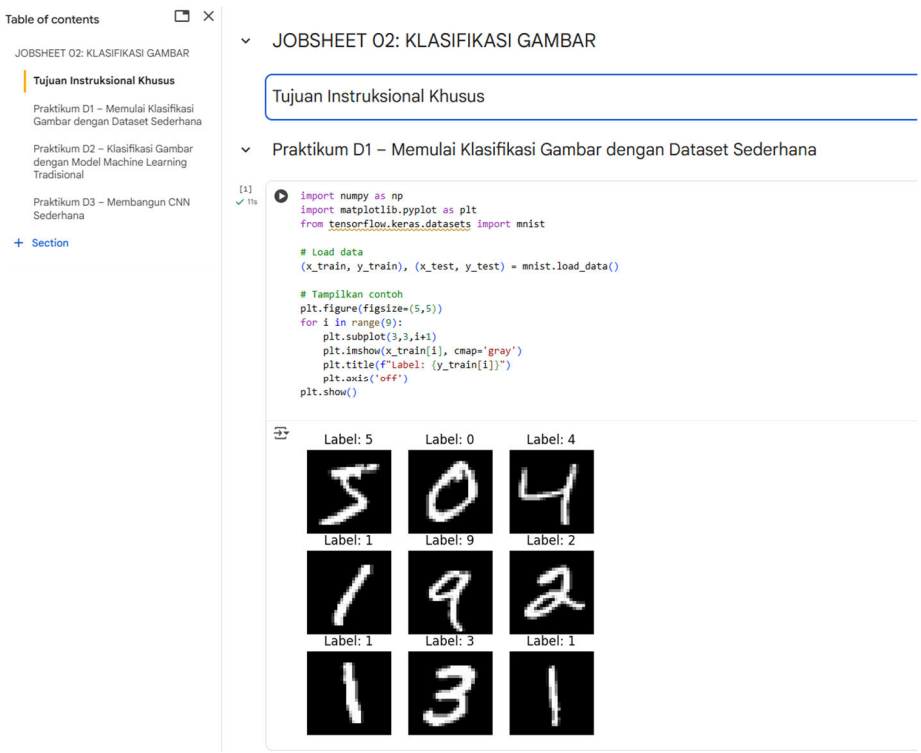
POLITEKNIK NEGERI SEMARANG
JURUSAN TEKNIK ELEKTRO
PROGRAM STUDI STR TEKNOLOGI REKAYASA KOMPUTER

JOBSHEET 02:
KLASIFIKASI GAMBAR



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

Nama Mahasiswa : _____
NIM Mahasiswa : _____

Tahun Akademik 2025

Klasifikasi gambar merupakan salah satu tugas fundamental dalam bidang *computer vision*. Tujuan utamanya adalah memberikan label semantik pada sebuah citra dari sekumpulan kelas yang telah ditentukan. Perkembangan teknologi *machine learning* dan *deep learning*, khususnya Convolutional Neural Networks (CNN), telah membawa kemajuan signifikan pada performa sistem klasifikasi gambar, sehingga mampu diaplikasikan di berbagai bidang seperti kesehatan, kendaraan otonom, keamanan, hingga e-commerce.
Pada praktikum ini mahasiswa akan mempelajari konsep dasar klasifikasi gambar, mengimplementasikan algoritma dengan bahasa pemrograman Python menggunakan Google Colab, serta melakukan analisis hasil klasifikasi menggunakan berbagai model, baik tradisional maupun modern. Dengan praktikum ini, mahasiswa diharapkan tidak hanya memahami teori, tetapi juga memiliki keterampilan teknis dalam membangun sistem klasifikasi gambar yang efisien dan akurat.

A. Tujuan Instruksional Khusus

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu untuk:

1. Menjelaskan perumusan masalah klasifikasi gambar.
2. Menerapkan pendekatan tradisional seperti *Support Vector Machine (SVM)*, *k-NN*, dan *Random Forest* untuk klasifikasi gambar.
3. Mendesain dan melatih model CNN sederhana untuk tugas klasifikasi.
4. Menerapkan *transfer learning* dengan memanfaatkan model *pre-trained* (misalnya ResNet, VGG, EfficientNet).
5. Mengevaluasi performa sistem klasifikasi menggunakan metrik seperti akurasi, presisi, recall, F1-score, dan confusion matrix.
6. Menangani tantangan praktis pada klasifikasi gambar seperti *class imbalance* dan *overfitting*.

B. Alat dan Bahan

Untuk melaksanakan praktikum ini, diperlukan:

1. **Perangkat keras**
   - Laptop/PC dengan koneksi internet stabil.
2. **Perangkat lunak**
   - Google Colab (Python 3, Jupyter-like environment).
   - Library Python: `numpy`, `pandas`, `matplotlib`, `scikit-learn`, `tensorflow/keras`, `torch` (opsional).
3. **Dataset**
   - Dataset citra sederhana seperti *MNIST*, *CIFAR-10*, atau dataset kustom sesuai kebutuhan praktikum.

C. Dasar Teori

Bayangkan Anda sedang membuka aplikasi kamera di ponsel. Anda memotret seekor kucing yang sedang duduk di kursi. Ketika foto itu dikirim ke media sosial, sistem secara otomatis dapat mengenali bahwa gambar tersebut berisi seekor kucing. Proses sederhana yang terlihat sepele ini sebenarnya merupakan hasil kerja panjang dari bidang yang disebut **klasifikasi gambar**.

Pada dasarnya, klasifikasi gambar adalah upaya memberi "nama" pada suatu gambar. Komputer harus belajar membedakan apakah sebuah citra termasuk ke dalam kategori *anjing*, *kucing*, *mobil*, atau bahkan *daun yang sehat* dan *daun yang sakit*. Seolah-olah kita mengajarkan komputer untuk "melihat" dan memahami isi dunia visual.

Namun, perjalanan ini tidak mudah. Coba bayangkan seekor anjing dari berbagai sudut pandang: dari depan, samping, atau bahkan dari atas. Penampilannya bisa sangat berbeda, tetapi tetap harus dikenali sebagai anjing. Ini yang disebut **variasi sudut pandang (viewpoint variation)**. Belum lagi ketika kondisi cahaya berubah – foto diambil siang hari tentu berbeda dengan foto di malam hari – inilah tantangan **iluminasi**. Bahkan, seekor kucing berbulu putih bisa tampak mirip dengan seekor anjing kecil berbulu putih (**inter-class similarity**), membuat komputer mudah salah tebak.

1. Konsep Dasar Klasifikasi Gambar

Klasifikasi gambar adalah proses memberikan label pada sebuah citra berdasarkan kategori yang sudah ditentukan. Misalnya, sistem harus dapat mengklasifikasikan gambar sebagai "kucing", "anjing", atau "burung". Dalam formulasi matematis, klasifikasi gambar bertujuan mempelajari fungsi f:I→Cf: I \rightarrow Cf:I→C, yang memetakan sebuah citra III ke kelas CCC.

Ada beberapa jenis klasifikasi:

- **Binary classification**: hanya dua kelas, misalnya "defect" dan "non-defect".
- **Multi-class classification**: banyak kelas, tetapi setiap gambar hanya memiliki satu label.
- **Multi-label classification**: sebuah gambar bisa memiliki lebih dari satu label, misalnya "orang", "sepeda", dan "jalan".

2. Tantangan dalam Klasifikasi Gambar

Beberapa faktor membuat klasifikasi gambar menjadi sulit:

- **Intra-class variation**: objek dalam kelas yang sama bisa sangat berbeda, misalnya mobil sedan dan truk.
- **Inter-class similarity**: objek dari kelas berbeda tampak mirip, seperti anjing dan serigala.
- **Viewpoint variation**: objek terlihat berbeda dari sudut pandang yang berbeda.
- **Iluminasi dan skala**: cahaya, ukuran, serta jarak kamera memengaruhi hasil citra.
- **Occlusion dan clutter**: objek bisa tertutup sebagian atau muncul dalam latar belakang yang rumit.

3. Pendekatan Tradisional

Sebelum era *deep learning*, klasifikasi gambar dilakukan dengan cara:

- **Ekstraksi fitur manual**, seperti SIFT, SURF, atau HOG.
- **Klasifikasi dengan algoritma ML**, misalnya *Support Vector Machine (SVM)*, *Random Forests*, atau *k-Nearest Neighbors (k-NN)*.
- **Bag of Visual Words (BoVW)**: merepresentasikan citra sebagai histogram kemunculan fitur visual.

Meskipun sederhana, pendekatan ini memiliki keterbatasan: membutuhkan *feature engineering* manual, sulit menangkap pola kompleks, dan kurang efektif untuk dataset besar.

4. Deep Learning dan CNN

Kemunculan *Convolutional Neural Networks (CNN)* membawa perubahan besar. CNN mampu secara otomatis belajar mengekstraksi fitur dari data mentah tanpa perlu *feature engineering* manual. CNN terdiri dari lapisan:

- **Convolutional layer**: mengekstrak pola sederhana seperti tepi dan tekstur.
- **Pooling layer**: mereduksi dimensi sambil mempertahankan informasi penting.
- **Fully connected layer**: menggabungkan fitur untuk klasifikasi.
- **Softmax output**: menghasilkan probabilitas kelas.

CNN modern seperti **AlexNet, VGG, GoogLeNet, ResNet, DenseNet, dan EfficientNet** menunjukkan peningkatan signifikan dalam akurasi dan efisiensi.

5. Transfer Learning

Transfer learning memungkinkan penggunaan model yang sudah dilatih pada dataset besar (misalnya ImageNet) untuk menyelesaikan tugas baru dengan data terbatas. Ada dua pendekatan:

- **Feature extraction**: menggunakan CNN pra-latih sebagai ekstraktor fitur, lalu melatih classifier baru.
- **Fine-tuning**: menyesuaikan (update) bobot CNN pra-latih dengan dataset baru.

Teknik ini sangat berguna karena lebih hemat data, waktu, dan komputasi.

6. Evaluasi Model Klasifikasi

Kinerja sistem klasifikasi dinilai menggunakan metrik berikut:

- **Accuracy**: rasio prediksi benar.
- **Precision**: seberapa tepat prediksi positif.
- **Recall (Sensitivity)**: kemampuan mendeteksi semua data positif.
- **F1-score**: harmonisasi precision dan recall.
- **Confusion matrix**: menampilkan detail kesalahan per kelas.
- **ROC & AUC**: mengukur performa pada berbagai ambang batas.

D. Langkah Praktikum

Praktikum D1 – Memulai Klasifikasi Gambar dengan Dataset Sederhana

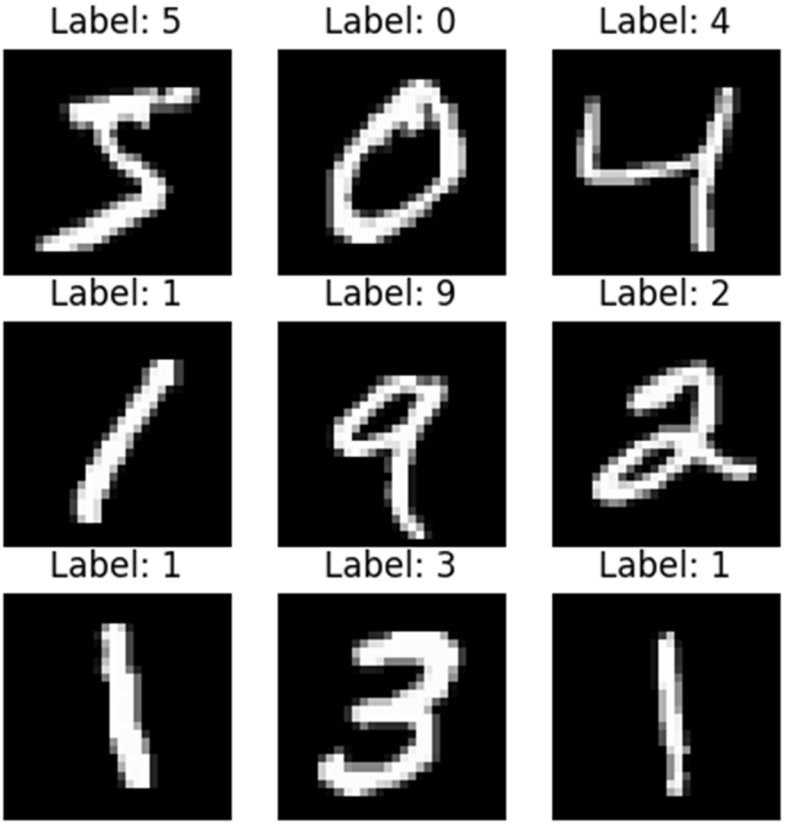
Bayangkan Anda baru pertama kali mengenalkan komputer pada dunia visual. Sebelum komputer bisa mengenali gambar kompleks seperti mobil atau wajah manusia, mari kita ajarkan dulu konsep sederhana: mengenali angka tulisan tangan. Dataset **MNIST** yang berisi digit 0–9 adalah titik awal yang bagus.

Langkah di Google Colab:

1. Buat notebook baru di Google Colab.
2. Import library dasar Python (`numpy`, `matplotlib`, `tensorflow`).
3. Load dataset MNIST dari `keras.datasets`.
4. Visualisasikan beberapa gambar untuk melihat bentuk data.
5. Normalisasi data agar siap diproses model.

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist

# Load data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Tampilkan contoh
plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.show()
```

Tugas kecil: Cobalah ganti `range(9)` dengan `range(25)` untuk menampilkan lebih banyak contoh. Apa yang Anda perhatikan dari bentuk tulisan tangan manusia?



Praktikum D2 – Klasifikasi Gambar dengan Model Machine Learning Tradisional

Sebelum CNN populer, pendekatan klasik digunakan. Mari kita coba **Support Vector Machine (SVM)**. Dataset yang kita pakai tetap MNIST, tapi gambar harus diubah menjadi *vektor 1D*.

Langkah di Google Colab:

1. Flatten setiap gambar 28x28 menjadi vektor 784.
2. Gunakan `sklearn.svm.SVC` untuk melatih model sederhana.
3. Uji model pada data test dan lihat akurasinya.

```
from sklearn import svm
from sklearn.metrics import accuracy_score

# Flatten
x_train_flat = x_train.reshape(len(x_train), -1) / 255.0
x_test_flat = x_test.reshape(len(x_test), -1) / 255.0

# SVM
clf = svm.SVC(kernel='linear', gamma='scale')
clf.fit(x_train_flat[:5000], y_train[:5000])  # gunakan subset karena SVM
berat
y_pred = clf.predict(x_test_flat)

print("Akurasi:", accuracy_score(y_test, y_pred))
```

Tugas kecil: Ubah kernel dari `linear` menjadi `rbf`. Bagaimana pengaruhnya terhadap akurasi?

Praktikum D3 – Membangun CNN Sederhana

Sekarang saatnya mencoba **Convolutional Neural Network**. CNN akan belajar mengenali pola dari gambar MNIST secara otomatis.

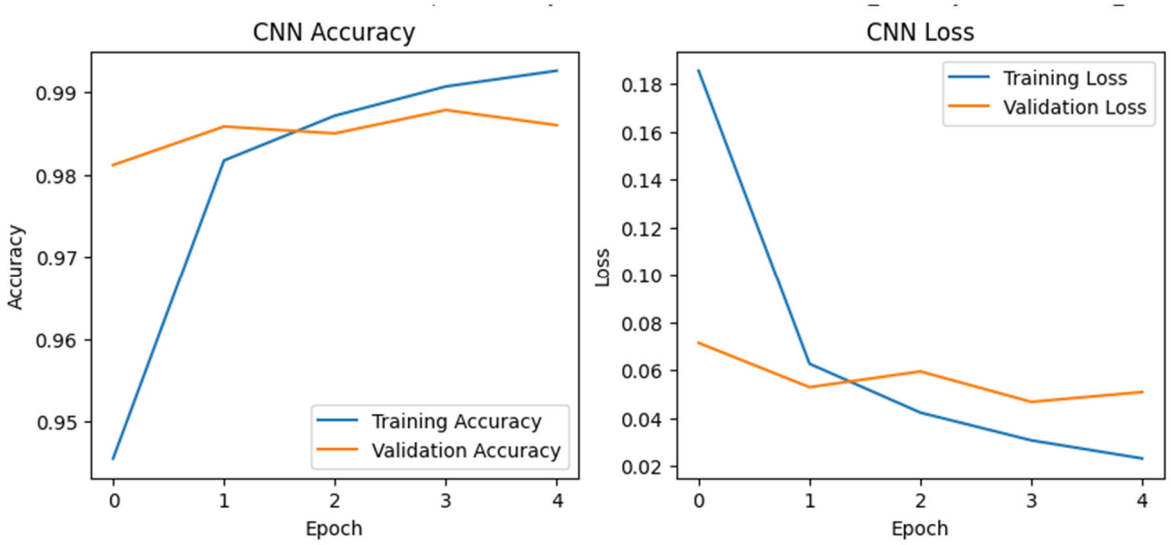
Langkah di Google Colab:

1. Ubah data menjadi bentuk `[samples, height, width, channels]`.
2. Bangun CNN sederhana dengan `Conv2D`, `MaxPooling2D`, `Flatten`, dan `Dense`.
3. Latih model dan amati hasilnya.

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

x_train_cnn = x_train.reshape(-1,28,28,1) / 255.0
x_test_cnn = x_test.reshape(-1,28,28,1) / 255.0

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

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

history = model.fit(x_train_cnn, y_train, epochs=5, validation_split=0.1)

# ===== Plot history =====
import matplotlib.pyplot as plt
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('CNN Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('CNN Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Tugas kecil: Tambahkan satu lapisan `Conv2D` lagi sebelum `Flatten`. Apakah akurasinya meningkat?



Praktikum D4 – Eksperimen dengan Dataset Lebih Kompleks (CIFAR-10)

MNIST mudah. Sekarang mari coba **CIFAR-10**, yang berisi gambar berwarna (mobil, pesawat, anjing, kucing, dll). Ini membuat model CNN bekerja lebih keras.

Langkah di Google Colab:

1. Load CIFAR-10 dataset dari `keras.datasets`.
2. Bangun arsitektur CNN yang lebih dalam.
3. Latih model dan bandingkan akurasi dengan MNIST.

```
from tensorflow.keras.datasets import cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train/255.0, x_test/255.0

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

history = model.fit(x_train, y_train, epochs=10, validation_split=0.1)

# ===== Plot history =====
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('CNN CIFAR-10 Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('CNN CIFAR-10 Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Tugas kecil: Coba tambahkan `Dropout(0.5)` sebelum lapisan Dense terakhir. Apa pengaruhnya pada overfitting?

Praktikum D5 – Transfer Learning dengan Model Pra-Latih

Bagaimana jika kita gunakan "otak" yang sudah cerdas? Misalnya **VGG16** atau **ResNet50** yang sudah dilatih di *ImageNet*. Kita tinggal memanfaatkannya.

Langkah di Google Colab:

1. Gunakan `tensorflow.keras.applications.VGG16` dengan `weights='imagenet'`.
2. Bekukan lapisan convolutional dan tambahkan classifier baru.
3. Latih hanya lapisan classifier dengan CIFAR-10.

```
from tensorflow.keras.applications import VGG16

base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(32,32,3))
base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

history = model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# ===== Plot history =====
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Transfer Learning Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Transfer Learning Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Tugas kecil: Coba aktifkan 1–2 lapisan terakhir dari `base_model` (fine-tuning). Bagaimana akurasinya berubah?

Praktikum D6 – Evaluasi dengan Confusion Matrix dan Metrik Lain

Model sudah dilatih, tapi bagaimana cara mengevaluasi secara lebih detail? Akurasi saja tidak cukup. Mari tampilkan **confusion matrix** dan hitung precision, recall, serta F1-score.

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

y_pred = model.predict(x_test).argmax(axis=1)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Tugas kecil: Dari confusion matrix, identifikasi kelas apa yang paling sering salah diklasifikasikan. Menurut Anda, mengapa hal itu bisa terjadi?

E. Hasil Praktikum

Lengkapi hasil tabel praktikum berikut:

No	Nama Praktikum	Hasil Praktikum
1	**D1 – Memulai Klasifikasi Gambar (MNIST)**	
2	**D2 – Klasifikasi Tradisional dengan SVM (MNIST)**	
3	**D3 – CNN Sederhana (MNIST)**	
4	**D4 – CNN dengan Dataset CIFAR-10 (RGB)**	
5	**D5 – Transfer Learning (VGG16/ResNet50)**	
6	**D6 – Evaluasi dengan Confusion Matrix**	

F. 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. **Mencoba klasifikasi tulisan angka**:

    - Tulis 1–3 digit (0–9) di kertas putih dengan spidol/bolpen tebal (ukuran besar, kontras jelas).
    - Foto lurus dari atas, pencahayaan merata, **tanpa bayangan** besar.
    - **Upload** foto ke Google Colab.
    - **Pre-process** gambar agar sesuai format MNIST (28×28, grayscale).
    - **Klasifikasi** dengan **salah satu model**:
        - **CNN (D3)** → butuh tensor shape `(1, 28, 28, 1)`
        - **SVM (D2)** → butuh vektor **784** (flatten 28×28)

Catatan penting: Jalankan dulu praktikum **D2** (untuk `clf` SVM) atau **D3** (untuk `model` CNN) sehingga objek model **sudah ada** di memori notebook. Pilih salah satu sesuai yang Anda gunakan.

Kode Colab: Upload & Pre-process

Tempatkan sel ini **setelah** Anda selesai melatih model (D2 atau D3).

```
# ===== 1) Upload file foto tulisan angka =====
from google.colab import files
uploaded = files.upload()  # pilih 1 atau lebih file gambar (jpg/png)

# ===== 2) Utilitas Preprocess agar mirip MNIST (28x28, putih-di-atas-
hitam) =====
import numpy as np
from PIL import Image, ImageOps

def preprocess_to_mnist_28x28(img_pil):
    """
    Langkah:
    - Konversi ke grayscale
    - Auto-contrast
    - (Opsional) invert bila latar terang (agar digit jadi putih, latar
jadi gelap seperti MNIST)
    - Crop ke bounding box digit
    - Resize mempertahankan rasio ke (20x20), lalu pad ke (28x28)
    - Normalisasi ke [0,1] dan tambah axis channel
    """
    # Grayscale + autocontrast
    img = img_pil.convert('L')
    img = ImageOps.autocontrast(img)

    arr = np.array(img).astype(np.uint8)
    # Jika rata-rata terang (kertas putih), invert supaya digit menjadi
putih di atas latar gelap (gaya MNIST)
    if arr.mean() > 127:
        img = ImageOps.invert(img)
        arr = np.array(img)

    # Binarisasi ringan untuk cari bbox digit
    thr = np.mean(arr) * 0.8  # ambang adaptif sederhana
    mask = arr > thr
    if mask.any():
        ys, xs = np.where(mask)
        y0, y1 = ys.min(), ys.max()
        x0, x1 = xs.min(), xs.max()
        img = img.crop((x0, y0, x1+1, y1+1))

    # Resize ke 20x20 dengan aspect ratio
    img.thumbnail((20, 20), Image.Resampling.LANCZOS)
    w, h = img.size

    # Pad ke 28x28 dan center
    canvas = Image.new('L', (28, 28), color=0)
    canvas.paste(img, ((28 - w)//2, (28 - h)//2))

    # Normalisasi ke [0,1]
    arr = np.array(canvas).astype('float32') / 255.0
    # Tambah channel dim (28,28,1)
```

```
    arr = arr[..., None]
    return canvas, arr
```

(Pilihan A) Prediksi dengan CNN dari D3

Pastikan Anda sudah **menjalankan D3** dan memiliki variabel `model` (Keras) terlatih pada MNIST.

```
import matplotlib.pyplot as plt

results = []
for fname in uploaded.keys():
    img_pil = Image.open(fname)
    disp, x = preprocess_to_mnist_28x28(img_pil)   # disp: PIL untuk
ditampilkan, x: (28,28,1)
    x_batch = np.expand_dims(x, axis=0)            # (1,28,28,1)
    probs = model.predict(x_batch, verbose=0)[0]   # shape (10,)
    pred = int(np.argmax(probs))
    conf = float(np.max(probs))

    results.append((fname, pred, conf))

    # Tampilkan hasil
    plt.figure(figsize=(3,3))
    plt.imshow(disp, cmap='gray')
    plt.title(f"Prediksi: {pred}  (p={conf:.2f})")
    plt.axis('off')
    plt.show()

# Rekap ringkas
print("Rekap Prediksi (CNN):")
for r in results:
    print(f"- {r[0]} -> {r[1]} (p={r[2]:.3f})")
```

Apa yang diharapkan mahasiswa tuliskan di laporan:

- 1–2 gambar hasil (display 28×28) + label prediksi & probabilitas.
- 2–3 kalimat refleksi: apakah prediksi sudah benar? jika salah, kira-kira kenapa (pencahayaan, ketebalan tulisan, pemrosesan, dsb.).

(Pilihan B) Prediksi dengan SVM dari D2

Pastikan Anda sudah **menjalankan D2** dan memiliki variabel `clf` (SVM) terlatih.
SVM butuh input **flatten (784)**, jadi kita ubah tensor `(28,28,1)` menjadi vektor `(784,)`.

```
from sklearn.metrics import accuracy_score

results = []
for fname in uploaded.keys():
    img_pil = Image.open(fname)
    _, x = preprocess_to_mnist_28x28(img_pil)   # x: (28,28,1) float [0,1]
    x_flat = x.reshape(1, -1)                   # (1,784)

    pred = int(clf.predict(x_flat)[0])
```

```
    # SVM default tidak punya proba kecuali SVC(probability=True). Jika
Anda ingin probabilitas:
    # clf = svm.SVC(kernel='rbf', gamma='scale', probability=True) saat
pelatihan.
    conf = None
    try:
        if hasattr(clf, "predict_proba"):
            conf = float(np.max(clf.predict_proba(x_flat)))
    except Exception:
        pass

    results.append((fname, pred, conf))

    # Tampilkan hasil (gunakan gambar 28x28 yang sudah diproses di tahap
CNN juga boleh)
    print(f"{fname} -> Prediksi SVM: {pred}" + (f" (p≈{conf:.2f})" if conf
is not None else ""))
```

Yang dilaporkan mahasiswa:

- 1–3 baris rekap file→prediksi (dan probabilitas jika diset `probability=True`).
- 2–3 kalimat refleksi: performa SVM vs CNN pada tulisan tangan mereka.

Tips kecil agar akurasi lebih baik

- Tulis angka **tebal dan besar**, biar stroke jelas.
- Usahakan latar **putih bersih**; hindari bayangan/tangan masuk frame.
- Jika hasilnya sering salah, coba **kontras** ditingkatkan (autocontrast sudah dilakukan), atau tulis angka ulang lebih jelas.
- Untuk SVM, hasil sangat sensitif pada preprocessing—pastikan ukuran & posisi digit konsisten.

G. Kesimpulan

Melalui rangkaian praktikum klasifikasi gambar ini, mahasiswa telah menempuh sebuah perjalanan belajar dari dasar hingga penerapan tingkat lanjut:

- **D1–D2** memperkenalkan konsep dasar klasifikasi dengan dataset sederhana (MNIST), sekaligus membandingkan pendekatan tradisional (*Support Vector Machine*) dengan representasi manual. Dari sini mahasiswa memahami bahwa model klasik bisa bekerja, tetapi terbatas dalam menangkap kompleksitas data visual.
- **D3–D4** menunjukkan kekuatan **Convolutional Neural Networks (CNN)**. CNN mampu secara otomatis mengekstrak fitur dari citra, sehingga performanya jauh melampaui metode tradisional. Dengan dataset CIFAR-10, mahasiswa juga belajar menghadapi tantangan data berwarna, kompleks, dan penuh variasi.
- **D5** memperlihatkan bagaimana **transfer learning** memanfaatkan model pra-latih (VGG16, ResNet, dsb.) sehingga mahasiswa tidak harus melatih model dari nol. Strategi *fine-tuning* memungkinkan peningkatan akurasi signifikan dengan efisiensi waktu dan data.
- **D6** melatih mahasiswa untuk mengevaluasi model secara lebih mendalam melalui confusion matrix, classification report, dan analisis salah klasifikasi. Dengan ini mahasiswa tidak hanya melihat angka akurasi, tetapi juga memahami pola kesalahan model.
- Tugas tambahan berupa **unggah tulisan angka sendiri** memberikan pengalaman nyata bagaimana sistem klasifikasi berinteraksi dengan data dunia nyata, yang seringkali tidak “seindah” dataset standar.

Dari seluruh proses ini, mahasiswa diharapkan menyadari bahwa **membangun sistem klasifikasi gambar bukan hanya soal kode, tetapi juga soal pemahaman data, arsitektur model, strategi pelatihan, dan cara mengevaluasi hasil secara kritis**.

H. 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 Large-Scale 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.