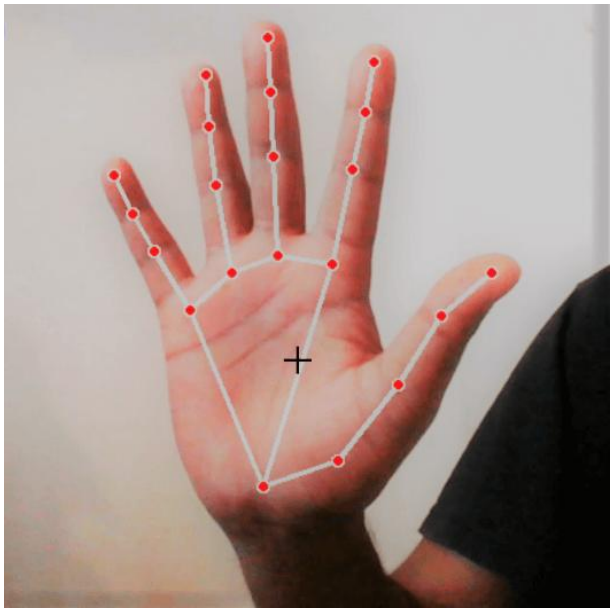
الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
العام الدراسي 2024– 2025

GESTURE RECOGNITION

2024-2025



الفهرس

الملخص

يتناول هذا المشروع تصميم وتطوير نظام متكامل للتعرف على إيماءات اليد في الزمن الحقيقي باستخدام تقنيات الرؤية الحاسوبية والذكاء الاصطناعي، مع تطبيق عملي على بطاقة BeagleBone Black وكاميرا ويب بسيطة. يبدأ النظام بالتقاط صورة لليد المستخدم، وقلبها كما في المرآة ،ثم إجراء سلسلة من خطوات المعالجة التي تشمل كشف لون البشرة في فضاء الألوان HSV، واستخراج محيط اليد باستخدام خوارزميات الكونتور ، وتحديد الهيكل الهندسي لليد عبر بناء المضلع المحدب Convex Hull لليد واكتشاف عيوب التحدب Convexity Defects، والتي تُستخدم لاحقًا لحساب عدد الأصابع المرفوعة أو التعرف على إشارات معينة بلغة ASL وقد تم دعم النظام بمجموعة من التحسينات التقنية لتحقيق أداء فعّال في الزمن الحقيقي، منها تقليل دقة الفيديو لتحسين سرعة المعالجة، وتطبيق تقنيات تنعيم النتائج لتقليل التذبذب في الفراغات. يثبت المشروع القدرة على بناء واجهة تفاعلية بسيطة لكنها فعّالة، يمكن توظيفها في العديد من التطبيقات العملية مثل المساعدات الذكية، وأدوات التعليم التفاعلي، وأنظمة التحكم عن بعد.

المقدمة

في العقود الأخيرة، شهد العالم قفزة نوعية في مجال التفاعل بين الإنسان والآلة، مدفوعة بتطور تقنيات الذكاء الصنعي والرؤية الحاسوبية، الأمر الذي أدى إلى ظهور أنظمة أكثر ذكاءً ومرونة، قادرة على فهم سلوك المستخدم والتجاوب معه بشكل طبيعي. يندرج هذا المشروع ضمن إطار بناء نظام ذكي قادر على التعرف على إيماءات اليد في الزمن الحقيقي، اعتمادًا على خوارزميات متقدمة لمعالجة الصور وتحليل الأشكال باستخدام مكتبة OpenCV ، وتنفيذه على بطاقة BeagleBone Black، وهي بطاقة مدمجة منخفضة الكلفة وعالية الأداء. ويهدف المشروع إلى توفير حل عملي وتفاعلي يمكّن الأجهزة من تفسير حركات اليد المختلفة، مثل عدّ الأصابع والتعرف على رموز لغة الإشارة الأمريكية (ASL)، مما يمهد الطريق أمام تطوير أنظمة تواصل جديدة تتجاوز الأساليب التقليدية المعتمدة على اللمس أو الأوامر الصوتية، ويعزز من فعالية النظام و التفاعل مع الآلة، ويخدم فئات واسعة من المستخدمين، ولا سيما ذوي الاحتياجات الخاصة.

التعريف بالمشروع

يُعنى هذا المشروع بتصميم وتنفيذ نظام ذكي للتعرف على إيماءات اليد باستخدام تقنيات الرؤية الحاسوبية في الزمن الحقيقي. يتم التقاط صورة ليد المستخدم عبر كاميرا ويب موصولة إلى بطاقة BeagleBone Black، ثم تحليل هذه الصورة باستخدام خوارزميات معالجة الصور لتحديد عدد الأصابع المرفوعة أو التعرف على لغة الإشارة ASL، وذلك بالاعتماد على المكتبة OpenCV. ويتضمن العمل خطوات تشمل اكتشاف لون الجلد، وتحليل الإطار الخارجي لليد، وحساب المضلع المحدب وعيوب التحدب لتحديد عدد الأصابع الظاهرة، وتدريب شبكة عصبونية للتعرف على لغة الإشارة. وبالتالي نحصل على واجهة تفاعلية تُتيح للمستخدم التحكم أو التواصل مع الأنظمة الرقمية بطريقة طبيعية وسلسة من خلال الإيماءات اليدوية.

هدف المشروع

يهدف هذا المشروع إلى تطوير نظام رؤية حاسوبية قادر على التعرف على إيماءات اليد في الزمن الحقيقي، عن طريق توظيف تقنيات معالجة الصور والرؤية الحاسوبية والمعالجات المضمنة في تصميم واجهات تفاعلية، بحيث يكون قادراً على تفسير بعض الإيماءات اليدوية، مثل عدّ الأصابع المرفوعة، والتعرف على حركات لغة الإشارة الأمريكيةASL.

مكونات النظام

الأجهزة (Hardware)

1. A single-board computer BeagleBone Black لمعالجة الصور وتنفيذ الخوارزميات.

2. Web camera التقاط الصور بالزمن الحقيقي وإرسالها لBBB.

3. حاسب للتحكم والمراقبة.

البرمجيات (Software)

4. على BBB
   - كود بلغة بايثون لتحصيل الصور ومعالجتها.
   - مكتبة OpenCV لمعالجة الصور واكتشاف اليد.
   - مكتبة NumPy للعمليات العددية والهندسية.
5. على الحاسب
   - برنامج MobaXterm لتأمين اتصال من النمط SSH مع BBB.
   - موقع Colab لتدريب الشبكة العصبونية.
   - Dataset التي سنحتاجها لتدريب النموذج.

القسم العملي

التوصيل وتهيئة البطاقة

- تم بداية توصيل البطاقة إلى الحاسب بحيث نستطيع العمل على النظام الموجود في كرت الذاكرة.
- تم تثبيت المكتبات اللازمة باستخدام الانترنت لاستخدامها في البرنامج.

```
import numpy as np
import cv2
import time
import math
from collections import deque
```

المكتبة	الاستخدام
numpy	حساب المسافات والزوايا
OpenCV مكتبةcv2	معالجة الصور والفيديو
Time	تأخير تشغيل الكاميرا قليلاً عند التشغيل
Math	حساب الزوايا بين النقاط
Deque	تخزين تاريخ عدد الأصابع وتنعيم النتائج

- تم وصل كاميرا الويب مع BBB والتأكد من سلامة الاتصال.

التقاط الصور باستخدام الكاميرا وعرضها على الشاشة

سنقوم بدايةً بالتقاط الصور يدوياً ومعالجتها للتمكن من تقييم أداء التوابع المستخدمة بشكل أفضل، وسنبدأ بمعالجة الصور بالوقت الحقيقي لاحقاً عند اعتماد التوابع والبارامترات التي تعطي أفضل أداء. قمنا بعرض فيديو يحوي ما تراه الكاميرا بشكل مستمر على الشاشة، إضافة إلى الصورة التي يتم التقاطها من قبل المستخدم.

لاحظنا في البداية أن الفيديو الذي يتم عرضه فيه تأخير كبير عن الحركة في الواقع، لذلك قمنا بتحديد أبعاد الصورة 240x320، وهذه الأبعاد جيدة جداً في تطبيقنا (الكشف عن الإماءات)، كما قمنا بتقليل عدد الصور المعالجة والمعروضة بالفيديو إلى 1 من كل 5 صور، وبذلك حصلنا على عرض سلس للفيديو بتأخير أقل بكثير وحركة مشابهة في السرعة للواقع الذي تلتقطه الكاميرا.

معالجة الصور من أجل كشف اليد

عرفنا التابع detect_skin الذي يقوم بمعالجة الصورة ويعيد لنا القناعmask .

- أولا قلبنا الصورة (flipping) لجعل التفاعل مع التطبيق أكثر واقعية كما في المرآة.
- حولنا الصورة الملتقطة من الكاميرا بصيغة RGB إلى HSV، وذلك لأن الصيغة RGB حساسة لتغيرات الإضاءة وبالتالي لا يمكننا تحديد لون البشرة في جميع ظروف الإضاءة، لذا يجب تحويلها للصيغة HSV لنقلها لفضاء لوني مختلف يفصل الإضاءة عن باقي خواص الصورة، وبالتالي يمكننا تحديد حدود لونية للجلد بغض النظر عن الإضاءة

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

- عرفنا بعد ذلك مصفوفة أو صورة ثنائية (أبيض وأسود) تحدد الأماكن ذات اللون البشري من الصورة عن طريق مقارنة كل بيكسل مع قيم حدية للون البشرة، وبذلك نفصل اليد عن باقي الصورة لنتمكن من إكمال المعالجة

```
lower_skin = np.array([0, 20, 70], dtype=np.uint8)
upper_skin = np.array([20, 255, 255], dtype=np.uint8)
mask = cv2.inRange(hsv, lower_skin, upper_skin)
```

وبذلك حصلنا على قناع يسمح بتحديد مكان اليد، ولكن بالتأكيد يوجد العديد من مصادر الضجيج للصورة السابقة فيجب تقليلها قدر المستطاع قبل إيجاد محيط اليد.

- التخلص من الضجيج

عرفنا التابع advanced_noise_processing الذي يعدل على القناع ويعيده بعد الترشيح.

نريد أن نقلل الضجيج ذو الترددات العالية من الصورة الثنائية السابقة لأنها تعطينا معلومات خاطئة، جربنا في البداية استخدام المرشح الوسيط

```
blurred= cv2.medianBlur(mask, 9)
```

إلا أن النتائج لم تكن فعالة في الحفاظ على تفاصيل اليد، حيث أدى إلى طمس بعض الحواف الدقيقة المهمة لتحديد عدد الأصابع. لذلك قمنا باستبداله بالمرشح الغاوسي لتحسين نعومة الصورة دون فقدان التفاصيل الدقيقة. يقوم هذا التابع بتقليل التغيرات اللونية في البكسلات عن طريق تطبيق نواة Gaussian kernel على الصورة أي إيجاد جداء التلاف معها للحصول على النتيجة المطلوبة

```
blurred = cv2.GaussianBlur(mask, (5, 5), 0)
```

بعد ذلك، تم استخدام عمليات مورفولوجية مثلMorphological Closing لملء الفجوات الصغيرة داخل العناصر، وMorphological Opening لإزالة الضجيج الخارجي الصغير.

تابعنا المعالجة عبر تطبيق adaptiveThreshold لتحسين تمييز الحواف، تلتها عملية استخراج الحواف باستخدام التدرج المورفولوجي morphological gradient وأخيراً، تم استخدام تحليل المساحات عبر الدالة findContours مع ترشيح بناءً على المساحة، حيث تم الاحتفاظ فقط بالمناطق ذات المساحة الأكبر من 500 بكسل، لضمان عزل اليد عن الضجيج في الخلفية بشكل دقيق. وأدى هذا التسلسل المتكامل من المرشحات والخطوات المورفولوجية إلى تحسين جودة القناع الناتج وزيادة دقة اكتشاف اليد وعدّ الأصابع المرفوعة.

استخراج الإطار المحيط باليد

عرفنا التابع count_fingers الذي يقوم بعد الأصابع

- نوجد حدود اليد contour عن طريق تحديد النقاط المتجاورة المستمرة التي تفصل بين الحد اللوني والكثافة لليد والخلفية. وذلك بعد إيجاد الصورة الثنائية لليد التي تظهر اليد باللون الأبيض والباقي أسود عن طريق استبعاد الألوان والكثافات التي لا تمثل لون البشرة من الصورة الأصلية.

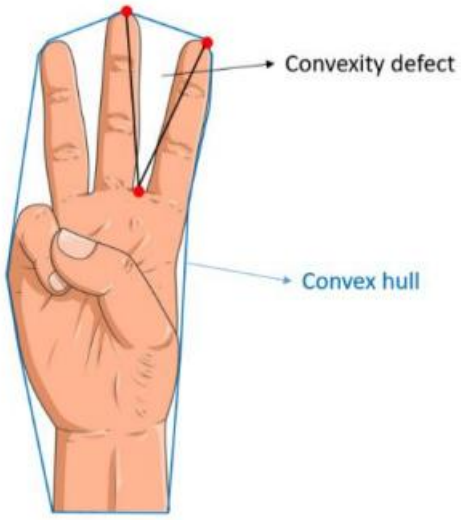
```
contours,_=cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

حيث يعيد هذا التابع الحدود الخارجية للمساحات البيضاء.

- والآن لعزل حدود اليد نقوم باختيار المحيط الذي يضم أكبر مساحة باعتباره حد اليد contour وذلك لحذف الضجيج الذي قد يؤثر لاحقا بحساب عدد الأصابع، باعتبار اليد قريبة ما يكفي من الكاميرا وهي العنصر الأساسي في الصورة.

```
if contours:
   max_contour = max (contours, key=cv2.contourArea)
```

تحليل شكل اليد لتحديد عدد الأصابع



بعد الحصول على نقاط حدود الصورة contour، سنستخدمها الآن لإيجاد عدد الأصابع عن طريق استخدام كل من Convex Hull و Convexity Defects.

إن Convex Hull هي خوارزمية هندسية دخلها مجموعة نقاط وتعطي النقاط التي تشكل المضلع المحدب (الزاوية بين أي ضلعين متجاورين أصغر من 180 درجة) ذي أقل عدد من الأضلاع ويحوي جميع باقي النقاط. سنقوم هنا بالاعتماد عليها لإيجاد المضلع الأصغري الذي يحوي جميع نقاط اليد أو contour. ومن ثم الاعتماد عليه لإيجاد عدد الأصابع في الصورة.

```
hull = cv2.convexHull(contour)
```

هذه العملية تعطينا تمثيل مبسط لشكل اليد بعد حذف التعرجات، وخاصة الفراغات بين الأصابع. وهذا التبسيط يساعد في إيجاد بعض الخواص المفيدة مثل فراغات الأصابع والاتجاهات.

والآن للاستفادة من الشكل الذي حصلنا عليه نستعمل Convexity Defects، وهي تمثل المناطق التي يبتعد فيها محيط اليد contour عن الشكل المحدب hull، حيث تظهر هذه المناطق في الفراغات بين الأصابع وتكون الأبعد عن المضلع المحيط بها في هذه الحالة.

```
hull_indices = cv2.convexHull(contour, returnPoints=False)
defects = cv2.convexityDefects(contour, hull_indices)
```

حيث إن كل defect ينتج من التوابع السابقة عبارة عن شعاع طوله 4 يحوي

- نقطة من المضلع المحدب hull.
- النقطة التي تجاورها من المضلع المحدب hull.
- النقطة الأبعد من نقاط contour عن الضلع بين النقطتين السابقتين وتقع بينهما.
- العمق وهو البعد بين الضلع والنقطة السابقة.

والآن يمكننا الاستفادة من المعلومات السابقة في إيجاد عدد الأصابع عن طريق إيجاد الزاوية بين النقاط السابقة. لكل عنصر من defects إذا كانت الزاوية بين النقاط الثلاثة أصغر من حد معين (90 درجة) والعمق أكبر من حد معين فيمكننا اعتباره فراغ بين أصبعين. لحساب الزاوية

```
import math
angle = math.acos((a**2 + b**2 - c**2) / (2*a*b))
angle = math.degrees(angle)
```

حيث كل من a, b, c المسافات بين النقاط الثلاثة السابقة.

وبعد المرور على جميع عناصر defects والتأكد من تحقق شرطي الزاوية والعمق لكل منها نكون حصلنا على عدد الفراغات بين الأصابع في الصورة وبالتالي فإن عدد الأصابع هو عدد الفراغات+1.

التقاط الصورة بالزمن الحقيقي

الآن بعد تأكدنا من أداء التوابع والخوارزميات المستخدمة والتعديل على البارامترات بما يناسبنا تم تعديل الكود الأصلي ليعمل بالزمن الحقيقي (Real-Time) وقد شملت التعديلات الجوانب التالية:

1. التحويل من الالتقاط الثابت إلى الفيديو اللحظي

في النسخة الأولى، كان المستخدم بحاجة إلى الضغط على مفتاح (Space) لالتقاط صورة واحدة وتحليلها أما في النسخة المعدلة تم تحويل التطبيق ليقوم بتحليل مستمر لفيديو مباشر، وإلغاء الحاجة إلى تدخل المستخدم لالتقاط الصورة، حيث أصبحت المعالجة تلقائية ضمن الحلقة الزمنية، ما يجعل التطبيق أكثر تفاعلية وملاءمة للاستخدام العملي.

2. تقليل عدد الإطارات المعالجة لتخفيف الضغط على المعالج

تم تغيير قيمة المتحول display_every_n لمعالجة 1 من كل 10 إطارات فقط مما يُحسن الأداء على المتحكم.

3. تحسين استقرار قراءة عدد الأصابع

تمت إضافة آلية تخزين القراءات الأخيرة لعدد الأصابع باستخدام بنية deque (ذاكرة دائرية) حيث يتم حساب متوسط آخر 5 قراءات واعتمادها لتخفيف التذبذب الناتج عن الضوضاء أو تغير الإضاءة، مما ينتج عنه قراءة أكثر دقة وثباتاً.

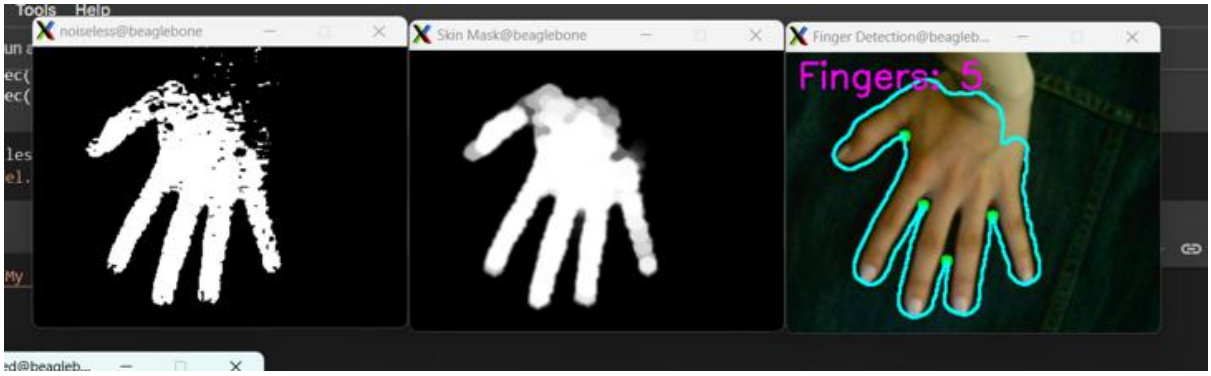
4. ضبط دقة الكاميرا إلى قيم منخفضة (320×240 then 160×120)

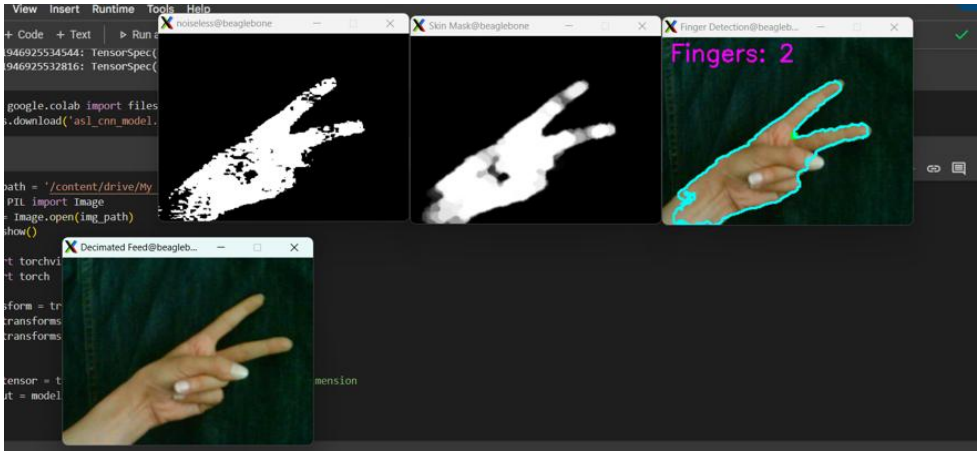
تم تقليل دقة الإطارات لتسريع عملية المعالجة وتقليل الحمل الحوسبي.

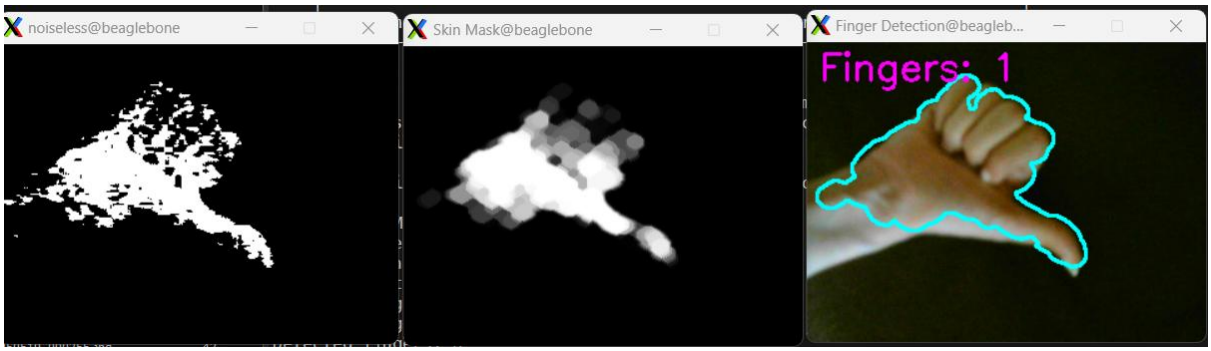
5. تحسين عرض الصور

أصبح يتم عرض صورة اليد، القناع، وعدد الأصابع بشكل مباشر داخل النوافذ الرسومية (cv2.imshow) بدون تدخل المستخدم.

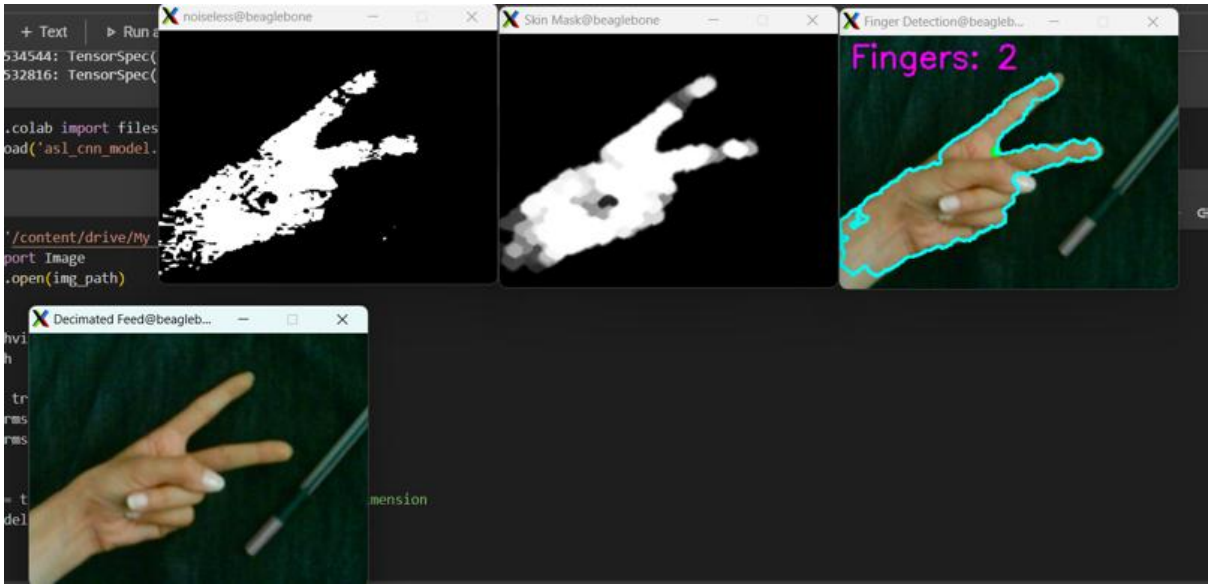
6. التجريب

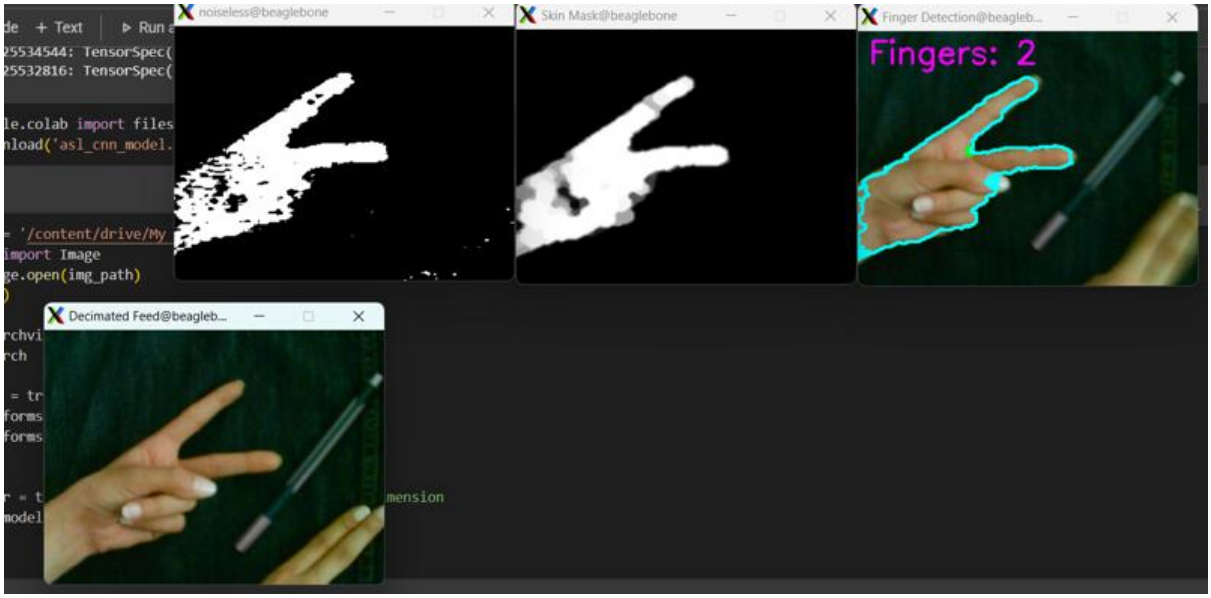






عند إضافة عنصر خارجي بجانب اليد:





6. التحسينات على المشروع

- تدريب نموذج للتعرف على لغة الإشارة

ضمن التوسعات التطويرية للمشروع، تم تدريب نموذج تصنيف يعتمد على الشبكات العصبية الالتفافية (CNN) للتعرف على الحروف الأبجدية في لغة الإشارة الأمريكية (ASL)، وذلك باستخدام مجموعة البيانات المتاحة على منصة Kaggle بعنوان:

[ASL Alphabet Test Dataset]:

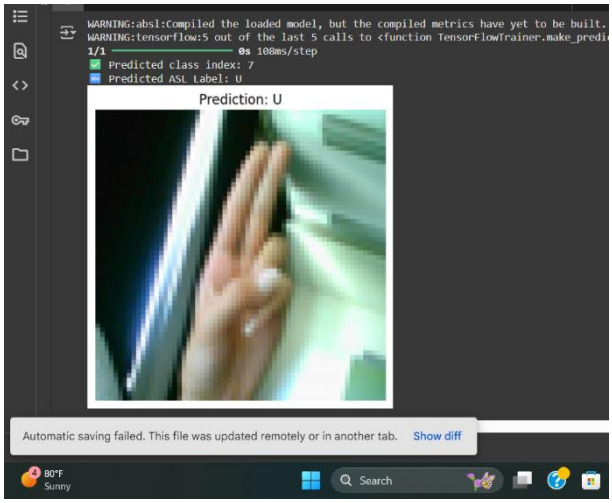
(https://www.kaggle.com/datasets/danrasband/asl-alphabet-test).

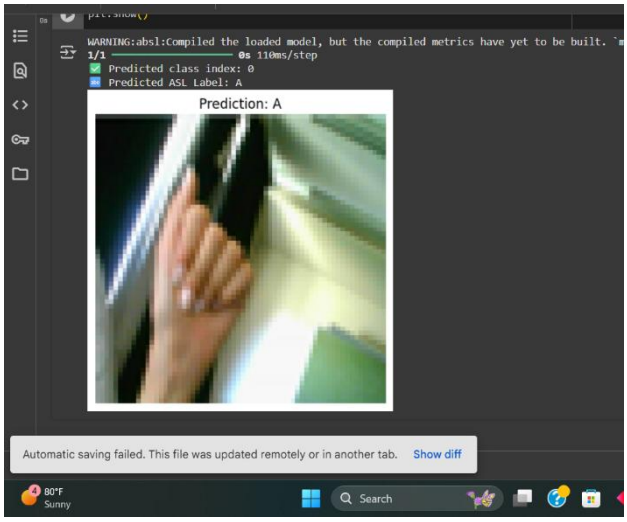
تم تنفيذ التدريب باستخدام مكتبة TensorFlow ضمن بيئة Google Colab، حيث تم تحميل جميع صور البيانات ومعالجتها باستخدام `ImageDataGenerator، مع إعادة تحجيم الصور إلى أبعاد (64×64) بكسل وتطبيع القيم اللونية. وقد تم استخدام كامل البيانات المتاحة لتدريب النموذج، دون فصل مجموعة تحقق.

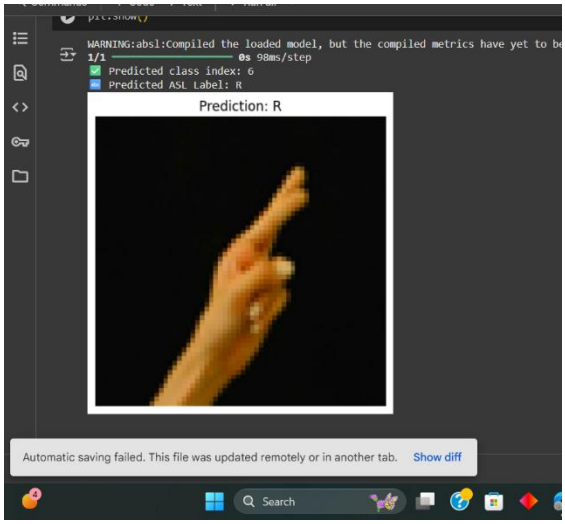
تم بناء النموذج باستخدام تسلسل من الطبقات، يتضمن طبقتين من الالتفاف (Convolutional) يلي كلٌّ منهما طبقة تجميع (MaxPooling)، ثم طبقة تسطيح (Flatten)، تليها طبقة كثيفة (Dense) مكونة من 128 خلية عصبية مع دالة تفعيل ReLU، وطبقة إسقاط Dropout لتقليل فرط التعلّم. وانتهى النموذج

بطبقة إخراج Softmax بعدد فئات يعادل عدد الحروف. تم تدريب النموذج باستخدام خوارزمية التحسين Adam ودالة الخسارة `categorical_crossentropy`، وتم حفظ النموذج بعد التدريب بصيغة H5.

لأغراض النشر والاستخدام على الأنظمة محدودة الموارد، تم تحويل النموذج إلى صيغة TensorFlow Lite باستخدام TFLiteConverter، وتم تصدير النموذج بصيغة tflite بنجاح، مما يتيح إمكانية استخدامه مستقبلاً على الأجهزة المضمنة مثل BeagleBone Black أو غيرها سواء بشكل مباشر أو ضمن بيئة معالجة خارجية مرتبطة بالنظام عبر الشبكة.







الخاتمة والنتائج

يمثل المشروع مثالاً عملياً على كيفية توظيف الإمكانات المتاحة لتطوير أنظمة ذكية تتفاعل مع الإنسان بشكل أكثر طبيعية ومرونة. ومن خلال استخدام أدوات مفتوحة المصدر ومنصة منخفضة التكلفة، أظهر المشروع قابلية التطبيق في بيئات حقيقية، مع تحقيق نتائج واعدة من حيث الدقة والسرعة وسهولة الاستخدام. وقد تم تجاوز العديد من التحديات التقنية المرتبطة بالضجيج البصري وتغير الإضاءة بفضل تطبيق خوارزميات متقدمة للتحليل ومعالجة الصورة، مما مكّن من بناء نظام مستقر وفعّال.

رغم أن النظام في شكله الحالي يقتصر على عد الأصابع والتعرف على إشارات محددة من لغة ASL ، إلا أنه يؤسس لقاعدة يمكن البناء عليها لتطوير أنظمة أكثر ذكاءً، مدعومة بتقنيات التعلم العميق والتعلم الآلي، ما يفتح المجال أمام استخدامات أوسع مثل الترجمة الآلية للغات الإشارة، أو التحكم بالأجهزة المنزلية الذكية. وبهذا، فإن المشروع لا يكتفي بتقديم حل تقني، بل يسهم في تمهيد الطريق نحو مستقبل رقمي أكثر إنسانية وشمولية.

المراجع

[1] A. Khanal, "Hand pose estimation and gesture detection from webcam images," *arXiv preprint arXiv:2103.01743*, 2021.

[2] R. P. L. B. Dey and S. K. Ray, "Real-time finger counting based on convex hull defects of hand contour," *2016 International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, India, 2016, pp. 0634–0638, doi: 10.1109/ICCSP.2016.7754182.

[3] D. S. Alex and R. S. Raj, "A comparative study between HSV and YCbCr color models in skin color detection," *International Journal of Computational Engineering Research (IJCER)*, vol. 2, no. 5, pp. 125–128, 2012.