerBlock(
              (norm1): GroupNorm1(1, 128, eps=1e-05, affine=True)
              (attn): LinearSelfAttention(
                (qkv_proj): Conv2d(128, 257, kernel_size=(1, 1),
stride=(1, 1))
                (attn_drop): Dropout(p=0.0, inplace=False)
                (out_proj): Conv2d(128, 128, kernel_size=(1, 1),
stride=(1, 1))
                (out_drop): Dropout(p=0.0, inplace=False)
```

```
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 128, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(128, 256, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(256, 128, kernel_size=(1, 1), stride=(1,
1))
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
          (2): LinearTransformerBlock(
            (norm1): GroupNorm1(1, 128, eps=1e-05, affine=True)
            (attn): LinearSelfAttention(
              (qkv_proj): Conv2d(128, 257, kernel_size=(1, 1),
stride=(1, 1))
              (attn_drop): Dropout(p=0.0, inplace=False)
              (out_proj): Conv2d(128, 128, kernel_size=(1, 1),
stride=(1, 1))
              (out_drop): Dropout(p=0.0, inplace=False)
            )
            (drop_path1): DropPath(drop_prob=0.000)
            (norm2): GroupNorm1(1, 128, eps=1e-05, affine=True)
            (mlp): ConvMlp(
              (fc1): Conv2d(128, 256, kernel_size=(1, 1), stride=(1,
1))
              (norm): Identity()
              (act): SiLU()
              (drop): Dropout(p=0.0, inplace=False)
              (fc2): Conv2d(256, 128, kernel_size=(1, 1), stride=(1,
1))
            )
            (drop_path2): DropPath(drop_prob=0.000)
          )
        )
        (norm): GroupNorm1(1, 128, eps=1e-05, affine=True)
        (conv_proj): ConvNormAct(
          (conv): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
          (bn): BatchNormAct2d(
            256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
        )
      )
```

```
    )
  )
  (final_conv): Identity()
  (head): ClassifierHead(
    (global_pool): SelectAdaptivePool2d(pool_type=avg,
flatten=Flatten(start_dim=1, end_dim=-1))
    (drop): Dropout(p=0.0, inplace=False)
    (fc): Linear(in_features=256, out_features=1000, bias=True)
    (flatten): Identity()
  )
)
```

1. FineTuning Vision Transformer for Object Detection Melakukan fine-tuning model pretrained Vision Transformer untuk mendeteksi objek.

```
from transformers import ViTForImageClassification import
torch
# Memuat model Vision Transformer (ViT) yang sudah pretrained
model = ViTForImageClassification.from_pretrained("google/vit-base- patch16-
224-in21k")

# Menyesuaikan output untuk jumlah kelas baru (misalnya, 20 kelas) model.classifier =
torch.nn.Linear(in_features=model.classifier.in_features, out_features=20)

# Menampilkan arsitektur model
print(model)
```

```
Some weights of ViTForImageClassification were not initialized from the
model checkpoint at google/vit-base-patch16-224-in21k and are newly
initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use
it for predictions and inference.
```

```
ViTForImageClassification(
  (vit): ViTModel(
    (embeddings): ViTEmbeddings(
      (patch_embeddings): ViTPatchEmbeddings(
        (projection): Conv2d(3, 768, kernel_size=(16, 16), stride=(16,
16))
      )
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): ViTEncoder(
      (layer): ModuleList(
        (0-11): 12 x ViTLayer(
          (attention): ViTSdpaAttention(
            (attention): ViTSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768,
```

```
bias=True)
                (key): Linear(in_features=768, out_features=768,
bias=True)
                (value): Linear(in_features=768, out_features=768,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
              (output): ViTSelfOutput(
                (dense): Linear(in_features=768, out_features=768,
bias=True)
                (dropout): Dropout(p=0.0, inplace=False)
              )
            )
            (intermediate): ViTIntermediate(
              (dense): Linear(in_features=768, out_features=3072,
bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): ViTOutput(
              (dense): Linear(in_features=3072, out_features=768,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (layernorm_before): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (layernorm_after): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
          )
        )
      )
      (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    )
    (classifier): Linear(in_features=768, out_features=20, bias=True)
)
```

1. DETR (DEtection TRansformer) DETR menggunakan Vision Transformer untuk tugas

```
from transformers import DetrForObjectDetection

# Memuat model DETR
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-
50")
print(model)
```

{"model_id":"1d56dbc9a0a349968e4519c82b4a0393","version_major":2,"vers
ion_minor":0}

{"model_id":"c8d64519570040daa8ae4cb09349c864","version_major":2,"vers
ion_minor":0}

deteksi objek.

{"model_id":"36de5aada4a642a2926e6da53624a196","version_major":2,"version_minor":0}

Some weights of the model checkpoint at facebook/detr-resnet-50 were not used when initializing DetrForObjectDetection:
['model.backbone.conv_encoder.model.layer1.0.downsample.1.num_batches_tracked',
'model.backbone.conv_encoder.model.layer2.0.downsample.1.num_batches_tracked',
'model.backbone.conv_encoder.model.layer3.0.downsample.1.num_batches_tracked',
'model.backbone.conv_encoder.model.layer4.0.downsample.1.num_batches_tracked']
- This IS expected if you are initializing DetrForObjectDetection from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing DetrForObjectDetection from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

```
DetrForObjectDetection(
  (model): DetrModel(
    (backbone): DetrConvModel(
      (conv_encoder): DetrConvEncoder(
        (model): FeatureListNet(
          (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
          (bn1): DetrFrozenBatchNorm2d()
          (act1): ReLU(inplace=True)
          (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
          (layer1): Sequential(
            (0): Bottleneck(
              (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
              (bn1): DetrFrozenBatchNorm2d()
              (act1): ReLU(inplace=True)
              (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity()
              (act2): ReLU(inplace=True)
              (aa): Identity()
              (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
              (bn3): DetrFrozenBatchNorm2d()
              (act3): ReLU(inplace=True)
              (downsample): Sequential(
```

```
                (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (1): DetrFrozenBatchNorm2d()
              )
            )
            (1): Bottleneck(
              (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1,
1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True) (aa):
              Identity()
              (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1,
1), bias=False)
              (bn3): DetrFrozenBatchNorm2d() (act3):
              ReLU(inplace=True)
            )
            (2): Bottleneck(
              (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1,
1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True) (aa):
              Identity()
              (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1,
1), bias=False)
              (bn3): DetrFrozenBatchNorm2d() (act3):
              ReLU(inplace=True)
            )
          )
          (layer2): Sequential( (0):
            Bottleneck(
              (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1,
1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2,
2), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True)
```

```
                (aa): Identity()
                (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn3): DetrFrozenBatchNorm2d() (act3):
                ReLU(inplace=True) (downsample):
                Sequential(
                  (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2,
2), bias=False)
                  (1): DetrFrozenBatchNorm2d()
                )
              )
              (1): Bottleneck(
                (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity() (act2):
                ReLU(inplace=True) (aa):
                Identity()
                (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn3): DetrFrozenBatchNorm2d() (act3):
                ReLU(inplace=True)
              )
              (2): Bottleneck(
                (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity() (act2):
                ReLU(inplace=True) (aa):
                Identity()
                (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn3): DetrFrozenBatchNorm2d() (act3):
                ReLU(inplace=True)
              )
              (3): Bottleneck(
                (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1,
1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1,
```

```
1), padding=(1, 1), bias=False)
                    (bn2): DetrFrozenBatchNorm2d()
                    (drop_block): Identity() (act2):
                    ReLU(inplace=True) (aa):
                    Identity()
                    (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1,
1), bias=False)
                    (bn3): DetrFrozenBatchNorm2d() (act3):
                    ReLU(inplace=True)
                  )
                )
                (layer3): Sequential( (0):
                  Bottleneck(
                    (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1,
1), bias=False)
                    (bn1): DetrFrozenBatchNorm2d() (act1):
                    ReLU(inplace=True)
                    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2,
2), padding=(1, 1), bias=False)
                    (bn2): DetrFrozenBatchNorm2d()
                    (drop_block): Identity() (act2):
                    ReLU(inplace=True) (aa):
                    Identity()
                    (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                    (bn3): DetrFrozenBatchNorm2d() (act3):
                    ReLU(inplace=True) (downsample):
                    Sequential(
                      (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2,
2), bias=False)
                      (1): DetrFrozenBatchNorm2d()
                    )
                  )
                  (1): Bottleneck(
                    (conv1): Conv2d(1024, 256, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                    (bn1): DetrFrozenBatchNorm2d() (act1):
                    ReLU(inplace=True)
                    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                    (bn2): DetrFrozenBatchNorm2d()
                    (drop_block): Identity() (act2):
                    ReLU(inplace=True) (aa):
                    Identity()
                    (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                    (bn3): DetrFrozenBatchNorm2d() (act3):
                    ReLU(inplace=True)
                  )
```

```
            (2): Bottleneck(
              (conv1): Conv2d(1024, 256, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True) (aa):
              Identity()
              (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn3): DetrFrozenBatchNorm2d() (act3):
              ReLU(inplace=True)
            )
            (3): Bottleneck(
              (conv1): Conv2d(1024, 256, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True) (aa):
              Identity()
              (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn3): DetrFrozenBatchNorm2d() (act3):
              ReLU(inplace=True)
            )
            (4): Bottleneck(
              (conv1): Conv2d(1024, 256, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn1): DetrFrozenBatchNorm2d() (act1):
              ReLU(inplace=True)
              (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
              (bn2): DetrFrozenBatchNorm2d()
              (drop_block): Identity() (act2):
              ReLU(inplace=True) (aa):
              Identity()
              (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
              (bn3): DetrFrozenBatchNorm2d() (act3):
              ReLU(inplace=True)
            )
            (5): Bottleneck(
```

```
                (conv1): Conv2d(1024, 256, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity() (act2):
                ReLU(inplace=True) (aa):
                Identity()
                (conv3): Conv2d(256, 1024, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn3): DetrFrozenBatchNorm2d() (act3):
                ReLU(inplace=True)
              )
            )
            (layer4): Sequential( (0):
              Bottleneck(
                (conv1): Conv2d(1024, 512, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2,
2), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity() (act2):
                ReLU(inplace=True) (aa):
                Identity()
                (conv3): Conv2d(512, 2048, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn3): DetrFrozenBatchNorm2d() (act3):
                ReLU(inplace=True) (downsample):
                Sequential(
                  (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2,
2), bias=False)
                  (1): DetrFrozenBatchNorm2d()
                )
              )
              (1): Bottleneck(
                (conv1): Conv2d(2048, 512, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn1): DetrFrozenBatchNorm2d() (act1):
                ReLU(inplace=True)
                (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity() (act2):
                ReLU(inplace=True) (aa):
                Identity()
```

```
                (conv3): Conv2d(512, 2048, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn3): DetrFrozenBatchNorm2d()
                (act3): ReLU(inplace=True)
              )
              (2): Bottleneck(
                (conv1): Conv2d(2048, 512, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn1): DetrFrozenBatchNorm2d()
                (act1): ReLU(inplace=True)
                (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), bias=False)
                (bn2): DetrFrozenBatchNorm2d()
                (drop_block): Identity()
                (act2): ReLU(inplace=True)
                (aa): Identity()
                (conv3): Conv2d(512, 2048, kernel_size=(1, 1),
stride=(1, 1), bias=False)
                (bn3): DetrFrozenBatchNorm2d()
                (act3): ReLU(inplace=True)
              )
            )
          )
        )
        (position_embedding): DetrSinePositionEmbedding()
      )
      (input_projection): Conv2d(2048, 256, kernel_size=(1, 1),
stride=(1, 1))
      (query_position_embeddings): Embedding(100, 256)
      (encoder): DetrEncoder(
        (layers): ModuleList(
          (0-5): 6 x DetrEncoderLayer(
            (self_attn): DetrAttention(
              (k_proj): Linear(in_features=256, out_features=256,
bias=True)
              (v_proj): Linear(in_features=256, out_features=256,
bias=True)
              (q_proj): Linear(in_features=256, out_features=256,
bias=True)
              (out_proj): Linear(in_features=256, out_features=256,
bias=True)
            )
            (self_attn_layer_norm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
            (activation_fn): ReLU()
            (fc1): Linear(in_features=256, out_features=2048, bias=True)
            (fc2): Linear(in_features=2048, out_features=256, bias=True)
            (final_layer_norm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
```

```
          )
        )
      )
      (decoder): DetrDecoder(
        (layers): ModuleList(
          (0-5): 6 x DetrDecoderLayer(
            (self_attn): DetrAttention(
              (k_proj): Linear(in_features=256, out_features=256,
bias=True)
              (v_proj): Linear(in_features=256, out_features=256,
bias=True)
              (q_proj): Linear(in_features=256, out_features=256,
bias=True)
              (out_proj): Linear(in_features=256, out_features=256,
bias=True)
            )
            (activation_fn): ReLU()
            (self_attn_layer_norm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
            (encoder_attn): DetrAttention(
              (k_proj): Linear(in_features=256, out_features=256,
bias=True)
              (v_proj): Linear(in_features=256, out_features=256,
bias=True)
              (q_proj): Linear(in_features=256, out_features=256,
bias=True)
              (out_proj): Linear(in_features=256, out_features=256,
bias=True)
            )
            (encoder_attn_layer_norm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
            (fc1): Linear(in_features=256, out_features=2048, bias=True)
            (fc2): Linear(in_features=2048, out_features=256, bias=True)
            (final_layer_norm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
          )
        )
        (layernorm): LayerNorm((256,), eps=1e-05,
elementwise_affine=True)
      )
    )
    (class_labels_classifier): Linear(in_features=256, out_features=92,
bias=True)
    (bbox_predictor): DetrMLPPredictionHead(
      (layers): ModuleList(
        (0-1): 2 x Linear(in_features=256, out_features=256, bias=True)
        (2): Linear(in_features=256, out_features=4, bias=True)
      )
```

```
)
)
```

1. Vision Transformers for Image Segmentation Segmentasi gambar menggunakan Vision Transformers untuk membagi gambar menjadi segmen-segmen kecil.
2. OneFormer OneFormer menggabungkan pendekatan Vision Transformer untuk berbagai tugas penglihatan komputer dalam satu model.
3. Knowledge Distillation with Vision Transformers Distilasi pengetahuan memungkinkan Vision Transformers besar untuk mengajarkan model yang lebih kecil.

```
!pip install transformers --upgrade Requirement

already satisfied: transformers in
/usr/local/lib/python3.10/dist-packages (4.47.1)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.27.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.2) Requirement
already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.10/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.24.0->transformers) (2024.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.24.0->transformers) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
```

```
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2024.12.14)

from transformers import AutoImageProcessor,
AutoModelForImageClassification

# Use a model designed for image classification, e.g., 'google/vit- base-patch16-224'
distil_model = AutoModelForImageClassification.from_pretrained("google/vit-base-
patch16-224")
print(distil_model)

{"model_id":"6012e44344f24df68300a8beb857a950","version_major":2,"vers
ion_minor":0}

{"model_id":"c3f17c2d82b1460bbee0d4e950c9acda","version_major":2,"vers
ion_minor":0}

ViTForImageClassification(
  (vit): ViTModel(
    (embeddings): ViTEmbeddings(
      (patch_embeddings): ViTPatchEmbeddings(
        (projection): Conv2d(3, 768, kernel_size=(16, 16), stride=(16,
16))
      )
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): ViTEncoder(
      (layer): ModuleList(
        (0-11): 12 x ViTLayer(
          (attention): ViTSdpaAttention(
            (attention): ViTSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (output): ViTSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
```

```
            )
          )
          (intermediate): ViTIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): ViTOutput(
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (dropout): Dropout(p=0.0, inplace=False)
          )
          (layernorm_before): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
          (layernorm_after): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
        )
      )
    )
    (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
  )
  (classifier): Linear(in_features=768, out_features=1000, bias=True)
)
```

UNIT 4

1. Exploring Multimodal Text and Vision Models - Uniting Senses in AI Multimodal models menggabungkan teks dan gambar untuk memahami konteks gabungan. Contoh aplikasi: CLIP (menghubungkan teks dan gambar), DALL-E (membuat gambar dari teks), dan BLIP (menghasilkan teks dari gambar).

2. A Multimodal World Mengacu pada dunia di mana model memahami berbagai modalitas data, seperti teks, gambar, audio, dan video. Contoh: Sistem AI yang dapat menjelaskan gambar (teks), mengenali suara, dan memahami kombinasi semuanya.

3. Introduction to Vision Language Models Model Vision Language mempelajari hubungan antara teks dan gambar. Contoh: CLIP (Contrastive Language–Image Pre-training) yang menghubungkan representasi gambar dan teks dalam ruang vektor yang sama.

```
from transformers import CLIPProcessor, CLIPModel
from PIL import Image
import requests
import os

# URL gambar dari web (ganti dengan URL yang valid)
image_url =
"https://upload.wikimedia.org/wikipedia/commons/4/47/PNG_transparency_
demonstration_1.png"
```

```
image_path = "example.jpg"

# Mengunduh gambar jika tidak ada
if not os.path.exists(image_path):
    print("Gambar tidak ditemukan, mengunduh dari web...") response =
    requests.get(image_url)
    if response.status_code == 200:
        with open(image_path, "wb") as f:
            f.write(response.content)
    else:
        print("Gagal mengunduh gambar. Periksa URL.")
        exit()

# Memastikan file yang diunduh adalah gambar
try:
    image = Image.open(image_path)
    image.verify()          # Memeriksa apakah file adalah gambar yang valid
    image = Image.open(image_path)          # Buka ulang setelah verifikasi
except Exception as e:
    print(f"File yang diunduh bukan gambar yang valid: {e}")
    exit()

# Memuat model CLIP
print("Memuat model CLIP...")
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32") processor =
CLIPProcessor.from_pretrained("openai/clip-vit-base- patch32")

# Input gambar dan teks
text = ["a cat", "a dog"]

# Memproses data
print("Memproses data...")
inputs = processor(text=text, images=image, return_tensors="pt", padding=True)

# Mendapatkan skor kesesuaian antara teks dan gambar
print("Mendapatkan skor kesesuaian...") outputs =
model(**inputs)
logits_per_image = outputs.logits_per_image          #Skorgambarterhadap
teks
probs = logits_per_image.softmax(dim=1)          # Probabilitas

# Menampilkan hasil
for i, t in enumerate(text):
    print(f"Probabilitas bahwa gambar adalah '{t}': {probs[0,
i].item():.4f}")
```

```
Gambar tidak ditemukan, mengunduh dari web... Memuat
model CLIP...
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings
tab (https://huggingface.co/settings/tokens), set it as secret in your
Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks. Please note
that authentication is recommended but still optional to access public models
or datasets.
  warnings.warn(

{"model_id":"20abe3d117194d9c94886d39a2258645","version_major":2,"vers
ion_minor":0}

{"model_id":"7e895085b37d4875a5d2050513213070","version_major":2,"vers
ion_minor":0}

{"model_id":"034b822b77504141a6a837bb391c3307","version_major":2,"vers
ion_minor":0}

{"model_id":"993f397fea004600b4efb21e59f608d7","version_major":2,"vers
ion_minor":0}

{"model_id":"f483d154a9e84679a0a5fcea1889244e","version_major":2,"vers
ion_minor":0}

{"model_id":"834f70f8cfdf404283f2a480f27cf7ce","version_major":2,"vers
ion_minor":0}

{"model_id":"db88b37b5d644f47ad1a6273b933a889","version_major":2,"vers
ion_minor":0}

{"model_id":"f47730d39ff64ba6b579323b597c115c","version_major":2,"vers
ion_minor":0}

Memproses data...
Mendapatkan skor kesesuaian...
Probabilitas bahwa gambar adalah 'a cat': 0.3104 Probabilitas
bahwa gambar adalah 'a dog': 0.6896
```

Penjelasan:

CLIPProcessor: Memproses gambar dan teks agar sesuai dengan model CLIP. logits_per_image: Skor kecocokan gambar terhadap teks.

1. Multimodal Tasks and Models Tugas multimodal meliputi klasifikasi teks dan gambar, pencocokan teks-gambar, deteksi objek multimodal, dll. Contoh: Visual Question Answering (VQA) di mana model menjawab pertanyaan tentang gambar.

2. CLIP and Relatives CLIP adalah model utama yang mendasari banyak model multimodal lainnya, seperti DALL-E dan BLIP. Relatives-nya mencakup model-model seperti ALIGN dan Florence.

3. Losses Loss functions dalam model multimodal, seperti Contrastive Loss untuk mengoptimalkan kesesuaian antara teks dan gambar.

```
import torch
import torch.nn as nn

# Skor antara teks dan gambar
logits = torch.tensor([[0.8, 0.2], [0.3, 0.7]])
labels = torch.tensor([0, 1])

# Loss fungsi
loss_fn = nn.CrossEntropyLoss()
loss = loss_fn(logits, labels)
print(f"Loss: {loss.item()}")
```

```
Loss: 0.47525161504745483
```

1. Contrastive Language-Image Pre-training (CLIP) CLIP mempelajari hubungan antara teks dan gambar dengan memaksimalkan kesesuaian pasangan teks-gambar yang benar.
2. Multimodal Text Generation (BLIP) BLIP (Bootstrapping Language-Image Pre-training) adalah model untuk menghasilkan teks dari gambar, seperti menghasilkan deskripsi gambar.

```
from transformers import BlipProcessor, BlipForConditionalGeneration from
PIL import Image
import urllib.request from
io import BytesIO

# URL gambar
image_url = "https://upload.wikimedia.org/wikipedia/commons/4/47/PNG_transparency_
demonstration_1.png"

# Mengunduh gambar dari URL
try:
    with urllib.request.urlopen(image_url) as response: image
        = Image.open(BytesIO(response.read()))
except Exception as e:
    print(f"Error mengunduh gambar: {e}")
    exit()

# Memuat model BLIP
print("Memuat model BLIP...")
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-
captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip- image-
captioning-base")
```

```
# Memproses gambar
print("Memproses gambar...")
inputs = processor(images=image, return_tensors="pt")

# Menghasilkan teks deskripsi print("Menghasilkan
deskripsi gambar...") output =
model.generate(**inputs)
caption = processor.decode(output[0], skip_special_tokens=True)

# Menampilkan hasil
print(f"Caption: {caption}")
```

```
Memuat model BLIP...
```


```
{"model_id":"101b0405538f4082b531f031ec334e7f","version_major":2,"vers
ion_minor":0}

{"model_id":"13a98e18cb3a4f738c238a63e81440c4","version_major":2,"vers
ion_minor":0}

{"model_id":"2110ef432b974afdbadcc798ba59e6fb","version_major":2,"vers
ion_minor":0}

{"model_id":"0530da626d2647f08b2d0ac2a1f39da6","version_major":2,"vers
ion_minor":0}

{"model_id":"a4a524c36f2d4418bc93a237023a9d93","version_major":2,"vers
ion_minor":0}

{"model_id":"c27347809e0e4ed4bcdb7dbb34f2e288","version_major":2,"vers
ion_minor":0}

{"model_id":"493aa740c9e44da3ba774d31094d79e1","version_major":2,"vers
ion_minor":0}
```


```
Memproses gambar... Menghasilkan
deskripsi gambar...
Caption: two dice with one red and one green
```

1. Multimodal Object Detection (OWL-ViT) OWL-ViT adalah model deteksi objek multimodal yang menggunakan teks dan gambar untuk mendeteksi objek dalam gambar.

```
from transformers import OwlViTProcessor, OwlViTForObjectDetection
from PIL import Image

# Memuat model OWL-ViT
processor = OwlViTProcessor.from_pretrained("google/owlvit-base-
patch32")
model = OwlViTForObjectDetection.from_pretrained("google/owlvit-base-
patch32")
```

```
# Input gambar dan teks
image = Image.open("example.jpg").convert("RGB")  # Convert to RGB
texts = ["a cat", "a dog"]

# Memproses input
inputs = processor(text=texts, images=image, return_tensors="pt")

# Deteksi objek
outputs = model(**inputs)
print(outputs)
```

```
OwlViTObjectDetectionOutput(loss=None, loss_dict=None,
logits=tensor([[[ -8.3236,  -9.6619],
         [ -8.5080,  -9.4863],
         [ -8.9768,  -9.9662],
         [ -8.7752,  -9.9611],
         [-14.1851, -14.3139]]], grad_fn=<WhereBackward0>),
pred_boxes=tensor([[[0.1196, 0.0982, 0.2367, 0.2081],
         [0.1488, 0.0213, 0.3097, 0.0430],
         [0.1934, 0.0200, 0.3668, 0.0399],
         ...,
         [0.8724, 0.9997, 0.2512, 0.2077],
         [0.9999, 0.9999, 0.2641, 0.1440]]],
grad_fn=<SigmoidBackward0>), text_embeds=tensor([[[ 0.0116,  0.0415, -
0.0899,  ...,  0.0203, -0.0425, -0.0035],
         [ 0.0131,  0.0636, -0.0904,  ...,  0.0226, -0.0090, -
0.0083]]],
       grad_fn=<ViewBackward0>), image_embeds=tensor([[[[ 3.5582e-01,
-2.8430e-01,  1.4926e-01,  ...,  6.2385e-02,
           7.2613e-01,  1.0921e+00],
         [ 4.1576e-01, -1.7261e-01,  9.2260e-02,  ...,  1.5630e-03,
           5.8427e-01,  7.6387e-01],
         [ 3.7357e-01, -1.8184e-01,  1.1331e-01,  ..., -1.0034e-02,
           5.3163e-01,  8.1191e-01],
         ...,
         [ 2.3372e-01,  4.2824e-02,  5.2720e-02,  ..., -1.1133e-01,
           3.0231e-01,  8.2453e-01],
         [ 1.1568e-01,  4.1856e-02,  1.3733e-01,  ...,  2.0058e-02,
           1.4116e-01,  7.3146e-01],
         [ 6.0391e-02,  3.1123e-01, -1.6099e-01,  ...,  2.7281e-02,
           2.8968e-02,  4.5994e-01]],

        [[ 3.1388e-01, -1.0650e-01,  5.2082e-02,  ..., -3.0755e-01,
           6.5623e-01, -2.2183e-03],
         [ 3.1975e-01, -6.5208e-02,  3.1178e-02,  ..., -3.0910e-01,
           5.1345e-01,  4.9320e-01],
```

```
                3.7668e-01,  7.3501e-01]],
       [[ 2.1611e-01, -1.9925e-01, -1.2372e-02, ..., -2.5222e-01,
         ...,
        [ 1.0870e-01,  2.2828e-01,  1.0118e-02, ..., -5.5721e-01,
          4.4434e-01,  4.6520e-01],
        [-2.4009e-01,  2.5897e-01,  1.4455e-01, ..., -6.5058e-01,
          4.3856e-01,  5.1929e-01],
        [-2.8475e-01,  3.7478e-01,  1.3252e-01, ..., -2.7003e-01,
          7.6027e-02,  1.0940e+00]],

       [[ 4.6160e-01,  1.1588e-01,  1.6027e-01, ..., -1.5656e-01,
          6.1706e-01, -4.9265e-02],
        [ 1.2408e-01,  9.9920e-03,  1.1253e-01, ..., -3.3554e-01,
          5.1455e-01,  6.1423e-01],
        [ 9.9816e-02, -1.9030e-01,  2.2319e-01, ..., -5.1750e-01,
          3.1600e-01,  8.7069e-01],
         ...,
        [ 6.0955e-04,  3.6975e-01,  1.5490e-01, ..., -4.3214e-01,
          3.1591e-01,  4.5725e-01],
        [-8.5913e-03,  1.7954e-01,  1.6426e-01, ..., -5.0522e-01,
          1.2322e-01,  1.1432e+00],
        [ 2.7707e-01,  3.7983e-01,  2.2342e-01, ..., -7.5790e-01,
          2.4677e-01,  3.5034e-01]],

       [[ 6.0080e-01, -4.3290e-02,  6.2004e-01, ..., -4.6292e-01,
          4.8503e-01, -8.2820e-02],
        [ 6.2004e-01,  1.9054e-01,  1.5089e-01, ..., -4.2148e-01,
          3.7270e-01,  1.4703e-02],
        [ 3.9698e-01, -4.4684e-01,  3.4714e-02, ..., -5.4062e-01,
         -9.3736e-03,  2.7538e-01],
         ...,
        [ 6.1001e-01,  1.4925e-01,  1.0002e-01, ..., -7.8703e-01,
          1.8165e-01,  2.6218e-01],
        [ 3.3832e-01,  6.0157e-02,  1.7844e-01, ..., -4.2127e-01,
          6.0030e-02,  5.5429e-01],
        [ 7.9885e-02,  3.2766e-01, -2.4102e-01, ..., -1.2570e-02,
         -4.2046e-02,  3.3182e-01]],

       [[ 3.9865e-01, -9.5197e-02,  2.1538e-01, ..., -4.3487e-02,
          6.0242e-01,  5.9516e-01],
        [ 3.0424e-01, -2.2531e-01,  1.4977e-01, ...,  1.4248e-02,
          2.2055e-01,  4.9476e-01],
        [ 8.0732e-01,  1.6269e-01,  2.6407e-01, ..., -2.0115e-02,
          2.3399e-01,  4.2938e-01],
         ...,
        [ 6.9091e-01,  1.3940e-01,  1.2246e-01, ..., -2.0469e-01,
         -9.2186e-02,  1.2488e-01],
        [ 5.1892e-01,  1.5400e-01,  1.9718e-01, ..., -2.2689e-01,
```

```
                4.5454e-02,  3.5248e-01],
              [-2.5721e-02,  3.1963e-01, -5.5130e-03,  ...,  8.7767e-02,
               -8.1640e-02,  6.5097e-01]],

             [[ 1.0761e-02,  1.1647e-01, -2.5931e-01,  ..., -7.9030e-02,
                5.1821e-02,  4.7146e-01],
              [ 4.2574e-01,  1.9252e-01,  3.3106e-01,  ..., -1.0216e-01,
                2.4519e-01,  8.4105e-01],
              [ 4.2899e-01,  2.1127e-01,  2.7107e-01,  ..., -1.8511e-02,
                2.3509e-01,  7.6581e-01],
              ...,
              [ 8.2224e-03,  1.7046e-01,  9.0105e-02,  ..., -5.7398e-02,
               -1.0615e-01,  6.7680e-01],
              [ 6.7372e-02,  2.3852e-01, -2.9456e-01,  ..., -1.2261e-01,
               -1.2838e-01,  3.4907e-01],
              [ 2.0724e-01,  2.5018e-01, -4.3864e-01,  ..., -2.6876e-02,
                6.5065e-02, -1.7811e-01]]]), grad_fn=<ViewBackward0>),
class_embeds=tensor([[[-0.0324, -0.0150, -0.0136,  ..., -0.0411,
0.0187,  0.0384],
              [-0.0293,  -0.0334, -0.0491,  ..., -0.0463,  0.0484,  0.0784],
              [-0.0300,  -0.0327, -0.0546,  ..., -0.0417,  0.0384,  0.0883],
              ...,
              [-0.0325,  0.0173, -0.0004,  ..., -0.0333,  0.0067, -0.0505],
              [-0.0472,  -0.0002,  0.0424,  ..., -0.0501,  0.0206, -0.0578],
             [-0.0444, -0.0196,  0.0521,  ..., -0.0448,  0.0584, -
0.0238]]],
            grad_fn=<DivBackward0>),
text_model_output=BaseModelOutputWithPooling(last_hidden_state=tensor( [[[
0.2850, 0.1107,  0.1510,  ...,  0.2730,  0.6234,  0.1871],

             [ 0.1997,  0.8008,  0.4034,  ...,  0.7576,  0.9788,  0.1406],
             [ .,4550, -1.0130,  1.4609,  -0.2548,  0.1609, -0.9661],
            [[0.2401,  0.1273,  0.2129,  ...,  0.2735,  0.6116,  0.1732],
            [ 0.2396,  0.1275,  0.2121,  ...,  0.2734,  0.6115,  0.1729],
0.1735]],

            [[ 0.2850,  0.1107,  0.1510,  ...,  0.2730,  0.6234,  0.1871],
             [ 0.1997,  0.8008,  0.4034,  ...,  0.7576,  0.9788,  0.1406],
              [-0.2384,  0.0783,  1.6396,  ...,  0.4354,  0.2103,  0.0416],
             ...,
             [ 0.2697,  0.1778,  0.2767,  ...,  0.2657,  0.6164,  0.2532],
             [ 0.2689,  0.1775,  0.2755,  ...,  0.2656,  0.6162,  0.2524],
              [ 0.2696,  0.1759,  0.2799,  ...,  0.2671,  0.6155,
0.2543]]],
            grad_fn=<NativeLayerNormBackward0>), pooler_output=tensor([[-
0.2785,  0.0856,  1.4236,  ...,  0.7624,  0.3804, -0.1104],
             [ 0.1095,  0.6405,  2.6134,  ...,  0.4068,  0.1522,  0.5055]],
            grad_fn=<IndexBackward0>),  hidden_states=None,
```

attentions=None),
vision_model_output=BaseModelOutputWithPooling(last_hidden_state=tenso r([[[
22.1317, -13.6000, 10.0231, ..., -21.3690, 16.2190,
18.4783],
[0.8209, 1.5430, 0.8662, ..., -0.3643, 2.1661,
2.7722],
[1.2603, 1.3748, 0.6200, ..., -0.1053, 2.1263,
2.3480],
...,
[-0.1617, -0.1146, 0.5836, ..., -0.0584, -0.5572,
1.6379],
[0.1795, -0.5457, -1.3137, ..., -0.1313, -0.4502,
0.9588],
[1.0838, -1.0439, -2.5113, ..., -0.9885, 0.6093, -
0.5448]]],
grad_fn=<AddBackward0>), pooler_output=tensor([[1.3454, - 1.0544,
0.6831, -0.4825, -0.5693, 1.0428, 1.3054, 0.9947,
-1.3114, 1.9017, -1.4265, -1.0055, 1.2163, 1.1747,
1.0455, 0.4001,
-1.4837, 0.8065, 1.3712, 0.9853, 0.6108, 0.3397,
1.2000, 1.1622,
0.2706, 1.1122, 1.0824, 0.9774, -0.9521, 1.1782,
0.3986, -0.5751,
-0.8805, -1.1644, -0.6660, 0.8834, -0.4203, -1.3948,
0.9104, -0.2646,
-0.7002, -0.7261, -0.2947, 0.1348, 0.9735, 0.8185,
0.7039, 1.3040,
0.7913, 0.1930, -1.4600, 0.4501, 1.0061, -1.3729, -
1.7223, -0.5026,
1.1064, -0.9804, -0.5842, -0.4743, 0.9661, -0.7153,
0.9737, -0.7714,
1.1683, -0.0855, -1.8538, 0.1426, 1.1678, 0.6559, -
1.0399, 0.7908,
1.2099, 0.9796, 1.3043, 1.0245, 0.6029, 0.9183, -
0.4997, 0.4970,
0.7293, 0.2646, 0.9436, 0.1467, -0.0241, -0.4934, -
1.3269, -1.0465,
0.8140, 1.0607, 1.0269, 1.1194, -0.6040, 0.0711, -
0.4731, 0.7830,
1.2406, 1.2760, 0.7908, -1.6138, 0.9796, 1.1673, -
1.0609, 0.1020,
0.7011, 1.0413, 0.8997, 0.5039, -0.2886, 0.5515, -
1.0038, 1.1965,
-0.7445, -0.2622, -0.4495, 0.0651, 2.3454, -0.6412,
1.0101, -1.2278,
-1.0205, 1.3483, -0.1588, 1.2017, -0.5859, -1.2193, -
0.6386, -0.9985,
-1.2105, 1.1206, 1.2856, 0.6722, 1.4530, -0.5922, -
1.1426, -1.3673,

-0.5542, -0.1382, -1.1271, -1.0592, 1.0232, -0.5105,
0.6625, 0.8489,
0.9474, -0.3818, 0.9308, -0.7194, -0.9134, -1.0070, -
0.7855, -1.0178,
-1.3195, 0.5391, -0.3089, 0.7398, 1.1478, 1.1645,
1.1754, -1.1764,
1.1264, 0.7873, 0.9979, 1.0723, 1.0524, 0.6876,
0.6573, 0.0597,
-0.7940, -1.1800, -0.4979, -1.3269, 1.2571, -0.6451,
1.3084, 0.8675,
0.9311, -0.4966, -1.2817, 1.6671, 0.9993, 1.3484,
1.2996, -1.0199,
1.0017, 0.9661, -0.8473, -1.3070, 1.1510, -0.9891,
1.1721, -1.2845,
0.6956, 1.2241, 0.0457, 1.3220, -1.0105, 1.1720, -
0.2011, -1.3824,
-0.9277, 1.0366, -0.5447, 1.1476, -0.1856, -1.2935, -
0.3119, 0.0391,
-1.1569, -1.2138, 0.9092, -1.2128, 1.7030, 1.3062,
1.3341, 1.1576,
1.9564, 0.8803, 0.2769, 1.2588, 1.2495, -1.2247, -
0.1016, 0.8782,
-0.4188, 1.1749, 0.5200, 0.7737, -0.9461, 0.7419,
0.9922, -1.0753,
1.2689, -1.2729, -1.2359, 0.2563, -0.6669, -0.2291, -
0.0097, 1.0530,
-0.9564, 0.4153, 0.0608, 0.9283, -1.3227, -0.9181,
0.5230, -0.2382,
1.0592, -1.5625, 1.1349, -0.4338, -1.2555, 0.4688, -
1.3955, -0.5901,
0.2075, 1.6170, 1.0053, 1.3747, 1.2807, -0.6833,
0.8636, 1.1333,
-1.1853, 1.0623, -1.1400, -0.5290, 0.8306, -1.1518,
1.1602, 0.7709,
0.6650, -0.7769, 0.8346, -0.4362, 0.7246, -0.2611,
0.8608, -1.1788,
0.7900, 0.7591, -1.2509, 0.5635, -0.3020, 1.4281, -
0.1796, -1.2660,
-0.9154, 1.0872, 1.1143, 1.1856, -1.0749, -0.4230,
0.4398, -0.9201,
-0.2939, 0.7924, 0.8199, 1.1654, -1.2859, 0.6806, -
1.3177, 0.6781,
-1.3219, 0.4333, -0.3271, 1.0693, 0.6021, 5.9944,
0.6751, -1.1324,
-0.6423, -0.5081, -0.3915, -0.6703, 0.0988, -0.0245,
1.5204, 1.0850,
1.6329, -1.0554, -1.0677, 1.2374, -1.4658, -0.3832, -
1.1571, 0.6528,
0.6724, 1.1749, 1.2642, 0.7615, -1.1472, 1.2882,

```
1.2601, -1.0013,
        -0.7856,  0.9880,  1.0114, -1.2108, -1.0997, -0.6609, -
1.2311,  0.9001,
         0.0067, -0.9758, -0.4330,  1.1049, -0.5373,  0.2225,
1.1270,  0.2713,
         0.5749, -0.4861,  1.2907, -0.0409,  1.1705, -0.9824,
1.3097,  1.2868,
         0.7993,  0.9733, -0.5762, -1.2035,  0.6486, -1.2507, -
1.2548,  0.7396,
        -1.1864,  0.9829,  1.1089, -1.3009,  1.0099, -0.6685, -
1.0620,  0.3579,
         1.0483,  0.8770, -0.9417,  0.6121, -0.8729,  0.7837,
0.8675, -0.1003,
         0.0807, -1.4298,  1.1516, -1.1571, -0.1375,  0.9799,
0.8952,  0.4309,
         1.2446, -1.3540,  1.2592,  1.2669, -0.8894,  1.1212,
1.0178,  0.8411,
        -1.2621, -1.1167,  1.1254,  1.3112,  1.3235,  1.3577,
0.8762, -1.2720,
         1.1139,  0.7816, -1.2387,  0.6757, -0.8276, -0.0753,
1.0204, -0.4302,
         1.1595,  1.3638,  1.3654,  1.0884, -0.3945,  1.2671, -
1.1826, -1.1072,
         1.6447, -0.1253,  0.0141, -0.3538,  0.5820,  0.6034,
1.2672,  1.0176,
         0.6553,  1.1150,  0.9244, -0.8923,  1.0819,  1.3992,
1.0378, -0.3411,
        -1.4627, -1.0570,  0.8280,  1.1882, -1.3589, -1.0749, -
0.9855,  0.9372,
         1.1456,  1.0801,  0.6629, -1.4385,  0.8050,  1.0476,
1.2395, -1.0450,
        -0.1498, -1.0458, -0.4971,  0.9035,  0.7928,  1.3007, -
0.5024,  0.7651,
        -1.1324,  1.1983,  1.0533, -1.2749,  1.0870, -1.2070,
0.9589,  1.4618,
         1.1284, -0.6232,  1.1881, -0.7113, -0.4698,  1.1465,
0.8326,  1.0215,
         1.3051,  1.0511, -1.0284,  1.2328,  1.1469,  1.1876, -
0.5732,  0.9238,
        -1.1869,  0.9889,  1.0520,  1.1740,  1.3219, -0.5374,
1.1315,  0.7807,
        -0.9109, -0.9027,  0.6881,  1.1068,  0.3469,  0.3239, -
1.2442, -1.1838,
        -0.7024,  1.3709,  0.8274,  1.0734,  1.2666,  0.7667, -
1.2350, -1.0785,
         0.7760, -0.2697,  1.1674,  0.9512,  0.7297,  1.2121,
0.9222,  0.7841,
        -0.5238, -1.1074,  0.8300,  1.1162, -0.6536,  0.9144,
1.3114, -0.0133,
```

```
          -0.8819,  1.2364,  0.7427,  0.5229,  0.4261,  0.9683, -
0.4401, -0.4050,
           1.1333, -1.0185,  1.0888,  0.8275,  0.8262,  0.9905, -
0.3950,  0.0925,
           1.0745,  1.2794, -1.0763,  0.8971,  1.2417,  1.2310, -
0.3377, -0.0602,
          -0.6534,  1.1376, -1.2818,  0.9341, -0.0463,  1.9507,
1.2923,  1.1477,
           0.7969,  0.7250, -0.8371,  1.0344,  0.9755,  0.4688,
1.4543, -1.2025,
           0.1835, -0.3066,  0.0552, -0.3094, -1.2306,  0.8935,
0.8154, -0.2883,
           0.2345,  1.1927,  0.3487,  1.7668,  0.9483,  1.1902,
0.7699,  0.5760,
          -0.1077, -0.1476,  0.6967,  0.9416, -0.8350,  0.7245,
1.0363,  0.2315,
           0.6526, -0.8144,  0.8976, -0.3623,  0.3706,  1.1336, -
1.0973, -1.0399,
          -1.3007, -1.1218, -0.4903,  1.1863,  2.3659, -1.8301,
0.6891,  0.8081,
          -1.0644, -1.1551,  1.2128,  0.7699,  0.5822, -0.8037,
1.1317,  0.9959,
          -0.1637,  0.8323,  1.1654,  0.7830, -0.7202,  1.4069,
0.4894,  0.6297,
          -0.9195,  4.6587, -0.7664, -1.0718, -1.2824, -1.0093,
0.5507, -1.0094,
           0.8575,  0.7909, -1.5164, -0.5052,  0.9614,  1.0426,
1.1114,  0.8266,
           0.8518, -0.1112,  1.3333, -0.5222, -1.1908, -0.5468,
0.9133, -1.2650,
          -0.4309, -0.8321,  1.1541,  0.9589, -1.3366, -0.9352, -
1.2021,  1.1675,
          -0.9999,  1.0273,  0.7066, -0.3524,  1.0002, -0.7203, -
0.6906, -1.0926,
          -0.5253, -1.1197,  0.6073, -1.1342, -0.2808, -0.9214, -
1.1386, -0.7596,
          -0.6724, -1.1240, -0.2167, -1.1169,  0.5483,  1.0474, -
0.9110, -1.1606,
          -0.2699, -1.2281,  1.3454, -1.1183,  1.2652,  0.5296,
0.9155,  0.8604,
           1.1033,  1.4239, -0.7368,  1.0769, -1.1239,  0.2889, -
1.1212,  0.9161,
           1.1078,  1.1868, -0.6631,  0.6401,  1.0821,  0.9152,
0.2904,  0.5435,
           1.1464, -0.9250,  1.8443,  0.7968,  1.0508, -1.2130,
1.2371, -1.4894,
           0.9858,  0.9485,  0.5747,  0.8121, -1.6350, -1.1534, -
1.2852, -1.5433,
           0.7149,  0.7482, -0.7036,  1.2287,  1.1060,  0.7354,
```

```
0.5347,  0.3929,
          1.0140, -1.0760,   0.6813,   0.4842,   0.7254,   0.6762,
0.0606, -0.8701,
         -0.4802,   1.2077,   1.2099, -1.2456,   1.1782,   1.0627,
1.0289, -1.0287,
          1.2851, -1.0945,   1.0848, -0.4002,   0.4263,   1.1901,
0.7812, -0.1926,
          1.0457,   1.3768,   0.6721, -1.0472,   0.5068,   1.0484,
1.2187,  1.0342,
          1.0524, -0.1800, -1.4739, -0.7040,   0.7009, -1.1226,
1.0932,  1.2487]],
        grad_fn=<NativeLayerNormBackward0>),   hidden_states=None,
attentions=None))
```

1. Transfer Learning of Multimodal Models Transfer learning memungkinkan model multimodal seperti CLIP atau BLIP dilatih ulang untuk tugas spesifik dengan data baru.

2. Supplementary Reading and Resources Bagian ini mencakup sumber bacaan tambahan untuk mempelajari model-model multimodal lebih lanjut, seperti makalah penelitian dan dokumentasi model. Kesimpulan Kode di atas mencakup implementasi beberapa model utama, seperti:

CLIP untuk mencocokkan teks dan gambar. BLIP untuk menghasilkan teks dari gambar. OWL-ViT untuk deteksi objek multimodal.

lampiran

https://youtu.be/dFmBRlYovlQ?si=LPBHQHEg1EGhdMy0