

تثبيت المكتبات المطلوبة

في هذه الخطوة نقوم بتثبيت المكتبات اللازمة لتنفيذ مشروع تصنيف الصور باستخدام التعلم العميق. تشمل هذه المكتبات أدوات لمعالجة الصور، بناء النماذج، وتقييم الأداء.

```
In [1]: !pip install torch torchvision scikit-learn pillow matplotlib seaborn tqdm

Requirement already satisfied: torch in c:\users\bashr\miniconda3\lib\site-packages (2.10.0)
Requirement already satisfied: torchvision in c:\users\bashr\miniconda3\lib\site-packages (0.25.0)
Requirement already satisfied: scikit-learn in c:\users\bashr\miniconda3\lib\site-packages (1.8.0)
Requirement already satisfied: pillow in c:\users\bashr\miniconda3\lib\site-packages (12.0.0)
Requirement already satisfied: matplotlib in c:\users\bashr\miniconda3\lib\site-packages (3.10.8)
Requirement already satisfied: seaborn in c:\users\bashr\miniconda3\lib\site-packages (0.13.2)
Requirement already satisfied: tqdm in c:\users\bashr\miniconda3\lib\site-packages (4.67.1)
Requirement already satisfied: filelock in c:\users\bashr\miniconda3\lib\site-packages (from torch) (3.20.3)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\bashr\miniconda3\lib\site-packages (from torch) (4.15.0)
Requirement already satisfied: sympy>=1.13.3 in c:\users\bashr\miniconda3\lib\site-packages (from torch) (1.14.0)
Requirement already satisfied: networkx>=2.5.1 in c:\users\bashr\miniconda3\lib\site-packages (from torch) (3.6.1)
Requirement already satisfied: Jinja2 in c:\users\bashr\miniconda3\lib\site-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in c:\users\bashr\miniconda3\lib\site-packages (from torch) (2026.2.0)
Requirement already satisfied: setuptools in c:\users\bashr\miniconda3\lib\site-packages (from torch) (80.9.0)
Requirement already satisfied: numpy in c:\users\bashr\miniconda3\lib\site-packages (from torchvision) (2.4.1)
Requirement already satisfied: scipy>=1.10.0 in c:\users\bashr\miniconda3\lib\site-packages (from scikit-learn) (1.17.0)
Requirement already satisfied: joblib>=1.3.0 in c:\users\bashr\miniconda3\lib\site-packages (from scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.2.0 in c:\users\bashr\miniconda3\lib\site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=3 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (3.3.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\bashr\miniconda3\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\users\bashr\miniconda3\lib\site-packages (from seaborn) (2.3.3)
Requirement already satisfied: colorama in c:\users\bashr\miniconda3\lib\site-packages (from tqdm) (0.4.6)
Requirement already satisfied: pytz>=2020.1 in c:\users\bashr\miniconda3\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\bashr\miniconda3\lib\site-packages (from pandas>=1.2->seaborn) (2025.3)
Requirement already satisfied: six>=1.5 in c:\users\bashr\miniconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\bashr\miniconda3\lib\site-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\bashr\miniconda3\lib\site-packages (from Jinja2->torch) (3.0.2)
```

تحديد مسار البيانات

في هذه الخطوة نقوم بتحديد مسار مجلد البيانات الذي يحتوي على صور الزهور. سيتم استخدام هذا المسار لاحقًا لتحميل الصور وتدريب النموذج عليها.

```
In [2]: data_dir = "Data"
```

استيراد المكتبات

في هذه الخطوة نقوم باستيراد المكتبات اللازمة لتنفيذ مشروع تصنيف صور الزهور باستخدام التعلم العميق. PyTorch تساعد هذه المكتبات في تحميل البيانات، معالجة الصور، بناء النموذج، وتدريب النموذج باستخدام مكتبة

```
In [3]: import torch
import numpy as np
import matplotlib.pyplot as plt

from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
from torch import nn, optim
```

تحديد مسار بيانات التدريب والتحقق

سيتم استخدام هذه البيانات لاحقًا في عملية تدريب النموذج وتقييم أدائه (Validation) ومجلد بيانات التحقق (Train) في هذه الخطوة نقوم بتحديد مسار مجلد بيانات التدريب.

```
In [4]: train_dir = "Data/train"
val_dir = "Data/val"
```

Data Augmentation تجهيز الصور وتطبيق

في هذه الخطوة نقوم بتجهيز الصور قبل إدخالها إلى نموذج التعلم العميق. مثل قلب الصورة أفقيًا وتدويرها، وذلك لزيادة تنوع البيانات وتحسين قدرة النموذج على التعلم والتعميم (Data Augmentation) كما نقوم بتطبيق بعض تقنيات زيادة البيانات.

```
In [5]: train_transform = transforms.Compose([
    transforms.Resize((128,128)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor()
])

val_transform = transforms.Compose([
    transforms.Resize((128,128)),
    transforms.ToTensor()
])
```

DataLoader تحميل البيانات وإنشاء

(Batches) لتنظيم البيانات وتقسيمها إلى مجموعات صغيرة DataLoader كما نقوم بإنشاء ImageFolder في هذه الخطوة نقوم بتحميل صور الزهور من مجلد البيانات باستخدام مكتبة لتسهيل عملية تدريب النموذج وتحسين كفاءة المعالجة.

```
In [6]: train_dataset = datasets.ImageFolder(train_dir, transform=train_transform)
        val_dataset = datasets.ImageFolder(val_dir, transform=val_transform)

        train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
        val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

عرض الفئات الموجودة في مجموعة البيانات

الموجودة في مجموعة بيانات التدريب (Classes) في هذه الخطوة نقوم باستخراج وعرض أسماء الفئات Dataset. يساعد ذلك في معرفة أنواع الزهور التي سيتعلم النموذج تصنيفها، وكذلك معرفة عدد الفئات الموجودة في

```
In [7]: classes = train_dataset.classes
        print(classes)
        print("Number of classes:", len(classes))

['dandelion', 'iris', 'rose', 'sunflower', 'tulip']
Number of classes: 5
```

عرض بعض الصور من مجموعة البيانات

في هذه الخطوة نقوم بعرض بعض الصور من مجموعة بيانات التدريب للتأكد من أن البيانات تم تحميلها بشكل صحيح. كما يساعد ذلك على التعرف على أنواع الزهور المختلفة Dataset.

```
In [8]: images, labels = next(iter(train_loader))

        plt.figure(figsize=(8,8))

        for i in range(6):
            plt.subplot(2,3,i+1)
            plt.imshow(images[i].permute(1,2,0))
            plt.title(classes[labels[i]])
            plt.axis("off")

        plt.show()
```



عرض صور إضافية من مجموعة البيانات

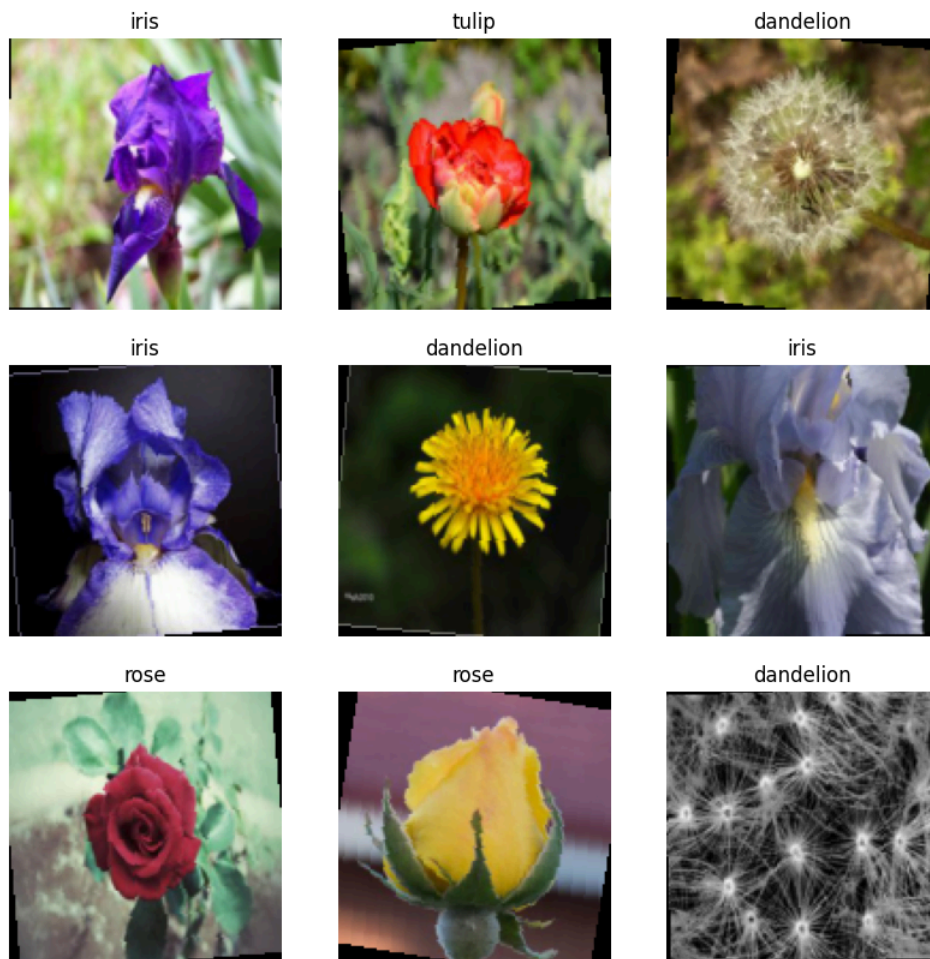
Dataset. في هذه الخطوة نقوم بعرض مجموعة أخرى من صور الزهور من بيانات التدريب للتعرف على الأنواع المختلفة الموجودة في

```
In [9]: images, labels = next(iter(train_loader))

        plt.figure(figsize=(10,10))

        for i in range(9):
            plt.subplot(3,3,i+1)
            plt.imshow(images[i].permute(1,2,0))
            plt.title(classes[labels[i]])
            plt.axis("off")

        plt.show()
```



```
In [10]: images, labels = next(iter(train_loader))

plt.figure(figsize=(15,15))

for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(images[i].permute(1,2,0))
    plt.title(classes[labels[i]])
    plt.axis("off")

plt.show()
```

rose



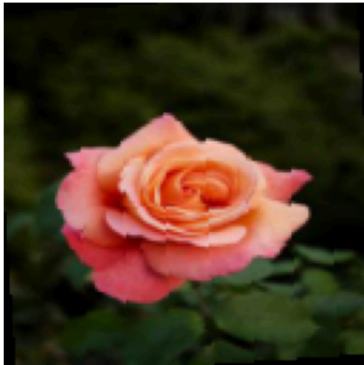
dandelion



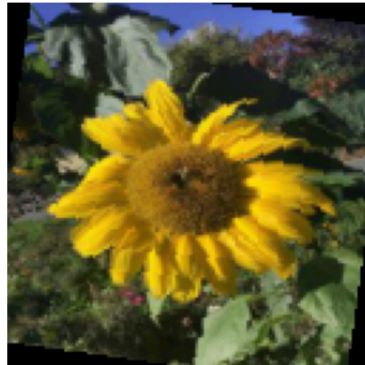
dandelion



rose



sunflower



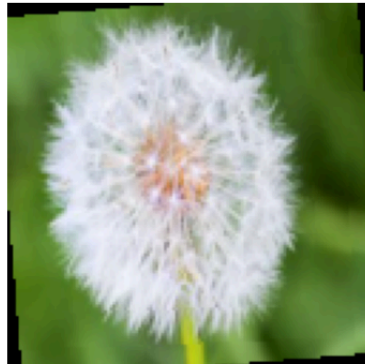
sunflower



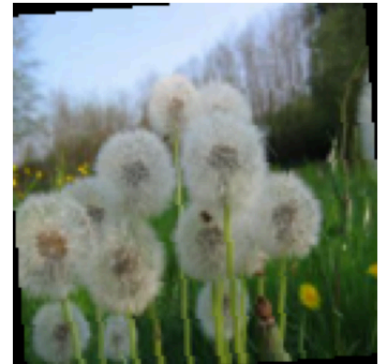
sunflower



dandelion



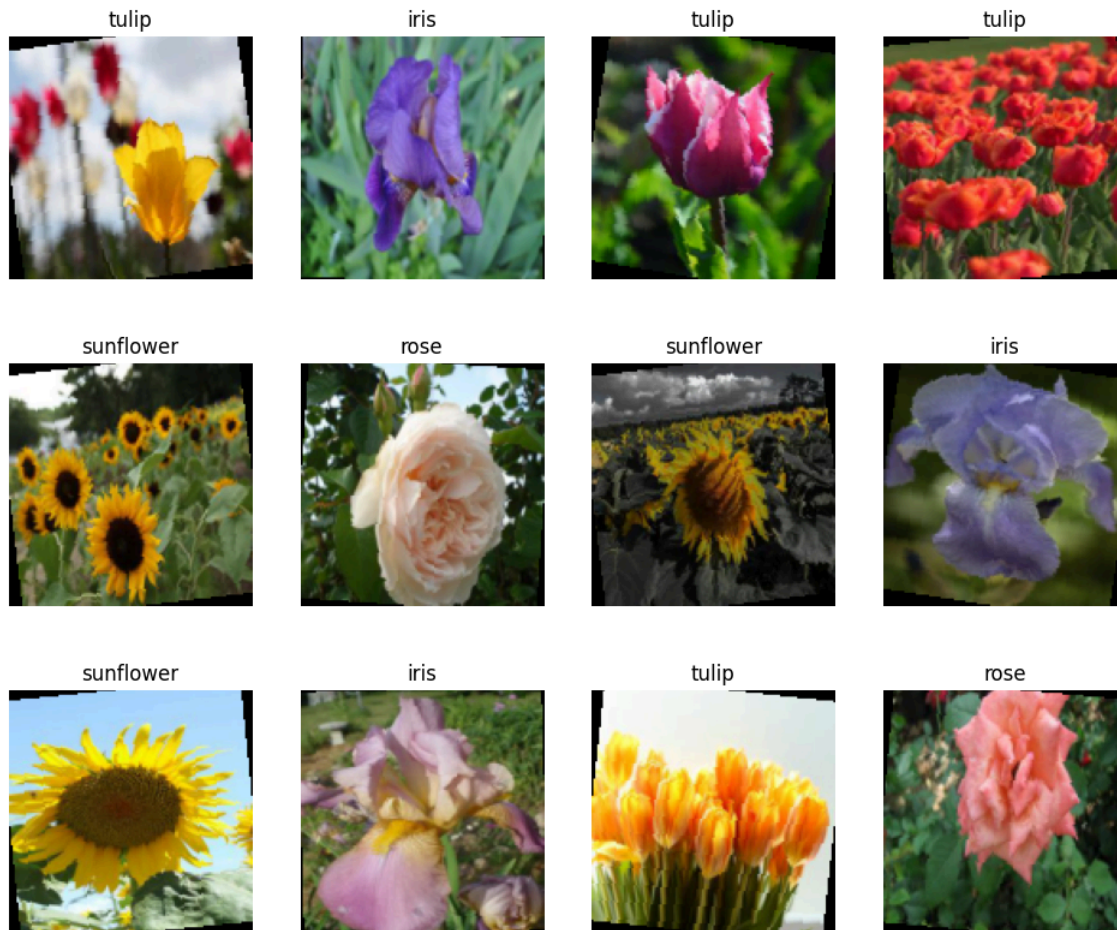
dandelion



```
In [11]: images, labels = next(iter(train_loader))
plt.figure(figsize=(12,10))

for i in range(12):
    plt.subplot(3,4,i+1)
    plt.imshow(images[i].permute(1,2,0))
    plt.title(classes[labels[i]])
    plt.axis("off")

plt.show()
```



بناء نموذج التعلم العميق

MobileNetV2 في هذه الخطوة نقوم بإنشاء نموذج التعلم العميق باستخدام الشبكة العصبية الجاهزة

من النماذج الخفيفة والسريعة والمناسبة لتصنيف الصور MobileNetV2 تُعد

أو بطاقة (CPU) بعد ذلك نقوم بتعديل الطبقة الأخيرة من النموذج لتناسب مع عدد الفئات الموجودة في مجموعة البيانات، ثم نحدد الجهاز المستخدم في التدريب سواء كان المعالج (GPU) الرسومات

```
In [12]: model = models.mobilenet_v2(pretrained=True)

# تجميد الطبقات الجاهزة لتسريع التدريب
for param in model.parameters():
    param.requires_grad = False

# تعديل الطبقة الأخيرة فقط
model.classifier[1] = nn.Linear(model.classifier[1].in_features, len(classes))

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

C:\Users\bashr\miniconda3\Lib\site-packages\torchvision\models_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
warnings.warn(
C:\Users\bashr\miniconda3\Lib\site-packages\torchvision\models_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing `weights=MobileNet\_V2\_Weights.IMAGENET1K\_V1`. You can also use `weights=MobileNet\_V2\_Weights.DEFAULT` to get the most up-to-date weights.
warnings.warn(msg)

إعداد عملية التدريب

التي سيتم استخدامها أثناء تدريب النموذج (Optimizer) وخوارزمية التحسين (Loss Function) في هذه الخطوة نقوم بتحديد دالة الخطأ

```
In [13]: criterion = nn.CrossEntropyLoss()

# تدريب الطبقة الأخيرة فقط
optimizer = optim.Adam(model.classifier[1].parameters(), lr=0.001)
```

تدريب النموذج

في هذه الخطوة يتم تدريب النموذج على بيانات التدريب ليتعلم التمييز بين أنواع الزهور المختلفة، وذلك من خلال تمرير الصور على النموذج وحساب الخطأ وتحديث الأوزان

```
In [14]: epochs = 1

for epoch in range(epochs):
    model.train()
    running_loss = 0
    correct = 0
```

```
total = 0

for images, labels in train_loader:
    images = images.to(device)
    labels = labels.to(device)

    optimizer.zero_grad()

    outputs = model(images)
    loss = criterion(outputs, labels)

    loss.backward()
    optimizer.step()

    running_loss += loss.item()

_, predicted = torch.max(outputs, 1)
total += labels.size(0)
correct += (predicted == labels).sum().item()

train_accuracy = correct / total
print("Epoch:", epoch + 1)
print("Training Loss:", running_loss)
print("Training Accuracy:", train_accuracy)
```

Epoch: 1

Training Loss: 53.70178801193833

Training Accuracy: 0.8984741784037559

اختبار النموذج

بعد الانتهاء من التدريب يتم اختبار النموذج باستخدام بيانات التحقق لمعرفة مدى دقته في تصنيف صور الزهور بشكل صحيح.

```
In [15]: model.eval()

correct = 0
total = 0

with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)

        outputs = model(images)
        _, predicted = torch.max(outputs, 1)

        total += labels.size(0)
        correct += (predicted == labels).sum().item()

val_accuracy = correct / total
print("Validation Accuracy:", val_accuracy)
```

Validation Accuracy: 0.9714285714285714

الخلاصة

بعد تجهيز البيانات وتدريب النموذج تم اختبار الأداء على بيانات التحقق، حيث حقق النموذج دقة MobileNetV2. تم في هذا المشروع تصنيف صور الزهور باستخدام نموذج التعلم العميق بلغت حوالي 97%، مما يدل على قدرة النموذج على تصنيف أنواع الزهور المختلفة بكفاءة.