

جمهورية العراق — وزارة الصحة

مديرية التشخيص • برنامج التدريب المهني

الأدوات العملية للذكاء الاصطناعي

دليل مصوّر خطوة بخطوة — الخطة العشرية المحدثة
سطر الأوامر • Miniconda • PyTorch/TensorFlow GPU • Colab • تدريب YOLO



د. حسام صلاح مهدي

رئيس قسم الذكاء الاصطناعي — جامعة المصطفى • مدير الدورة

ماذا سنتعلم في هذا الدليل؟

1. الخطة العشرية المحدثه

2. سطر الأوامر (CLI)

3. Miniconda والبيئات المعزولة

4. تثبيت PyTorch وTensorFlow مع GPU

5. Google Colab السحابي

6. التكامل: حاسوبك وColab

7. تدريب YOLO خطوة بخطوة

8. الخلاصة وبطاقة الأوامر

القسم 1

الخطة العشرية المحدثه

عشرة أيام تمزج علم البيانات بأدوات التدريب الحديثة

الأسبوع الأول — التأسيس والبنية التحتية



ناتج الأسبوع: بيئة عمل جاهزة على الحاسوب وفي
السحابة + أول تحليل لبيانات مرضى حقيقية البنية

- كل يوم 90 دقيقة: نظرية قصيرة ثم تطبيق حي ثم تحدّ سريري
- من اليوم الثاني تعمل الأوامر على حاسوبك ومن اليوم الرابع في السحابة

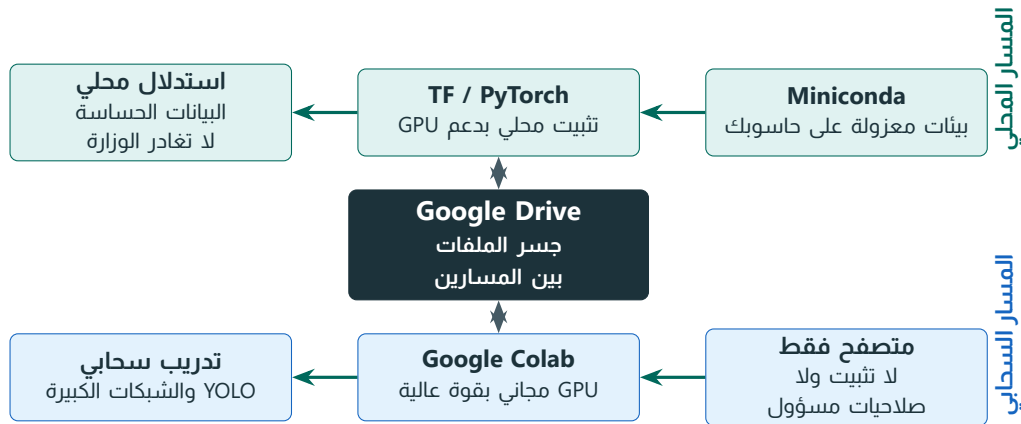
الأسبوع الثاني — النماذج والتدريب الحقيقي



ناتج الأسبوع: نموذج YOLO مدرب بأيديكم يكشف
الخلايا في صورة لطاقة دم — سحابياً ومحلياً

- اليوم التاسع هو ذروة الدورة: تدريب كامل بسطر أوامر واحد على GPU مجاني
- اليوم العاشر مشروع جماعي يختار كل فريق مساره: محلي أو سحابي

مساران متوازيان طوال الدورة — والجسر بينهما



- لا نختار مساراً واحداً — نتقن الاثنين ونمرّر الملفات بينهما عبر Drive
- القاعدة: البيانات الحساسة محلياً، والتدريب الثقيل سحابياً على بيانات اصطناعية

إيقاع الجلسة الثابت — نظرية ثم تطبيق ثم تحدّ

الدقيقة 90

الدقيقة 60

الدقيقة 20

الدقيقة 0



90 دقيقة يومياً

في أيام الأدوات (2، 4، 7، 8، 9): كتلة التطبيق كلها أوامر
تُنفَّذ حرفياً — كل متدرب يكتب الأمر بيده ويرى نتيجته

● الإيقاع نفسه كل يوم — بعد اليوم الثالث يختفي الشكل ويبقى المحتوى

القسم 2

سطر الأوامر (CLI)

لوحة تحكم المحترفين — نكتب أمراً فينفذ حرفياً

لماذا سطر الأوامر وليس الفأرة؟

سطر الأوامر (CLI)

أمر نصي يُنسخ ويُلصق ويُحفظ

نفس الأمر يعمل عند الجميع

قابل للأتمتة والتكرار بلا خطأ



الواجهة الرسومية (GUI)

نقرات لا يمكن توثيقها

تختلف من نسخة إلى نسخة

يصعب تكرارها 100 مرة

- كل أدوات الذكاء الاصطناعي الحديثة (yolo، pip، conda) تُدار بسطر الأوامر
- في هذه الدورة: كل خطوة تثبيت وتدريب = أمر نصي مكتوب أمامكم تنسخونه حرفياً

أين نجد الطرفية (Terminal)؟

3

خلية Colab تبدأ بـ !

طرفية لينكس داخل
المتصفح — بلا أي تثبيت

2

CMD / PowerShell

طرفية ويندوز الأصلية —
تعمل بعد `conda init`

1

Anaconda Prompt

يُثَبَّت مع Miniconda —
خيارنا الأول على ويندوز

الأوامر واحدة في الأماكن الثلاثة — تتعلمها مرة وتستعملها في كل مكان

- على حواسيب الوزارة: ابحث في قائمة ابدأ عن Anaconda Prompt بعد تثبيت Miniconda
- في Colab: أي خلية تبدأ بعلامة التعجب ! تُنفَّذ كأمر طرفية لا كسيفرة بايثون

تشرح الأمر — ثلاث قطع لا غير

```
conda install pandas
```

pandas



الهدف — على ماذا؟

install



الأمر الفرعي
— ماذا يفعل؟

conda



البرنامج — من ننادي؟

- القالب نفسه في كل الأدوات: `yolo train`, `pip install`, `conda create`
- بعض الأوامر تضيف خيارات بصيغة `key=value` — سنشرّحها عند أمر التدريب

أوامر التنقل الثلاثة — أين أنا؟ ماذا هنا؟ إلى أين؟

```
C:\Users\lab> cd Desktop\projects      # move into a folder
C:\Users\lab\Desktop\projects> dir     # list what is here
C:\Users\lab\Desktop\projects> mkdir yolo_data  # new folder
```

— أنشئ مجلداً `mkdir`

— اعرض المحتوى `dir`

— انتقل إلى مجلد `cd`

- على لينكس (Colab و WSL): `ls` بدل `dir`، والمسارات بشرطة أمامية /
- لماذا يهم؟ لأن أمر التدريب يحتاج مسار البيانات — ولن تجده إن لم تعرف أين أنت
- خدعة مفيدة: اكتب `cd` ثم اسحب المجلد من مستكشف الملفات إلى الطرفية فيُلصق مساره

سطر الموجّه يخبرك بكل شيء

```
(medai) C:\Users\lab\projects>
```

>

جاهز — اكتب أمرك

C:\Users\lab\projects

المجلد الحالي — منه
تُقرأ المسارات النسبية

(medai)

البيئة النشطة الآن —
هنا ستُنبت المكتبات

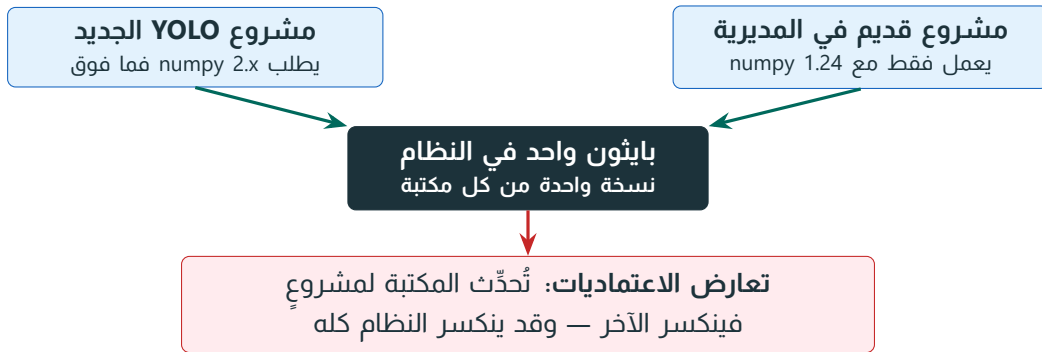
- إن لم تر اسم البيئة بين قوسين فأنت خارج بيئتك — نفذ `conda activate medai` أولاً
- هذه أول نظرة تلقيها قبل أي أمر تثبيت أو تدريب — عادة المحترفين

القسم 3

Miniconda والبيئات المعزولة

غرفة مختبر مستقلة لكل مشروع — ننشئها ونفعلها بالأوامر

المشكلة: مشروعان يتنازعان بايثون واحداً



- الظاهرة أشهر من أن تُحصى حتى إن لها اسماً: جحيم الاعتماديات (Dependency Hell)
- الحل ليس الحذر — الحل بنيوي: عزل كل مشروع في بيئة مستقلة

الحل: بيئة معزولة لكل مشروع (Environment)

بيئة yolo — الرؤية الحاسوبية

Python 3.11

torch 2.x + CUDA

ultralytics • opencv

بيئة medai — تحليل البيانات

Python 3.11

numpy 2.2 • pandas 2.3

matplotlib • scikit-learn

غرفتا مختبر منفصلتان: لكل أدواتها ونسخها
— ولا تلمس إحداها أدوات الأخرى أبداً

- البيئة (Environment): نسخة بايثون + مجموعة مكتبات معزولة في مجلد خاص
- ننقل بين البيئات بأمر واحد — والأداة التي تدير كل هذا اسمها conda

Anaconda أم Miniconda؟

Miniconda

نحو 100 ميغابايت فقط

conda + بايثون — لا غير

نثبت ما نحتاجه فقط عند الحاجة

Anaconda

حجم يتجاوز 4 غيغابايت

مئات المكتبات مثبتة سلفاً

مناسب لمن يريد كل شيء جاهزاً

اختيار الدورة: Miniconda — خفيف على أجهزة الوزارة، والأمر conda فيهما واحد بلا أي فرق

تثبيت Miniconda على ويندوز — ثلاث خطوات



```
:: Option A - one command on Windows 10/11:
```

```
winget install -e --id Anaconda.Miniconda3
```

```
:: Option B - download the official installer, then run it:
```

```
:: https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86\_64.exe
```

```
:: verify (in a NEW Anaconda Prompt window):
```

```
conda --version
```

- لا يحتاج صلاحيات مسؤول عند اختيار Just Me — مناسب لحواسيب الدوائر
- إن ظهر رقم نسخة مثل 25.x conda فالتثبيت سليم وجاهز

دورة حياة البيئة — خمس محطات



- الإنشاء والتثبيت مرة واحدة في العمر — التفعيل في بداية كل جلسة عمل
- كل ما يُثَبَّت وأنت داخل البيئة يبقى محصوراً فيها

تشرح أمر الإنشاء

```
conda create -n medai python=3.11
```

python=3.11



نسخة بايثون
المثبتة داخلها

-n medai



الاسم الذي سنناديها به

conda create



أنشئ بيئة جديدة

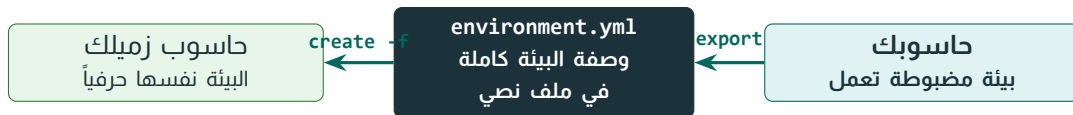
- تحديد نسخة بايثون هو مفتاح التوافق: بعض الأدوات تشترط نسخة بعينها
- أضف -y في نهاية الأمر ليجيب بنعم تلقائياً عن أسئلة التأكيد

بطاقة أوامر البيئات — كل ما ستحتاجه

```
conda create -n medai python=3.11 -y      # create (once)
conda activate medai                      # enter (every session)
conda install numpy pandas matplotlib -y
pip install ultralytics                  # pip works inside too
conda list                                # what is inside this env?
conda env list                            # all my environments
conda deactivate                          # leave the environment
conda remove -n medai --all -y           # delete it completely
```

- `conda install` و `pip install` كلاهما يعمل داخل البيئة — والقاعدة العملية اليوم: جَرِّب `conda` أولاً للحزم العلمية، و `pip` لكل ما سواها (مثل `ultralytics`)
- علامة النجمة: نجمة * في ناتج `conda env list` تشير إلى البيئة النشطة الآن

مشاركة البيئة مع الزملاء — قابلية التكرار



```
conda env export > environment.yml      # on your machine
conda env create -f environment.yml      # on your colleague's machine
```

- هكذا تُجهَّز قاعة تدريب كاملة ببيئة موحدة — ملف واحد يُوزَّع على الجميع
- الملف نصي بسيط يمكن فتحه وقراءته: أسماء الحزم ونسخها سطراً سطراً

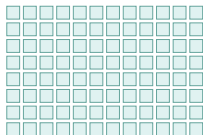
القسم 4

تثبيت PyTorch و TensorFlow بدعم GPU

من فحص البطاقة إلى أمر التثبيت الصحيح والتحقق منه

لماذا يحتاج التدريب إلى GPU؟

المسرّع الرسومي GPU



آلاف الأنوية الصغيرة معاً
ممتاز لحساب المصفوفات المتوازي

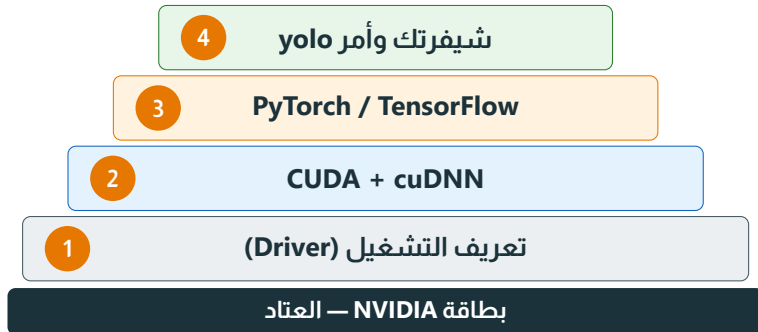
المعالج CPU



أنوية قليلة قوية جداً
ممتاز للمهام المتسلسلة

- تدريب الشبكة العصبية = ملايين عمليات ضرب مصفوفات — كلها قابلة للتوازي
- الفرق عملياً: تدريب يستغرق ساعات على CPU ينجز بدقائق على GPU

سلم GPU — ماذا يقف بين شيفرتك والبطاقة؟



لا يتغير محلياً أو سحابياً

أمر تثبيت واحد صحيح

يصل مع حزمة pip تلقائياً

بنته أنت — حدّته من موقع NVIDIA

- الخبر الجيد: حزم pip الحديثة تحمل CUDA و cuDNN في داخلها — كل ما يلزمك تعريف NVIDIA حديث ثم أمر التثبيت الصحيح

الفحص الأول دائماً: nvidia-smi

```
C:\> nvidia-smi
```

```
+-----+
| NVIDIA-SMI 560.94    Driver Version: 560.94    CUDA: 12.6    |
| GPU   Name                               Memory-Usage   |
|  0   NVIDIA GeForce RTX 3060            1024MiB / 12288MiB |
+-----+
```

البطاقة وذاكرتها
الذاكرة تحدد حجم الدفعة

أعلى CUDA مدعومة
هنا: 12.6

نسخة التعريف
يجب أن تكون حديثة

- نجح الأمر وظهر الجدول؟ لديك GPU جاهز — انتقل إلى أمر التثبيت
- فشل الأمر؟ إما لا بطاقة NVIDIA في الجهاز أو التعريف غير مثبت — استعمل مسار CPU أو Colab

تثبيت PyTorch — في بيئة conda نظيفة

```
conda create -n torchgpu python=3.11 -y
conda activate torchgpu

# GPU (NVIDIA) build - from the official PyTorch index:
pip install torch torchvision --index-url https://download.pytorch.org/whl/cu126

# CPU-only build (works on any machine):
pip install torch torchvision
```

- cu126 تعني نسخة مبنية على CUDA 12.6 — ولّد الأمر المناسب لجهازك من صفحة pytorch.org (اختر Windows + pip + CUDA)
- القناة الرسمية القديمة في conda أوقفت — الطريقة المعتمدة اليوم: pip داخل بيئة conda معزولة
- التنزيل نحو 2-3 غيغابايت لنسخة GPU — مرة واحدة لكل بيئة

التحقق: هل يرى PyTorch البطاقة فعلياً؟

```
(torchgpu) C:\> python -c "import torch; print(torch.__version__)"
2.7.1+cu126
(torchgpu) C:\> python -c "import torch; print(torch.cuda.is_available())"
True
(torchgpu) C:\> python -c "import torch; print(torch.cuda.get_device_name(0))"
NVIDIA GeForce RTX 3060
```

False ← راجع التعريف وأمر
التثبيت، أو اعمل بـ CPU / Colab

True ← جاهز — دَرَبَ
على GPU بـ device=0

● لاحظ +cu126 في رقم النسخة: دليل أنك تثبت نسخة GPU لا نسخة CPU

TensorFlow بدعم GPU — الخريطة الصادقة

أين تشغل
TensorFlow
على GPU؟

ويندوز الأصلي

النسخ الحديثة تعمل بـ
CPU فقط — دعم
GPU توقف عند 2.10

**WSL2 (لينكس
داخل ويندوز)**

دعم GPU كامل وحديث
عبر tensorflow[and-cuda]

Google Colab

جاهز فوراً بلا تثبيت
— خيار الدورة الأول

**قاعدة الدورة: تدريب TensorFlow على GPU يتم في
Colab — ومحلياً نكتفي بنسخة CPU للتجارب الصغيرة**

أوامر TensorFlow الثلاثة + التحقق

```
# 1) Colab: nothing to install - it is already there with GPU.
```

```
# 2) WSL2 / Linux with NVIDIA GPU:  
pip install tensorflow[and-cuda]
```

```
# 3) Native Windows (CPU only):  
conda create -n tfcpu python=3.11 -y  
conda activate tfcpu  
pip install tensorflow
```

```
python -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"  
# GPU found -> [PhysicalDevice(name='/physical_device:GPU:0', ...)]  
# no GPU      -> []      (training will silently run on CPU)
```

● القائمة الفارغة [] ليست خطأً — لكنها تعني أن التدريب سيجري على CPU ببطء

أي إطار عمل لأي مهمة في الدورة؟

PyTorch

معيّار الأبحاث الطبية الحديثة

الأساس الذي يقف عليه YOLO

الأيام 8-9: محلياً وسحابياً

TensorFlow + Keras

واجهة Keras سهلة للمبتدئين

اليوم 7: تصنيف بيانات الجداول

ندربه في Colab

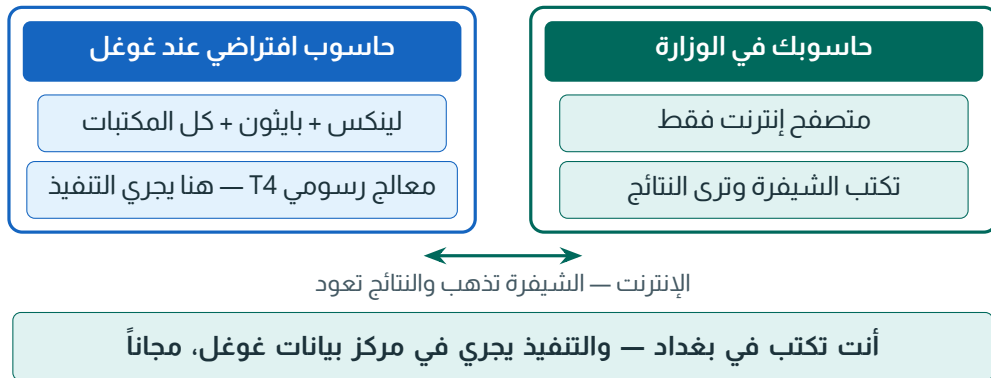
لا صراع بينهما: المفاهيم واحدة — Tensors وطبقات
وتدريب — ومن أتقن أحدهما فهم الآخر في ساعة

القسم 5

Google Colab — مختبر التدريب السحابي

حاسوب قوي بمعالج رسومي مجاني — يعمل من متصفحك

ما الذي يحدث فعلاً عند فتح Colab؟



● العنوان: colab.research.google.com — يلزم فقط حساب Google

لماذا يناسب كوادر الوزارة؟ — وبأي شرط؟

GPU مجاني يوازي
جهازاً بآلاف الدولارات

صفر تثبيت وصفر صلاحيات مسؤول

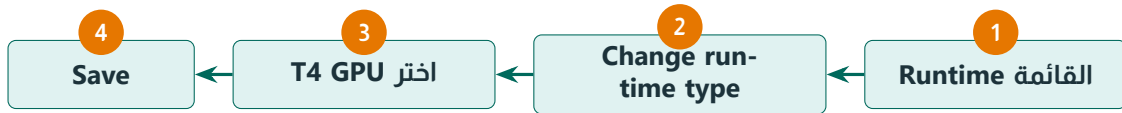
الدفتر يُشارك كمستند — تعاون فوري

كل مكتبات الدورة مثبتة مسبقاً

الشرط الأحمر: لا تُرفع أي بيانات مرضى حقيقية إلى السحابة — في التدريب نستعمل بيانات اصطناعية أو مموّهة بالكامل (Anonymized)

هذا ليس عائقاً بل منهج عمل: التعلم سحابياً على
بيانات آمنة، والتشغيل الحقيقي محلياً داخل الوزارة

تفعيل الـ GPU — أربع نقرات وفحص واحد



```
!nvidia-smi
# Tesla T4, 15360MiB total memory - free of charge
```

- بلا هذه الخطوة يعمل الدفتر على CPU — والتدريب يصبح أبطأ عشرات المرات
- الحصة المجانية محدودة بساعات يومياً — فعّل الـ GPU عند التدريب فقط لا طوال الجلسة

علامة ! — الخلية تتحول إلى طرفية

الحاسوب الافتراضي
المفسر والطرفية يعملان على
الجهاز نفسه ويريان الملفات نفسها

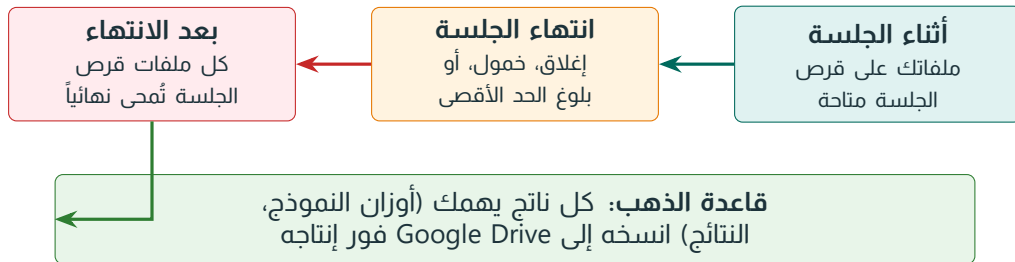
خلية عادية: $x = 5$
تذهب إلى مفسر بايثون

خلية تبدأ بـ **!pip install**
تذهب إلى طرفية لينكس

```
!pip install ultralytics -q      # install a library
!ls /content                     # list session files
!unzip -q blood_cells.zip -d /content/
```

● كل أوامر سطر الأوامر التي تعلمناها تعمل هنا حرفياً — مسبقة بعلامة ! فقط

تحذير مصيري: قرص الجلسة مؤقت



- الجلسة المجانية تنتهي بعد ساعات أو عند الخمول — هذا سلوك مقصود لا عطل
- Drive دائم وقرص الجلسة زائل: هذه الجملة تختصر ثلث مشكلات المبتدئين مع Colab

ربط Google Drive بالجلسة

```
from google.colab import drive
drive.mount('/content/drive')
# after approving access, your files appear under:
# /content/drive/MyDrive/...
```



- عند التنفيذ تظهر نافذة استئذان — وافق عليها بحساب Google نفسه
- القراءة والكتابة عبر هذا المسار تصل إلى Drive مباشرة — وتبقى بعد موت الجلسة

ثلاث طرق لنقل الملفات من وإلى الجلسة

3

Google Drive

الطريقة المعتمدة للبيانات
والنماذج — دائمة ومنظمة

2

files.upload()

من داخل الشيفرة —
مناسبة للملفات الصغيرة

1

لوحة الملفات

سحب وإفلات من الجهاز —
سريعة لكنها تُمحي مع الجلسة

```
from google.colab import files
files.upload()           # pick a file from your PC
files.download('best.pt') # push a file back to your PC
```

● في مشروع التدريب سنستخدم Drive حصراً — وهو محور القسم التالي

القسم 6

التكامل: حاسوبك ↔ Colab

جهاز محلياً — درّب سحابياً — شغل محلياً

الرحلة الكاملة في صورة واحدة



● هذا هو نمط العمل المهني القياسي: قوة السحابة للتدريب وخصوصية المحلي للتشغيل

الخطوات 1-3 على حاسوبك: اضغط وارفع

```
:: in the folder that CONTAINS blood_cells\ (Windows 10/11):  
tar -a -c -f blood_cells.zip blood_cells  
  
:: or with PowerShell:  
Compress-Archive -Path blood_cells -DestinationPath blood_cells.zip
```

أو تطبيق **Drive for Desktop**: انسخ
الملف إلى مجلد Drive فيزَامَن تلقائياً

الرفع بالمتصفح: افتح
drive.google.com
واسحب الملف المضغوط

- ملف مضغوط واحد يُرفع أسرع بكثير من آلاف الصور الصغيرة المتفرقة
- سَمِّ الملف باسم واضح بلا فراغات وبالأحرف اللاتينية: `blood_cells.zip`

الخطوة 4 في Colab: اربط وفك الضغط

```
from google.colab import drive
drive.mount('/content/drive')
```

```
!unzip -q /content/drive/MyDrive/blood_cells.zip -d /content/
!ls /content/blood_cells
# data.yaml  images  labels
```

لماذا ننسخ إلى قرص الجلسة بدل التدريب من Drive مباشرة؟ لأن قرص الجلسة أسرع بأضعاف — والتدريب يقرأ كل صورة عشرات المرات

• تحقق دائماً بأمر 1s من ظهور data.yaml والمجلدين قبل التدريب

الخطوة 5: درّب — ثم أنقذ الأوزان فوراً

```
!pip install ultralytics -q

!yolo detect train model=yolo11n.pt \
  data=/content/blood_cells/data.yaml \
  epochs=50 imgsz=640 device=0

# THE most important line of the day:
!cp /content/runs/detect/train/weights/best.pt \
  /content/drive/MyDrive/blood_cells_best.pt
```

تذكير القرص المؤقت: إن انتهت الجلسة قبل نسخ best.pt إلى Drive ضاع التدريب كله — انسخ فور انتهاء التدريب لا في نهاية اليوم

الخطوة 6 على حاسوبك: نَزِّل وشغِّل

```

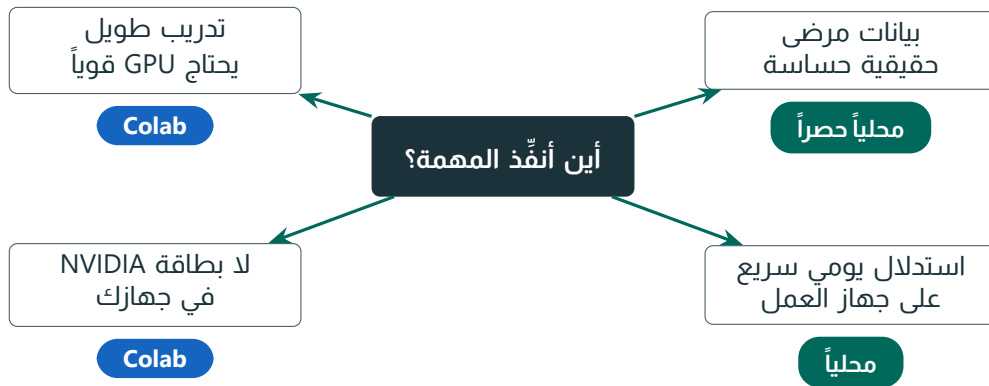
:: download blood_cells_best.pt from drive.google.com (browser),
:: or via CLI if the file is shared "anyone with the link":
pip install gdown
gdown https://drive.google.com/uc?id=FILE_ID -O best.pt

:: run it locally - prediction is light, CPU is enough:
conda activate yolo
yolo detect predict model=best.pt source=new_smear.jpg save=True

```

- FILE_ID تجده في رابط مشاركة الملف بين المقطعين d و view
- النتيجة: صورة بالصناديق المكتشفة في مجلد runs\detect\predict
- الدرس الكبير: التدريب الثقيل يحتاج السحابة — لكن نموذجك المدرب يعمل على أي جهاز في المديرية

متى محلياً ومتى سحابياً؟ — خريطة القرار



● القاعدة المختصرة: الحساسية تفرض المحلي، والحسابات الثقيلة ترجّح السحابة

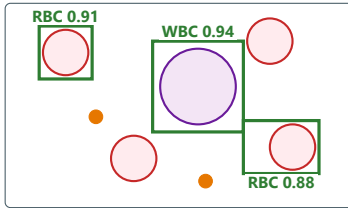
القسم 7

تدريب YOLO خطوة بخطوة

كشف خلايا الدم في الصور المجهرية — بسطر أوامر واحد

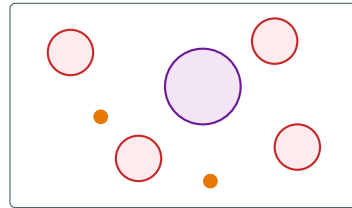
تصنيف أم كشف؟ — الفرق الذي يهم المختبر

كشف (Detection)



لكل خلية: الصنف +
الموقع + نسبة الثقة

تصنيف (Classification)



جواب واحد للصورة
كلها: «توجد خلايا بيضاء»

● الكشف يجيب عن سؤال المختبري الحقيقي: أين الخلايا؟ وكم عددها من كل نوع؟

ما YOLO؟ — نظرة واحدة تكفي



● نستعمل حزمة ultralytics — أشهر تطبيق عملي لعائلة YOLO وأسهلها تعليمياً

هيكل مجلد البيانات — قانون YOLO الصارم

```

blood_cells/
|-- data.yaml          <- the dataset ID card
|-- train/
|   |-- images/        205 x .jpg    (learning set)
|   `-- labels/        205 x .txt
|-- valid/
|   |-- images/        87 x .jpg    (honest exam set)
|   `-- labels/        87 x .txt
`-- test/  images/ + labels/  72 x (final exam)
  
```

القاعدة الذهبية: لكل صورة ملف تسمية بالاسم نفسه — IMG_0042.jpg ↔ IMG_0042.txt

- train للتعليم و valid امتحان نزيه لم تره الشبكة و test للاختبار الختامي — وتقسيم مجموعتنا الحقيقي: 205 و 87 و 72
- صورة بلا أجسام؟ ملف تسمية فارغ — لكنه موجود

تشرح سطر التسمية — خمسة أرقام لكل خلية

1 0.512 0.430 0.220 0.180

0.180



ارتفاع الصندوق

0.220



عرض الصندوق

0.430



مركز الصندوق
عمودياً

0.512



مركز الصندوق أفقياً

1



رقم الصنف:
1 = WBC

الأربعة كلها نسب من 0 إلى 1 من أبعاد الصورة — فلا يهم
أكانت الصورة 640 بكسل أم 4000: التسمية نفسها صالحة

● كل سطر في ملف التسمية = جسم واحد — خمس خلايا في الصورة تعني خمسة أسطر

بطاقة التعريف data.yaml — أربعة أسطر جوهرية

```
path: /content/blood_cells      # dataset root folder
train: images/train             # relative to path
val: images/val
names:
  0: RBC
  1: WBC
  2: Platelets
```

- path هو السطر الوحيد الذي يتغير بين البيئتين: في كولا ب مسار لينكس للجلسة كما في الأعلى، وعلى ويندوز مثلاً C:\datasets\blood_cells
- أرقام names هي نفسها الأرقام في ملفات التسمية — الرقم 1 هنا يعني WBC هناك
- خطأ شائع: تغيير ترتيب الأصناف بعد التوسيم — الأرقام عندئذ تشير إلى أصناف خاطئة

من أين نأتي بالبيانات الموسومة؟

1

التوسيم اليدوي

أدوات CVAT و Labellmg
Roboflow: ترسم الصناديق
بالفأرة وتصدّر بصيغة YOLO

2

مجموعات طبية جاهزة

مثل BCCD لخلايا الدم:
مئات الصور المجهرية
الموسومة، عامة ومجانية

3

للتجربة الفورية

coco8: ثماني صور تنزل تلقائياً
— لاختبار السلسلة قبل بياناتك

مشروع الدورة: مجموعة خلايا دم بنمط BCCD
بثلاثة أصناف — RBC و WBC و Platelets

● جودة التوسيم سقف جودة النموذج — صندوق مهزوز يعلم الشبكة خطأً مهزوزاً

التثبيت وفحص الدخان — قبل البيانات الحقيقية

```
pip install ultralytics
yolo checks          # versions, torch, GPU detected?

# 2-minute smoke test on the tiny built-in dataset:
yolo detect train model=yolo11n.pt data=coco8.yaml epochs=3 imgsz=640
```

- أول تشغيل ينزل أوزان yolo11n.pt تلقائياً (نحو 5 ميغابايت) — مرة واحدة
- coco8.yaml مجموعة مدمجة من 8 صور تنزل وحدها — هدفها إثبات أن السلسلة كلها تعمل: القراءة والتدريب والحفظ
- نجح فحص الدخان؟ بَدِّل data= إلى ملفك وابدأ التدريب الحقيقي

تشرح أمر التدريب — المعاملات السبعة

```
yolo detect train model=yolo11n.pt data=data.yaml epochs=50 imgsz=640 batch=16 device=0
```

data=data.yaml
بطاقة تعريف بياناتك

model=yolo11n.pt
نقطة البداية: أوزان مدربة
مسبقاً (Transfer Learning)

yolo detect train
الأداة + مهمة الكشف
+ فعل التدريب

device=0
أول GPU — أو CPU

batch=16
كم صورة معاً
في الخطوة

imgsz=640
توحيد حجم
الصور بكسلاً

epochs=50
مرور على كامل
البيانات 50 مرة

● نفدت ذاكرة الـ GPU؟ أنقص batch إلى 8 ثم 4 — هذا أول مقبض ضبط

الوصفة الكاملة في Colab — أربع خلايا

```
# cell 1 - install the tool
!pip install ultralytics -q

# cell 2 - connect Drive and unpack the dataset
from google.colab import drive
drive.mount('/content/drive')
!unzip -q /content/drive/MyDrive/blood_cells.zip -d /content/

# cell 3 - train on the free GPU (about 15-20 min on T4)
!yolo detect train model=yolo11n.pt data=/content/blood_cells/data.yaml epochs=50 imgsz=640
device=0

# cell 4 - rescue the weights BEFORE the session ends
!cp /content/runs/detect/train/weights/best.pt /content/drive/MyDrive/blood_cells_best.pt
```

● لا تنسَ تفعيل GPU T4 من قائمة Runtime قبل الخلية الثالثة

الوصفة الكاملة على حاسوبك — الأمر نفسه

```
:: once: env with GPU PyTorch (section 4), then the tool:
conda activate yolo
pip install ultralytics

:: train - identical command, Windows paths:
yolo detect train model=yolo11n.pt data=C:\datasets\blood_cells\data.yaml epochs=50 imgsz=640
device=0

:: no NVIDIA card? train on the CPU (slower - overnight run):
yolo detect train model=yolo11n.pt data=C:\datasets\blood_cells\data.yaml epochs=50 imgsz=640
device=cpu
```

**لاحظ: الفرقان الوحيدان عن كولاب هما صيغة المسارات
وغياب علامة ! — المهارة واحدة تماماً هنا وهناك**

ماذا يترك التدريب خلفه؟ — مجلد runs

```
runs/detect/train/
|-- weights/
|   |-- best.pt          <- THE product: deploy this one
|   |-- last.pt          resume training from here
|-- results.png          loss + mAP curves per epoch
|-- confusion_matrix.png which classes get confused
`-- args.yaml            every setting used (reproducibility)
```

- `best.pt` يلتقط عند أفضل أداء على مجموعة `val` — لا في الحقبة الأخيرة، فالحقبة المتأخرة قد تكون فرط تخصيص (Overfitting)
- افتح `results.png` بعد كل تدريب: منحنى خسارة يهبط ومنحنى `mAP` يصعد = تدريب سليم
- `args.yaml` يوثق تجربتك كاملة — أرفقه بأي تقرير للمديرية

هل النموذج جيد؟ — التقييم ثم الاستخدام

```
yolo detect val model=best.pt data=data.yaml
# -> mAP50: 0.91      mAP50-95: 0.63

yolo detect predict model=best.pt source=smeat_042.jpg conf=0.4 save=True
# -> annotated image saved under runs/detect/predict/
```

mAP50-95: المعدل عبر معايير
أشد صرامة — أدنى دائماً،
وهو مقياس الأوراق البحثية

mAP50: دقة الكشف بمعيار
تطابق متساهل — المؤشر الأول
للمبتدئ؛ فوق 0.85 بداية ممتازة

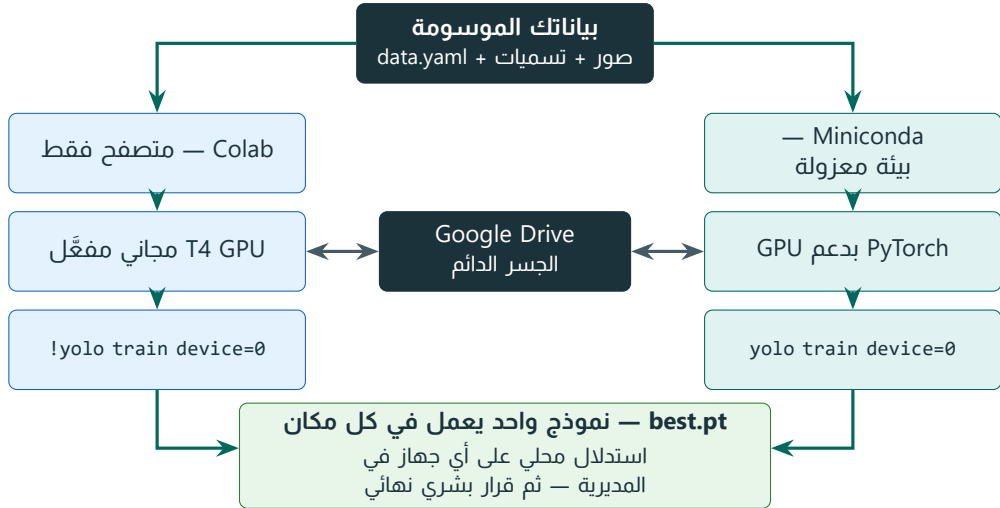
- $conf=0.4$ عتبة الثقة: ارفعها فتقل الإنذارات الكاذبة، وأنزلها فيقل الفائت — قرار موازنة يعرفه أهل المختبرات جيداً
- الحكم الأخير دائماً لعين المختبري على صور جديدة لم يرها النموذج

القسم 8

الخلاصة وخارطة الطريق

كل الأدوات في صورة واحدة — وبطاقة أوامر تكفيك

المنظومة كاملة — من البيانات إلى القرار



بطاقة الأوامر النهائية — علّقها قرب الشاشة

```
conda create -n yolo python=3.11 -y      # once per machine
conda activate yolo                      # every session
pip install torch torchvision --index-url https://download.pytorch.org/whl/cu126
pip install ultralytics                  # once per environment

yolo checks                             # everything OK?
yolo detect train model=yolo11n.pt data=data.yaml epochs=50 device=0
yolo detect val      model=runs/detect/train/weights/best.pt data=data.yaml
yolo detect predict model=best.pt source=image.jpg conf=0.4 save=True
```

- في Colab: الأسطر نفسها مسبوقة بعلامة ! وبمسارات لينكس — ولا حاجة لسطري conda ولا لتثبيت torch
- بلا بطاقة NVIDIA: أسقط سطر torch الخاص وثبّته بلا فهرس، ودّرّب بـ device=cpu

الأخطاء الخمسة التي تنتظركم — وعلاجها الجاهز

ثَبَّتَ الحزم خارج البيئة ⇐ تحقق من وجود (yolo)
في أول سطر الموجّه قبل أي pip install

أغلقت Colab قبل النسخ إلى Drive ⇐ انسخ
best.pt فور انتهاء التدريب — لا في نهاية اليوم

مسار ويندوز في data.yaml داخل كولا ب (أو
العكس) ⇐ عدّل سطر path وحده عند تبديل البيئة

device=0 يفشل رغم وجود البطاقة ⇐ ثَبَّتَ نسخة CPU من
torch؛ أعد التثبيت بفهرس cu126 وتحقق بـ cuda.is_available

نفدت ذاكرة الـ GPU أثناء التدريب ⇐ أنقص batch إلى 8 ثم 4، أو صغّر imgsز إلى 480

**كل خطأ من هذه واجهه فوج قبلكم — من
يحفظ هذه الشريحة يوفر على نفسه أسبوع حيرة**

الأدوات بأيديكم الآن

بيئة معزولة على حاسوبكم، وGPU مجاني في السحابة، وجسر بينهما —
وفي الدورة العشرية سندرب بها نموذج كشف حقيقي على صور الخلايا الدموية.

د. حسام صلاح مهدي • رئيس قسم الذكاء الاصطناعي — جامعة المصطفى