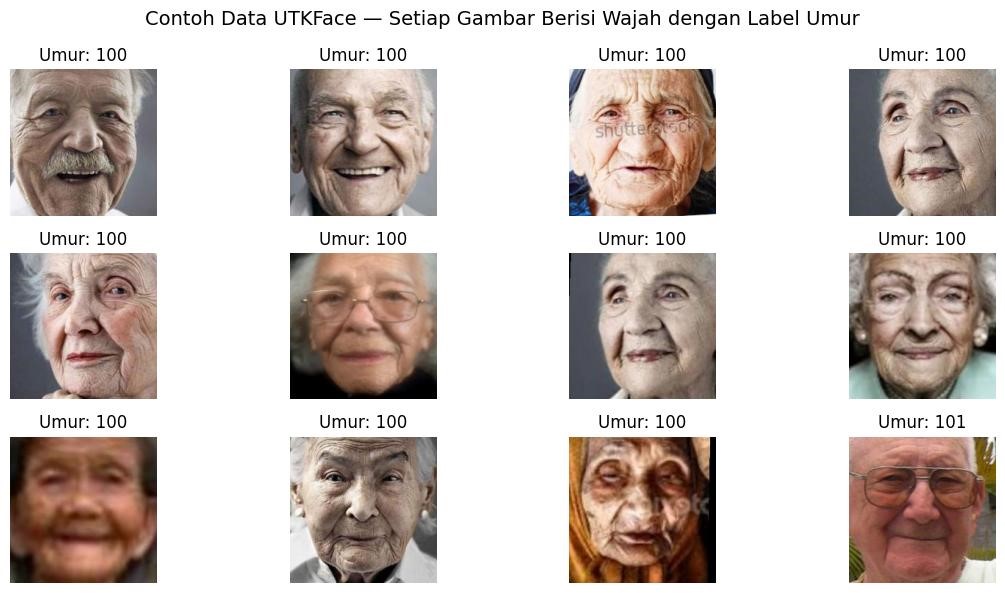
**JOBSHEET 03:**

**TEKNIK REGRESI GAMBAR**



**Dosen:**

Ir. Prayitno, S.ST., M.T., Ph.D.

**Nama Mahasiswa :** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NIM Mahasiswa :** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Tahun Akademik 2025

Dalam bidang visi komputer, *regresi gambar* (image regression) merupakan teknik untuk memprediksi nilai numerik berdasarkan informasi visual dari suatu gambar. Berbeda dengan *klasifikasi gambar* yang mengelompokkan ke dalam kategori diskret (misalnya “anjing”, “kucing”, “mobil”), regresi gambar mempelajari hubungan antara fitur gambar dan nilai kontinu, misalnya memprediksi **umur seseorang dari wajah**, **tinggi air sungai dari citra drone**, atau **intensitas emisi gas dari citra termal**.

Melalui praktikum ini, mahasiswa akan belajar bagaimana model *deep learning* seperti **Convolutional Neural Network (CNN)** dapat digunakan untuk menyelesaikan masalah regresi, dimulai dari persiapan dataset, pelatihan model, hingga evaluasi performa menggunakan metrik seperti **MAE (Mean Absolute Error)** dan **R² Score**.

# A. Tujuan Instruksional Khusus

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu untuk:

1. **Menjelaskan konsep regresi gambar** dan membedakannya dari klasifikasi serta segmentasi citra.
2. **Membangun dataset sintetis** untuk memahami hubungan antara representasi visual dan nilai kontinu (Praktikum D1).
3. **Menerapkan regresi gambar menggunakan CNN** pada data dunia nyata untuk memprediksi nilai kuantitatif seperti umur manusia (Praktikum D2).
4. **Menerapkan transfer learning** untuk memprediksi nilai estetika visual (Pawpularity) dengan data citra hewan peliharaan (Praktikum D3).
5. **Mengevaluasi performa model regresi gambar** menggunakan metrik MAE, RMSE, dan R², serta memvisualisasikan hasil prediksi terhadap nilai sebenarnya.
6. **Menafsirkan hasil regresi gambar secara kritis** dengan memperhatikan bias data dan interpretasi hasil model.

# B. Alat dan Bahan

Untuk melaksanakan praktikum ini, diperlukan:

|  |  |  |
| --- | --- | --- |
| **No** | **Alat/Bahan** | **Keterangan** |
| 1 | Google Colab | Lingkungan coding Python berbasis GPU |
| 2 | Python 3.x | Bahasa pemrograman utama |
| 3 | TensorFlow / Keras | Framework deep learning |
| 4 | NumPy, Matplotlib, scikit-learn | Analisis data dan visualisasi |
| 5 | Kaggle API (kaggle.json) | Akses dataset publik UTKFace dan Pawpularity |
| 6 | Dataset: UTKFace | Prediksi umur dari wajah |
| 7 | Dataset: Pawpularity | Prediksi skor estetika foto hewan |
| 8 | Dataset sintetis (dibuat sendiri) | Untuk pengantar regresi citra (Praktikum D1) |

# C. Dasar Teori

Bayangkan sebuah sistem komputer yang mampu menebak umur seseorang hanya dari potret wajahnya, atau memperkirakan kedalaman sungai dari citra udara yang diambil drone. Sistem seperti ini bekerja bukan dengan menebak kategori seperti “muda” atau “tua”, melainkan dengan memprediksi **nilai numerik yang terus-menerus berubah** — misalnya 22,5 tahun atau 3,2 meter. Nah, inilah yang disebut **teknik regresi gambar** (*image regression*).

Dalam visi komputer, regresi gambar bertujuan menemukan fungsi 𝑓(𝑥) yang mampu memetakan citra 𝑥 menjadi nilai kontinu 𝑦. Secara matematis, hubungan tersebut dapat ditulis sederhana sebagai:

𝑦 = 𝑓(𝑥) + 𝜀

di mana 𝜀 adalah kesalahan atau *noise* yang muncul karena faktor lingkungan atau keterbatasan model. Tugas utama kita adalah mencari parameter terbaik dari 𝑓(𝑥) agar selisih antara nilai prediksi dan nilai sebenarnya sekecil mungkin, biasanya dengan meminimalkan **fungsi kehilangan** (loss function):

𝑁

1

min∑(𝑦𝑖 − 𝑓(𝑥𝑖;𝜃))

𝜃 𝑁

𝑖=1

Tujuan utama teknik regresi gambar adalah **menerjemahkan informasi visual menjadi angka bermakna**. Kalau klasifikasi ibarat menjawab “apa ini?”, maka regresi menjawab “berapa nilainya?”. Aplikasinya sangat luas di dunia nyata. Dalam bidang **pengolahan wajah**, regresi gambar digunakan untuk **estimasi umur manusia** berdasarkan ekspresi, tekstur kulit, dan bentuk wajah. Di dunia **otomotif dan robotika**, teknik ini membantu memperkirakan **kedalaman (depth)** dari citra kamera tunggal agar kendaraan otonom dapat mengenali jarak objek di sekitarnya ([Eigen et al.,](https://arxiv.org/abs/1411.4734?utm_source=chatgpt.com) *[NIPS](https://arxiv.org/abs/1411.4734?utm_source=chatgpt.com)*[, 2014)](https://arxiv.org/abs/1411.4734?utm_source=chatgpt.com). Sementara dalam **rekayasa material**, para peneliti menggunakan regresi gambar untuk memprediksi **sifat mekanik logam atau komposit** dari citra mikroskopis struktur bahan (Gao et al., *Additive Manufacturing Letters*, 2023). Tiga contoh ini menggambarkan betapa luasnya peran regresi gambar dalam menjembatani dunia visual dan data kuantitatif.

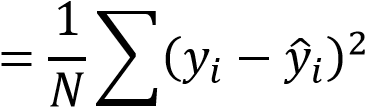
Sebelum sampai pada model canggih, tentu ada tahapan yang harus dijalani. Pertama, mahasiswa perlu **mengumpulkan data citra dan label numeriknya** — misalnya foto wajah dan usia asli seseorang. Kedua, dilakukan **pra-pemrosesan**: menyesuaikan ukuran gambar, menormalkan nilai piksel, dan memperbanyak variasi melalui *augmentasi* agar model tidak mudah “menghafal”. Langkah berikutnya adalah **merancang model**, biasanya menggunakan arsitektur *Convolutional Neural Network (CNN)* karena jaringan ini sangat baik dalam mengenali pola visual. Setelah model dibuat, tahap **pelatihan (training)** dilakukan untuk menyesuaikan bobot agar kesalahan prediksi semakin kecil. Proses ini diikuti **validasi** untuk memastikan model tidak *overfitting*, lalu diakhiri dengan **evaluasi performa** menggunakan data uji yang belum pernah dilihat model sebelumnya. Tahapan ini bukan sekadar langkah teknis, melainkan perjalanan logis dari “gambar mentah” hingga “prediksi yang bermakna”.

Seiring berkembangnya penelitian, berbagai teknik *state-of-the-art* untuk regresi gambar pun bermunculan. Salah satunya adalah **Adaptive Contrastive Learning for Regression (AdaCon)** yang diperkenalkan oleh Lee dan kolega (2021), di mana model belajar menyesuaikan jarak antar fitur sesuai perbedaan label kontinu ([arXiv:2112.11700)](https://arxiv.org/abs/2112.11700?utm_source=chatgpt.com). Pendekatan ini membuat model tidak hanya fokus pada nilai akhir, tetapi juga pada *hubungan antarsampel*. Dalam penelitian lain, **Spatially-Varying Neural Network Prior (SV-NN)** digunakan untuk menemukan bagian penting dari gambar yang paling relevan dengan nilai target ([Yin et al.,](https://jmlr.org/papers/volume26/22-0246/22-0246.pdf?utm_source=chatgpt.com) *[Journal of Machine Learning Research](https://jmlr.org/papers/volume26/22-0246/22-0246.pdf?utm_source=chatgpt.com)*[, 2024)](https://jmlr.org/papers/volume26/22-0246/22-0246.pdf?utm_source=chatgpt.com). Bahkan, pendekatan berbasis **Neural Ordinary Differential Equations (NODE)** kini digunakan untuk memprediksi nilai kontinu dari urutan citra medis, menggabungkan aspek spasial dan temporal secara elegan. Semua inovasi ini menunjukkan betapa regresi gambar terus berkembang dari sekadar “fitting angka” menjadi upaya memahami struktur visual secara mendalam.

Untuk mengetahui seberapa baik model kita bekerja, kita perlu alat ukur yang tepat. Dalam regresi, tidak ada istilah “akurasi” seperti klasifikasi. Sebaliknya, performa diukur berdasarkan **seberapa dekat nilai prediksi dengan nilai sebenarnya**. Beberapa metrik yang umum digunakan antara lain:

* **Mean Squared Error (MSE)**:

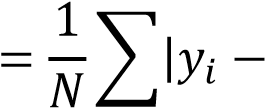
𝑁

𝑀𝑆𝐸 

𝑖

* **Mean Absolute Error (MAE)**:

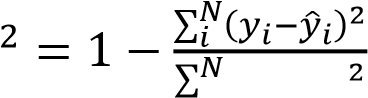
𝑁

𝑀𝐴𝐸  𝑦̂𝑖|2

𝑖

* **Koefisien Determinasi (R²)**:

𝑅

(

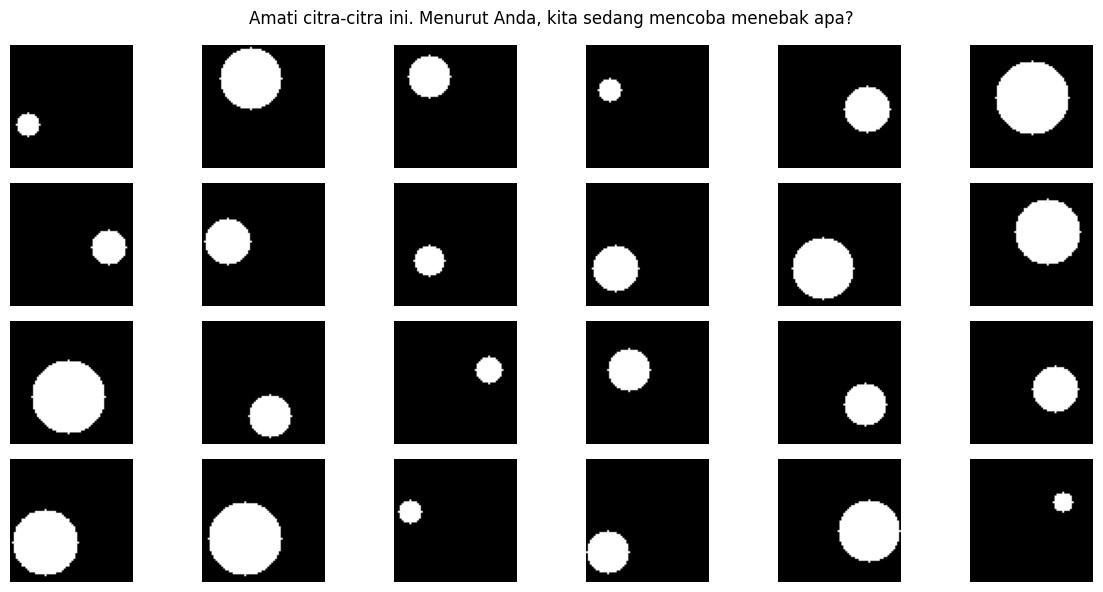
𝑖 𝑦𝑖−𝑦̅𝑖)

MSE dan MAE mengukur rata-rata kesalahan prediksi, sementara R² menunjukkan seberapa besar variasi data yang mampu dijelaskan oleh model. Semakin kecil MSE dan MAE serta semakin mendekati 1 nilai R², semakin baik pula performa model regresi gambar yang kita bangun.

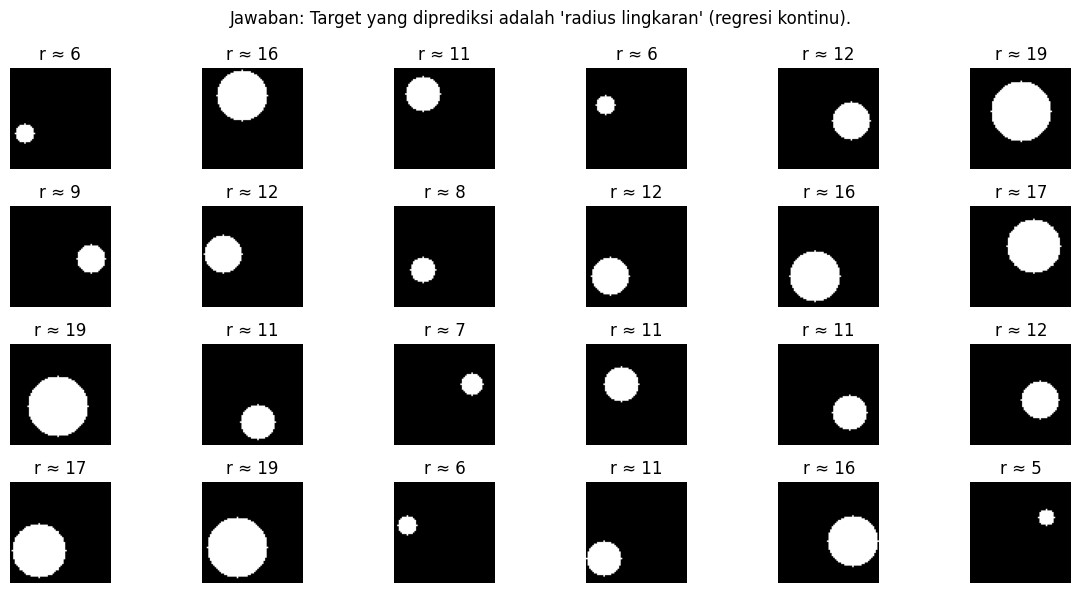
**D. Langkah Praktikum**

# Praktikum D1 – Regresi dari Citra Sintetis (Prediksi Radius Lingkaran)

Bayangkan Anda memiliki ribuan gambar hitam-putih sederhana yang masing-masing berisi satu lingkaran dengan ukuran berbeda. Dari gambar-gambar itu, bisakah komputer menebak **berapa besar radius lingkarannya** hanya dengan melihat citra tersebut?



Di praktikum ini, kita akan membuat dataset sendiri menggunakan Python. Setiap gambar dihasilkan secara acak dengan lingkaran berwarna putih di atas latar hitam. Labelnya? Nilai radius lingkaran itu sendiri! Inilah **pintu masuk** untuk memahami konsep dasar regresi gambar: kita akan melatih model *Convolutional Neural Network (CNN)* kecil agar mampu memetakan gambar → angka.



Setelah dataset selesai dibuat, model CNN akan dibangun menggunakan Keras. Jaringan ini akan belajar dari ribuan contoh yang kita hasilkan, dan Anda akan melihat bagaimana loss berkurang di setiap epoch. Pada akhir latihan, grafik *loss vs epoch* akan menunjukkan apakah model Anda sudah “mengerti” bentuk lingkaran.

Menariknya, Anda dapat langsung melihat hasil prediksi dengan membuat plot antara radius sebenarnya dan hasil prediksi. Jika garis hasil hampir menyentuh garis diagonal, artinya model sudah cukup baik dalam melakukan regresi.

**Tujuan belajar:**

* Memahami alur penuh regresi gambar tanpa dataset eksternal.
* Melatih CNN sederhana untuk prediksi nilai kontinu.
* Mengevaluasi performa dengan MAE, RMSE, dan R².

# 1) Setup & Generator Dataset

Kita akan membuat dataset citra 64×64 berisi **satu lingkaran putih** di latar hitam. Tiap gambar punya **radius** dan **titik pusat** acak.

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt import cv2  from sklearn.model\_selection import train\_test\_split from sklearn.metrics import mean\_absolute\_error, r2\_score    import tensorflow as tf  from tensorflow.keras import layers, models    # Generator 1 sample def make\_sample(img\_size=64, min\_r=5, max\_r=20):  r = np.random.randint(min\_r, max\_r + 1) # radius acak img = np.zeros((img\_size, img\_size), dtype=np.uint8) cx = np.random.randint(r, img\_size - r) # center-x cy = np.random.randint(r, img\_size - r) # center-y  cv2.circle(img, (cx, cy), r, (255,), -1) # lingkaran putih terisi img = (img / 255.0).astype(np.float32) # 3-channel biar kompatibel CNN img3 = np.stack([img, img, img], axis=-1) return img3, float(r), (cx, cy) |

# 2) “Tebak Apa?” — Tampilkan Contoh Gambar TANPA Label

Tugas kecil: Lihat gambar-gambar berikut. **Kira-kira variabel kontinu apa yang ingin kita prediksi?**

|  |
| --- |
| # Buat 24 contoh untuk visualisasi N\_show = 24  samples = [make\_sample() for \_ in range(N\_show)] imgs = [s[0] for s in samples] rads = [s[1] for s in samples] centers = [s[2] for s in samples] |
| # Grid gambar tanpa label: cols = 6  rows = N\_show // cols plt.figure(figsize=(12, 6)) for i in range(N\_show):  plt.subplot(rows, cols, i+1)  plt.imshow(imgs[i].squeeze(), cmap='gray') plt.axis('off')  plt.suptitle("Amati citra-citra ini. Menurut Anda, kita sedang mencoba menebak apa?") plt.tight\_layout() plt.show() |

Diskusi cepat (di kelas):

* Apakah yang berubah dari gambar ke gambar? • Apakah ukuran lingkaran berkaitan dengan sesuatu yang bisa “diukur” sebagai angka?
* Apakah posisi pusat juga bisa jadi target?

# 3) Buka Jawaban — Target yang Ingin Diprediksi

Di D1 ini, target utamanya adalah **radius lingkaran** (nilai kontinu). Mari **ungkap labelnya** di gambar yang sama.

|  |
| --- |
| # Tampilkan kembali, sekarang tampilkan radius (label) di judul tiap subplot  plt.figure(figsize=(12, 6)) for i in range(N\_show):  plt.subplot(rows, cols, i+1)  plt.imshow(imgs[i].squeeze(), cmap='gray') plt.title(f"r ≈ {int(rads[i])}") plt.axis('off')  plt.suptitle("Jawaban: Target yang diprediksi adalah 'radius lingkaran'  (regresi kontinu).") plt.tight\_layout() plt.show() |

Catatan:

* **Bisa diperluas**: selain radius, kita juga bisa membuat variasi praktikum untuk menebak koordinat pusat (cx,cy)(c\_x, c\_y)(cx,cy) → **multi-output regression**.
* Namun, di D1 kita fokus dulu pada satu target kontinu: **radius**.

# 4) (Opsional) Latih CNN Kecil untuk Memprediksi Radius

Kalau sudah paham “tebakannya”, mari buktikan bahwa **CNN** bisa mempelajari hubungan “pola lingkaran → angka radius”.

|  |
| --- |
| # Siapkan dataset lebih besar untuk training N = 3000  X, y, C = zip(\*[make\_sample() for \_ in range(N)]) X = np.array(X, dtype=np.float32) y = np.array(y, dtype=np.float32)    Xtr, Xte, ytr, yte = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Model CNN sederhana model = models.Sequential([ layers.Input((64,64,3)), |
| layers.Conv2D(32, 3, activation='relu'), layers.MaxPooling2D(), layers.Conv2D(64, 3, activation='relu'), layers.MaxPooling2D(), layers.Conv2D(128, 3, activation='relu'), layers.GlobalAveragePooling2D(), layers.Dense(64, activation='relu'), layers.Dense(1) # output regresi  ])  model.compile(optimizer='adam', loss='mse', metrics=['mae']) history = model.fit(Xtr, ytr, validation\_data=(Xte, yte), epochs=12, batch\_size=64, verbose=0)  # Evaluasi  y\_pred = model.predict(Xte).ravel() mae = mean\_absolute\_error(yte, y\_pred)  rmse = float(np.sqrt(np.mean((yte - y\_pred)\*\*2))) r2 = r2\_score(yte, y\_pred)  print(f"MAE={mae:.3f} | RMSE={rmse:.3f} | R²={r2:.3f}") |

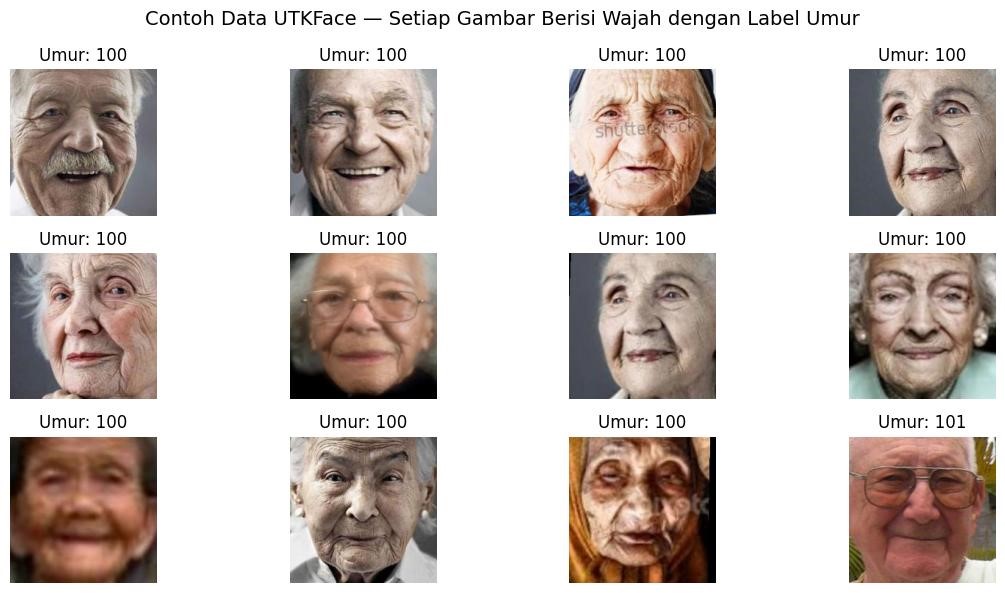
**Plot history & True vs Predicted:**

|  |
| --- |
| # Plot loss  plt.figure(figsize=(5,4))  plt.plot(history.history['loss'], label='train') plt.plot(history.history['val\_loss'], label='val') plt.title("D1: Training History (MSE)")  plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend(); plt.show()  # Scatter True vs Pred plt.figure(figsize=(5,5))  plt.scatter(yte, y\_pred, s=10, alpha=0.6)  lims = [min(yte.min(), y\_pred.min()), max(yte.max(), y\_pred.max())] plt.plot(lims, lims, '--')  plt.xlabel("True Radius"); plt.ylabel("Predicted Radius") plt.title("D1: True vs Predicted") plt.show() |

# 5) Tantangan Mini (Opsional untuk Mahasiswa)

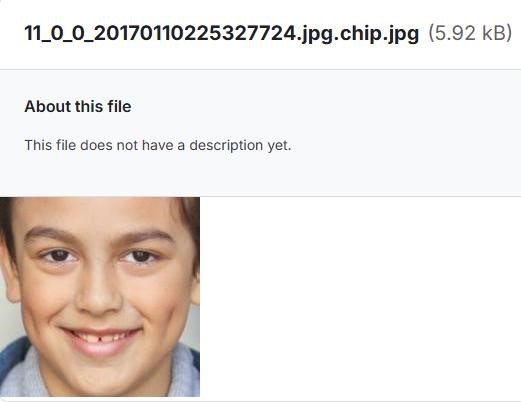
* Ubah **rentang radius** (mis. 8–28) dan lihat dampaknya ke MAE/RMSE/R².
* Tambahkan **noise** (blur, noise Gaussian) ke citra → bagaimana performanya?
* Jadikan tugas **multi-output**: prediksikan [r,cx,cy][r, c\_x, c\_y][r,cx,cy] sekaligus; metrik evaluasi apa yang cocok?

**Praktikum D2 – Menebak Umur Manusia dari Foto Wajah (UTKFace)** Pernahkah Anda memperhatikan wajah seseorang dan mencoba menebak usianya? Mungkin dari garis senyum, bentuk rahang, atau tekstur kulit. Nah, komputer juga bisa melakukan hal yang sama — **dengan belajar dari ribuan contoh wajah berlabel umur**. Di praktikum ini, kita akan membuat sistem sederhana yang mampu memperkirakan **umur seseorang dari fotonya** menggunakan **teknik regresi gambar**.



Dataset yang digunakan bernama **UTKFace**, berisi ribuan foto wajah manusia dari berbagai usia, jenis kelamin, dan ras. Setiap foto memiliki nama file yang sudah menyimpan label umur. Contohnya:

11\_0\_0\_20170110225327724.jpg.chip.jpg(5.92 kB)



Artinya, usia = 11 tahun, gender = laki-laki (0), ras = Asia (0). Namun kita hanya akan fokus pada **nilai pertama (umur)** sebagai target regresi.

**Tujuan belajar:**

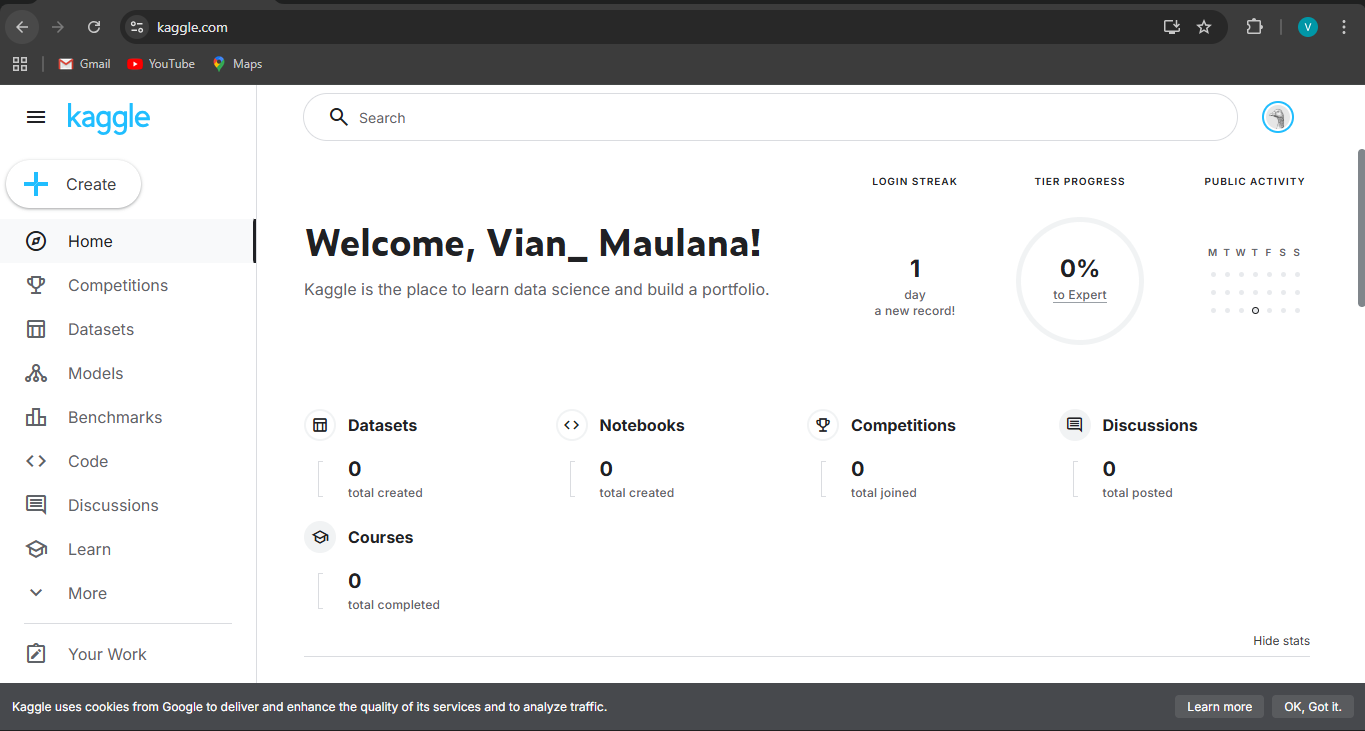
* Mengenal dataset wajah dan ekstraksi label dari nama file.
* Menggunakan *transfer learning* pada model CNN.
* Menerapkan regresi untuk masalah nyata (estimasi umur).

# Langkah 1 — Membuat Akun Kaggle dan Mengunduh kaggle.json

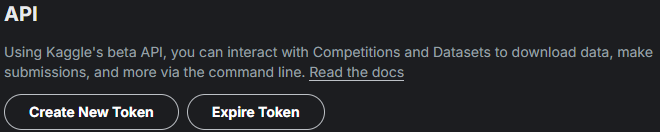
Agar dapat mengakses dataset publik Kaggle, mahasiswa perlu membuat akun terlebih dahulu.

Ikuti langkah berikut:

1. Buka situs https://www.kaggle.com
2. Daftar menggunakan email Google atau GitHub.



1. Setelah login, klik profil di kanan atas → pilih **“Account”**.
2. Gulir ke bawah ke bagian **API** → klik **“Create New API Token”**.



1. File bernama **kaggle.json** akan otomatis terunduh ke komputer Anda.

File ini berisi API key pribadi agar Colab bisa mengakses dataset Kaggle Anda. Simpan file kaggle.json dengan baik (jangan dibagikan ke orang lain).

# Langkah 2 — Mengunggah kaggle.json ke Colab

|  |
| --- |
| # Jalankan ini di awal notebook from google.colab import files  files.upload() # pilih file kaggle.json dari komputer Anda |

Setelah diunggah, jalankan sel berikut untuk menyiapkan koneksi Kaggle:

|  |
| --- |
| import os, shutil if os.path.exists("kaggle.json"):  os.makedirs(os.path.expanduser("~/.kaggle"), exist\_ok=True)  shutil.copy("kaggle.json", os.path.expanduser("~/.kaggle/kaggle.json")) os.chmod(os.path.expanduser("~/.kaggle/kaggle.json"), 0o600)  !pip -q install kaggle print("✅ Kaggle API siap digunakan.") else:  print(" kaggle.json belum ditemukan. Upload terlebih dahulu.") |

# Langkah 3 — Mengunduh Dataset UTKFace dari Kaggle

|  |
| --- |
| # Unduh dataset UTKFace (sekali saja)  !kaggle datasets download -d jangedoo/utkface-new -p /content -q  !unzip -q /content/utkface-new.zip -d /content/utk print("✅ Dataset UTKFace berhasil diekstrak.") |

# E. Langkah 4 — Menampilkan Contoh Gambar Dataset

Sebelum melatih model, mari **lihat langsung seperti apa data wajahnya**. Ini membantu mahasiswa memahami **apa masalah yang sedang diselesaikan**.

|  |
| --- |
| import matplotlib.pyplot as plt import os, glob from PIL import Image    # Ambil 12 gambar acak dari dataset  files = glob.glob("/content/utk/UTKFace/\*.jpg") files = sorted(files)  print(f"Total gambar ditemukan: {len(files)}") |
| plt.figure(figsize=(12, 6)) for i, f in enumerate(files[:12]):  # Ambil umur dari nama file  age = int(os.path.basename(f).split("\_")[0]) img = Image.open(f) plt.subplot(3, 4, i + 1) plt.imshow(img) plt.title(f"Umur: {age}") plt.axis("off")  plt.suptitle("Contoh Data UTKFace — Setiap Gambar Berisi Wajah dengan Label  Umur", fontsize=14) plt.tight\_layout() plt.show() |

**Hasil visualisasi:** Mahasiswa akan melihat wajah-wajah dari berbagai usia (anak, remaja, dewasa, hingga lansia). Inilah **masalah regresi** yang akan kita selesaikan:

“Dari citra wajah, perkirakan umur seseorang dalam satuan tahun.”

# Langkah 5 — Siapkan Dataset untuk Model

Berikut langkah singkat memuat gambar ke TensorFlow Dataset.

|  |
| --- |
| import tensorflow as tf import numpy as np  from sklearn.model\_selection import train\_test\_split  def parse\_age\_from\_name(fp):  return int(os.path.basename(fp).split('\_')[0])    ages = np.array([parse\_age\_from\_name(f) for f in files], dtype=np.float32) train\_files, test\_files, y\_train, y\_test = train\_test\_split( files, ages, test\_size=0.2, random\_state=42  )    IMG\_SIZE = 160 def load\_img(fp, label):  img = tf.io.read\_file(fp)  img = tf.image.decode\_jpeg(img, channels=3) img = tf.image.resize(img, (IMG\_SIZE, IMG\_SIZE)) return img / 255.0, label  train\_ds = tf.data.Dataset.from\_tensor\_slices((train\_files, y\_train)).map(load\_img).batch(64)  test\_ds = tf.data.Dataset.from\_tensor\_slices((test\_files, y\_test)).map(load\_img).batch(64)  print("✅ Dataset siap dilatih.") |

# Langkah 6 — Membangun Model dengan Transfer Learning

Setelah memahami seperti apa data wajah di UTKFace, kita kini akan melangkah ke tahap inti: **membangun model deep learning** untuk memprediksi umur dari foto.

Karena wajah memiliki pola kompleks — mata, kulit, ekspresi, pencahayaan — kita tidak akan melatih CNN dari nol.

Sebaliknya, kita akan menggunakan pendekatan **transfer learning**: memanfaatkan jaringan yang sudah belajar banyak tentang dunia visual (dari ImageNet), lalu menyesuaikannya sedikit untuk tugas regresi umur.

Model dasar yang kita gunakan adalah **MobileNetV2** — ringan, cepat, dan cukup akurat untuk dataset wajah berukuran menengah seperti UTKFace.

|  |
| --- |
| import tensorflow as tf  from tensorflow.keras import layers, models  from sklearn.metrics import mean\_absolute\_error, r2\_score import matplotlib.pyplot as plt import numpy as np    # Gunakan GPU jika tersedia  print("Hardware:", "GPU" if tf.config.list\_physical\_devices('GPU') else "CPU")    # Buat arsitektur model  base\_model = tf.keras.applications.MobileNetV2( include\_top=False,  input\_shape=(IMG\_SIZE, IMG\_SIZE, 3), weights='imagenet'  )  base\_model.trainable = False # tahap awal: freeze backbone  # Tambahkan head regresi  inputs = tf.keras.Input(shape=(IMG\_SIZE, IMG\_SIZE, 3))  x = tf.keras.applications.mobilenet\_v2.preprocess\_input(inputs \* 255.0) x = base\_model(x, training=False) x = layers.GlobalAveragePooling2D()(x) x = layers.Dropout(0.2)(x)  x = layers.Dense(128, activation='relu')(x) outputs = layers.Dense(1)(x) # output tunggal: umur model = tf.keras.Model(inputs, outputs)    # Kompilasi model  model.compile(optimizer=tf.keras.optimizers.Adam(1e-3), loss='mse', metrics=['mae'])    model.summary() |

**Penjelasan singkat arsitektur:**

* MobileNetV2 sudah “terlatih melihat dunia”, kita hanya menyesuaikan layer akhir untuk memetakan wajah → umur.
* GlobalAveragePooling2D mengubah fitur spasial menjadi vektor global.
* Dropout membantu mencegah overfitting.
* Output 1 neuron: prediksi umur dalam skala kontinu.

# Langkah 7 — Melatih Model (Tahap 1 – Frozen)

Tahap pertama ini adalah **pemanasan**.

Kita hanya melatih bagian *head* regresi, sementara backbone-nya tetap beku. Tujuannya agar bagian atas bisa “belajar membaca” fitur wajah tanpa mengganggu pengetahuan umum MobileNetV2.

|  |
| --- |
| # Callback untuk pelatihan yang lebih stabil cb = [  tf.keras.callbacks.EarlyStopping(patience=3, restore\_best\_weights=True, monitor='val\_loss'),  tf.keras.callbacks.ReduceLROnPlateau(patience=2, factor=0.5, min\_lr=1e-  5, monitor='val\_loss')  ]  history = model.fit( train\_ds,  validation\_data=test\_ds, epochs=10, callbacks=cb, verbose=1  )    # Visualisasi perubahan loss dan MAE selama pelatihan: plt.figure(figsize=(8,4)) plt.subplot(1,2,1)  plt.plot(history.history['loss'], label='train') plt.plot(history.history['val\_loss'], label='val')  plt.xlabel('Epoch'); plt.ylabel('MSE'); plt.title('Training vs Validation Loss') plt.legend()    plt.subplot(1,2,2)  plt.plot(history.history['mae'], label='train') plt.plot(history.history['val\_mae'], label='val') plt.xlabel('Epoch'); plt.ylabel('MAE') plt.title('Training vs Validation MAE') plt.legend() plt.tight\_layout() plt.show() |

*Interpretasi:*

* Jika **val\_loss** menurun dan stabil, model mulai memahami hubungan antara fitur wajah dan umur.
* Jika val\_loss naik terus, berarti model mulai **overfitting** — butuh dropout atau augmentasi tambahan.

# Langkah 8 — Fine-tuning Backbone (Tahap 2)

Setelah head regresi beradaptasi, kita bisa **membuka sebagian lapisan MobileNetV2** (finetuning) agar fitur internal menyesuaikan karakter wajah manusia.

# Aktifkan kembali sebagian layer terakhir untuk fine-tuning base\_model.trainable = True for layer in base\_model.layers[:-30]:

|  |
| --- |
| layer.trainable = False # beku sebagian besar layer  # Recompile dengan learning rate lebih kecil  model.compile(optimizer=tf.keras.optimizers.Adam(1e-4), loss='mse', metrics=['mae'])    history\_ft = model.fit( train\_ds,  validation\_data=test\_ds, epochs=5, callbacks=cb, verbose=1  )    # Visualisasi gabungan training dan fine-tuning: plt.plot(history.history['val\_loss'] + history\_ft.history['val\_loss'], label='val (combined)')  plt.title("Validasi Loss – Tahap 1 + Fine-tuning") plt.xlabel("Epoch"); plt.ylabel("MSE") plt.legend(); plt.show() |

# Langkah 9 — Evaluasi Akhir (MAE, RMSE, R²)

Setelah model selesai, mari kita ukur seberapa dekat prediksi dengan kenyataan.

|  |
| --- |
| from math import sqrt    y\_pred = np.concatenate([model.predict(batch[0], verbose=0).ravel() for batch in test\_ds])  mae = mean\_absolute\_error(y\_test, y\_pred) rmse = sqrt(np.mean((y\_test - y\_pred)\*\*2)) r2 = r2\_score(y\_test, y\_pred)  print(f"MAE = {mae:.2f} tahun") print(f"RMSE = {rmse:.2f} tahun") print(f"R² = {r2:.3f}")    # Plot “umur sebenarnya vs umur prediksi”:  plt.figure(figsize=(5,5))  plt.scatter(y\_test, y\_pred, s=8, alpha=0.6, color='royalblue')  lims = [min(y\_test.min(), y\_pred.min()), max(y\_test.max(), y\_pred.max())] plt.plot(lims, lims, '--', color='red') plt.xlabel("Umur Sebenarnya") plt.ylabel("Umur Prediksi")  plt.title("True vs Predicted Age (UTKFace)") plt.grid(True, linestyle='--', alpha=0.4) plt.show() |

**Interpretasi hasil:**

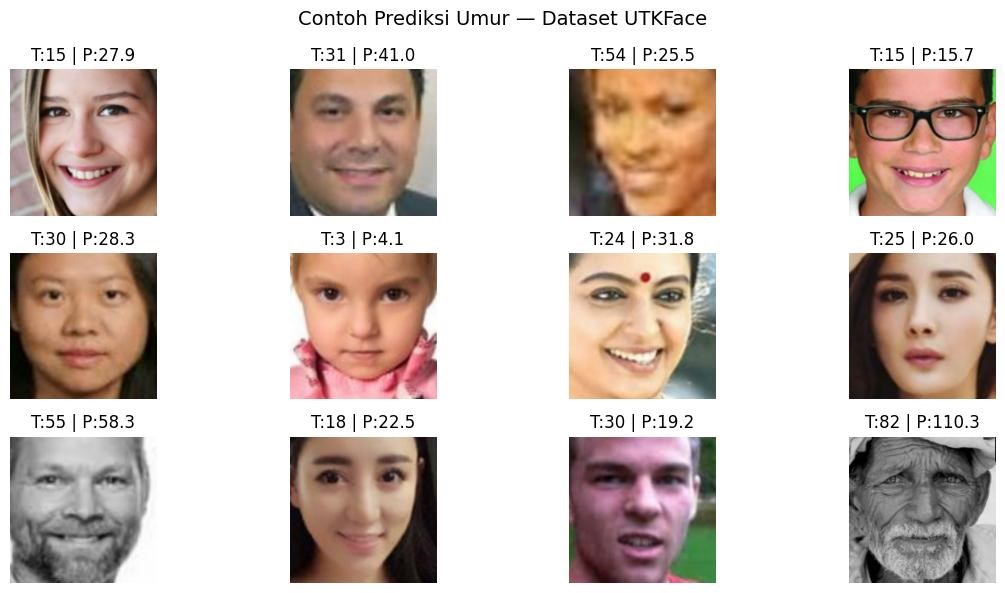
* Semakin rapat titik ke garis merah → semakin akurat prediksi.
* Nilai R² mendekati 1 menunjukkan model menjelaskan variasi data dengan baik.
* MAE kecil → rata-rata selisih umur prediksi dengan kenyataan kecil (mis. ±3 tahun).

# Langkah 10 — Melihat Contoh Prediksi Nyata

Mari lihat beberapa foto wajah berikut:

Berapa umur sebenarnya, dan berapa yang diprediksi model?

|  |
| --- |
| import random  sample\_paths = random.sample(test\_files, 12)  plt.figure(figsize=(12,6)) for i, path in enumerate(sample\_paths):  img = tf.io.read\_file(path)  img = tf.image.decode\_jpeg(img, channels=3)  img = tf.image.resize(img, (IMG\_SIZE, IMG\_SIZE))/255.0 true\_age = int(os.path.basename(path).split('\_')[0])  pred\_age = model.predict(tf.expand\_dims(img, 0), verbose=0).ravel()[0] plt.subplot(3,4,i+1) plt.imshow(img.numpy())  plt.title(f"T:{true\_age} | P:{pred\_age:.1f}") plt.axis('off')  plt.suptitle("Contoh Prediksi Umur — Dataset UTKFace", fontsize=14) plt.tight\_layout() plt.show() |



Mahasiswa dapat mengamati bahwa prediksi cenderung akurat untuk usia dewasa, namun bisa sedikit meleset untuk anak kecil atau lanjut usia — ini membuka diskusi tentang **bias dataset dan distribusi label.** **Tantangan Mini (Opsional)**

1. Tambahkan *data augmentation* ekstra (kontras, brightness, zoom).

→ Bagaimana pengaruhnya pada MAE?

1. Ubah *learning rate schedule* atau optimizer.

→ Apakah model lebih stabil?

1. Pisahkan dataset per kelompok umur (0–10, 11–20, dst).

→ Mana kelompok dengan error terbesar?

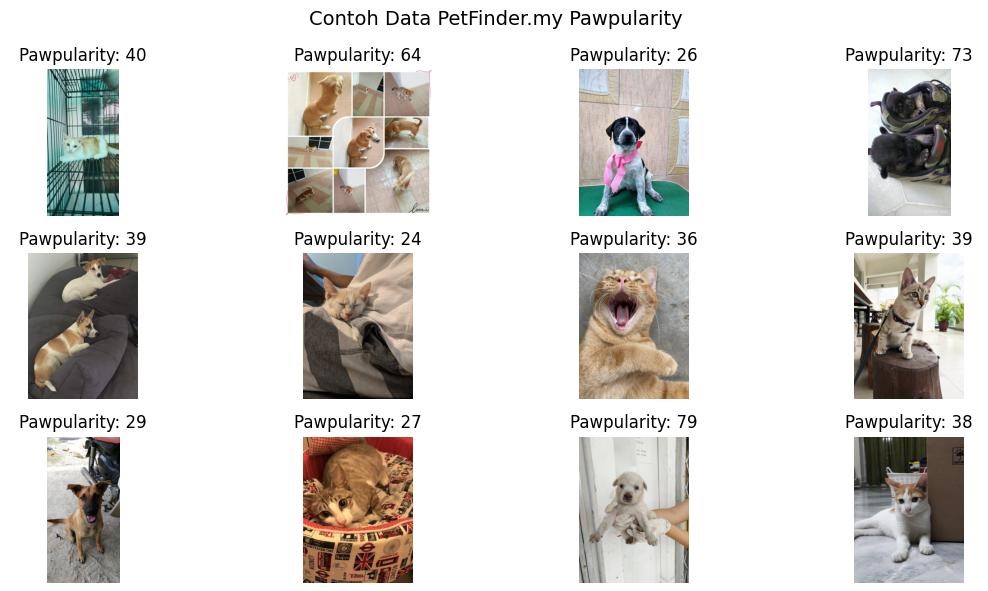
1. Uji model Anda pada wajah sendiri (setelah resize).

→ Apakah prediksinya realistis?

**Praktikum D3 — Menilai “Kepopuleran Hewan Peliharaan” dari Foto** Bayangkan Anda membuka media sosial dan melihat dua foto hewan: • seekor anak anjing dengan mata besar yang menatap kamera,

• seekor kucing yang sedang tidur di ruangan gelap.

Foto mana yang lebih menarik perhatian? Nah, para peneliti di **PetFinder.my** melakukan eksperimen serupa — mereka meminta ribuan orang untuk menilai **seberapa menarik foto hewan peliharaan**, dan setiap foto diberi skor antara **0–100** yang disebut *Pawpularity Score*.



Dalam praktikum ini, mahasiswa akan membangun sistem visi komputer yang dapat **memperkirakan skor kepopuleran hewan peliharaan dari fotonya.** Tugas ini adalah contoh nyata dari **image regression** di bidang *computational aesthetics* — mengajarkan mesin untuk memahami “daya tarik visual”.

# Tujuan Praktikum

* Memahami regresi gambar dengan target kontinu (0–100).
* Mengenal dataset dunia nyata: PetFinder.my Pawpularity.
* Melatih model **EfficientNetB0** dengan transfer learning.
* Mengevaluasi hasil dengan **MAE, RMSE, R²** dan visualisasi *True vs Predicted*.

# Langkah 1 — Menyiapkan Kaggle API

Dataset ini tersedia di kompetisi Kaggle: PetFinder.my Pawpularity Contest. Cara Membuat Akun dan Mendapatkan kaggle.json

1. Buka situs https://www.kaggle.com
2. Login dengan akun Google.
3. Klik profil → **Account** → gulir ke bawah ke bagian **API**.
4. Klik **Create New API Token**.

File kaggle.json akan otomatis terunduh.

1. Unggah file tersebut ke Colab dengan:

|  |
| --- |
| from google.colab import files  files.upload() # pilih kaggle.json dari komputer Anda import os, shutil if os.path.exists("kaggle.json"):  os.makedirs(os.path.expanduser("~/.kaggle"), exist\_ok=True)  shutil.copy("kaggle.json", os.path.expanduser("~/.kaggle/kaggle.json")) os.chmod(os.path.expanduser("~/.kaggle/kaggle.json"), 0o600)  !pip -q install kaggle print("✅ Kaggle API siap digunakan.") else:  print(" kaggle.json belum ditemukan. Upload terlebih dahulu.") |

# Langkah 2 — Mengunduh dan Mengekstrak Dataset

sebelum Anda dapat mengunduh datasetnya, Anda harus mengikuti kompetisinya terlebih dahulu. Buka <https://www.kaggle.com/competitions/petfinder-pawpularity-score/data>kemudian join.

# Unduh dataset Pawpularity (sekitar 800MB)

!kaggle competitions download -c petfinder-pawpularity-score -p /content -q !unzip -q /content/petfinder-pawpularity-score.zip -d /content/paw print("✅ Dataset Pawpularity berhasil diekstrak.")

Struktur folder:

/content/paw/train/ → berisi ribuan foto hewan (.jpg)

/content/paw/train.csv → berisi skor Pawpularity setiap foto

# Langkah 3 — Melihat Contoh Data

Sebelum melatih model, mari kita **melihat seperti apa bentuk datanya.**

Kita akan memuat file train.csv dan menampilkan beberapa foto hewan dengan skor Pawpularity-nya.

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt import os  from PIL import Image    # Muat CSV  df = pd.read\_csv('/content/paw/train.csv')  df['path'] = df['Id'].apply(lambda x: f"/content/paw/train/{x}.jpg") print(df.head())    # Tampilkan 12 contoh gambar plt.figure(figsize=(12, 6)) for i, row in enumerate(df.sample(12, random\_state=42).itertuples()):  img = Image.open(row.path) plt.subplot(3, 4, i + 1) |

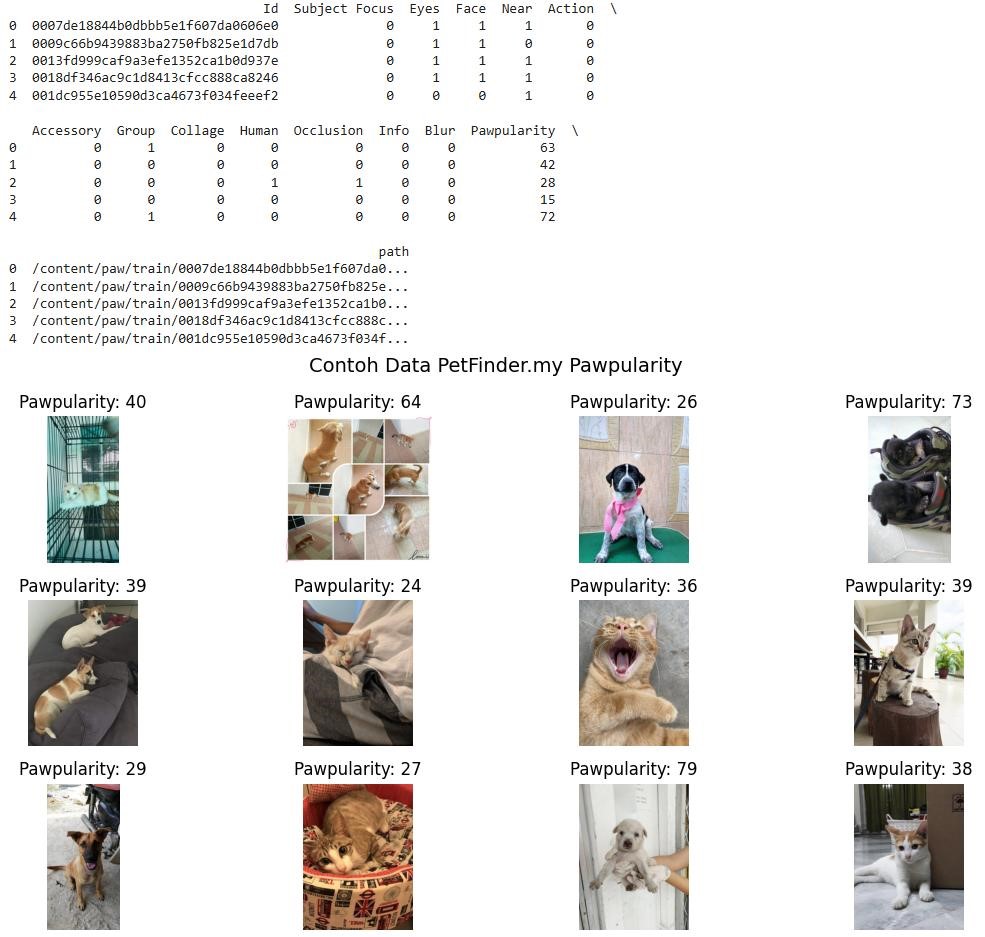
plt.imshow(img)

plt.title(f"Pawpularity: {row.Pawpularity}") plt.axis('off')

plt.suptitle("Contoh Data PetFinder.my Pawpularity", fontsize=14) plt.tight\_layout() plt.show()

**Interpretasi:**

Mahasiswa akan melihat berbagai foto — ada yang cerah, fokus, ekspresif, ada pula yang gelap atau blur. Coba perhatikan: foto dengan pencahayaan baik dan ekspresi jelas biasanya memiliki skor lebih tinggi. Inilah **hubungan visual** yang akan dipelajari oleh model kita.



# Langkah 4 — Persiapan Dataset

Sebelum memulai pelatihan, data perlu disiapkan dalam format yang bisa diproses oleh TensorFlow.

Kita akan membagi data menjadi dua bagian:

* **Train set (80%)** → untuk melatih model.
* **Validation set (20%)** → untuk menguji kemampuan generalisasi.

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split import tensorflow as tf    IMG\_SIZE = 224  train\_df, val\_df = train\_test\_split(df, test\_size=0.2, random\_state=42)  def load\_image(path, label): img = tf.io.read\_file(path)  img = tf.image.decode\_jpeg(img, channels=3) img = tf.image.resize(img, (IMG\_SIZE, IMG\_SIZE)) img = tf.cast(img, tf.float32) / 255.0 return img, tf.cast(label, tf.float32)    train\_ds = tf.data.Dataset.from\_tensor\_slices((train\_df['path'], train\_df['Pawpularity']))\  .map(load\_image, num\_parallel\_calls=tf.data.AUTOTUNE)\ .shuffle(4096).batch(64).prefetch(tf.data.AUTOTUNE)    val\_ds = tf.data.Dataset.from\_tensor\_slices((val\_df['path'], val\_df['Pawpularity']))\  .map(load\_image, num\_parallel\_calls=tf.data.AUTOTUNE)\  .batch(64).prefetch(tf.data.AUTOTUNE)    print(f"Dataset siap digunakan — {len(train\_df)} untuk training, {len(val\_df)} untuk validasi.") |

**Langkah 4 — Membangun Model (EfficientNetB0)** Sekarang, bayangkan kita ingin membangun "otak digital" yang bisa menilai keindahan foto. Kita tidak mulai dari nol, karena sudah ada model yang sangat baik dalam mengenali gambar — **EfficientNetB0**.

Model ini sebelumnya dilatih mengenali jutaan objek di ImageNet, dan kini kita hanya perlu “mengajarinya” hal baru: menilai daya tarik foto hewan berdasarkan skor 0–100.

Kita akan **membekukan** bagian awal model (agar pengetahuan dasar visual tidak hilang) dan menambahkan beberapa lapisan baru di atasnya untuk tugas regresi.

|  |
| --- |
| from tensorflow.keras import layers, models    base = tf.keras.applications.EfficientNetB0( include\_top=False,  input\_shape=(IMG\_SIZE, IMG\_SIZE, 3), weights='imagenet'  ) base.trainable = False # freeze sementara    inputs = tf.keras.Input((IMG\_SIZE, IMG\_SIZE, 3))  x = tf.keras.applications.efficientnet.preprocess\_input(inputs \* 255.0) |
| x = base(x, training=False)  x = layers.GlobalAveragePooling2D()(x) x = layers.Dropout(0.3)(x)  x = layers.Dense(256, activation='relu')(x) outputs = layers.Dense(1)(x)    model = tf.keras.Model(inputs, outputs)  model.compile(optimizer='adam', loss='mse', metrics=['mae']) model.summary() |

**Penjelasan:**

* *GlobalAveragePooling2D* mengubah citra menjadi vektor fitur.
* *Dropout(0.3)* membantu mencegah model menghafal.
* Layer terakhir (Dense(1)) memproduksi skor *pawpularity* secara kontinu.

**Langkah 5 — Melatih Model** Sekarang saatnya memberi “pengalaman belajar” pada model.

Setiap gambar akan dilihat, diproses, dan dibandingkan dengan skor sebenarnya. Kesalahan (*error*) digunakan untuk memperbaiki bobot jaringan agar prediksi semakin mendekati nilai asli.

|  |
| --- |
| cb = [  tf.keras.callbacks.EarlyStopping(patience=3, restore\_best\_weights=True),  tf.keras.callbacks.ReduceLROnPlateau(patience=2, factor=0.5)  ]  history = model.fit(train\_ds, validation\_data=val\_ds, epochs=10, callbacks=cb, verbose=1) |

**Langkah 6 — Melihat Proses Belajar** Grafik ini ibarat “rekam medis” dari proses belajar model kita.

Jika garis *val\_loss* turun dan stabil, berarti model mulai memahami pola yang benar.

Namun jika malah naik, model perlu regularisasi lebih kuat.

|  |
| --- |
| import matplotlib.pyplot as plt    plt.figure(figsize=(8,4)) plt.subplot(1,2,1)  plt.plot(history.history['loss'], label='Train') plt.plot(history.history['val\_loss'], label='Validation') plt.title("Training vs Validation Loss (MSE)") plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend()    plt.subplot(1,2,2)  plt.plot(history.history['mae'], label='Train') plt.plot(history.history['val\_mae'], label='Validation') plt.title("Training vs Validation MAE")  plt.xlabel("Epoch"); plt.ylabel("MAE"); plt.legend() plt.tight\_layout() plt.show() |

**Analisis:**

* Garis *train* turun tajam → model belajar cepat.
* Garis *validation* stabil → model mulai memahami pola umum.
* Jika *validation* naik → overfitting, perlu dropout lebih besar atau augmentasi.

# Langkah 7 — Evaluasi Model

Sekarang mari lihat bagaimana model “menilai” foto-foto yang belum pernah ia lihat. Apakah hasilnya mendekati skor manusia, atau masih meleset jauh?

|  |
| --- |
| import numpy as np  from sklearn.metrics import mean\_absolute\_error, r2\_score from math import sqrt    y\_true = val\_df['Pawpularity'].values.astype(np.float32)  y\_pred = np.concatenate([model.predict(batch[0], verbose=0).ravel() for batch in val\_ds])  mae = mean\_absolute\_error(y\_true, y\_pred) rmse = sqrt(np.mean((y\_true - y\_pred)\*\*2)) r2 = r2\_score(y\_true, y\_pred)  print(f"MAE = {mae:.2f}") print(f"RMSE = {rmse:.2f}") print(f"R² = {r2:.3f}")    # Plot hasil prediksi vs nilai sebenarnya: plt.figure(figsize=(5,5))  plt.scatter(y\_true, y\_pred, s=10, alpha=0.5)  lims = [min(y\_true.min(), y\_pred.min()), max(y\_true.max(), y\_pred.max())] plt.plot(lims, lims, '--', color='red') plt.xlabel("True Pawpularity") plt.ylabel("Predicted Pawpularity") plt.title("True vs Predicted Pawpularity") plt.grid(True, linestyle='--', alpha=0.4) plt.show() |

**Interpretasi:**

Titik-titik yang mendekati garis merah berarti prediksi model mendekati nilai aslinya. Jika R² mendekati 1 → model sangat baik; jika < 0.5 → model masih perlu dilatih lebih lama atau data diperluas.

# Langkah 8 — Melihat Contoh Prediksi

Saatnya mengintip bagaimana “otak model” menilai foto hewan-hewan lucu ini.

Apakah penilaiannya mirip dengan skor manusia?

|  |
| --- |
| import random  from PIL import Image    sample\_rows = val\_df.sample(12, random\_state=1)  plt.figure(figsize=(12,6)) for i, row in enumerate(sample\_rows.itertuples()):  img = Image.open(row.path)  pred = model.predict(tf.expand\_dims(load\_image(row.path, row.Pawpularity)[0], 0), verbose=0).ravel()[0] plt.subplot(3,4,i+1) plt.imshow(img)  plt.title(f"True: {row.Pawpularity}\nPred: {pred:.1f}") plt.axis('off')  plt.suptitle("Contoh Prediksi Skor Pawpularity", fontsize=14) plt.tight\_layout() plt.show() |

**Analisis Visual:**

* Foto yang terang, fokus, dan ekspresif cenderung mendapat skor tinggi.
* Foto gelap, buram, atau dengan cropping aneh — biasanya mendapat skor lebih rendah.

Model kita perlahan-lahan belajar “rasa visual” seperti manusia.

# Refleksi Akhir

Melalui praktikum ini, mahasiswa telah melihat bagaimana komputer:

* Menerjemahkan **gambar menjadi fitur numerik.**
* Mempelajari **pola estetika dan emosi** tanpa diberi aturan eksplisit.
* Menghasilkan prediksi numerik yang merepresentasikan *persepsi manusia*.

Di dunia nyata, teknik ini digunakan untuk:

* Rekomendasi foto terbaik,
* Sistem penilaian kualitas gambar otomatis, • Pencarian konten visual yang menarik (*content-based retrieval*).

# Tantangan Mini

1. Coba tambahkan *data augmentation* agar model tidak bosan dengan pola yang sama.
2. Gunakan **EfficientNetB3** atau **ResNet50**, lalu bandingkan hasilnya.
3. Buat eksperimen kecil: pisahkan anjing vs kucing — apakah model bias ke salah satu spesies? 4. Tambahkan fitur non-visual (misal brightness rata-rata) ke model regresi.

# F. Hasil Praktikum

Lengkapi hasil tabel praktikum berikut:

|  |  |  |
| --- | --- | --- |
| **No** | **Nama Praktikum** | **Hasil Praktikum** |
| 1 | **D1 – Prediksi Radius Lingkaran** | Plot dataset sintetis, model CNN, dan grafik True vs Predicted radius. |
| 2 | **D2 – Prediksi Umur dari Wajah** | Visualisasi dataset  UTKFace, training log MobileNetV2, hasil evaluasi MAE & R². |
| 3 | **D3 – Prediksi Pawpularity Foto Hewan** | Contoh foto dengan skor asli dan prediksi model, grafik True vs Predicted Pawpularity. |

# G. Penugasan

1. Kumpulkan Laporan Praktikum dari jobsheet ini dalam bentuk Microsoft word sesuai dengan format jobsheet praktikum dan dikumpulkan di web LMS. (JANGAN DALAM BENTUK PDF)
2. Kumpulkan luaran kode praktikum dalam bentuk ipynb yang sudah diunggah pada akun github masing-masing. Lampirkan tautan github yang sudah di unggah melalui laman LMS.
3. Gunakan foto pribadi untuk menampilkan ujicoba prediksi usia kalian — tampilkan hasilnya di laporan!
4. gunakan foto pribadi (atau hewan peliharaan sendiri) dan uji model Pawpularity — tampilkan hasilnya di laporan!

# H. Kesimpulan

Melalui rangkaian praktikum Teknik regresi gambar ini, mahasiswa telah menelusuri perjalanan lengkap dari konsep dasar hingga penerapan nyata **regresi gambar dalam visi komputer**. Setiap praktikum memiliki makna pembelajaran tersendiri yang membentuk pemahaman bertahap.

Pada **Praktikum D1**, mahasiswa belajar dari hal paling mendasar — menciptakan dataset citra sintetis berupa lingkaran dan mencoba menebak radiusnya. Dari eksperimen sederhana ini, muncul kesadaran bahwa **citra dapat diterjemahkan menjadi angka** jika kita dapat membangun fungsi pemetaan yang tepat menggunakan CNN.

Lanjut ke **Praktikum D2**, mahasiswa berhadapan dengan data dunia nyata: **foto wajah manusia** dari dataset UTKFace. Di sini diperkenalkan konsep **transfer learning**, di mana model yang sudah “berpengalaman melihat dunia” (MobileNetV2) disesuaikan untuk tugas baru — menebak umur seseorang. Mahasiswa belajar bahwa meski model dapat belajar pola visual kompleks, hasilnya tetap bergantung pada distribusi data dan cara evaluasi menggunakan MAE, RMSE, serta R².

Pada **Praktikum D3**, konteks menjadi lebih menarik secara emosional: memprediksi **daya tarik visual hewan peliharaan (Pawpularity)**. Di sini mahasiswa menyadari bahwa regresi gambar tidak hanya digunakan untuk pengukuran fisik (seperti jarak atau umur), tetapi juga untuk mempelajari **persepsi manusia terhadap keindahan dan ekspresi visual**. Model EfficientNetB0 dilatih untuk memahami unsur subjektif — terang, fokus, ekspresi, dan komposisi.

Secara keseluruhan, mahasiswa kini memahami bahwa:

* **Regresi gambar adalah proses menerjemahkan fitur visual menjadi nilai kontinu.**
* **Jaringan konvolusional (CNN) mampu mempelajari hubungan non-linear** antara pola piksel dan nilai target.
* **Evaluasi model tidak hanya angka, tetapi juga interpretasi makna hasil visual.**

Dengan pengalaman praktikum ini, mahasiswa telah memperoleh dasar kuat untuk mengembangkan model-model lebih lanjut — seperti prediksi kedalaman citra, estimasi jarak, atau pengukuran emosi dari ekspresi wajah — yang semuanya berakar pada konsep **regresi gambar**.

# I. Daftar Pustaka

* LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278–2324.
* Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems.
* Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for LargeScale Image Recognition*. arXiv:1409.1556.
* Szegedy, C., et al. (2015). *Going deeper with convolutions*. IEEE Conference on Computer Vision and Pattern Recognition.
* He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*.

IEEE Conference on Computer Vision and Pattern Recognition.

* Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). *Densely connected convolutional networks*. IEEE CVPR.
* Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking model scaling for convolutional neural networks*. ICML.
* Esteva, A., et al. (2017). *Dermatologist-level classification of skin cancer with deep neural networks*. Nature, 542(7639), 115–118.
* Bojarski, M., et al. (2016). *End to End Learning for Self-Driving Cars*. arXiv:1604.07316.
* Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
* Chollet, F. (2015). *Keras: Deep Learning for humans*. GitHub repository.