Preprocessing Requirements

☐ استخراج الفريمات أو ضبط الـ FPS

طب الفريمات وعرفنها_____ *??what is the meaning of FPS*

FPS = Frames Per Second

= عدد الفريمات (الصور) اللي بتتسجّل في الثانية الواحدة.

مهم لو هنشتغل على الفيديو نفسه
م مهم لو هنستخدم Mediapipe
محتاجه في الـ preprocessing لما تجهّز فيديوهات للموديل

لو عندك فيديو 1 ثانية:

- **لو FPS = 30**

يبقى الفيديو عبارة عن **30 صورة**.

- **لو FPS = 60**

يبقى الفيديو عبارة عن **60 صورة**.

ليه FPS مهم في الموديلات؟

لأن الموديل بياخد **عدد ثابت من الصور** من كل فيديو.

لو فيديو FPS بتاعه 60
وفيديو تاني FPS بتاعه 25
ده هيخلي:

- فيديو أطول من التاني
  وكده الموديل هيتلغبط لأن الـ input مختلف

علشان نفهم اكتر عملتلك جزء بسيط ممكن نفهم منه اكتر

☐ Resize الفيديوهات (أو الفريمات)

الفكرة

كل صورة (Frame) ليها حجم (W x H).
الموديل بيتدرّب على حجم ثابت، مثلاً 224 x 224.

ليه بنعملها؟

- الموديلات الجاهزة (مثل VideoMAE, ResNet, I3D) مصممة على أحجام معينة.
- لو كل فيديو بحجم مختلف ← الـ tensors مش هتدخل بعض.

عمليًا:

- بعد ما تطلع الفريمات ← تعمل لكل frame:
  - `resize` إلى 224x224 (أو 128x128 لو عايز أخف)
- ممكن كمان تعمل crop لو فى أجزاء زيادة مالهاش لازمة.

النتيجة: **كل فريم = صورة بنفس الحجم ← الموديل مبسوط.**

---

3) Normalization (تطبيع الـ Pixels)

الفكرة

الصورة عبارة عن قيم من 0 إلى 255 (pixel values).
الموديلات بتحب القيم تكون في range صغير زي 0–1 أو -1–1.

ليه بنعملها؟

- تسهّل عملية الـ optimization.
- تخلي gradient ثابت ومبقاش في انفجار في القيم.
- الموديلات الـ pretrained متعوّدة على نوع معينة من الـ normalization (زي mean/std بتاعة ImageNet).

عمليًا:

3 خطوات غالبًا:

1. تحوّل الصورة من `uint8` إلى `float32`

2. تقسم على 255 → القيم تبقى بين 0 و 1

3. تعمل normalization بالـ mean و std:
النتيجة: **الـ input للموديل في range صح → التدريب يبقى مستقر.**

Label Encoding (تحويل الكلمات لأرقام)

الفكرة

الكلاس عندك مكتوب كـ كلمة: `go`, `erase`, `dog` ...
بس الموديل بيشتغل على أرقام (0, 1, 2, ...).

ليه بنعملها؟

- الـ loss functions زي CrossEntropy بتتوقع labels أرقام.
- نحتاج mapping ثابت بين:
  - رقم الكلاس
  - واسم الكلمة

🧱 ترتيب التنفيذ بشكل منطقي

لو هتبني pipeline قوية، تقريبًا هتبقى:

1. تقرأ CSV → تجيب `word` + `full_path`

2. لكل فيديو:
   - تثبت الـ FPS والمدة
   - أو تستخرج عدد ثابت من الفريمات

3. لكل frame:

- `resize`
- `augmentation` (لو في Train)
- `normalization`

4. تحول `word` → label رقم
5. تقسّم Train / Val / Test
6. تبني DataLoader يطلع:
   - `video_tensor` (مثلاً: [num_frames, C, H, W])
   - `label_id`

`طب انا هعمل Preprocessing علي val , Test ?`

بعد تدوير كتير اه هنعملهم why?

ليه لازم نعمل **Preprocessing** على الـ **Test** و الـ **Val**؟

لأن الموديل لازم يستقبل نفس شكل البيانات اللي اتدرّب عليه.

يعني:

- نفس الـ `frame size`
- نفس الـ `normalization`
- نفس عدد الفريمات
- نفس ترتيب المعالجة

لو اديت للـ Test صور غير processed → الموديل هيتلغبط وحيعمل نتيجة وحشة.

طب ف سؤال حلو اوي هنا

طب كده ان لو اليوزر هيتست كده محتاج انه يتعمله بري بروسيسنج
طب والعمل ما كده المديل هيفشل
لا ما هبتم تظبيطه بس مش احنا ال هنظبطه
لازم حد يتواصل مع تيم الباك والله احنا عملنا الفريم كذا وكذا وكذا هيتم التظبيط من عندهم بسكده