الذكاء الاصطناعي للموبايل

تطبيقات موبايل تم انشاؤها وبرمجتها باستخدام فلاتر وتنسرفلو لايت

ترجمة واعداد: د. علاء طعيمة

بـــسم تعالى

الذكاء الاصطناعي للموبايل

تطبيقات موبايل تم انشاؤها وبرمجتها باستخدام فلاتر وتنسرفلو لايت

ترجمة واعداد:

د. علاء طعيمة

مقدمة المترجم

أصبح للتعلم الآلي والذكاء الاصطناعي تأثير متزايد على تكنولوجيا الهاتف المحمولفي الوقت الحاضر. أصبحت الأجهزة أكثر قدرة على استخدام الذكاء الاصطناعي، ويتم دمج التعلم الآلي وطرق الذكاء الاصطناعيفي تطبيقات الهاتف المحمول لتحسين تجارب المستخدم وتعزيزها.

لقد كان تطوير تطبيقات الهاتف المحمول التي تتضمن التعلم الآلي مهمة صعبة منذ فترة طويلة. ولكن بمساعدة المنصات وأدوات التطوير مثل TensorFlow Lite، أصبح القيام بذلك أسهل. تزودنا TensorFlow Lite بنماذج التعلم الآلي المدربة مسبقًا بالإضافة إلى أدوات لتدريب واستيراد نماذجنا المخصصة. ولكن كيف يمكننا بالفعل تطوير تجربة مقنعة فوق نماذج التعلم الآلي تلك؟ وهنا يأتي دور Flutter.

Flutter عبارة عن مجموعة أدوات محمولة لواجهة المستخدم تم إنشاؤها بواسطة Google ومجتمعها مفتوح المصدر لتطوير تطبيقات Android وIOS والويب وسطح المكتب. يجمع Flutterفي جوهره بين محرك رسومي عالي الأداء ولغة برمجة Dart.

في هذا الكتاب، سنجمع بين قوة Flutter وTensorFlow Lite على الجهاز لتطوير تطبيقات Flutter يمكنها القيام العديد من المهام مثل التعرف على الكلام والصور والإيماءات وغيرها.

لقد حاولت قدر المستطاع ان اترجم المقالات والمشاريع الأكثر طرحاًفي تطوير تطبيقات الموبايل باستخدام الذكاء الاصطناعي مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجال تطوير تطبيقات الموبايل باستخدام الذكاء الاصطناعي ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية / العراق

المحتويات

0) التعلم الآلي في فلاتر باستخدام تنسرفلو لايت Machine Learning in Flutter using TensorFlow Lite 11

تعاريف 11

الذكاء الاصطناعي (AI) 11

التعلم الآلي (ML) 12

التعلم العميق (DL) 12

أنواع تعلم الآلة 12

التعلم الخاضع الإشراف 13

التعلم غير الخاضع الإشراف 13

التعلم شبه الخاضع للإشراف 13

التعلم المعزز 13

تنسرفلو 14

تنسرفلو لايت 14

ميزات تنسرفلو لايت 14

فلاتر 14

بناء تطبيق ML-Flutter 15

TensorFlow Lite – الإعداد الأصلي (Android) 15

TensorFlow Lite – إعداد Flutter (Dart) 15

إنشاء نموذج (باستخدام موقع الويب) 16

إضافة ملفات/نماذج إلى مشروع Flutter 21

تحميل وتشغيل ML-Model 21

الاستنتاج 22

1) انشاء تطبيق موبايل للتعرف على النباتات باستخدام فلاتر وتنسرفلو لايت Build a Plant Recognition Mobile App using Flutter with TensorFlow Lite 24

البدء 25

مقدمة موجزة للتعلم الآلي **26**
ما هو التعلم الآلي 27
بناء نموذج باستخدام **Teachable Machine** **28**
إعداد مجموعة البيانات 28
تدريب النموذج 28
تدريب النموذج: كيف يعمل **33**
فهم تنبؤ **Tensor** و **TensorFlow** 34
تثبيت **TensorFlow Lite** في **Flutter** **35**
إنشاء مصنف الصور **37**
استيراد النموذج إلى **Flutter** 37
تحميل تسميات التصنيف 37
استيراد نموذج **TensorFlow Lite** 38
تنفيذ التنبؤ **TensorFlow** **40**
معالجة بيانات الصورة مسبقًا 40
تنفيذ التنبؤ 42
مرحلة المعالجة اللاحقة لنتيجة الإخراج 42
استخدام المصنف **44**
اختيار صورة من الجهاز 44
تهيئة المصنف 45
تحليل الصور باستخدام المصنف 46
2) أنشاء تطبيق موبايل لتصنيف الصور باستخدام فلاتر وتنسرفلو لايت **Build an Image Classification Mobile App using Flutter with TensorFlow Lite** **48**
حالات الاستخدام **48**
الحزم المطلوبة 48
تدريب نموذج باستخدام **Teachable Machine** 48
تطبيق **Flutter** 50
النتائج 53

الكود الكامل 53
الكود المصدري 55
الاستنتاج 55
3) أنشاء تطبيق موبايل لتصنيف الكلاب والقطط باستخدام فلاتر وتنسرفلو لايت Build a Cat-or-Dog Classification Mobile App using Flutter with TensorFlow Lite 56
إنشاء مشروع Flutter جديد 57
إنشاء عرض الصورة على الشاشة 58
دالة جلب وعرض الصورة 60
أداء تصنيف الصور باستخدام TensorFlow Lite 61
تثبيت TensorFlow Lite 62
استخدام TensorFlow Lite لتصنيف الصور 63
تحميل النموذج 63
تنفيذ تصنيف الصور 63
الاستنتاج 66
4) أنشاء تطبيق موبايل لتصنيف سلالات الكلاب باستخدام باستخدام فلاتر وتنسرفلو لايت Build a Dog Breed Classification Mobile App using Flutter with TensorFlow Lite 68
إنشاء مشروع Flutter جديد 69
إنشاء عرض الصورة على الشاشة 70
دالة جلب وعرض الصورة 71
إجراء تصنيف سلالات الكلاب باستخدام TensorFlow Lite 73
تثبيت TensorFlow Lite 74
تنفيذ نموذج التصنيف 74
تحميل النموذج 74
تنفيذ تصنيف الصور 75
الاستنتاج 78

5) أنشاء تطبيق موبايل لاكتشاف قناع الوجه باستخدام فلاتر وتنسرفلو لايت Build a face Mask Detection Mobile App using Flutter with TensorFlow Lite 79
تثبيت الحزم 79
تكوين المشروع 79
تنزيل مجموعة بيانات التدريب وتدريب النموذج 80
تهيئة الكاميرا 82
تحميل النموذج 82
تشغيل النموذج 82
معاينة الكاميرا 83
الكود الكامل 83
6) انشاء تطبيق لمصادقة التعرف على الوجوه باستخدام باستخدام فلاتر وتنسرفلو لايت Build an Face Recognition Authentication Mobile App using Flutter with TensorFlow Lite 86
ملخص العملية 86
التسجيل 86
تسجيل الدخول 86
مسح قاعدة البيانات 87
كيف تعمل 87
Tensorflow Lite 87
نماذج MobileFaceNet 87
Firebase ML vision 88
تثبيت 88
تنفيذ Flutter 88
الوصف 88
تعريف صديق للمبرمج 89
مثال 90
الاستنتاج 90

7) انشاء تطبيق موبايل للكشف المباشر عن الكائنات باستخدام فلاتر وتنسرفلو لايت Build a Live Object Detection Mobile App using Flutter with TensorFlow Lite 91
تثبيت الحزم 91
تكوين أندرويد 91
تهيئة الكاميرا 92
تحميل النموذج 92
تشغيل النموذج 92
عرض المربعات حول الكائنات التي تم التعرف عليها 93
معاينة الكاميرا 94
الكود الكامل 94
8) انشاء تطبيق لتقدير الوضعية في الوقت الفعلي باستخدام فلاتر وتنسرفلو لايت Build a Real-Time Pose Estimation Mobile App using Flutter with TensorFlow Lite 98
الحزم المطلوبة 98
TensorFlow Lite 98
Camera Plugin 98
PoseNet MobileNet V1 98
تطبيق Flutter 100
النتائج 105
9) انشاء تطبيق للتعرف على الأرقام باستخدام فلاتر وتنسرفلو لايت Build a Digit Recognition Mobile App using Flutter with TensorFlow Lite 106
الحزم المطلوبة 106
الاعداد 107
الاستنتاج 114
10) انشاء تطبيق للتعرف على الاصوات باستخدام فلاتر وتنسرفلو لايت Build an Audio recognition Mobile App using Flutter with TensorFlow Lite 115
إنشاء نموذج التصنيف 115
إنشاء تطبيق Flutter 117

دمج النموذج في التطبيق 118
تكوينات الاندرويد 119
تكوينات IOS 119
تحميل واستخدام النموذج 121
الكود 123
الاستنتاجات 123
11) انشاء تطبيق لتصنيف الرموز التعبيرية باستخدام فلاتر وتنسرفلو لايت Build an Emoji Classification Mobile App using Flutter with TensorFlow Lite 125
إنشاء مشروع Flutter جديد 125
إنشاء الشاشة الرئيسية 126
إنشاء واجهة المستخدم لعرض الصور 128
جلب وعرض صورة الرموز التعبيرية 130
إجراء تحديد الرموز التعبيرية باستخدام TensorFlow 131
تثبيت TensorFlow lite 132
تنفيذ TensorFlow Lite لتصنيف الرموز التعبيرية 133
تحميل النموذج 133
تنفيذ تصنيف الرموز التعبيرية 133
الاستنتاجات 137
12) انشاء تطبيق للكشف عن الالتهاب الرئوي باستخدام فلاتر وتنسرفلو لايت Build a Pneumonia Detection Mobile App using Flutter with TensorFlow Lite 138
الحصول على النموذج 138
بناء التطبيق 138
التكوينات والتبعيات 139
كتابة الكود 140
13) انشاء تطبيق للتعرف على ايماءات اليد باستخدام فلاتر وتنسرفلو لايت Build a Hand Gesture Detection Mobile App using Flutter with TensorFlow Lite 146
إعداد النموذج 146
جزء Flutter 147

الاستنتاج 156
14) انشاء تطبيق للكشف عن النص باستخدام فلاتر وتنسرفلو لايت Build a Text Mobile App using Flutter with TensorFlow Lite Classification 158
الإعداد الأولي 158
الحصول على البرنامج المساعد 158
تحميل النموذج 159
برمجة المصنف 159
المعالجة المسبقة 159
الاستدلال باستخدام tflite_flutter 161
إنشاء المفسر، تحميل النموذج 161
تنفيذ الاستدلال 162

0) التعلم الآلي في فلاتر باستخدام تنسرفلو لايت Machine Learning in Flutter using TensorFlow Lite

أصبح التعلم الآلي Machine learning(ML) جزءًا من الحياة اليومية. تستخدم المهام الصغيرة التعلم الآلي مثل البحث عن الأغاني على YouTube والاقتراحات على Amazon في الخلفية. هذا مجال تكنولوجي متطور مع إمكانيات هائلة. ولكن كيف يمكننا استخدامه؟

يهدف هذا البرنامج التعليمي إلى شرح مدى سهولة استخدام نماذج التعلم الآلي (التي ستكون بمثابة العقل) لإنشاء تطبيقات Flutter قوية تعتمد على التعلم الآلي. وسوف نتطرق بإيجاز إلى النقاط التالية:

تعاريف

دعنا ننتقل إلى الجزء الذي يشعر فيه معظم الناس بالارتباك. قد يعتقد الشخص الذي لا يتعامل مع صناعة تكنولوجيا المعلومات أن الذكاء الاصطناعي والتعلم الآلي والتعلم والعميق كلها متشابهة. لذلك، دعونا نفهم الفرق.

الذكاء الاصطناعي (AI)

الذكاء الاصطناعي Artificial Intelligence(AI)، هو مفهوم للآلات القادرة على تنفيذ المهام بطريقة أكثر ذكاءً. لا بد أنكم جميعًا استخدمتم موقع YouTube.في شريط البحث، يمكنك كتابة

كلمات أي أغنية، حتى الكلمات التي لا تمثل بالضرورة الجزء الأول من الأغنية أو عنوان الأغنية، وتحصل على نتائج مثالية تقريبًافي كل مرة. هذا هو عمل الذكاء الاصطناعي القوي جدًا.

الذكاء الاصطناعي هو قدرة الآلة على القيام بالمهام التي عادة ما يقوم بها البشر. وهذه القدرة مميزة لأن المهمة التي نتحدث عنها تتطلب ذكاء الإنسان وفطنته.

التعلم الآلي (ML)

التعلم الآلي Machine Learning (ML) هو مجموعة فرعية من الذكاء الاصطناعي. وهو يعتمد على فكرة أننا نعرض الأجهزة لبيانات جديدة، والتي يمكن أن تكون صفًا كاملاً أو جزئيًا، ونترك للآلة أن تقرر المخرجات المستقبلية. يمكننا أيضًا أن نقول إنه مجال فرعي من الذكاء الاصطناعي يتعامل مع استخراج الأنماط من مجموعات البيانات. وهذا يعني أن الآلة يمكنها إيجاد قواعد للسلوك البصري للحصول على مخرجات جديدة. ويمكنه أيضًا التكيف مع البيانات المتغيرة الجديدة تمامًا مثل البشر.

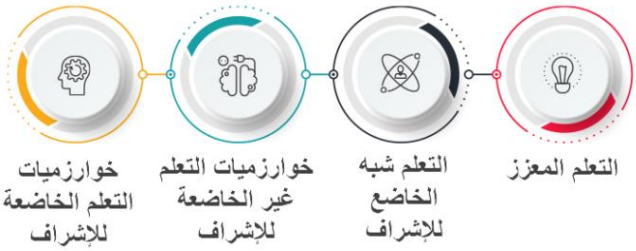
التعلم العميق (DL)

يعد التعلم العميق Deep Learning (DL) مرة أخرى مجموعة فرعية أصغر من التعلم الآلي، وهو في الأساس شبكة عصبية neural network ذات طبقات متعددة. تحاول هذه الشبكات العصبية محاكاة سلوك الدماغ البشري، لذلك يمكنك القول إننا نحاول إنشاء دماغ بشري اصطناعي. باستخدام طبقة واحدة من الشبكة العصبية، لا يزال بإمكاننا إجراء تنبؤات تقريبية، ويمكن أن تساعد الطبقات الإضافيةفي تحسين الدقة وتحسينها.

أنواع تعلم الآلة

قبل البدءفي التنفيذ، نحتاج إلى معرفة أنواع التعلم الآلي لأنه من المهم جدًا معرفة النوع الأكثر ملاءمة لوظائفنا المتوقعة.

أنواع التعلم الآلي



التعلم الخاضع الإشراف

كما يوحي الاسم،في التعلم الخاضع للإشراف supervised learning، يحدث التعلم الخاضع الإشراف. الإشراف يعني أن البيانات المقدمة إلى الجهاز هي بيانات مصنفة بالفعل، أي أن كل جزء من البيانات له تسميات ثابتة fixed labels، ويتم تعيين المدخلات بالفعل إلى المخرجات.

بمجرد تعلم الآلة، تصبح جاهزة لتصنيف البيانات الجديدة.

يعد هذا التعلم مفيدًا لمهام مثل اكتشاف الاحتيال fraud detection وتصفية البريد العشوائي spam filtering وما إلى ذلك.

التعلم غير الخاضع الإشراف

في التعلم غير الخاضع للاشراف unsupervised learning، تكون البيانات المقدمة للآلات للتعلم أولية تمامًا، بدون علامات tags أو تسميات labels. هنا، الآلة هي التي ستقوم بإنشاء فئات جديدة عن طريق استخراج الأنماط.

يمكن استخدام هذا التعلم للتجميع clustering والارتباط association وما إلى ذلك.

التعلم شبه الخاضع للإشراف

كل من الخاضع للإشراف وغير الخاضع للإشراف له حدوده الخاصة، لأن أحدهما يتطلب بيانات مصنفة، والآخر لا يتطلب ذلك، لذلك يجمع هذا التعلم بين سلوك كلا التعلمين، وبهذا يمكننا التغلب على القيود.

في التعلم شبه الخاضع للإشراف Semi-Supervised Learning، نقوم بتغذية الجهاز ببيانات الصف والبيانات المصنفة حتى يتمكن من تصنيف بيانات الصف، وإذا لزم الأمر، إنشاء مجموعات جديدة.

التعلم المعزز

بالنسبة للتعلم المعزز Reinforcement Learning، نقوم بتغذية تعليقات المخرجات الأخيرة بالبيانات الجديدة الواردة إلى الأجهزة حتى تتمكن من التعلم من أخطائها. ستستمر هذه العملية المبنية على التغذية الراجعة feedback-based process حتى تصل الآلة إلى المخرجات المثالية. يتم تقديم هذه التغذية الراجعة من قبل البشري شكل عقاب أو مكافأة. يشبه هذا عندما تمنحك خوارزمية البحث قائمة بالنتائج، لكن المستخدمين لا ينقرون على غير النتيجة الأولى. إنه كالطفل البشري الذي يتعلم من كل خيار متاح، وبتصحيح أخطائه ينمو.

تنسرفلو

يعد التعلم الآلي عملية معقدة حيث نحتاج إلى تنفيذ أنشطة متعددة مثل معالجة البيانات المجمعة ونماذج التدريب وخدمة التنبؤات وتحسين النتائج المستقبلية.

ولتنفيذ مثل هذه العمليات، قامت كوكل بتطوير إطار عملفي نوفمبر 2015 يسمى TensorFlow. يمكن أن تصبح جميع العمليات المذكورة أعلاه سهلة إذا استخدمنا إطار عمل TensorFlow.

في هذا المشروع، لن نستخدم إطار عمل TensorFlow كاملًا ولكن أداة صغيرة تسمى TensorFlow Lite

تنسرفلو لايت

يتيح لنا تنسرفلو لايت TensorFlow Lite تشغيل نماذج التعلم الآلي على الأجهزة ذات الموارد المحدودة، مثل ذاكرة الوصول العشوائي أو الذاكرة المحدودة.

ميزات تنسرفلو لايت

- تم تحسينه للتعلم الآلي على الجهاز من خلال معالجة خمسة قيود رئيسية:
  - **التاخير Latency**: لأنه لا توجد رحلة ذهابًا وإيابًا no round-trip إلى الخادم.
  - **الخصوصية Privacy**: لأنه لا توجد بيانات شخصية تخرج من الجهاز.
  - **الاتصال Connectivity**: لأن الاتصال بالإنترنت غير مطلوب.
  - **الحجم Size**: بسبب النموذج المصغر والحجم الثنائي.
  - **استهلاك الطاقة Power consumption**: بسبب الاستدلال الفعال ونقص اتصالات الشبكة.
- دعم أجهزة Android وiOS، وLinux المضمنة، ووحدات التحكم الدقيقة.
- دعم لغات البرمجة Java وSwift وObjective-C وC++ وPython.
- أداء عالي، مع تسريع الأجهزة وتحسين النموذج.
- أمثلة شاملة لمهام التعلم الآلي الشائعة مثل تصنيف الصور image classification، واكتشاف الكائنات object detection، وتقدير الوضعية pose estimation، والإجابة على الأسئلة question answering، وتصنيف النص text classification، وما إلى ذلك، على منصات متعددة multiple platforms.

فلاتر

فلاتر Flutter هو إطار تطوير مفتوح المصدر open source ومتعدد المنصات cross-platform. بمساعدة Flutter باستخدام قاعدة تعليمات برمجية واحدة، يمكننا إنشاء تطبيقات لنظام Android وiOS والويب بالإضافة إلى سطح المكتب. تم إنشاؤها بواسطة Google وتستخدم Dart كلغة

تطوير. تم إصدار أول نسخة مستقرة من Flutterفي أبريل 2018، ومنذ ذلك الحين، تم إجراء العديد من التحسينات.

بناء تطبيق ML-Flutter

سنقوم الآن ببناء تطبيق Flutter الذي يمكننا من خلاله معرفة الحالة الذهنية للشخص من خلال تعبيرات وجهه. توضح الخطوات التالية التحديث الذي يتعين علينا إجراؤه لتطبيق Android الأصلي. بالنسبة لتطبيق iOS، يرجى الرجوع إلى الروابط المتوفرةفي الخطوات.

TensorFlow Lite - الإعداد الأصلي (Android)

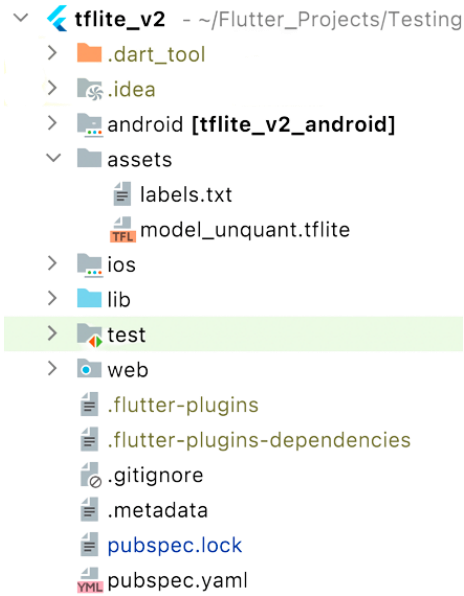
- في android/app/build.gradle، أضف الإعداد التاليفي كتلة android:

```
aaptOptions {
        noCompress 'tflite'
        noCompress 'lite'
    }
```

TensorFlow Lite - إعداد Flutter (Dart)

- قم بإنشاء مجلد assets ووضع ملف التسمية label file وملف النموذج model file فيه. (سننشئ هذه الملفات قريبًا.) أضففي pubspec.yaml:

```
assets:
   - assets/labels.txt
   - assets/<file_name>.tflite
```



- قم بتشغيل هذا الأمر (تثبيت حزمة TensorFlow Light):

```
$ flutter pub add tflite
```

- أضف السطر التالي إلى pubspec.yaml الخاص بحزمتك (وقم بتشغيل flutter pub get المضمنة):

```
dependencies:
     tflite: ^0.9.0
```

- الآن في كود Dart الخاص بك، يمكنك استخدام:

```
import 'package:tflite/tflite.dart';
```

- أضف تبعيات الكاميرا إلى pubspec.yaml الخاص بحزمتك (اختياري):

```
dependencies:
     camera: ^0.10.0+1
```

- الآن في كود Dart الخاص بك، يمكنك استخدام:

```
import 'package:camera/camera.dart';
```

- نظرًا لأن الكاميرا هي إحدى ميزات الأجهزة،في الكود الأصلي، هناك بعض التحديثات التي يتعين علينا إجراؤها لكل من Android وiOS. لمعرفة المزيد، قم بزيارة:

https://pub.dev/packages/camera

- فيما يلي الكود الذي سيظهر ضمن التبعيات في pubspec.yaml بمجرد اكتمال الإعداد.

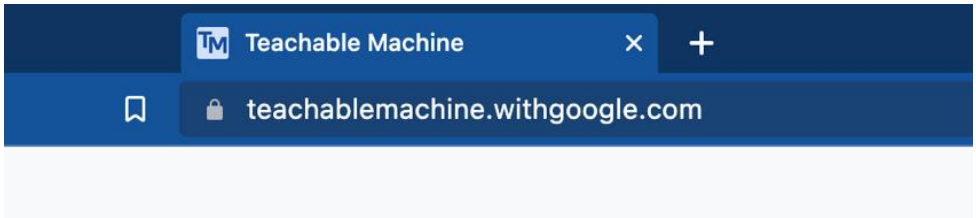
```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
  camera:
  tflite:

flutter:
  assets:
    - assets/
```

- سيقوم Flutter تلقائيًا بتنزيل أحدث إصدار إذا تجاهلت رقم إصدار الحزم.
- لا تنس إضافة مجلد assets في الدليل الجذر.

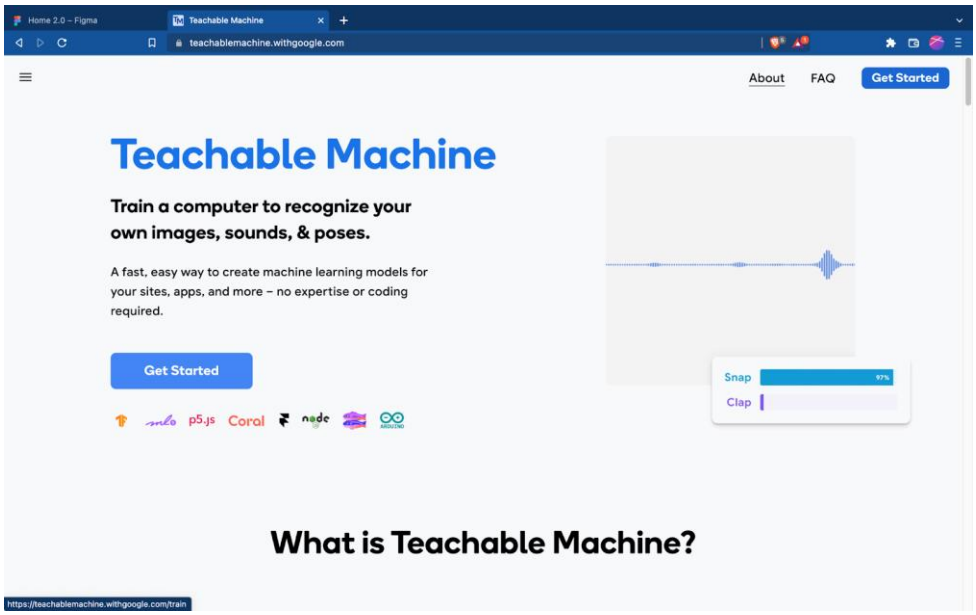
إنشاء نموذج (باستخدام موقع الويب)

- قم بزيارة الموقع التالي

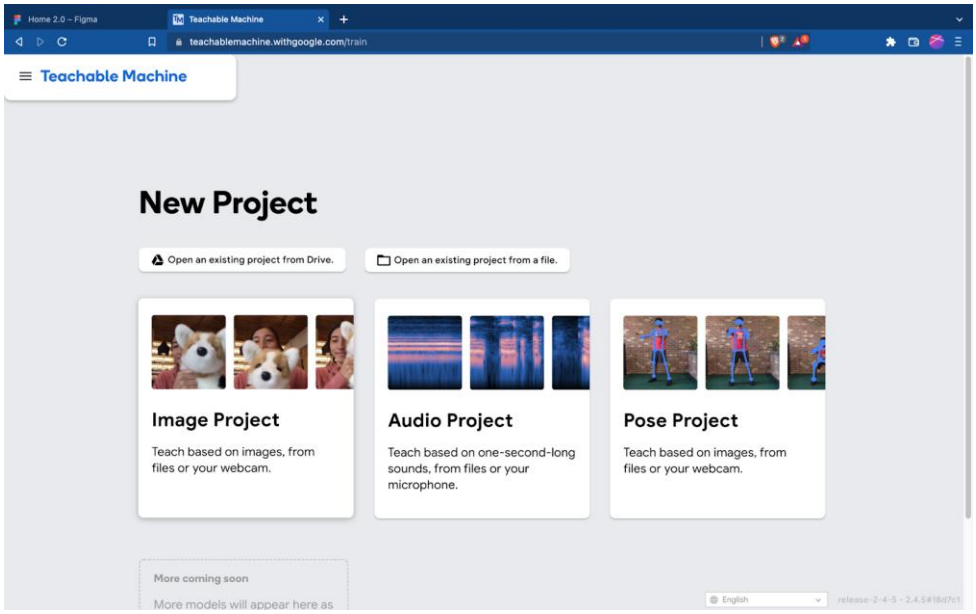
https://teachablemachine.withgoogle.com/



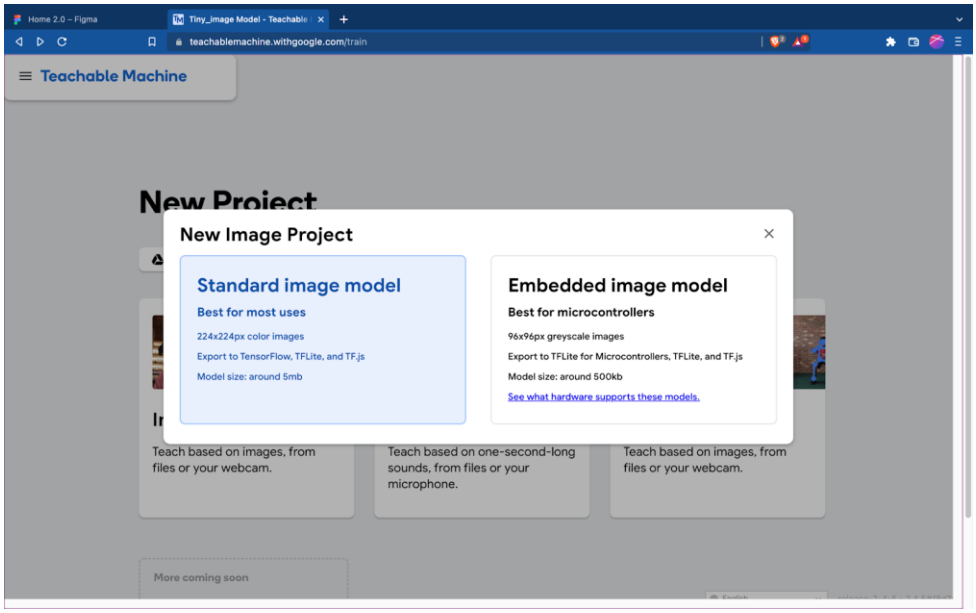
- انقر على البدء Get Started.



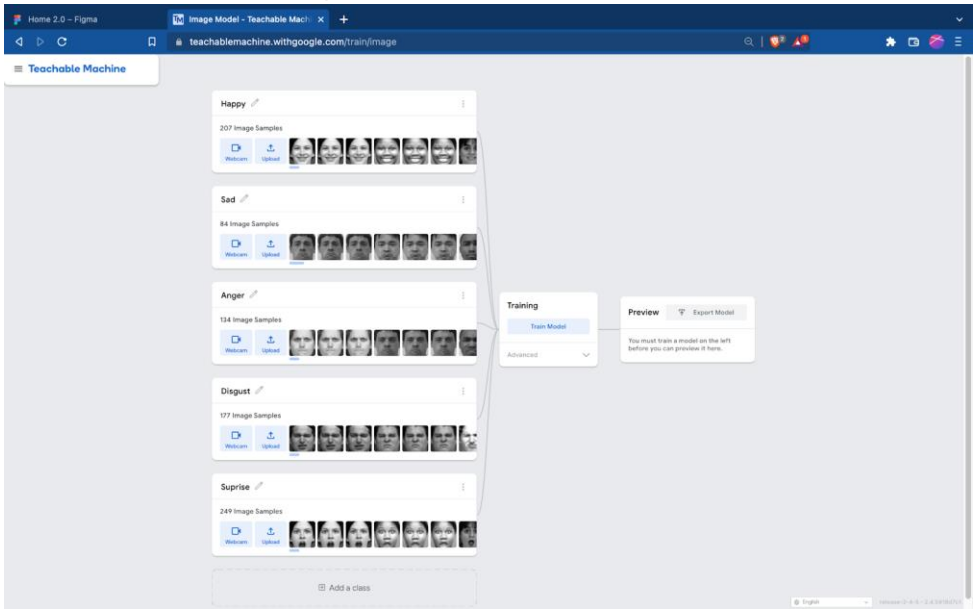
- حدد مشروع الصورة Image project.
- هناك ثلاث فئات مختلفة من مشاريع التعلم الآلي المتاحة. سنختار مشروع صورة لأننا سنقوم بتطوير مشروع يحلل تعبيرات وجه الشخص لتحديد حالته العاطفية.
- النوعان الآخران، المشروع الصوتي audio project ومشروع الوضعية pose project، سيكونان مفيدينفي إنشاء المشاريع التي تتضمن التشغيل الصوتي وإشارة الوضعية البشرية، على التوالي.



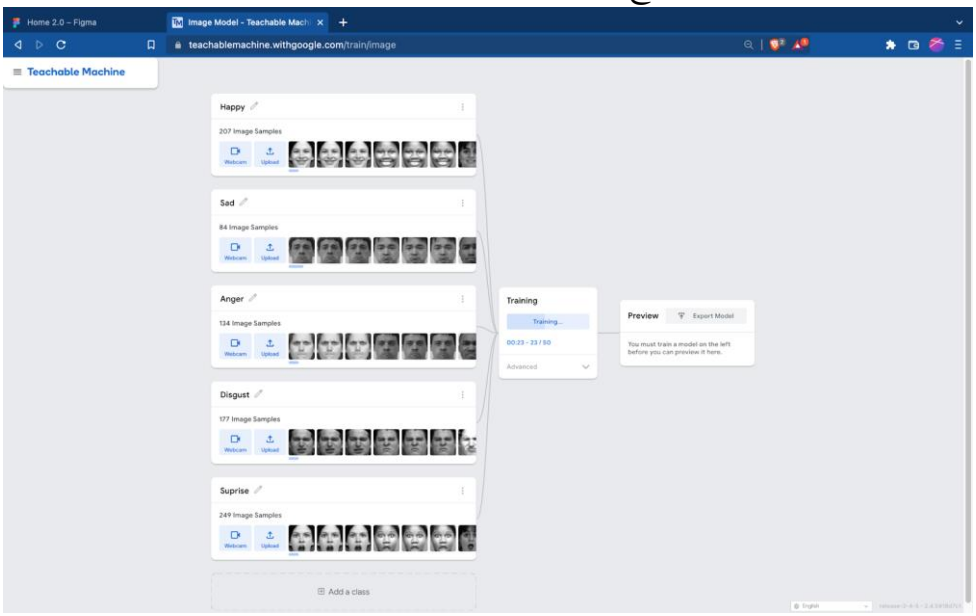
- حدد نموذج الصورة القياسية Standard Image model.
- مرة أخرى، هناك مجموعتان متميزتان من مشاريع التعلم الآلي للصور. نظرا لأننا نقوم بإنشاء مشروع لهاتف ذكي يعمل بنظام Android، فسوف نختار مشروع صورة قياسي.
- أما النوع الآخر، وهو مشروع Embedded Image Model، فهو مصمم للأجهزة ذات الذاكرة وقدرة الحوسبة المنخفضة نسبيًا.



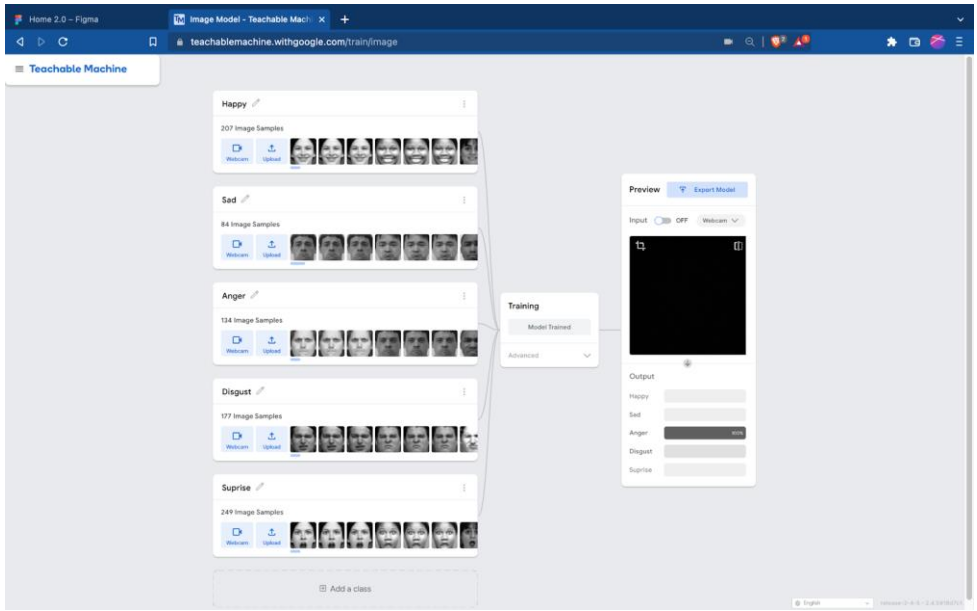
- تحميل الصور لتدريب الفئات classes.
- سنقوم بإنشاء فئات جديدة من خلال النقر على "إضافة فئة Add a class".
- يجب علينا تحميل الصور الفوتوغرافية إلى هذه الفئات بينما نعمل على تطوير مشروع يحلل الحالة العاطفية للشخص من خلال تعبيرات وجهه.
- كلما زاد عدد الصور التي نقوم بتحميلها، أصبحت النتائج أكثر دقة.



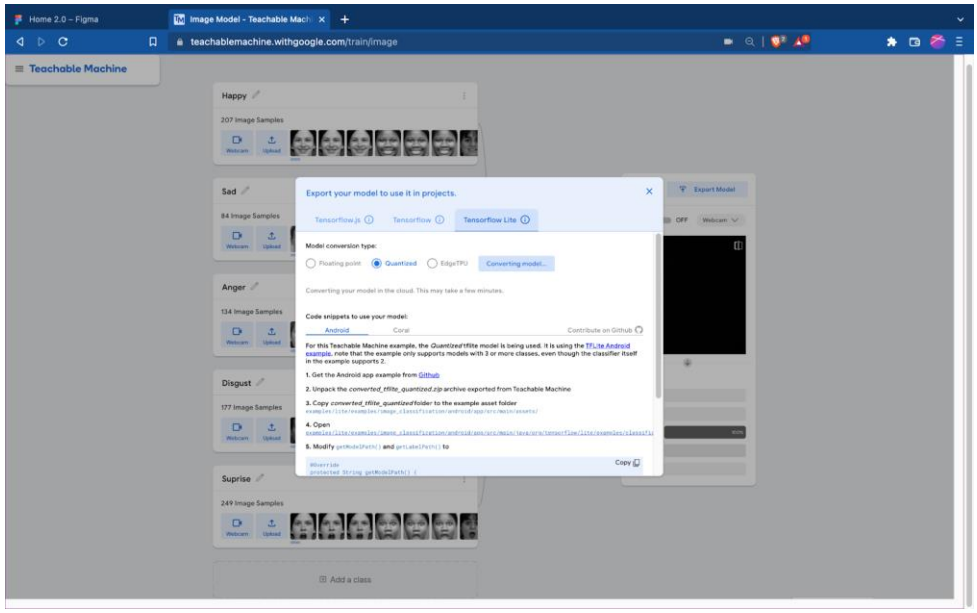
- انقر على تدريب النموذج train model وانتظر حتى انتهاء التدريب.



- انقر على تصدير النموذج Export model.



- حدد علامة التبويب TensorFlow Lite -> الزر الكمي Quantized button-> تنزيل النموذج الخاص بي Download my model



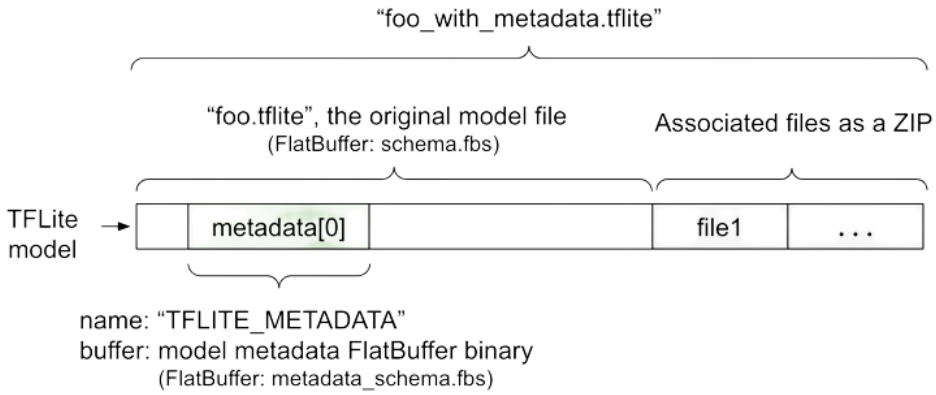
إضافة ملفات/نماذج إلى مشروع Flutter

- **Labels.txt**
- يحتوي الملف على كافة أسماء الفئات التي قمت بإنشائها أثناء إنشاء النموذج.

labels.txt

```
0 Happy
1 Angry
2 Sad
3 Fear
4 Surprise
5 Disgust
```

- ***.tflite**
- يحتوي الملف على ملف النموذج الأصلي بالإضافة إلى الملفات المرتبطة بتنسيق ZIP.



تحميل وتشغيل ML-Model

- نحن نقوم باستيراد النموذج من assets، لذا يعد هذا السطر من التعليمات البرمجية أمرًا بالغ الأهمية. سيكون هذا النموذج بمثابة عقل المشروع.

```
loadModel() async {
  await Tflite.loadModel(
      model: "assets/model_unquant.tflite", labels: "assets/labels.txt");
}
```

- نقوم هنا بتكوين الكاميرا باستخدام وحدة تحكم الكاميرا camera controller والحصول على بث مباشر (Cameras[0] هي الكاميرا الأمامية).

```
loadCamera() {
  cameraController = CameraController(cameras![0], ResolutionPreset.medium);
  cameraController!.initialize().then((value) {
    if (!mounted) {
      return;
    } else {
      setState(() {
        cameraController!.startImageStream((image) {
          cameraImage = image;
          runModel();
        });
      });
    }
  });
}

runModel() async {
  if (cameraImage != null) {
    var predictions = await Tflite.runModelOnFrame(
        bytesList: cameraImage!.planes.map((plane) {
          return plane.bytes;
        }).toList(),
        imageHeight: cameraImage!.height,
        imageWidth: cameraImage!.width,
        imageMean: 127.5,
        imageStd: 127.5,
        rotation: 90,
        numResults: 2,
        threshold: 0.1,
        asynch: true);
    predictions!.forEach((element) {
      setState(() {
        output = element['label'];
      });
    });
  }
}
```

الاستنتاج

يمكننا تحقيق أداء جيد لتطبيق Flutter باستخدام بُنية مناسبة، كما تمت مناقشتهفي هذه البرنامج التعليمي.

المصدر:

https://www.velotio.com/engineering-blog/machine-learning-in-flutter-using-tensorflow

1) انشاء تطبيق موبايل للتعرف على النباتات باستخدام فلاتر وتنسرفلو لايت Build a Plant Recognition Mobile App using Flutter with TensorFlow Lite

يعد التعلم الآلي أحد أهم التقنياتفي العقد الماضي. قد لا تدرك حتى أنه موجودفي كل مكان.

اعتمدت تطبيقات مثل الواقع المعزز، والمركبات ذاتية القيادة، وبوتات الدردشة، والرؤية الحاسوبية، ووسائل التواصل الاجتماعي، من بين تطبيقات أخرى، تكنولوجيا التعلم الآلي لحل المشكلات.

والخبر السار هو أن العديد من موارد وأطر التعلم الآلي متاحة للجمهور. اثنان منهم هما TensorFlow وTeachable Machine.

في هذا البرنامج التعليمي Flutter، ستقوم بتطوير تطبيق يسمى Plant Recognizer الذي يستخدم التعلم الآلي للتعرف على النباتات بمجرد النظر إلى صورها. ستحقق ذلك باستخدام منصة Teachable Machine، وTensorFlow Lite، وحزمة Flutter المسماة tflite_flutter.

بنهاية هذا البرنامج التعليمي، ستتعلم كيفية:

- استخدام التعلم الآليفي تطبيق الهاتف المحمول.
- تدريب نموذج باستخدام Teachable Machine.
- دمج واستخدام TensorFlow Lite مع حزمة tflite_flutter.
- أنشاء تطبيقًا للهاتف المحمول للتعرف على النباتات من خلال الصورة.

TensorFlow هي مكتبة شهيرة للتعلم الآلي للمطورين الذين يرغبونفي إنشاء نماذج تعليمية لتطبيقاتهم. TensorFlow Lite هو إصدار محمول من TensorFlow لنشر النماذج على الأجهزة المحمولة. وتعد Teachable Machine عبارة عن منصة مناسبة للمبتدئين لتدريب نماذج التعلم الآلي.

ملاحظة: يفترض هذا البرنامج التعليمي أن لديك فهمًا أساسيًا لـ Flutter وأنك قمت بتثبيت Android Studio أو Visual Studio Code. إذا كنت تستخدم نظام التشغيل macOS، فيجب أن يكون لديك Xcode مثبتًا أيضًا. إذا كنت جديدًافي استخدام Flutter، فيجب أن تبدأ بالبرنامج التعليمي الخاص بنا حول كيفية البدء باستخدام Flutter.

البدء

قم بتنزيل المشروع بالنقر فوق "تنزيل المواد Download Materials "في الجزء العلوي أو السفلي من البرنامج التعليمي واستخرجه إلى موقع مناسب.

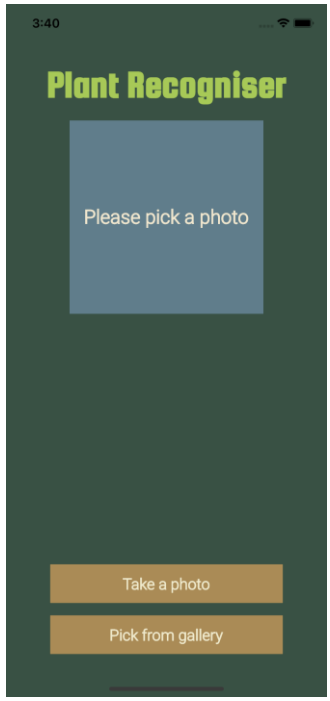
بعد فك الضغط ستظهر لك المجلدات التالية:

1. **final**: يحتوي على كود المشروع المكتمل.
2. **samples**: تحتوي على نماذج من الصور التي يمكنك استخدامها لتدريب النموذج الخاص بك.
3. **samples-test**: يضم عينات يمكنك استخدامها لاختبار التطبيق بعد اكتماله.
4. **starter**: المشروع المبدئي. ستعمل مع هذافي البرنامج التعليمي.

افتح المشروع المبدئيفي VS Code. لاحظ أنه يمكنك استخدام Android Studio، ولكن سيتعين عليك تعديل التعليمات بنفسك.

يجب أن يطالبك VS Code بالحصول على التبعيات dependencies ـ انقر فوق الزر للحصول عليها. يمكنك أيضًا تشغيل Flutter pub get من المحطة الطرفية للحصول على التبعيات.

البناء Build والتشغيل run بعد تثبيت التبعيات. يجب أن تشاهد الشاشة التالية:



يتيح لك المشروع بالفعل اختيار صورة من الكاميرا أو مكتبة الوسائط. اضغط على اختيار من المعرض Pick from gallery لتحديد صورة.

ملاحظة: قد تحتاج إلى نسخ الصور من samples-test إلى جهازك للاختبار. إذا كنت تستخدم محاكي iPhone أو محاكي Android، فما عليك سوى سحب وإفلات الصور من مجلد samples-test فيه. بخلاف ذلك، يمكنك العثور على تعليمات حول نسخ الملفات من جهاز كمبيوتر إلى جهاز محمول من الشركة المصنعة لجهازك.

كما ترون، التطبيق لا يتعرف على الصور. ستستخدم TensorFlow Lite لحل هذه المشكلةفي الأقسام التالية. ولكن أولاً، إليك نظرة عامة اختيارية من المستوى العالي على التعلم الآلي لتعطيك فكرة عما ستفعله.

مقدمة موجزة للتعلم الآلي

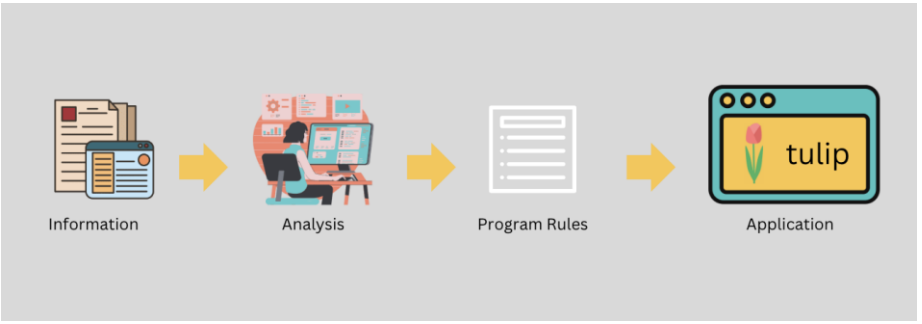
يعد هذا القسم اختياريًا لأن المشروع المبدئي يحتوي على نموذج مُدرب model_unquant.tflite وتسميات التصنيففي ملف labels.txt.

إذا كنت تفضل التعمق في تكامل TensorFlow Lite، فلا تترددفي الانتقال إلى تثبيت TensorFlow Lite.

ما هو التعلم الآلي

في هذا البرنامج التعليمي Flutter، تحتاج إلى حل مشكلة التصنيف classification problem: التعرف على النباتات plant recognition.في النهج التقليدي، يمكنك تحديد القواعد لتحديد الصور التي تنتمي إلى أي فئة.

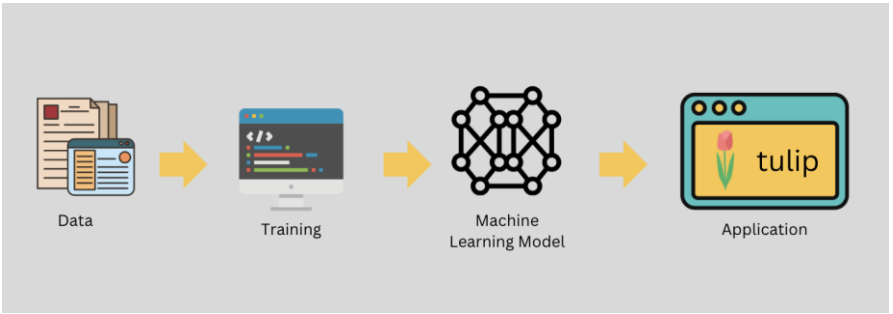
ستستند القواعد إلى أنماط مثل زهرة عباد الشمس sunflower، التي تحتوي على دائرة كبيرةفي المنتصف، أو الوردة، التي تشبه إلى حد ما كرة ورقية paper ball. تسير الأمور على النحو التالي:



النهج التقليدي لديه العديد من المشاكل:

- هناك العديد من القواعد التي يجب ضبطها عند وجود العديد من تسميات التصنيف.
- انها ذاتية.
- يصعب تحديد بعض القواعد بواسطة البرنامج. على سبيل المثال، لا يمكن تحديد القاعدة "مثل الكرة الورقية" بواسطة برنامج لأن الكمبيوتر لا يعرف كيف تبدو الكرة الورقية.

يقدم التعلم الآلي طريقة أخرى لحل المشكلة. بدلاً من تحديد القواعد، يحدد الجهاز قواعده الخاصة بناءً على البيانات المدخلة التي تقدمها:



تتعلم الآلة من البيانات، ولهذا السبب يسمى هذا النهج بالتعلم الآلي.

قبل المتابعة، إليك بعض المصطلحات التي قد تحتاج إلى معرفتها:

التدريب Training: العملية التي يتعلم الكمبيوتر من خلالها البيانات ويشتق القواعد.

بناء نموذج باستخدام Teachable Machine

إعداد مجموعة البيانات

تدريب النموذج

ستتعلم الآن كيفية تدريب نموذج باستخدام Teachable Machine. تتضمن الخطوات التي ستتبعها ما يلي:

خطوتك الأولى هي إعداد مجموعة البيانات الخاصة بك ـ يحتاج المشروع إلى صور نباتية. لذا فإن مجموعة البيانات الخاصة بك عبارة عن مجموعة من النباتات التي تريد التعرف عليها.

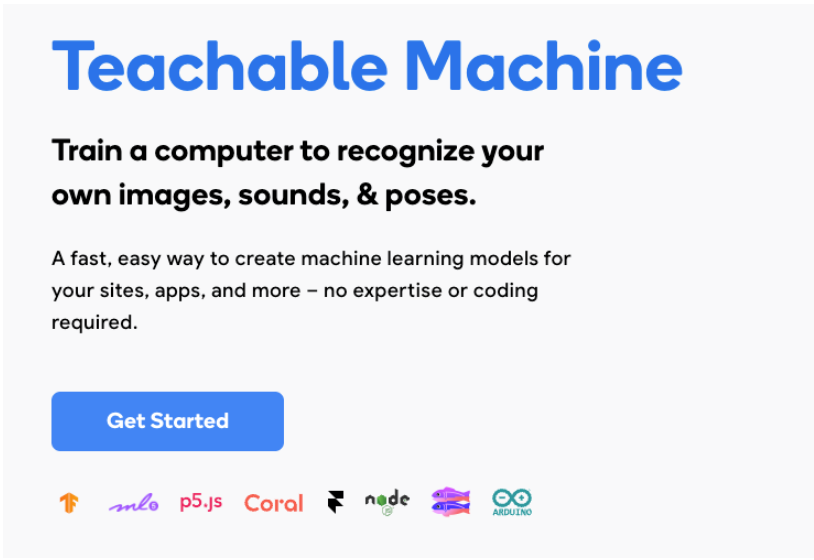
في تطبيق جاهز للإنتاج، قد ترغبفي جمع أكبر عدد ممكن من أنواع النباتات وأكبر عدد ممكن من النباتات لمجموعة البيانات الخاصة بك لضمان دقة أعلى. يمكنك القيام بذلك عن طريق استخدام كاميرا هاتفك لالتقاط صور لهذه النباتات أو تنزيل الصور من مصادر مختلفة عبر الإنترنت تقدم مجموعات بيانات مجانية مثل هذه من Kaggle.

ومع ذلك، يستخدم هذا البرنامج التعليمي النباتات من مجلد samples، لذا يمكنك أيضًا استخدامه كنقطة بداية.

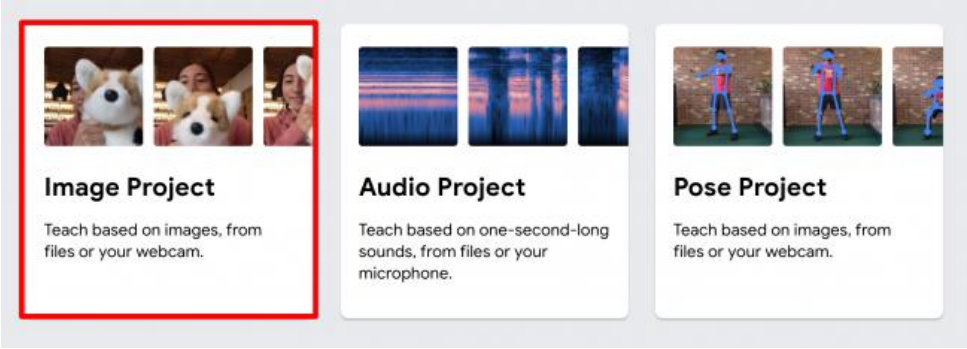
أيًا كان النوع الذي تستخدمه، فمن المهم الحفاظ على عدد العينات لكل تسمية عند مستويات مماثلة لتجنب إدخال التحيزفي النموذج.

بعد ذلك، ستتعلم كيفية تدريب النموذج باستخدام Teachable Machine.

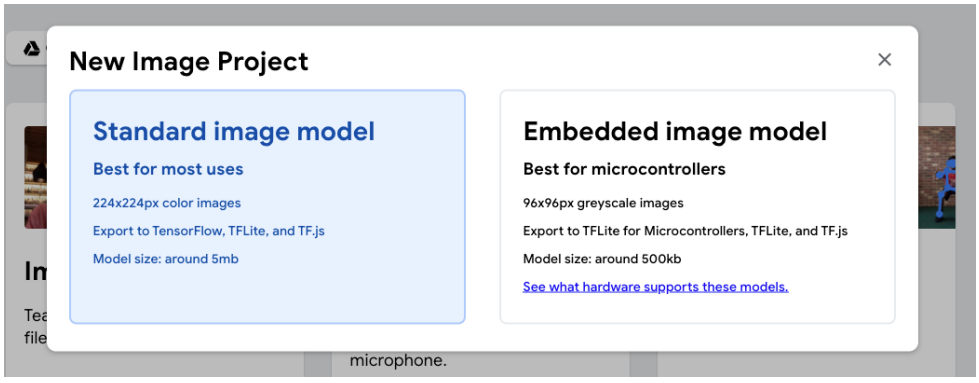
أولاً، انتقل إلى https://teachablemachine.withgoogle.com/ وانقر على "البدء Get Started" لفتح أداة التدريب:



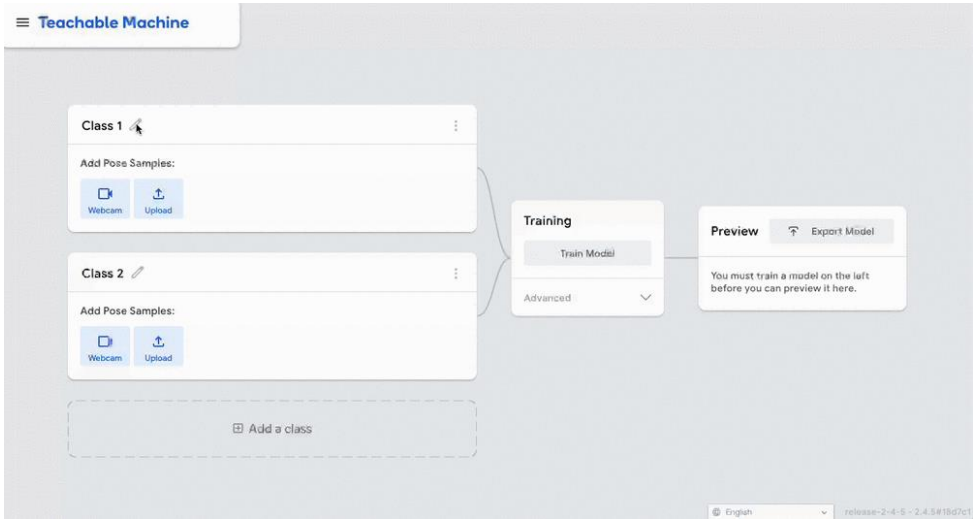
ثم حدد Image Project:



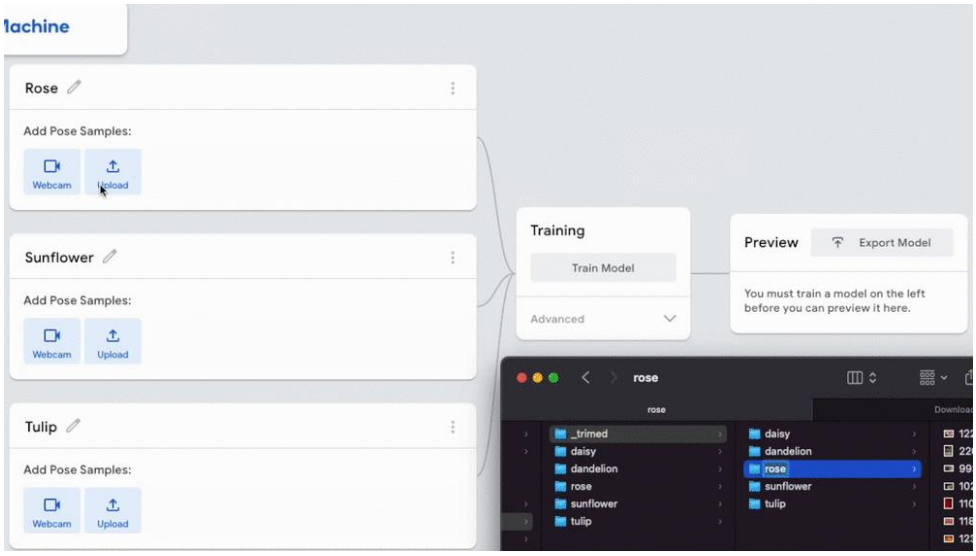
اختر Standard Image Model، لأنك لا تقوم بتدريب نموذج ليتم تشغيله على وحدة تحكم دقيقة microcontroller:



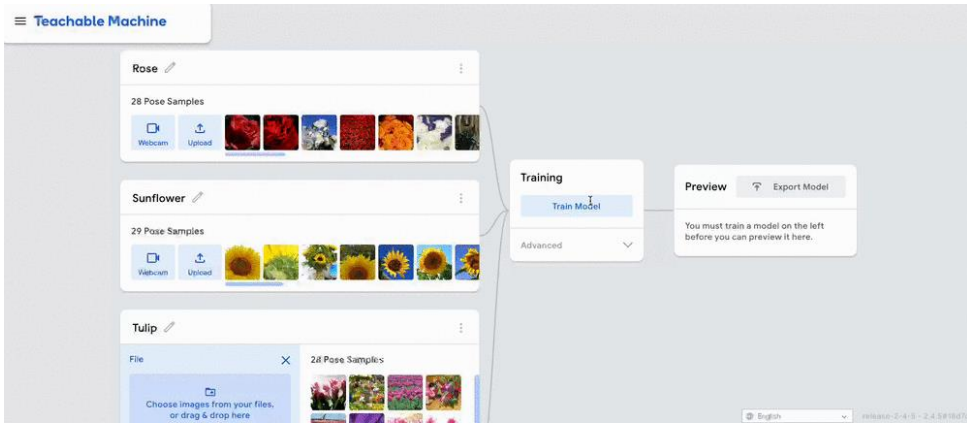
بمجرد دخولك إلى أداة التدريب، أضف الفئات classes وقم بتحرير تسميات labels كل فئة، كما هو موضح أدناه:



بعد ذلك، قم بإضافة نماذج التدريب الخاصة بك عن طريق النقر فوق تحميل Upload أسفل كل فئة. ثم اسحب المجلد الخاص بنوع النبات المناسب من مجلد samples إلى لوحة اختيار الصور من ملفاتك Choose images from your files ....

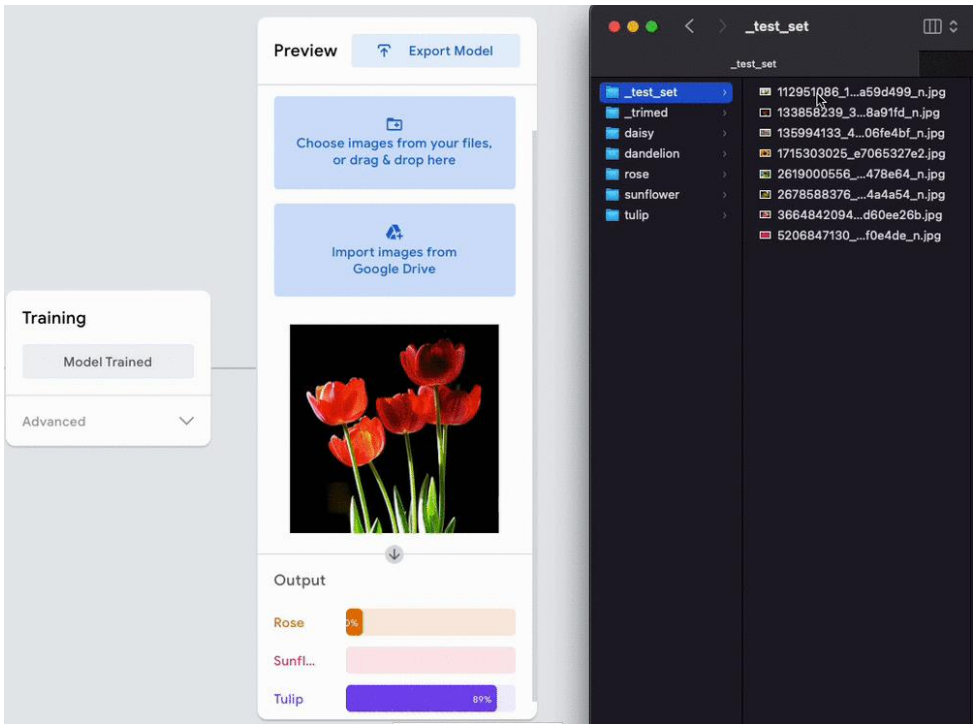


بعد إضافة جميع نماذج التدريب، انقر فوق Train Model لتدريب النموذج:

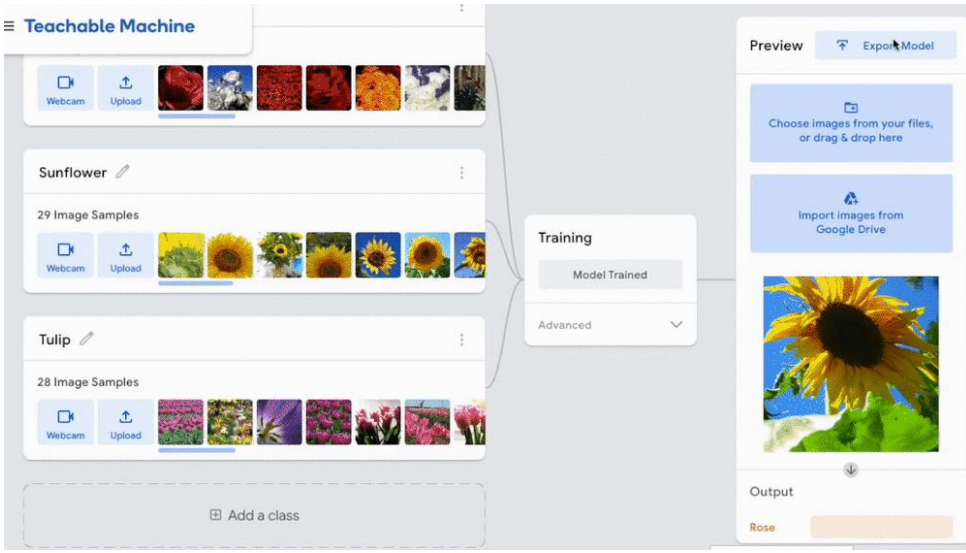


بعد اكتمال التدريب، اختبر النموذج باستخدام صور نباتية أخرى.

استخدم الصور الموجودةفي مجلد test-samples، كما يلي:



وأخيرًا، قم بتصدير النموذج بالنقر فوق "تصدير النموذج Export Model "في لوحة "المعاينة Preview panel ". يعرض مربع حوار:



في مربع الحوار، اختر TensorFlow Lite. وذلك لأن النظام الأساسي الذي تستهدفه هو الهاتف المحمول.

بعد ذلك، حدد نوع تحويل النقطة العائمة Floating point للحصول على أفضل أداء تنبؤي. ثم انقر فوق تنزيل النموذج الخاص بي Download my model لتحويل النموذج وتنزيله.

قد يستغرق الأمر عدة دقائق لإكمال عملية تحويل النموذج. بمجرد الانتهاء من ذلك، سيتم تنزيل ملف النموذج تلقائيًا إلى نظامك.

بعد أن يكون لديك ملف النموذج converter_tflite.zip في متناول يدك، قم بفك ضغطه وانسخ labels.txt وmodel_unquant.tflite إلى المجلد assets/. في مشروعك المبدئي.

إليك ما يحتويه كل ملف من هذه الملفات:

النموذج Model: الكائن الذي تم إنشاؤه من التدريب. وهو يتألف من الخوارزمية المستخدمة لحل مشكلة الذكاء الاصطناعي والقواعد المستفادة.

1. إعداد مجموعة البيانات Preparing the dataset.
2. تدريب النموذج Training the model.
3. تصدير النموذج Exporting the model.

ملاحظة: تأكد دائمًا من التحقق من شروط الخدمة terms of service (TOS) إذا كنت تقوم بتنزيل الصور من إحدى الخدمات. مع تزايد شعبية التعلم الآلي، تعمل الكثير من الخدمات على تعديل شروط الخدمة الخاصة بها لمعالجة بياناتها المضمنةفي نماذج التعلم الآلي على وجه التحديد.

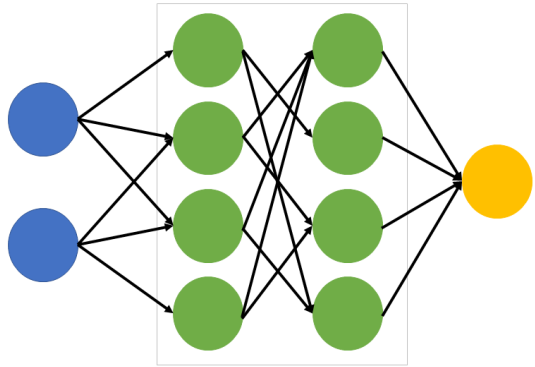
ملاحظة: أنواع التحويل الأخرى، quantized وEdge TPU، هي الأفضل للأجهزة التي تتمتع بقدرة حوسبة أقل من الهاتف المحمول. يتمثل الاختلاف الرئيسيفي أن البيانات الرقمية المستخدمةفي النموذج يتم تحويلها إلى أنواع بيانات أقل دقة يمكن لهذه الأجهزة التعامل معها، مثل عدد صحيح أو نقطة عائمة 16 بت.

labels.txt: تسمية كل فئة.

model_unquant.tflite: نموذج التعلم الآلي المُدرب للتعرف على النباتات.

تدريب النموذج: كيف يعمل

يستخدم TensorFlow نهجًا يسمى التعلم العميق deep learning، وهو مجموعة فرعية من التعلم الآلي. يستخدم التعلم العميق بُنية شبكة ذات طبقات عديدة، مشابهة لما هو موضح أدناه:



ولتوضيح الأمر أكثر:

- **يتم تغذية بيانات الإدخالفي الطبقة الأولى**: إذا كانت بيانات الإدخال عبارة عن صورة، فسيتم تغذية وحدات البكسل الخاصة بالصورة إلى الطبقة الأولى.
- **يتم تخزين نتيجة الإخراجفي الطبقة الأخيرة**: إذا كانت الشبكة تحل مشكلة تصنيف، فإن هذه الطبقة تخزن إمكانية كل فئة.
- **تسمى الطبقات الموجودة بينهما بالطبقات المخفية hidden layers**. أنها تحتوي على صيغ ذات معلمات موجودةفي العقدة. تتدفق قيم الإدخال إلى تلك الطبقات، والتي تقومفي النهاية بحساب النتائج النهائية.

يقوم التعلم العميق بضبط المعلماتفي الطبقات المخفية لتحقيق نتائج تنبؤية مماثلة للنتيجة المقدمة. هناك حاجة إلى العديد من التكرارات لعملية تدريب الآلة لتحقيق معلمات مضبوطة جيداً.

يتضمن كل تكرار الإجراءات التالية:

- قم بتشغيل خطوة التنبؤ باستخدام عينة الإدخال.
- قارن نتيجة التنبؤ بالنتيجة المقدمة. سيقوم النظام بحساب مقدار الفرق بينهما، وتسمى هذه القيمة **بالخسارة (الخطأ) loss**.
- قم بتعديل المعلماتفي الطبقات المخفية لتقليل الخسارة.

بعد اكتمال التكرارات، سيكون لديك معلمات محسنة، وستكون نتائجك بأعلى دقة ممكنة.

فهم تنبؤ Tensor و TensorFlow

بالنسبة لعملية التدريب والتنبؤ، يستخدم TensorFlow هيكل بيانات تسمى موترات Tensors كمدخلات ومخرجات ـ ولهذا السبب يعتبر Tensor جزءًا من اسم TensorFlow.

الموتر Tensor : عبارة عن مصفوفة متعددة الأبعاد تمثل بياناتك المدخلة ونتائج التعلم الآلي.

قد تساعدك التعريفات التالية على فهم ماهية الموتر، مقارنة بما تعرفه بالفعل:

- **العددي Scalar:** قيمة فردية، على سبيل المثال: 1، 2، 3.3
- **المتجه Vector:** قيمة متعددة المحاور، أمثلة: (0، 0)، (1، 2)، (3
- **الموتر Tensor**: قيمة متعددة الأبعاد. المثال هو: (((0 ,0), (0 ,1)), ((1,1), ((2,2)

في مشكلة تصنيف الصور image classification problem، يكون موتر الإدخال عبارة عن مصفوفة تمثل صورة، على غرار ما يلي:

```
[
 // First line of the first image
 [
   // First Pixel of the first line
   [0.0, 0.0, 1.0],
   // Second Pixel of the first line
   [0.0, 0.0, 1.0],
   [1.0, 1.0, 0.0], ...
 ]
 // Second line of the first image
 ...
 ...
[
```

للتوضيح أكثر:

- تمثل الطبقة الأولى من المصفوفة كل سطر من الصورة.

- تمثل الطبقة الثانية من المصفوفة كل بكسل من الخط.
- تمثل الطبقة الأخيرة لون البكسل وهو الأحمر أو الأخضر أو الأزرق.

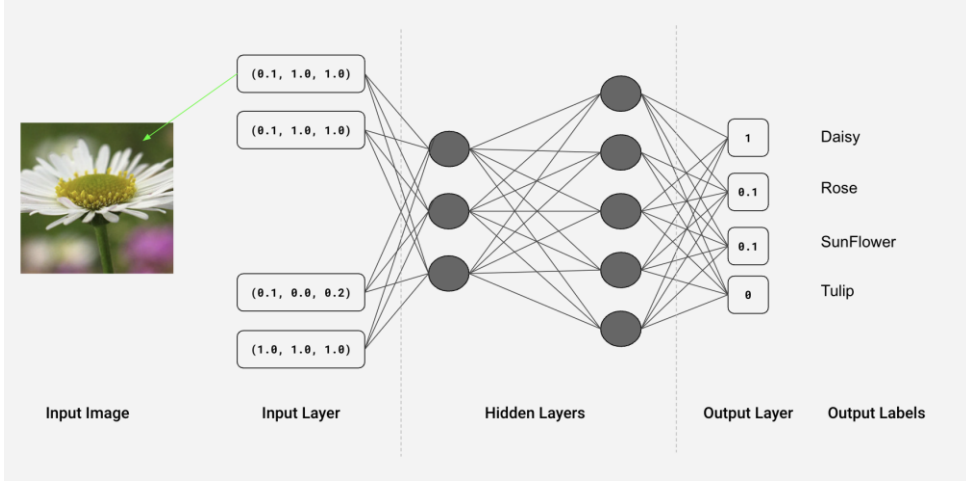
إذا قمت بإعادة تشكيل الصورة إلى 200×200، فإن شكل الموتر هو [200, 200, 3].

موتر الإخراج عبارة عن مصفوفة من النقاط لكل تسمية، على سبيل المثال:

[0.1، 0.8، 0.1، 0].في هذه الحالة، تتوافق كل قيمة مع تسمية، على سبيل المثال، الورد rose، والتيوليب tulip، وعباد الشمس sunflower ، والأقحوان daisy.

لاحظ أنهفي المثال، قيمة تسمية زهرة التيوليب هي 0.8 ـ مما يعني أن احتمال أن تظهر الصورة زهرة التيوليب هي 80%، والأخرى هي 10% وزهرة الأقحوان 0%. شكل الإخراج هنا هو [4].

ويوضح الرسم البياني التالي تدفق البيانات بشكل أكبر:



نظرًا لأن TensorFlow يستخدم الموترات للمدخلات والمخرجات، فأنت بحاجة إلى إجراء المعالجة المسبقة حتى يفهم TensorFlow بيانات الإدخال والمعالجة اللاحقة حتى يتمكن المستخدمون البشريون من فهم بيانات الإخراج. ستقوم بتثبيت TensorFlow Liteفي القسم التالي لمعالجة البيانات.

تثبيت TensorFlow Lite في Flutter

لاستخدام TensorFlowفي تطبيق Flutter، تحتاج إلى تثبيت الحزم التالية:

- **tflite_flutter:** يسمح لك بالوصول إلى مكتبة TensorFlow Lite الأصلية. عند استدعاء طرق tflite_flutter، فإنه يستدعي الطريقة المقابلة لـ TensorFlow Lite SDK الأصلي.

- **tflite_flutter_helper:** يمكّنك من معالجة مدخلات ومخرجات TensorFlow. على سبيل المثال، يقوم بتحويل بيانات الصورة إلى بُنية موتر. فهو يقلل من الجهد المطلوب لإنشاء منطق ما قبل وما بعد المعالجة لنموذجك.

افتح pubspec.yaml وأضفهافي قسم التبعيات dependencies:

```
tflite_flutter: ^0.9.0
tflite_flutter_helper: ^0.3.1
```

ثم قم بتشغيل Flutter Pub للحصول على الحزم.

ملاحظة: إذا رأيت خطأ مثل:

Class 'TfliteFlutterHelperPlugin' is not abstract and does not implement abstract member public abstract fun onRequestPermissionsResult(p0: Int, p1: Array<(out) String!>, p2:

للتغلب على هذه المشكلة، استبدل تبعية tflite_flutter_helper: ^0.3.1 باستدعاء git التالي:

```
tflite_flutter_helper:
 git:
  url: https://github.com/filofan1/tflite_flutter_helper.git
  ref: 783f15e5a87126159147d8ea30b98eea9207ac70
```

الحصول على الحزم مرة أخرى.

بعد ذلك، إذا كنت تقوم بالتصميم لنظام التشغيل Android، فقم بتشغيل السكريبت للتثبيت أدناه على نظام التشغيل macOS/Linux:

```
./install.sh
```

إذا كنت تستخدم نظام التشغيل Windows، فقم بتشغيل install.bat بدلاً من ذلك:

```
install.bat
```

ومع ذلك، للإنشاء لنظام iOS، تحتاج إلى تنزيل TensorFlowLiteC.framework، وفك ضغطه ووضع TensorFlowLiteC.framework في مجلد pub-cache. لـ tflite_flutter. موقع المجلد هو home/USER/.pub-cache/hosted/pub.dartlang.org/tflite_flutter-/0.9.0/ios/، حيث USER هو اسم المستخدم الخاص بك. إذا كنت لا تستخدم الإصدار 0.9.0، فضعهفي الإصدار المقابل.

أنت تقوم فقط بإضافة مكتبات Android وiOS الديناميكية حتى تتمكن من تشغيل TensorFlow Lite على النظام الأساسي المستهدف.

إنشاء مصنف الصور

في التعلم الآلي، يشير التصنيف إلى التنبؤ بفئة كائن ما من بين عدد محدود من الفئات، مع إعطاء بعض المدخلات.

المصنف المتضمنفي المشروع المبدئي هو هيكل عظمي لمصنف الصور الذي ستقوم بإنشائه للتنبؤ بفئة نبات معين.

بنهاية هذا القسم، سيكون المصنف مسؤولاً عن الخطوات التالية:

1. تحميل التسميات والنموذج.
2. المعالجة المسبقة للصورة.
3. استخدام النموذج.
4. المعالجة اللاحقة لإخراج TensorFlow.
5. تحديد وبناء إخراج الفئة.

سيقوم كود التهيئة الخاص بك بتحميل التسميات والنموذج من ملفاتك. وبعد ذلك، سيتم إنشاء هياكل TensorFlow وإعدادها لاستخدامها من خلال استدعاء ()predict.

سيتضمن إجراء التنبؤ الخاص بك عدة أجزاء. أولاً، سيتم تحويل صورة Flutter إلى موتر إدخال TensorFlow. وبعد ذلك، سيتم تشغيل النموذج وتحويل الإخراج إلى سجل الفئة المحدد النهائي الذي يحتوي على التسمية label والنتيجة score.

ملاحظة: يقوم المشروع المبدئي بالفعل بتنفيذ عناصر واجهة المستخدم واستخدام مثيل Classifier. يصف القسم الأخير من هذا البرنامج التعليمي، استخدام المصنف وكيفية تنفيذه.

استيراد النموذج إلى Flutter

هناك نوعان من البيانات التي ستقوم بتحميلهافي البرنامج: نموذج التعلم الآلي – model_unquant.tflite وتسميات التصنيف – labels.txt، التي حصلت عليها من منصة Teachable Machine.

للبدء، تأكد من تضمين مجلد assetsفي pubspec.yaml:

```
assets:
 -  assets/
```

سجل assets مسؤول عن نسخ ملفات الموارد الخاصة بك إلى حزمة التطبيق النهائية.

تحميل تسميات التصنيف

افتح lib/classifier/classifier.dart واستورد tflite_flutter_helper:

```
import 'package:tflite_flutter_helper/tflite_flutter_helper.dart;'
```

ثم أضف الكود التالي بعد التنبؤ predict:

```
static Future<ClassifierLabels> _loadLabels(String labelsFileName) async {
  // #1
  final rawLabels = await FileUtil.loadLabels(labelsFileName);

  // #2
  final labels = rawLabels
    .map((label) => label.substring(label.indexOf(' ')).trim())
    .toList();

  debugPrint('Labels: $labels');
  return labels;
{
```

إليك ما يفعله الكود أعلاه:

- يقوم بتحميل التسميات باستخدام الأداة المساعدة للملفات من tflite_flutter_helper.
- إزالة بادئة رقم الفهرس من التصنيفات التي قمت بتنزيلها مسبقًا. على سبيل المثال، يتغير 0 Rose إلى Rose.

بعد ذلك، استبدل _loadLabels :TODO // في LoadWith عن طريق استدعاء _loadLabels كما يلي:

```
final labels = await _loadLabels(labelsFileName);
```

يقوم هذا الكود بتحميل ملف التسمية.

احفظ التغييرات. لم يعد هناك أي شيء يتعلق بالتسميات الآن، لذا فقد حان الوقت لإجراء اختبار.

بناء وتشغيل المشروع.

انظر إلى إخراج وحدة التحكم:

```
flutter: Start loading of Classifier with labels at assets/labels.txt, model at model_unquant.tf
lite
flutter: Labels: [Rose, Daisy, Sunflower, Tulip]
```

تهانينا، لقد نجحتفي تحليل تسميات النموذج!

استيراد نموذج TensorFlow Lite

انتقل إلى lib/classifier/classifier_model.dart واستبدل المحتويات بالكود التالي:

```
import 'package:tflite_flutter/tflite_flutter.dart';

class ClassifierModel {
 Interpreter interpreter;

 List<int> inputShape;
 List<int> outputShape;
```

```
 TfLiteType inputType;
 TfLiteType outputType;

 ClassifierModel({
  required this.interpreter,
  required this.inputShape,
  required this.outputShape,
  required this.inputType,
  required this.outputType,
 });
{
```

يقوم ClassifierModel بتخزين جميع البيانات المتعلقة بالنموذج للمصنف الخاص بك. ستستخدم المفسر interpreter للتنبؤ بالنتائج. إن inputShape وoutputShape هما شكلان لبيانات الإدخال والإخراج على التوالي بينما inputType وoutputType هما أنواع البيانات لموترات الإدخال والإخراج.

الآن، قم باستيراد النموذج من الملف. انتقل إلى lib/classifier/classifier.dart وأضف الكود التالي بعد _loadLabels :

```
static Future<ClassifierModel> _loadModel(String modelFileName) async {
  // #1
  final interpreter = await Interpreter.fromAsset(modelFileName);

  // #2
  final inputShape = interpreter.getInputTensor(0).shape;
  final outputShape = interpreter.getOutputTensor(0).shape;

  debugPrint('Input shape: $inputShape');
  debugPrint('Output shape: $outputShape');

  // #3
  final inputType = interpreter.getInputTensor(0).type;
  final outputType = interpreter.getOutputTensor(0).type;

  debugPrint('Input type: $inputType');
  debugPrint('Output type: $outputType');

  return ClassifierModel(
   interpreter: interpreter,
   inputShape: inputShape,
   outputShape: outputShape,
   inputType: inputType,
   outputType: outputType,
  );
 }
```

لا تنس إضافة import 'package:tflite_flutter/tflite_flutter.dart'; في القمة.

إليك ما يحدث في الكود أعلاه:

1. إنشاء مفسر interpreter باستخدام ملف النموذج المقدم : المفسر عبارة عن أداة للتنبؤ بالنتيجة.

2. اقرأ أشكال الإدخال والإخراج، التي ستستخدمها لإجراء المعالجة المسبقة والمعالجة اللاحقة لبياناتك.
3. اقرأ أنواع المدخلات والمخرجات حتى تعرف نوع البيانات المتوفرة لديك.

بعد ذلك، استبدل TODO: _loadModel // في loadWith بما يلي:

```
final model = await _loadModel(modelFileName);
```

يقوم الكود أعلاه بتحميل ملف النموذج.

بناء وتشغيل المشروع. انظر إلى إخراج وحدة التحكم:

```
flutter: Start loading of Classifier with labels at assets/labels.txt, model at model_unquant.tf
lite
flutter: Labels: [Rose, Daisy, Sunflower, Tulip]
flutter: Input shape: [1, 224, 224, 3]
flutter: Output shape: [1, 4]
flutter: Input type: TfLiteType.float32
flutter: Output type: TfLiteType.float32
```

لقد قمت بتحليل النموذج بنجاح! إنها مجموعة متعددة الأبعاد من قيم float32.

أخيرًا، للتهيئة، استبدل TODO: build and return Classifier // في LoadWith بما يلي:

```
return Classifier._(labels: labels, model: model);
```

يؤدي ذلك إلى إنشاء مثيل Classifier الخاص بك، والذي يستخدمه PlantRecogniser للتعرف على الصور التي يقدمها المستخدم.

تنفيذ التنبؤ TensorFlow

قبل القيام بأي تنبؤ، تحتاج إلى إعداد المدخلات.

ستكتب طريقة لتحويل كائن Image من Flutter إلى TensorImage، وهي بُنية الموتر التي يستخدمها TensorFlow للصور. تحتاج أيضًا إلى تعديل الصورة لتناسب الشكل المطلوب للنموذج.

معالجة بيانات الصورة مسبقًا

بمساعدة tflite_flutter_helper، أصبحت معالجة الصور بسيطة لأن المكتبة توفر العديد من الدوال التي يمكنك استخدامها للتعامل مع إعادة تشكيل الصورة.

أضف طريقة _preProcessInput إلى lib/classifier/classifier.dart:

```
TensorImage _preProcessInput(Image image) {
  // #1
  final inputTensor = TensorImage(_model.inputType);
  inputTensor.loadImage(image);

  // #2
  final minLength = min(inputTensor.height, inputTensor.width);
```

```
  final cropOp = ResizeWithCropOrPadOp(minLength, minLength);

  // #3
  final shapeLength = _model.inputShape[1];
  final resizeOp = ResizeOp(shapeLength, shapeLength,
ResizeMethod.BILINEAR);

  // #4
  final normalizeOp = NormalizeOp(127.5, 127.5);

  // #5
  final imageProcessor = ImageProcessorBuilder()
    .add(cropOp)
    .add(resizeOp)
    .add(normalizeOp)
    .build();

  imageProcessor.process(inputTensor);

  // #6
  return inputTensor;
 }
```

يقوم _preProcessInput بمعالجة كائن الصورة مسبقًا بحيث يصبح TensorImage المطلوب. هذه هي الخطوات المعنية:

1. قم بإنشاء TensorImage وقم بتحميل بيانات الصورة إليه.
2. قص Crop الصورة إلى شكل مربع. يجب عليك استيراد dart:math في الأعلى لاستخدام الدالة min.
3. قم بتغيير حجم Resize عملية الصورة لتناسب متطلبات الشكل للنموذج.
4. قم بتسوية Normalize قيمة البيانات. تم تحديد الوسيطة 127.5 بسبب معلمات النموذج المُدرب. تريد تحويل قيمة بكسل الصورة 0ـ255 إلى نطاق ـ1...1.
5. قم بإنشاء معالج الصور image processor بالعملية المحددة وقم بمعالجة الصورة مسبقًا.
6. إرجاع الصورة المعالجة مسبقًا.
7. ثم، قم باستدعاء الطريقة داخل predict(...) في _preProcessInput :// TODO:

```
final inputImage = _preProcessInput(image);

debugPrint(
 'Pre-processed image: ${inputImage.width}x${image.height}, '
 'size: ${inputImage.buffer.lengthInBytes} bytes',
);
```

لقد قمت بتنفيذ منطق المعالجة المسبقة الخاص بك.

بناء وتشغيل المشروع.

اختر صورة من المعرض وانظر إلى وحدة التحكم:

```
flutter: Image: 320x240, size: 76800 bytes
flutter: Pre-processed image: 224x240, size: 602112 bytes
```

لقد نجحتفي تحويل الصورة إلى الشكل المطلوب للنموذج!

بعد ذلك، ستقوم بتشغيل التنبؤ.

تنفيذ التنبؤ

أضف الكود التاليفي TODO: run TF Lite // : قم بتشغيل TF Lite لتشغيل التنبؤ:

```
// #1
final outputBuffer = TensorBuffer.createFixedSize(
 _model.outputShape,
 _model.outputType,
);

// #2
_model.interpreter.run(inputImage.buffer, outputBuffer.buffer);
debugPrint('OutputBuffer: ${outputBuffer.getDoubleList()}');
```

إليك ما يحدثفي الكود أعلاه:

- يقوم TensorBuffer بتخزين النتائج النهائية للتنبؤ الخاص بك بتنسيق أولي.
- يقرأ المفسر صورة الموتر ويخزن الإخراجفي المخزن المؤقت.

بناء وتنفيذ المشروع.

حدد صورة من معرض الصور الخاص بك ولاحظ وحدة التحكم:

```
flutter: OutputBuffer: [0.004849598277360201, 0.22528165578842163, 0.7576797604560852, 0.0121890
18540084362]
```

عمل عظيم! لقد حصلت بنجاح على نتيجة تفسيرية من النموذج. ما عليك سوى بضع خطوات إضافية لجعل النتائج مناسبة للمستخدمين البشريين. ينقلك هذا إلى المهمة التالية: المعالجة اللاحقة للنتيجة.

مرحلة المعالجة اللاحقة لنتيجة الإخراج

نتيجة إخراج TensorFlow هي درجة تشابه similarity score لكل تسمية، وتبدو كما يلي:

```
[0.0, 0.2, 0.9, 0.0]
```

من الصعب بعض الشيء معرفة القيمة التي تشير إلى أي تسمية إلا إذا قمت بإنشاء النموذج.

أضف الطريقة التالية إلى lib/classifier/classifier.dart:

```
List<ClassifierCategory> _postProcessOutput(TensorBuffer outputBuffer) {
  // #1
  final probabilityProcessor = TensorProcessorBuilder().build();

  probabilityProcessor.process(outputBuffer);

  // #2
  final labelledResult = TensorLabel.fromList(_labels, outputBuffer);
```

```
  // #3
  final categoryList = <ClassifierCategory>[];
  labelledResult.getMapWithFloatValue().forEach((key, value) {
   final category = ClassifierCategory(key, value);
   categoryList.add(category);
   debugPrint('label: ${category.label}, score: ${category.score}');
  });

  // #4
  categoryList.sort((a, b) => (b.score > a.score ? 1 : -1));

  return categoryList;
 }
```

إليك منطق طريقة المعالجة اللاحقة الجديدة:

1. قم بإنشاء مثيل TensorProcessorBuilder لتحليل الإخراج ومعالجته.
2. قم بتعيين قيم الإخراج إلى التسميات الخاصة بك.
3. قم ببناء مثيلات الفئة باستخدام قائمة التصنيفات ـ سجلات score – label.
4. قم بفرز القائمة لوضع النتيجة الأكثر احتماليةفي الأعلى.
5. رائع، كل ما عليك الآن هو استدعاء (postProcessOutput_ للتنبؤ.

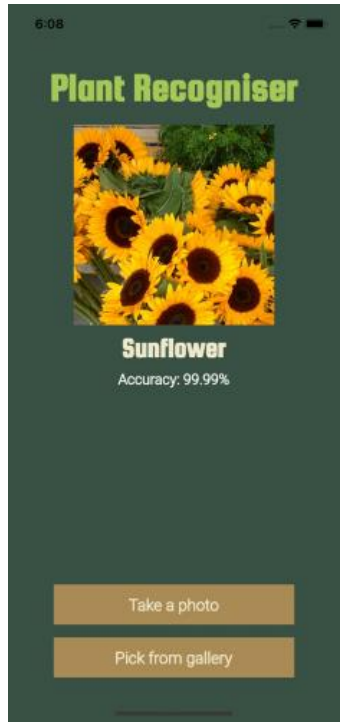
قم بتحديث (...)predict بحيث يبدو كما يلي:

```
ClassifierCategory predict(Image image) {
 // Load the image and convert it to TensorImage for TensorFlow Input
 final inputImage = _preProcessInput(image);

 // Define the output buffer
 final outputBuffer = TensorBuffer.createFixedSize(
  _model.outputShape,
   model.outputType,
 );

 // Run inference
 _model.interpreter.run(inputImage.buffer, outputBuffer.buffer);

 // Post Process the outputBuffer
 final resultCategories = _postProcessOutput(outputBuffer);
 final topResult = resultCategories.first;

 debugPrint('Top category: $topResult');

 return topResult;
}
```

لقد قمت بتطبيق طريقة ما بعد المعالجة الجديدةفي مخرجات TensorFlow، حتى تحصل على النتيجة الأولى والأكثر قيمة مرة أخرى.

بناء وتنفيذ المشروع.

قم بتحميل صورة وشاهدها تتنبأ بالنبات بشكل صحيح:



تهانينا! لقد كانت تلك رحلة جيدة.

بعد ذلك، ستتعرف على كيفية عمل المصنف Classifier للحصول على هذه النتيجة.

استخدام المصنف

الآن بعد أن تم إنشاؤه، ربما ترغبفي فهم كيفية استخدام هذا التطبيق لـ Classifier لتحديد اسم النبات وعرض النتائج.

تم بالفعل تنفيذ جميع التعليمات البرمجية من هذا القسمفي المشروع المبدئي، لذا اقرأ واستمتع فقط!

اختيار صورة من الجهاز

يحتاج جهازك إلى صورة لتحليلها، وتحتاج إلى السماح للمستخدمين بالتقاط صورة التقطوها إما من الكاميرا أو ألبوم الصور.

هذه هي الطريقة التي تفعل بها ذلك:

```
void _onPickPhoto(ImageSource source) async {
  // #1
  final pickedFile = await picker.pickImage(source: source);
```

```
  // #2
  if (pickedFile == null) {
   return;
  }

  // #3
  final imageFile = File(pickedFile.path);

  // #4
  setState(() {
   _selectedImageFile = imageFile;
  });


 }
```

وإليك كيفية عمل الكود أعلاه:

1. اختيار صورة من مصدر الصورة، إما الكاميرا أو ألبوم الصور.
2. تنفيذ المعالجةفي حالة قرر المستخدم الإلغاء.
3. لف مسار الملف المحدد بكائن File.
4. تغيير حالة _selectedImageFile لعرض الصورة.

تهيئة المصنف

إليك الكود المستخدم لتهيئة المصنف:

```
@override
void initState() {
 super.initState();
 // #1
 _loadClassifier();
}

Future _loadClassifier() async {
 debugPrint(
  'Start loading of Classifier with '
  'labels at $_labelsFileName, '
  'model at $_modelFileName',
 );

 // #2
 final classifier = await Classifier.loadWith(
  labelsFileName: _labelsFileName,
  modelFileName: _modelFileName,
 );

 // #3
 _classifier = classifier;
}
```

وإليك كيفية عمل ذلك:

1. قم بتشغيل التحميل غير المتزامن asynchronous loading لمثيل المصنف. لاحظ أن المشروع لا يحتوي على تعليمات برمجية كافية لمعالجة الأخطاء للإنتاج، لذلك قد يتعطل التطبيق إذا حدث خطأ ما.
2. قم باستدعاء loadWith(...) مع مسارات الملفات الخاصة بالتسميات وملفات النماذج.
3. احفظ المثيل في خاصية حالة عنصر واجهة المستخدم (ويدجت).

تحليل الصور باستخدام المصنف

انظر إلى الكود التالي في PlantRecogniser على lib/widget/plant_recogniser.dart.

```
void _analyzeImage(File image) async {

  // #1
  final image = img.decodeImage(image.readAsBytesSync())!;

  // #2
  final resultCategory = await _classifier.predict(image);

  // #3
  final result = resultCategory.score >= 0.8
    ?  ResultStatus.found
    : _ResultStatus.notFound;

  // #4
  setState(() {
    resultStatus = result;
   _plantLabel = resultCategory.label;
   _accuracy = resultCategory.score * 100;
  });
 }
```

المنطق أعلاه يعمل مثل هذا:

1. الحصول على الصورة من إدخال الملف.
2. استخدام Classifier للتنبؤ بأفضل فئة.
3. تحديد نتيجة التنبؤ. إذا كانت النتيجة منخفضة للغاية، أقل من 80%، فسيتم التعامل مع النتيجة على أنها غير موجودة.
4. تغيير حالة البيانات المسؤولة عن عرض النتيجة. تحويل النتيجة إلى نسبة مئوية عن طريق ضربها في 100.

قمت بعد ذلك باستدعاء هذه الطريقة في _onPickPhoto() بعد imageFile = File(pickedFile.path);:

```
void _onPickPhoto(ImageSource source) async {
  ...
  final imageFile = File(pickedFile.path);
  _analyzeImage(imageFile);
}
```

إليك التأثير عندما يتم ضبط كل شيء:

المصدر:

https://www.kodeco.com/37077010-tensorflow-lite-tutorial-for-flutter-image-classification

2) أنشاء تطبيق موبايل لتصنيف الصور باستخدام فلاتر وتنسرفلو لايت Build an Image Classification Mobile App using Flutter with TensorFlow Lite

(تطوير تطبيق جوال لتصنيف الصور باستخدام Flutter و TensorFlow Lite و Google's Teachable Machine)

في هذا البرنامج التعليمي ، سنعمل مع TensorFlow Lite، مع التركيز على تنفيذ تصنيف الصور لتصنيف الصور بين فئتين. سيكون التطبيق الذي سنقوم بإنشائه قادرًا على تصنيف ما إذا كانت الصورة المدخلة تحتوي على حصان أم إنسان.

حالات الاستخدام

يوفر لنا TensorFlow Lite نماذج مدربة مسبقًا ومحسنة لتحديد مئات فئات الكائنات، بمافي ذلك الأشخاص والأنشطة والحيوانات والنباتات والأماكن. باستخدام Teachable Machine من Google، يمكننا تطوير نموذجنا المخصص باستخدام بعض الصور الخاصة بنا. يتيح لك Teachable Machine 2.0 تدريب نماذج التعلم الآليفي المتصفح، دون أي كود تعلم آلي. يمكنك تدريب النماذج على الصور images والأصوات sounds والوضعيات poses، ومن ثم يمكنك حفظ النماذج التي قمت بتدريبها واستخدامهافي مشاريعك الخاصة.

الحزم المطلوبة

- TensorFlow Lite
- Image Picker
- Horse or Human Dataset

تحتوي المجموعة على 500 صورة لأنواع مختلفة من الخيولفي أوضاع مختلفة وفي مواقع مختلفة. ويحتوي أيضًا على 527 صورة معروضة لبشرفي أوضاع ومواقع مختلفة.

سيتم استخدام مجموعة البيانات هذه لإنشاء النموذج باستخدام Teachable Machine. أولاً، قم بتحميل مجموعة البيانات من هذا الرابط.

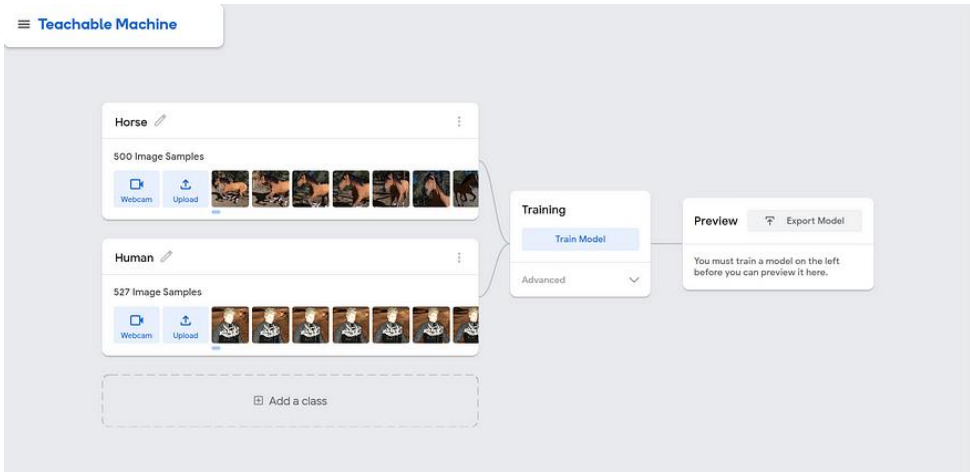
تدريب نموذج باستخدام Teachable Machine

قبل أن ندرب بياناتنا باستخدام Teachable Machine، نحتاج إلى جمع البيانات المطلوبة حتى تتعلم الآلة. تتوقف عملية جمع البيانات على المشكلة التي تحاول حلها. بالنسبة لمشكلات تصنيف

الصور، بعد جمع هذه البيانات، يجب ترتيبهافي فئات ذات صلة (أي تصنيفها). هذا هو المكان الذي تأتي فيه Teachable Machine.

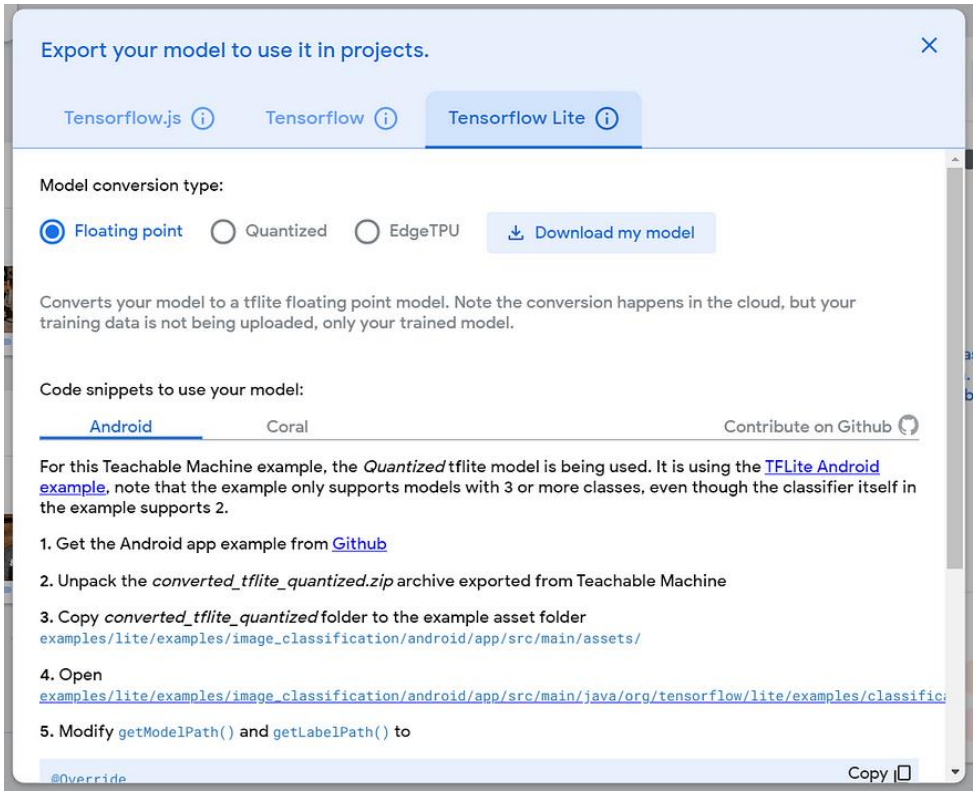
بعد أن قمنا بتنزيل البيانات المطلوبة من خلال Kaggle، نحتاج إلى تدريب نموذج TensorFlow Lite عليه.

بالنسبة لهذا التطبيق، نحتاج إلى تقسيم مجموعة البيانات الخاصة بنا إلى فئتين classes 2. (الخيول Horses والبشر Humans). تحتوي مجموعة البيانات التي تم تنزيلها من Kaggle على مجلدين مختلفين، اعتمادًا على الفئة. نحتاج إلى توفير مجلدات الصور هذه إلى Teachable Machine وتصنيفها على النحو التالي:



الخطوة الثانية هي تدريب النموذج، وهو ما يمكنك القيام به عن طريق النقر على زر تدريب النموذج Train Model. سوف يستغرق الأمر بضع دقائق لتدريب النموذج. يمكنك تجربة الإعدادات المتقدمة وإعطاء عدد مخصص من الفترات epochs (في الأساس، سيعتمد وقت تدريب النموذج على عدد الفترات التي تحددها).

بعد اكتمال التدريب، ستتمكن من النقر فوق زر تصدير النموذج Export Model.



قم بتنزيل نموذج TensorFlow Lite باستخدام نقطة تحويل النموذج Floating Point.

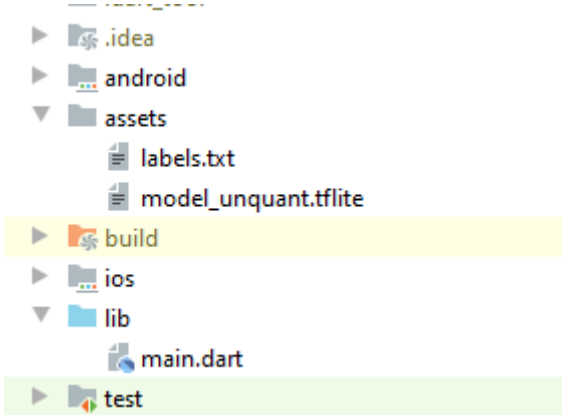
نظرة سريعة على جميع الخيارات المتاحة أمامك هنا:

- **Floating Point:** مؤهلة لوحدة المعالجة المركزية CPU ووحدة معالجة الرسومات GPU (أصغر بمقدار ضعفين ، وتسريع وحدة معالجة الرسومات GPU acceleration).
- **Quantized**: مؤهل لوحدة المعالجة المركزية (أصغر بمقدار اربع اضعاف 4x، تسريع بمقدار ضعفين الى ثلاثة 2x-3x).
- **EdgeTPU**: مؤهل لـ EdgeTPU ووحدات التحكم الدقيقة Microcontrollers (أصغر بمقدار اربع اضعاف 4x ، وتسريع بمقدار اربع اضعاف 4x).

تطبيق Flutter

الآن بعد أن حصلنا على النموذج المدرب، يمكننا تطوير تطبيق Flutter الخاص بنا لتصنيف الصورة على أنها تتضمن إما حصانًا أو إنسانًا. هيا بنا نبدأ.

أولاً، قم بإنشاء مجلد assets داخل مجلد المشروع وانسخ النموذج الذي تم تنزيله إلى هذا المجلد.



نظرًا لأن هذا تطبيق على مستوى المبتدئين، فيمكننا كتابة الكود داخل ملف main.dart دون إنشاء صفحة أخرى.

سنحتاج إلى بدء ثلاثة متغيرات لتنفيذ التطبيق بشكل صحيح. لتلقي الإخراج، سنحتاج إلى متغير القائمة List؛ للحصول على الصورة التي تم تحميلها، سنحتاج إلى متغير ملف File، ولإدارة الأخطاء سنحتاج إلى متغير منطقي Boolean:

```
List _outputs;
File _image;
bool _loading = false;
```

وبما أن النموذج سيعمل أيضًا دون الاتصال بالإنترنت (إحدى فوائد التعلم الآلي على الجهاز –on device machine learning)، فإننا نحتاج إلى تحميل النموذج عند تشغيل التطبيق:

```
loadModel() async {
await Tflite.loadModel(
model: "assets/model_unquant.tflite",
labels: "assets/labels.txt",
);
}
```

باستخدام الدالة أعلاه، يمكننا تحميل النموذج. لتشغيل هذه الدالة عند بدء تشغيل التطبيق، يمكننا استدعاء هذه الدالة داخل طريقة initState():

```
pickImage() async {
var image = await ImagePicker.pickImage(source: ImageSource.gallery);
if (image == null) return null;
setState(() {
_loading = true;
_image = image;
});
classifyImage(image);
}
```

باستخدام الدالة المذكورة أعلاه، يمكننا استخدام ImagePicker لاختيار صورة من المعرض. يمكننا بعد ذلك تمرير هذه الصورة إلى الدالة classifyImage() لتشغيلها من خلال نموذج التصنيف:

```
classifyImage(File image) async {
var output = await Tflite.runModelOnImage(
path: image.path,
numResults: 2,
threshold: 0.5,
imageMean: 127.5,
imageStd: 127.5,
);
setState(() {
_loading = false;
_outputs = output;
});
}
```

الآن بعد أن حددنا الدوال المطلوبة، يمكننا تطوير واجهة المستخدم حتى يتمكن تطبيقنا من عرض هذه النتائج للمستخدم:

```
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
backgroundColor: Colors.red,
title: Text('Horse/Human Detector'),
),
body: _loading
? Container(
alignment: Alignment.center,
child: CircularProgressIndicator(),
)
: Container(
width: MediaQuery.of(context).size.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
_image == null ? Container() : Image.file(_image),
SizedBox(
height: 20,
),
_outputs != null
? Text(
"${_outputs[0]["label"]}",
style: TextStyle(
color: Colors.black,
fontSize: 20.0,
background: Paint()..color = Colors.white,
),
)
: Container()
],
),
),
floatingActionButton: FloatingActionButton(
onPressed: pickImage,
backgroundColor: Colors.red,
child: Icon(Icons.image),
),
);
}
```

النتائج

الآن بعد أن قمنا بتنفيذ كود تطبيق Flutter، فلنلقِ نظرة على مخرجات التطبيق عندما يكون قيد التشغيل:

الكود الكامل

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:tflite/tflite.dart';

void main() => runApp(MaterialApp(
  home: MyApp(),
));

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}
```

```
class _MyAppState extends State<MyApp> {
  List _outputs;
  File _image;
  bool _loading = false;

  @override
  void initState() {
    super.initState();
    _loading = true;

    loadModel().then((value) {
      setState(() {
        _loading = false;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.red,
        title: Text('Horse/Human Detector'),
      ),
      body: _loading
          ? Container(
        alignment: Alignment.center,
        child: CircularProgressIndicator(),
      )
          : Container(
        width: MediaQuery.of(context).size.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            _image == null ? Container() : Image.file(_image),
            SizedBox(
              height: 20,
            ),
            _outputs != null
                ? Text(
              "${_outputs[0]["label"]}",
              style: TextStyle(
                color: Colors.black,
                fontSize: 20.0,
                background: Paint()..color = Colors.white,
              ),
            )
                : Container()
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: pickImage,
        backgroundColor: Colors.red,
        child: Icon(Icons.image),
      ),
    );
  }

  pickImage() async {
```

```
    var image = await ImagePicker.pickImage(source: ImageSource.gallery);
    if (image == null) return null;
    setState(() {
      _loading = true;
      _image = image;
    });
    classifyImage(image);
  }

  classifyImage(File image) async {
    var output = await Tflite.runModelOnImage(
      path: image.path,
      numResults: 2,
      threshold: 0.5,
      imageMean: 127.5,
      imageStd: 127.5,
    );
    setState(() {
      _loading = false;
      _outputs = output;
    });
  }

  loadModel() async {
    await Tflite.loadModel(
      model: "assets/model_unquant.tflite",
      labels: "assets/labels.txt",
    );
  }

  @override
  void dispose() {
    Tflite.close();
    super.dispose();
  }
}
```

الكود المصدري

https://github.com/ravindu9701/Image-Classification-Flutter-TensorFlow-Lite?source=post_page-----f7de39598e0c----------------------------

الاستنتاج

بمجرد أن تتقن الأمر، يمكنك معرفة مدى سهولة استخدام TensorFlow Lite مع Flutter لتطوير تطبيقات الهاتف المحمول القائمة على إثبات المفهوم للتعلم الآلي. تساعدنا Teachable machine من Google Creative Labs على إنشاء نماذج مخصصة بسهولة تامة وبدون خبرة كبيرةفي تعلم الآلة. علاوة على ذلك، يمكنك زيارة Kaggle وتنزيل مجموعات البيانات المختلفة لتطوير نماذج تصنيف مختلفة.

المصدر:

https://heartbeat.comet.ml/image-classification-on-mobile-with-flutter-tensorflow-lite-and-teachable-machine-f7de39598e0c

3) أنشاء تطبيق موبايل لتصنيف الكلاب والقطط باستخدام فلاتر وتنسرفلو لايت Build a Cat-or-Dog Classification Mobile App using Flutter with TensorFlow Lite

(استخدام نموذج TensorFlow Lite للتصنيف المُدرب مسبقًا لإنشاء تطبيق Flutter الذي يعمل بالتعلم الآلي)

التعرف على الكائنات Object detection، وتصنيف الصور image classification، والتعرف على الإيماءات gesture recognition – تعد مهام الرؤية الحاسوبية computer vision هذه كلها موضوعات ساخنةفي مشهد التعلم الآلي اليوم. هناك العديد من التطبيقات اليوم التي تستفيد من هذه التقنيات لتوفير حلول فعالة ومحسنة. وبشكل متزايد، تجد هذه التقنيات طريقها إلى تطبيقات الهاتف المحمول.

يهدف هذا البرنامج التعليمي إلى تقديم أحد هذه التطبيقات التوضيحية، باستخدام مكتبة التعلم الآلي TensorFlowفي مشروع Flutter لإجراء تصنيف ثنائي للصور binary image classification– القطط مقابل الكلاب cats vs dogs، وهي حالة استخدام أساسية.

للقيام بذلك، سنحتاج إلى نموذج تصنيف مُدرب مسبقًا pre-trained classification model ومخصص للاستخدام على الأجهزة المحمولة. إذا كنت تفضل ذلك، يمكنك أيضًا تدريب النموذج الخاص بك باستخدام Teachable Machine، وهي خدمة بناء نموذج بدون تعليمات برمجية تقدمها TensorFlow.

سيساعدنا استخدام مكتبة TensorFlow Liteفي تحميل نموذج تصنيف الصور على الهاتف المحمول وتطبيقه. تصنيف الصور هو مهمة رؤية حاسوبية تعمل على تحديد وتصنيف العناصر المختلفة للصور و/أو مقاطع الفيديو. يتم تدريب نماذج تصنيف الصور على التقاط صورة كمدخل وإخراج واحد أو أكثر من التصنيفات التي تصف الصورة.

الفكرة الرئيسية هي الاستفادة من البرنامج المساعد TensorFlow Lite وتصنيف صورة حيوان وتصنيف ما إذا كان كلبًا أو قطة. على طول الطريق، سنستفيد أيضًا من مكتبة Image Picker لجلب الصور من معرض الجهاز أو وحدة التخزين. ستكون العملية الرئيسية هي تحميل نموذج القط/الكلب المدرب مسبقًا باستخدام مكتبة TensorFlow Lite وتصنيف صورة حيوان الاختبار بناءً عليها.

اذا هيا بنا نبدأ!

إنشاء مشروع Flutter جديد

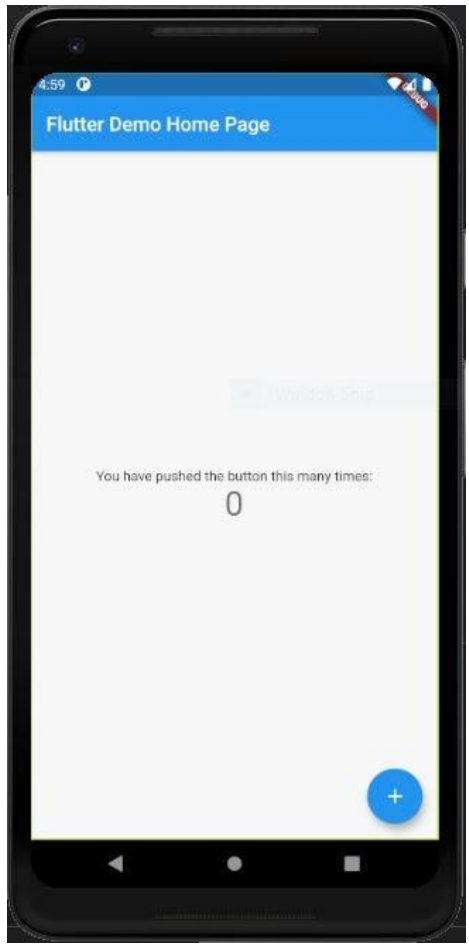
أولاً، نحتاج إلى إنشاء مشروع Flutter جديد. لذلك، تأكد من تثبيت Flutter SDK والمتطلبات الأخرى المتعلقة بتطوير تطبيق Flutter بشكل صحيح. إذا تم إعداد كل شيء بشكل صحيح، فمن أجل إنشاء مشروع، يمكننا ببساطة تشغيل الأمر التالي في الدليل المحلي المطلوب:

```
flutter create catDogIdentifier
```

بعد إعداد المشروع، يمكننا التنقل داخل دليل المشروع وتنفيذ الأمر التالي في الوحدة الطرفية terminal لتشغيل المشروع إما في محاكي متاح أو في جهاز فعلي:

```
flutter run
```

بعد البناء بنجاح، سنحصل على النتيجة التالية في شاشة المحاكي:



إنشاء عرض الصورة على الشاشة

سنقوم هنا بتنفيذ واجهة المستخدم لجلب صورة من مكتبة الجهاز وعرضها على شاشة التطبيق. لجلب الصورة من المعرض، سنستخدم مكتبة Image Picker. توفر هذه المكتبة وحدات لجلب مصادر الصور والفيديو من كاميرا الجهاز والمعرض وما إلى ذلك.

أولاً، نحتاج إلى تثبيت مكتبة image_picker. للقيام بذلك، نحتاج إلى نسخ النص الموجودفي مقتطف التعليمات البرمجية التالي ولصقهفي ملف pubspec.yaml الخاص بمشروعنا:

```
image_picker: ^0.6.7+14
```

الآن، نحن بحاجة إلى استيراد الحزم اللازمةفي ملف main.dart لمشروعنا:

```
import 'dart:io';import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
```

في ملف main.dart، سيكون لدينا فئة عناصر ذات الحالة (stateful widget class) MyHomePage. في كائن الفئة هذا، نحتاج إلى تهيئة ثابت constant لتخزين ملف الصورة بمجرد جلبه. هنا، سنقوم بذلك باستخدام متغير نوع الملف imageFile_ :

الآن، سنقوم بتنفيذ واجهة المستخدم، والتي ستمكن المستخدمين من اختيار الصورة وعرضها. ستحتوي واجهة المستخدم على قسم لعرض الصور وزر يسمح للمستخدمين باختيار الصورة من المعرض. يتم توفير قالب واجهة المستخدم الشاملفي مقتطف الكود أدناه:

```
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Cat Dog Identifier")
),
body: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
color: Colors.black12,
offset: Offset(2, 2),
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
Image.file(_imageFile) :
```

```
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
RaisedButton(
onPressed: (){
},
child: Icon(Icons.camera)
),
],
),
),
);
}
```

هنا، استخدمنا عنصر واجهة مستخدم الحاوية Container بنمط يشبه البطاقة Card لعرض الصورة. لقد استخدمنا العرض الشرطي conditional rendering لعرض صورة العنصر النائب حتى يتم تحديد الصورة الفعلية وتحميلها على الشاشة. لقد استخدمنا أيضًا عنصر واجهة المستخدم RaisedButton لعرض زر أسفل قسم عرض الصورة مباشرةً.

وبالتالي، يجب أن نحصل على النتيجة كما هو موضح في لقطة شاشة المحاكي أدناه:

دالة جلب وعرض الصورة

بعد ذلك، سنقوم بتنفيذ دالة تمكن المستخدمين من فتح المعرض، وتحديد صورة، ثم عرض الصورة في قسم عرض الصورة. يتم توفير التنفيذ الشامل للدالةفي مقتطف الكود أدناه:

```
Future selectImage() async {
final picker = ImagePicker();
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
setState(() {
if (image != null) {
_imageFile = File(image.path);
} else {
print('No image selected.');
}
});
}
```

هنا، قمنا بتهيئة مثيل ImagePicker واستخدمنا طريقة getImage التي يوفرها لجلب الصورة من المعرض إلى متغير الصورة. بعد ذلك، قمنا بتعيين حالة _imageFile على نتيجة الصورة التي تم جلبها باستخدام طريقة setState. سيؤدي هذا إلى إعادة عرض طريقة البناء الرئيسية وإظهار الصورة على الشاشة.

بعد ذلك، نحتاج إلى استدعاء دالة SelectImageفي خاصية onPressed لعنصر واجهة RaisedButton، كما هو موضحفي مقتطف الكود أدناه:

```
RaisedButton(
onPressed: (){
selectImage();
},
child: Icon(Icons.camera)
),
```

وبالتالي، سوف نحصل على النتيجة كما هو موضحفي لقطة شاشة المحاكي أدناه:

كما نرى، بمجرد تحديد الصورة من المعرض، يتم عرض الصورة المحددة على الشاشة بدلاً من صورة العنصر النائب.

أداء تصنيف الصور باستخدام TensorFlow Lite

حان الوقت الآن لتكوين مسار تصنيف صور القطط والكلاب. تذكر أن هدفنا هو تصنيف صورة معينة لحيوان على أنها قطة أو كلب أو كلب أو قطة. ولهذا، سنستخدم نموذجًا تم تدريبه باستخدام Teachable Machine من TensorFlow.

إذاكنت ترغبفي ذلك، يمكنك أيضًا تجربة التدريب على النموذج الخاص بك باستخدام Teachable Machine. يقدم هذا النموذج الذي نستخدمهفي هذا البرنامج التعليمي صورًا مدربة للقطط والكلاب من سلالات مختلفة بالإضافة إلى علامات التصنيف.

بعد التحميل سنحصل على ملفين:

- catdog_model.tflite
- cat_dog_labels.txt

يمكن لملف التصنيفات هنا التمييز بين القطة والكلب فقط. يتم ذلك لأغراض الاختبار السريع. يمكنك إضافة المزيد من التصنيفات لتحديد السلالات والحيوانات الأخرى وما إلى ذلك.

نحتاج إلى نقل الملفين المقدمين إلى المجلد assets/.في دليل المشروع الرئيسي.

ثم نحتاج إلى تمكين الوصول إلى ملفات assetsفي pubspec.yaml:

```
assets:
- assets/cat_dog_labels.txt
- assets/catdog_model.tflite
```

تثبيت TensorFlow Lite

سنقوم هنا بتثبيت حزمة TensorFlow Lite. إنه مكون إضافي لـ Flutter للوصول إلى واجهات برمجة تطبيقات TensorFlow Lite. تدعم هذه المكتبة تصنيف الصور واكتشاف الكائنات وPix2Pix وDeeplab وPoseNet على نظامي iOS وAndroid.

من أجل تثبيت البرنامج المساعد، نحتاج إلى إضافة السطر التالي إلى ملف pubspec.yaml لمشروعنا:

```
tflite: ^1.1.1
```

بالنسبة لنظام Android، نحتاج إلى إضافة الإعداد التالي إلى كائن android الخاص بملف :./android/app/build.gradle

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

هنا، نحتاج إلى التحقق لمعرفة ما إذا كان التطبيق قد تم إنشاؤه بشكل صحيح عن طريق تنفيذ أمر Flutter Run.

في حالة حدوث خطأ، قد نحتاج إلى زيادة الحد الأدنى لإصدار SDK إلى ≤19في ملف ./android/app/build.gradle. حتى يعمل المكون الإضافي tflite.

```
minSdkVersion 19
```

بمجرد إنشاء التطبيق بشكل صحيح، سنكون جاهزين لاستخدام حزمة TensorFlow Liteفي مشروع Flutter الخاص بنا.

استخدام TensorFlow Lite لتصنيف الصور

أولاً، نحتاج إلى استيراد الحزمة إلى ملف main.dart الخاص بنا، كما هو موضحفي مقتطف الكود أدناه:

```
import 'package:tflite/tflite.dart';
```

تحميل النموذج

الآن، نحن بحاجة إلى تحميل ملفات النموذجفي التطبيق. للقيام بذلك، سنقوم بتكوين دالة تسمى LoadImageModel. بعد ذلك، من خلال الاستفادة من طريقة loadModel التي يوفرها مثيل Tflite، سنقوم بتحميل ملفات النماذج الخاصة بنافي مجلد assets إلى تطبيقنا. نحتاج إلى تعيين معلمة النموذج model والتسميات labels داخل طريقة LoadModel، كما هو موضحفي مقتطف الكود أدناه:

```
Future loadImageModel() async {
Tflite.close();
String result;
result = await Tflite.loadModel(
model: "assets/catdog_model.tflite",
labels: "assets/cat_dog_labels.txt",
);
print(result);
}
```

بعد ذلك، نحتاج إلى استدعاء الدالة داخل طريقة initState بحيث يتم تشغيل الدالة بمجرد دخولنا إلى الشاشة:

```
@override
void initState() {
super.initState();
loadEmojiModel();
}
```

تنفيذ تصنيف الصور

الآن، سنقوم بكتابة التعليمات البرمجية لتنفيذ تصنيف الصور فعليًا. أولاً، نحتاج إلى تهيئة متغير لتخزين نتيجة التصنيف:

```
List _classifiedResult;
```

سيقوم متغير نوع قائمة _classifiedResult بتخزين نتيجة تصنيف النموذج.

بعد ذلك، نحتاج إلى انشاء دالة تسمى classifyImage تأخذ ملف الصورة image كمعلمة. يتم توفير التنفيذ الشامل للدالةفي مقتطف الكود أدناه:

```
Future classifyImage(image) async {
_classifiedResult = null;
// Run tensorflowlite image classification model on the image
print("classification start $image");
final List result = await Tflite.runModelOnImage(
path: image.path,
```

```
numResults: 6,
threshold: 0.05,
imageMean: 127.5,
imageStd: 127.5,
);
print("classification done");
setState(() {
if (image != null) {
_imageFile = File(image.path);
_classifiedResult = result;
} else {
print('No image selected.');
}
});
}
```

هنا، استخدمنا طريقة runModelOnImage التي يوفرها مثيل Tflite لتصنيف الصورة المحددة. كمعلمات، قمنا بتمرير مسار الصورة وكمية النتيجة وعتبة التصنيف والتكوينات الاختيارية الأخرى لتصنيف أفضل. بعد نجاح التصنيف، قمنا بتعيين النتيجة إلى قائمة _classfiedResult .

نحن الآن بحاجة إلى استدعاء الدالة داخل دالة SelectImage وتمرير ملف image كمعلمة:

```
Future selectImage() async {
final picker = ImagePicker();
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
classifyImage(image);
}
```

سيسمح لنا ذلك بضبط الصورة على عرض الصورة بالإضافة إلى تصنيف الصورة بمجرد تحديد صورة من المعرض.

الآن، نحن بحاجة إلى تكوين قالب واجهة المستخدم لعرض نتائج التصنيف. سنعرض نتيجة التصنيف بأسلوب البطاقة كقائمة أسفل عنصر واجهة المستخدم RaisedButton مباشرةً.

يتم توفير تنفيذ واجهة المستخدم الشاملة للشاشةفي مقتطف الكود أدناه:

```
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Cat Dog Identifier"),
),
body: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
```

```
color: Colors.black12,
offset: Offset(2, 2),
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
Image.file(_imageFile) :
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
RaisedButton(
onPressed: (){
selectImage();
},
child: Icon(Icons.camera)
),
SizedBox(height: 20),
SingleChildScrollView(
child: Column(
children: _classifiedResult != null
? _classifiedResult.map((result) {
return Card(
elevation: 0.0,
color: Colors.lightBlue,
child: Container(
width: 300,
margin: EdgeInsets.all(10),
child: Center(
child: Text(
"${result["label"]} : ${(result["confidence"] * 100).toStringAsFixed(1)}%",
style: TextStyle(
color: Colors.black,
fontSize: 18.0,
fontWeight: FontWeight.bold),
),
),
),
);
}).toList()
: [],
),
),
],
),
),
);
}
```

هنا، أسفل عنصر واجهة المستخدم RaisedButton مباشرةً، قمنا بتطبيق عنصر واجهة المستخدم SingleChildScrollView بحيث يكون المحتوى الموجود بداخله قابلاً للتمرير. لقد استخدمنا أيضًا عنصر واجهة مستخدم العمود Column لإدراج عناصر واجهة المستخدم بداخله أفقيًا.

داخل عنصر واجهة مستخدم العمود، قمنا بتعيين نتيجة التصنيف باستخدام طريقة map وعرضنا النتيجة بتنسيق النسبة المئوية داخل عنصر واجهة مستخدم Card.

وبالتالي سنحصل على النتيجة كما هو موضح في العرض التوضيحي أدناه:

يمكننا أن نرى أنه بمجرد اختيار الصورة من المعرض، يتم عرض نتيجة التصنيف على الشاشة أيضًا ـ وتسمح طبيعة النموذج الموجود على الجهاز بحدوث التنبؤاتفي الوقت الفعلي.

وهذا كل شيء! لقد نجحنافي تنفيذ مُصنف القط أو الكلب الخاص بنافي تطبيق Flutter باستخدام TensorFlow Lite.

الاستنتاج

في هذا البرنامج التعليمي، تمكنا من إنشاء تطبيق تجريبي يصنف بشكل صحيح صور القطط والكلاب.

تم تبسيط العملية برمتها وجعلها سهلة نظرًا لتوفر مكتبة TensorFlow Lite لـ Flutter، بالإضافة إلى نموذج تم تدريبه مسبقًا. تم توفير ملفات النماذج لهذا البرنامج التعليمي، ولكن يمكنك إنشاء

نماذجك المدربة باستخدام Teachable Machine، وهي خدمة بدون تعليمات برمجية تقدمها TensorFlow.

الآن، يمكن أن يتمثل التحديفي تدريب النموذج الخاص بك وتحميلهفي تطبيق Flutter، ثم تطبيق النموذج لتصنيف الصور. مكتبة TensorFlow Lite قادرة على إجراء عمليات تعلم الآلة الأخرى مثل اكتشاف الكائنات وتقدير الوضعية واكتشاف الإيماءات وما إلى ذلك.

جميع التعليمات البرمجية متاحة على GitHub.

المصدر:

https://heartbeat.comet.ml/build-a-cat-or-dog-classification-flutter-app-tensorflow-lite-7b57223d7754

4) أنشاء تطبيق موبايل لتصنيف سلالات الكلاب باستخدام باستخدام فلاتر وتنسرفلو لايت Build a Dog Breed Classification Mobile App using Flutter with TensorFlow Lite

أصبح للتعلم الآلي والذكاء الاصطناعي تأثير متزايد على تكنولوجيا الهاتف المحمولفي الوقت الحاضر. أصبحت الأجهزة أكثر قدرة على استخدام الذكاء الاصطناعي، ويتم دمج التعلم الآلي وطرق الذكاء الاصطناعيفي تطبيقات الهاتف المحمول لتحسين تجارب المستخدم وتعزيزها.

في هذا البرنامج التعليمي، سنقوم بتطبيق أساليب التعلم الآلي التي توفرها مكتبة TensorFlow Lite لغرض تصنيف الصورفي تطبيق Flutter. تميل Flutter ـ كما قدمتها واعترفت بها Google ـ إلى إنتاج ودعم المكتبات المتعلقة بالتعلم الآلي.

ستساعدنا مكتبة TensorFlow lite هذهفي تحميل نموذجنا وكذلك تطبيق النموذج لتصنيف الصور. تصنيف الصور هو مهمة رؤية حاسوبية تسمح لنا بالتعرف على الكائنات والجوانب المختلفة للصورة أو الفيديو وتحديدها.

لأغراض الاختبار، سنستخدم نموذجًا مُدربًا مسبقًا مقدمًا من TensorFlow لتصنيف صور الكلاب وبناءً على سلالاتها. تتمثل الفكرةفي الاستفادة من مكتبة Image Picker لجلب الصورة من معرض الجهاز ثم تطبيق تصنيف الصورة عليها لتحديد سلالة الكلاب الموجودةفي الصورة.

اذا هيا بنا نبدأ!

إنشاء مشروع Flutter جديد

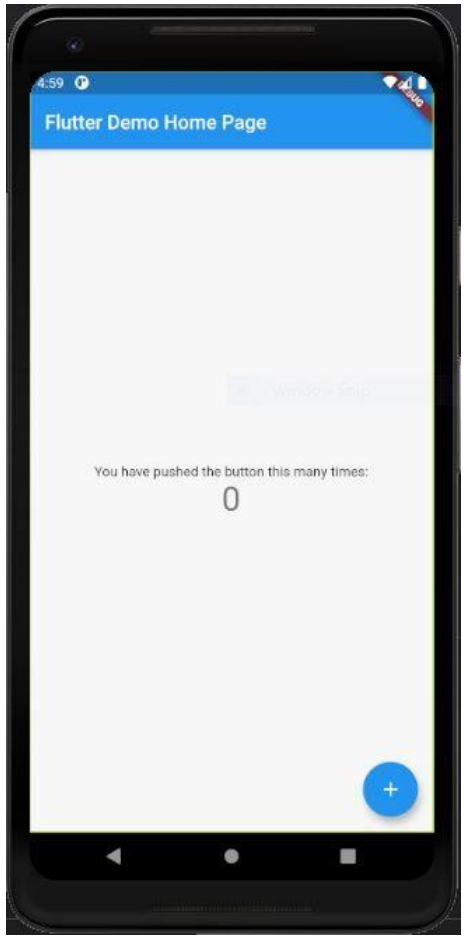
أولاً، نحتاج إلى إنشاء مشروع Flutter جديد. لذلك، تأكد من تثبيت Flutter SDK والمتطلبات الأخرى ذات الصلة بتطوير تطبيق Flutter بشكل صحيح. إذا تم إعداد كل شيء بشكل صحيح، فمن أجل إنشاء مشروع، يمكننا ببساطة تشغيل الأمر التاليفي الدليل المحلي المطلوب:

```
flutter create EstimatingDogBreed
```

بعد إعداد المشروع، يمكننا التنقل داخل دليل المشروع وتنفيذ الأمر التاليفي الوحدة الطرفية لتنفيذ المشروع إمافي محاكي متاح أو على جهاز فعلي:

```
flutter run
```

بعد نجاح البناء، سنحصل على النتيجة التالية على شاشة المحاكي:



إنشاء عرض الصورة على الشاشة

سنقوم هنا بتنفيذ واجهة المستخدم لجلب صورة من مكتبة الجهاز وعرضها على شاشة التطبيق. لجلب الصورة من المعرض، سنستخدم مكتبة Image Picker. توفر هذه المكتبة وحدات لجلب الصور ومقاطع الفيديو من كاميرا الجهاز أو المعرض أو ما إلى ذلك.

أولاً، نحتاج إلى تثبيت مكتبة image_picker. للقيام بذلك، نحتاج إلى نسخ النص الموجودفي مقتطف التعليمات البرمجية التالي ولصقهفي ملف pubspec.yaml الخاص بمشروعنا:

```
image_picker: ^0.6.7+14
```

بعد ذلك، نحتاج إلى استيراد الحزم الضروريةفي ملف main.dart الخاص بمشروعنا:

```
import 'dart:io';import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
```

في ملف main.dart، سيكون لدينا فئة عناصر واجهة المستخدم ذات الحالة MyHomePage.في كائن الفئة هذا، نحتاج إلى تهيئة ثابت لتخزين ملف الصورة بمجرد جلبه. هنا، سنفعل ذلكفي متغير نوع الملف imageFile_ :

```
File _imageFile;
```

نحن الآن بحاجة إلى تنفيذ واجهة المستخدم، والتي ستمكن المستخدمين من اختيار الصورة وعرضها. ستحتوي واجهة المستخدم على قسم لعرض الصور وزر يسمح للمستخدمين باختيار صورة من المعرض. يتم توفير قالب واجهة المستخدم الشاملفي مقتطف الكود أدناه:

```
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text(widget.title),
),
body: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
color: Colors.black12,
offset: Offset(2, 2),
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
```

```
Image.file(_imageFile) :
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
RaisedButton(
onPressed: (){},
child: Text('Select Image')
),
],
),
),
);
}
```

هنا، استخدمنا عنصر واجهة مستخدم الحاوية Container بنمط يشبه البطاقة لعرض الصورة. لقد استخدمنا العرض الشرطي لعرض صورة العنصر النائب حتى يتم تحديد الصورة الفعلية وتحميلها على الشاشة. لقد استخدمنا عنصر واجهة المستخدم RaisedButton لعرض زر أسفل قسم عرض الصورة مباشرةً.

وبالتالي، سنحصل على النتيجة كما هو موضح في لقطة شاشة المحاكي أدناه:

دالة جلب وعرض الصورة

سنقوم الآن بتنفيذ دالة تمكن المستخدمين من فتح المعرض، وتحديد صورة، ثم عرض الصورة في قسم عرض الصورة. يتم توفير التنفيذ الشامل للدالة في مقتطف الكود أدناه:

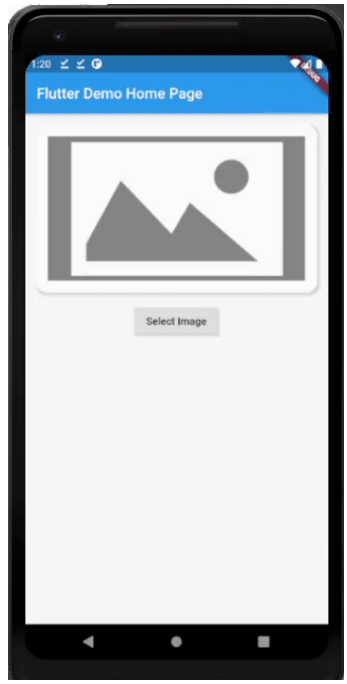
```
Future selectImage() async {
final picker = ImagePicker();
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
setState(() {
if (image != null) {
_imageFile = File(image.path);
} else {
print('No image selected.');
}
});
}
```

لقد قمنا هنا بتهيئة مثيل ImagePicker واستخدمنا طريقة getImage التي يوفرها لجلب الصورة من المعرض إلى متغير الصورة image. بعد ذلك، قمنا بتعيين حالة imageFile_ على نتيجة الصورة التي تم جلبها باستخدام طريقة setState. سيؤدي هذا إلى إعادة عرض طريقة build الرئيسية وإظهار الصورة على الشاشة.

الآن، نحن بحاجة إلى استدعاء دالة SelectImage في الخاصية onPressed لعنصر واجهة المستخدم RaisedButton، كما هو موضح في مقتطف الكود أدناه:

```
RaisedButton(
onPressed: (){
selectImage();
},
child: Text('Select Image')
),
```

يجب أن نحصل على النتيجة التالية:



كما نرى، بمجرد تحديد الصورة من المعرض، يتم عرضها على الشاشة بدلاً من صورة العنصر النائب.

إجراء تصنيف سلالات الكلاب باستخدام TensorFlow Lite

لقد حان الوقت لتكوين نموذج تصنيف الصور الخاص بنا. كما ذكرنا سابقًا، سنعمل مع نموذج "starter" مُدرب مسبقًا، والذي يمكننا تنزيله هنا. يتم توفير المعلومات الشاملة حول النموذجفي الوثائق نفسها.

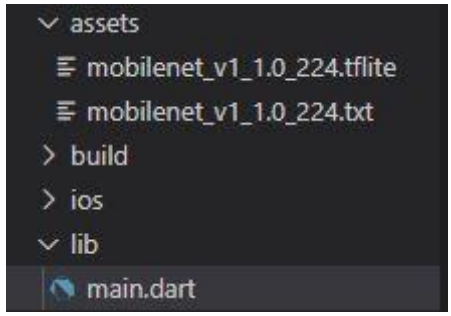
يقدم هذا النموذج صورًا مدربة للكلاب من سلالات مختلفة. ومن ثم، سنقوم بتصنيف صورة كلب لأغراض الاختبار. هناك أيضًا نماذج مخصصة للحيوانات الأخرى أيضًا، والتي يمكنك التحقق منهافي رابط النماذج المبدئية أعلاه.

ملاحظة: يمكننا أيضًا تدريب نموذجنا الخاص باستخدام أدوات مثل Teachable Machine، والتي تتيح لك إنشاء نماذج تصنيف مخصصة بسرعة.

بمجرد التنزيل، سنحصل على الملف المضغوط. سيحتوي الملف المضغوط على ملفين نموذجيين:

- mobilenet_v1_1.0_224.txt
- mobilenet_v1_1.0_224.tflite

انقل الملفين المقدمين إلى المجلد ./assets. في دليل المشروع الرئيسي:



ثم نحتاج إلى تمكين الوصول إلى ملفات ال assets في pubspec.yaml:

```
assets:
- assets/mobilenet_v1_1.0_224.txt
- assets/mobilenet_v1_1.0_224.tflite
```

تثبيت TensorFlow Lite

بعد ذلك، نحتاج إلى تثبيت حزمة TensorFlow Lite، وهي مكون إضافي لـ Flutter للوصول إلى واجهات برمجة تطبيقات TensorFlow Lite. تدعم هذه المكتبة تصنيف الصور واكتشاف الكائنات وPix2Pix وDeeplab وPoseNet لكل من منصات iOS وAndroid.

من أجل تثبيت الاضافة، نحتاج إلى إضافة السطر التالي إلى ملف pubspec.yaml لمشروعنا:

```
tflite: ^1.1.1
```

بالنسبة لنظام Android، نحتاج إلى إضافة الإعداد التالي إلى كائن Android الخاص بملف :./android/app/build.gradle

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

تحقق مرة واحدة لمعرفة ما إذا كان التطبيق قد تم إنشاؤه بشكل صحيح عن طريق تنفيذ أمر Flutter Run.

في حالة حدوث خطأ، قد نحتاج إلى زيادة الحد الأدنى لإصدار SDK إلى ≤19 حتى يعمل مكون TFLite الإضافي.

```
minSdkVersion 19
```

بمجرد إنشاء التطبيق بشكل صحيح، نكون جاهزين لتنفيذ نموذجنا.

تنفيذ نموذج التصنيف

أولاً، نحتاج إلى استيراد الحزمة إلى ملف main.dart الخاص بنا كما هو موضحفي مقتطف الكود أدناه:

```
import 'package:tflite/tflite.dart';
```

تحميل النموذج

بعد ذلك، نحتاج إلى تحميل ملفات النموذجفي التطبيق. للقيام بذلك، سنقوم بتكوين دالة تسمى LoadImageModal. بعد ذلك، من خلال الاستفادة من طريقة loadModel التي يوفرها مثيل Tflite، سنقوم بتحميل ملفات النماذج الخاصة بنافي مجلد assets إلى تطبيقنا. سنقوم بتعيين معلمة النموذج model والتسميات labels داخل طريقة LoadModel، كما هو موضحفي مقتطف الكود أدناه:

```
Future loadImageModel() async {
String result;
result = await Tflite.loadModel(
model: "assets/mobilenet_v1_1.0_224.tflite",
labels: "assets/mobilenet_v1_1.0_224.txt",
);
```

```
print("result: $result");
}
```

بعد ذلك، نحتاج إلى استدعاء الدالة داخل طريقة initState بحيث يتم تشغيل الدالة بمجرد دخولنا إلى الشاشة:

```
@override
void initState() {
super.initState();
loadImageModel();
}
```

تنفيذ تصنيف الصور

الآن، سنقوم بكتابة التعليمات البرمجية لتنفيذ التصنيف نفسه. أولاً، نحتاج إلى تهيئة متغير لتخزين نتيجة التصنيف كما هو موضح في مقتطف الكود أدناه:

```
List _classifiedResult;
```

سيقوم متغير نوع قائمة _classifiedResult بتخزين نتيجة التصنيف.

بعد ذلك، نحن بحاجة إلى انشاء دالة تسمى classifyImage التي تأخذ ملف image كمعلمة. يتم توفير التنفيذ الشامل للدالة في مقتطف الكود أدناه:

```
Future classifyImage(image) async {
_classifiedResult = null;
// Run tensorflowlite image classification model on the image
print("classification start $image");
final List result = await Tflite.runModelOnImage(
path: image.path,
numResults: 6,
threshold: 0.05,
imageMean: 127.5,
imageStd: 127.5,
);
print("classification done");
setState(() {
if (image != null) {
_imageFile = File(image.path);
_classifiedResult = result;
} else {
print('No image selected.');
}
});
}
```

هنا، استخدمنا طريقة runModelOnImage التي يوفرها مثيل Tflite لتصنيف الصورة المحددة. كمعلمات، قمنا بتمرير مسار الصورة وكمية النتيجة وعتبة التصنيف والتكوينات الاختيارية الأخرى لتصنيف أفضل. بعد تشغيل النموذج بنجاح، قمنا بتعيين النتيجة إلى قائمة _classfiedResult .

بعد ذلك، نحتاج إلى استدعاء الدالة داخل دالة SelectImage وتمرير ملف image كمعلمة، كما هو موضح في مقتطف الكود أدناه:

```
Future selectImage() async {
final picker = ImagePicker();
```

```
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
classifyImage(image);
}
```

سيسمح لنا ذلك بضبط الصورة على عرض الصورة، بالإضافة إلى تصنيف الصورة بمجرد تحديدها من المعرض.

نحتاج الآن إلى تكوين قالب واجهة المستخدم لعرض نتائج التصنيف. سنعرض نتيجة التصنيف بنمط البطاقة كقائمة أسفل عنصر واجهة المستخدم FlatButton مباشرةً.

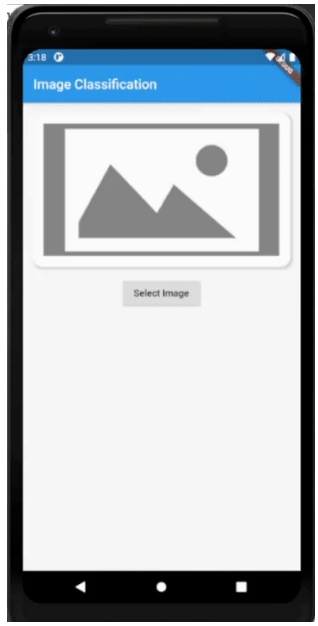
يتم توفير تنفيذ واجهة المستخدم الشاملة للشاشةفي مقتطف الكود أدناه:

```
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Image Classification"),
),
body: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
color: Colors.black12,
offset: Offset(2, 2),
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
Image.file(_imageFile) :
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
RaisedButton(
onPressed: (){
selectImage();
},
child: Text('Select Image')
),
SizedBox(height: 20),
SingleChildScrollView(
child: Column(
children: _classifiedResult != null ?
_classifiedResult.map((result) {
return Card(
elevation: 0.0,
color: Colors.lightBlue,
child: Container(
width: 300,
```

```
margin: EdgeInsets.all(10),
child: Center(
child: Text(
"${result["label"]} : ${(result["confidence"] * 100).toStringAsFixed(1)}%",
style: TextStyle(
color: Colors.black,
fontSize: 18.0,
fontWeight: FontWeight.bold),
),
),
),
);
}).toList()
: [],
),
),
],
),
),
);
}
```

هنا، أسفل عنصر واجهة المستخدم FlatButton مباشرةً، قمنا بتطبيق عنصر واجهة المستخدم SingleChildScrollView بحيث يكون المحتوى الموجود بداخله قابلاً للتمرير. بعد ذلك، استخدمنا عنصر واجهة مستخدم العمود Column لإدراج عناصر واجهة المستخدم بداخله أفقيًا. داخل عنصر واجهة مستخدم العمود، قمنا بتعيين نتيجة التصنيف باستخدام طريقة map وعرضنا النتيجة بتنسيق النسبة المئوية داخل عنصر واجهة مستخدم Card.

وبالتالي سنحصل على النتيجة كما هو موضح في العرض التوضيحي أدناه:



يمكننا أن نرى أنه بمجرد اختيار الصورة من المعرض، يتم عرض نتيجة التصنيف على الشاشة أيضًا.

وهذا كل شيء! لقد نجحنافي تنفيذ نموذج تصنيف سلالات الكلابفي تطبيق Flutter باستخدام TensorFlow Lite.

الاستنتاج

في هذا البرنامج التعليمي، تم تكليفنا بإجراء تصنيف لسلالات الكلاب على صورةفي تطبيق Flutter. إن توفر مكتبة TensorFlow Lite جعل العملية برمتها بسيطة نسبيًا وسهلة الفهم. يمكن أن تكون هذه خطوة أساسية نحو تعلم تطبيقات التعلم الآليفي تطبيق Flutter. يمكننا أيضًا تصنيف صور الحيوانات الأخرى باستخدام النماذج المعروضةفي النماذج المبدئية على TensorFlow.

يمكننا تصنيف صور الحيوانات الأخرى، أو استخدام النماذج الأخرى المتنوعة التي قمنا بتنزيلها من TensorFlow. التحدي الآن هو تدريب النموذج الخاص بك وتنفيذهفي تطبيق Flutter. مكتبة TensorFlow Lite قادرة على القيام بعمليات التعلم الآلي الأخرى مثل اكتشاف الكائنات وتقدير الوضعيات أيضًا.

الكود الكامل متاحفي GitHub repo.

المصدر:

https://heartbeat.comet.ml/dog-breed-classifier-on-mobile-with-flutter-and-tensorflow-lite-8bce10d29aa6

5) أنشاء تطبيق موبايل لاكتشاف قناع الوجه باستخدام فلاتر وتنسرفلو لايت Build a face Mask Detection Mobile App using Flutter with TensorFlow Lite

دعونا نتعلم كيفية إنشاء تطبيق Flutter يكتشف ما إذا كان الشخص يرتدي قناعًا أم لا على الكاميرا بشكل مباشر.

لقد أثرت أزمات كوفيد-19 بشكل كبير على سبل عيشنا. هناك العديد من الإرشادات التي قدمتها منظمة الصحة العالمية لمنع انتشار فيروس كورونا (COVID-19). هذا الفيروس خطير جدا ومعدي. في سيناريوهات اليوم، أصبح ارتداء الكمامة إلزامياًفي الأماكن العامة. وفي العديد من البلدان، يُفرض على الأشخاص أموال مقابل عدم ارتداء قناع. لذلك أصبح من واجبنا الأخلاقي ارتداء الكمامةفي الأماكن العامة، لكن بعض الأشخاص لا يتبعون الإرشادات. لا يستطيع النظام التعرف على الأشخاص الذين لا يرتدون قناعًا، لذلك أصبح من المهم بناء أداة للتعرف على الشخص سواء كان يرتدي قناعًا أم لا. بمساعدة التعلم الآلي والتعلم العميق، يمكننا بسهولة بناء نموذج وتدريبه باستخدام مجموعة بيانات لحل هذه المشكلة والمساعدةفي منع انتشار فيروس كوفيد-19.

في هذه البرنامج التعليمي، سنتعلم كيفية إنشاء تطبيق لكشف قناع الوجه Face Mask Detection مع Flutter باستخدام حزمة tflite لتحديد ما إذا كان الشخص يرتدي قناعًا أم لا.

تثبيت الحزم

لبناء هذا التطبيق سنحتاج إلى حزمتين:

- يتم استخدام camera للحصول على المخازن المؤقتة للصور المتدفقة.
- يتم استخدام tflite لتشغيل نموذجنا المدرب.

تكوين المشروع

- لأجهزة الأندرويد:

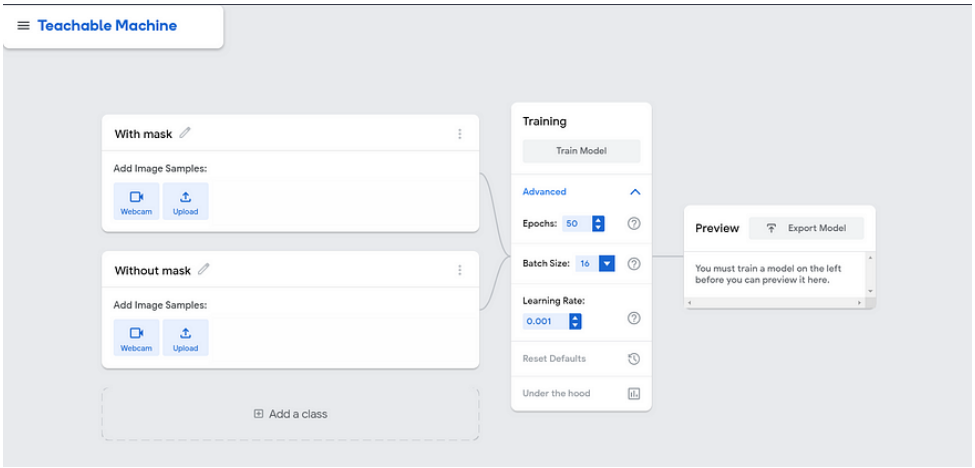
في android/app/build.gradle، أضف الإعداد التالي في كتلة android.

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

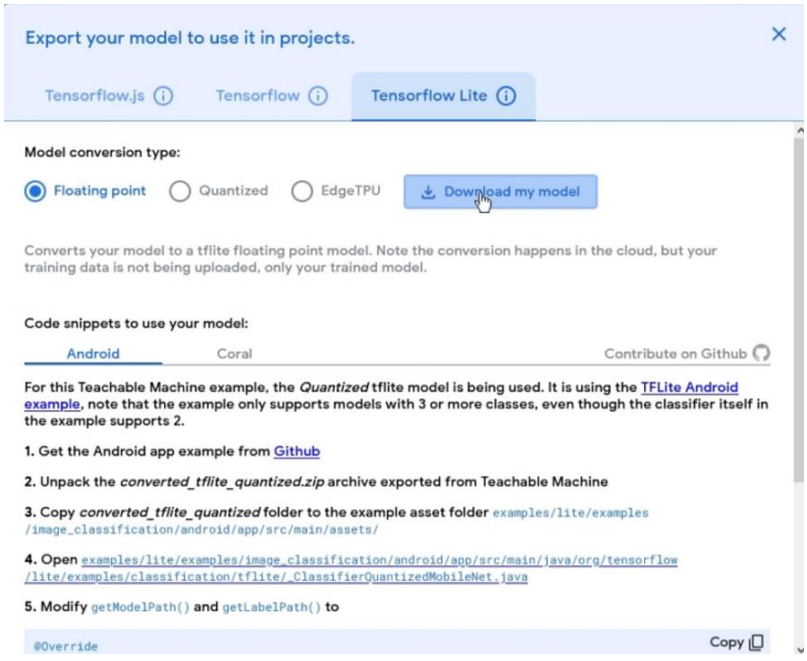
- تغيير minSdk الإصدار 21

تنزيل مجموعة بيانات التدريب وتدريب النموذج

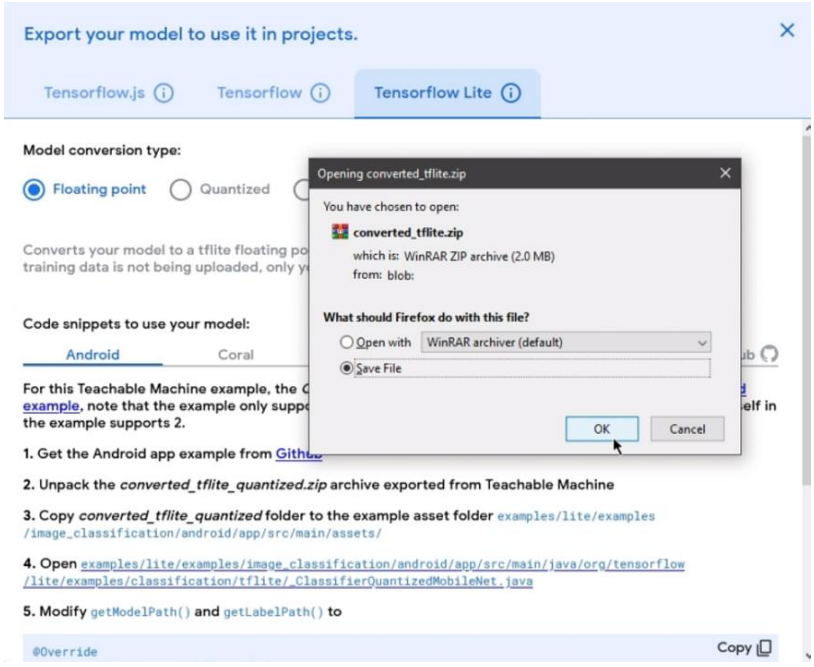
- لتنزيل مجموعة البيانات، قم بزيارة kaggle.com وابحث عن "الكشف عن قناع الوجه Face mask detection".
- قم بتنزيل مجموعة البيانات.
- لتدريب نموذجنا باستخدام مجموعة البيانات التالية، سنستخدم https://teachablemachine.withgoogle.com/لتدريب نموذجنا.
- انقر على البدء get started.



- قم بتحميل صور الأشخاص المقنعينفي فئة With mask وصور بدون قناعفي فئة Without mask.
- ثم انقر فوق تدريب نموذج Train Model، ولا تقم بتغيير الإعدادات.
- تصدير النموذج.
- انقر على Tensorflow Lite وقم بتنزيل النموذج.



- انقر فوق Ok.



- قم بتصدير ملف model.tflite وlable.txt وقم بتخزينهمافي مجلد assets في مشروعك.

لنبدأ بالبرمجة:

تهيئة الكاميرا

داخل طريقة main، قم بتهيئة الكاميرات المتاحة باستخدام availableCameras.

```
List<CameraDescription> cameras;

Future<void> main() async {
WidgetsFlutterBinding.ensureInitialized();
cameras = await availableCameras();
runApp(MyApp());
}
```

camera توفر لنا الحزمة الدعم لبث الصور المباشرة. قم أولاً بإنشاء كائن من CameraController. يأخذ CameraController وسيطتين CameraDescription وResolutionPreset. قم بتهيئة initialize وحدة التحكم بالكاميرا CameraController وبعد ذلك يمكننا بدء تدفق الصور باستخدام طريقة startImageStream. startImageStream توفر لنا الطريقة الصور، وسنعطي هذه الصور إلى CameraImage ثم سنقوم بتشغيل النموذج الخاص بنا.

```
CameraImage cameraImage;
CameraController cameraController;
String result = "";

initCamera() {
cameraController = CameraController(cameras[0], ResolutionPreset.medium);
cameraController.initialize().then((value) {
if (!mounted) return;
setState(() {
cameraController.startImageStream((imageStream) {
cameraImage = imageStream;
runModel();
});
});
});
}
```

تحميل النموذج

يوفر لنا Tflite طريقة LoadModel لتحميل نموذجنا. يستغرق الأمر مسار ملف نموذج قيمتين ومسار ملف التسميات.

```
loadModel() async {
await Tflite.loadModel(
model: "assets/model.tflite", labels: "assets/labels.txt");
}
```

تشغيل النموذج

في هذه الطريقة، سنقوم بتشغيل النموذج باستخدام Tflite. نحن هنا نستخدم البث المباشر للصورة لذا سيتعين علينا استخدام طريقة runModelOnFrame لتشغيل نموذجنا.

```
runModel() async {
if (cameraImage != null) {
```

```
var recognitions = await Tflite.runModelOnFrame(
bytesList: cameraImage.planes.map((plane) {
return plane.bytes;
}).toList(),
imageHeight: cameraImage.height,
imageWidth: cameraImage.width,
imageMean: 127.5,
imageStd: 127.5,
rotation: 90,
numResults: 2,
threshold: 0.1,
asynch: true);
recognitions.forEach((element) {
setState(() {
result = element["label"];
print(result);
});
});
}
}
```

recognitions هي قائمة المستقبل لذلك لكل عنصر أو نص تسمية، سنقوم بتعيينه على متغير النتيجة.

طريقة initState

```
@override
void initState() {
super.initState();
initCamera();
loadModel();
}
```

معاينة الكاميرا

توفر لنا حزمة Tflite أداة CameraPreview لمعاينة الكاميرا على شاشة التطبيق، فهي تتطلب وحدة تحكم الكاميرا cameraController.

```
AspectRatio(
aspectRatio: cameraController.value.aspectRatio,
child: CameraPreview(cameraController),
),
```

عرض النتيجة

```
Text(
result,
style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),
)
```

الكود الكامل

رابط الكود

```
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:tflite/tflite.dart';

List<CameraDescription> cameras;

Future<void> main() async {
```

```
WidgetsFlutterBinding.ensureInitialized();
cameras = await availableCameras();
runApp(MyApp());
}

class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
return MaterialApp(
theme: ThemeData.dark(),
home: MyHomePage(),
);
}
}

class MyHomePage extends StatefulWidget {
@override
_MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
CameraImage cameraImage;
CameraController cameraController;
String result = "";

initCamera() {
cameraController = CameraController(cameras[0], ResolutionPreset.medium);
cameraController.initialize().then((value) {
if (!mounted) return;
setState(() {
cameraController.startImageStream((imageStream) {
cameraImage = imageStream;
runModel();
});
});
});
}

loadModel() async {
await Tflite.loadModel(
model: "assets/model.tflite", labels: "assets/labels.txt");
}

runModel() async {
if (cameraImage != null) {
var recognitions = await Tflite.runModelOnFrame(
bytesList: cameraImage.planes.map((plane) {
return plane.bytes;
}).toList(),
imageHeight: cameraImage.height,
imageWidth: cameraImage.width,
imageMean: 127.5,
imageStd: 127.5,
rotation: 90,
numResults: 2,
threshold: 0.1,
asynch: true);
recognitions.forEach((element) {
setState(() {
result = element["label"];
print(result);
});
```

```
});
}
}

@override
void initState() {
super.initState();
initCamera();
loadModel();
}

@override
Widget build(BuildContext context) {
return SafeArea(
child: Scaffold(
appBar: AppBar(
title: Text("Face Mask Detector"),
),
body: Column(
children: [
Padding(
padding: const EdgeInsets.all(20),
child: Container(
height: MediaQuery.of(context).size.height - 170,
width: MediaQuery.of(context).size.width,
child: !cameraController.value.isInitialized
? Container()
: AspectRatio(
aspectRatio: cameraController.value.aspectRatio,
child: CameraPreview(cameraController),
),
),
),
Text(
result,
style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),
)
],
),
),
);
}
}
```

المصدر:

https://medium.flutterdevs.com/face-mask-detection-app-in-flutter-with-tensorflow-lite-89355032ccb6

6) انشاء تطبيق لمصادقة التعرف على الوجوه باستخدام باستخدام فلاتر وتنسرفلو لايت Build an Face Recognition Authentication Mobile App using Flutter with TensorFlow Lite

إن نمو قوة المعالجةفي الأجهزة والتعلم الآلي يسمح لنا بإنشاء حلول جديدة لم يكن من الممكن تحقيقها قبل بضع سنوات.في هذه الحالة، أريد أن أعرض طريقة مثيرة للاهتمام لإجراء المصادقة face authentication باستخدام Flutter وTensorflow Lite مع التعرف على الوجه recognition.

سأشرح خطوة بخطوة كيفية إنشاء تطبيق بسيط للتعرف على الوجه يحتوي على 3 وظائف: التسجيل Sign Up وتسجيل الدخول Sign In ومسح قاعدة البيانات Clear DB.

ملخص العملية

التسجيل

- يلتقط المستخدم صورة.
- تقوم نماذج التعلم الآلي بمعالجتها وإنشاء مخرجات (مجموعة من الأرقام) ليتم تخزينهافي قاعدة بيانات.
- يتم طلب اسم وكلمة مرور (الاسم ليس ضروريًافي الواقع، يتم طلبه فقط لإظهار التحيةفي ملفك الشخصي).

تسجيل الدخول

- يلتقط المستخدم صورة.
- تقوم نماذج التعلم الآلي بمعالجتها وإنشاء مخرجات.
- ستتم مقارنة المخرجات بالمخرجات المخزنة بالفعلفي قاعدة البيانات (تتم مقارنتها حسب القرب بالمخرجات التي تجدها). كشرط، يجب أن يكون القرب تحت العتبة threshold (الحد الأدنى للمسافة minimum distance)، فإذا تجاوزه، فسوف يقوم بمعالجته كمستخدم غير موجود.
- إذا كان المستخدم موجودًا (تمت معالجة الوجه بالفعل)، فإنه يطلب كلمة المرور لذلك المستخدم، ويتحقق من صحتها ويصادق عليها.

مسح قاعدة البيانات

- هذه الوظيفة مخصصة لتصحيح الأخطاء فقط، فهي تحذف جميع البيانات المحفوظةفي الذاكرة.

ملاحظة: الغرض من هذا التطبيق هو ببساطة إظهار الوظيفة الرئيسية (وهي التعرف على الوجه). ولهذا السبب لا يتم تخزين السجلاتفي قاعدة بيانات على الخادم، ولكن يتم حفظهافي ملف jsonفي ذاكرة الجهاز.

كيف تعمل

إنه يعمل مع نموذجين للرؤية الحاسوبية يعملان معًا، نموذج Firebase ML vision لإجراء اكتشاف الوجه والمعالجة المسبقةفي الصورة، ونموذج MobileFaceNet لمعالجة وتصنيف وتحويل إلى هيكل بيانات "قابل للحفظ savable" بواسطة قاعدة بيانات (مجموعة من أعداد).

Tensorflow Lite

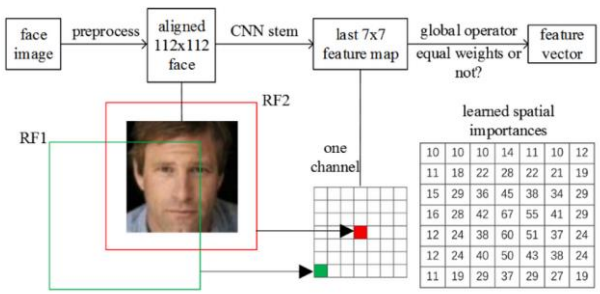
لدمج MobileFaceNet، من الضروري تحويل نموذج Tensorflow (امتداد pb.) إلى ملف بامتداد tflite.. إنها عملية تم شرحهافي هذه السلسلة: الجزء 1، الجزء 2. الجزء 3. ولكن إذا كنت تريد تخطيها، يمكنك تنزيل ملف tflite. مباشرة من الرابط الخاص بي.

نماذج MobileFaceNet

MobileFaceNets عبارة عن مجموعة من نماذج CNN، التي تستخدم أقل من مليون معلمة ومصممة خصيصًا للتحقق من الوجه عالي الدقةفي الوقت الفعلي على الأجهزة المحمولة mobile devices والأجهزة المدمجة embedded devices.

في ظل نفس الظروف التجريبية، تحقق MobileFaceNets دقة فائقة بشكل ملحوظ بالإضافة إلى سرعة فعلية تزيد عن ضعفي سرعة MobileNetV2 (وهو ما تم شرحهفي هذا البرنامج التعليمي).

لمزيد من المعلومات راجع هذه الورقة.



يتلقى مصفوفة من المدخلات 112x112 ويعيد كإخراج مصفوفة 7x7 مع تعديل القيم حسب الأهمية، كما ترون، يجب أن تكون وحدة الزاوية أقل أهمية من الوحدة المركزية، بسبب الضوضاءفي الحواف.

Firebase ML vision

باستخدام واجهة برمجة التطبيقات Face Detection API الخاصة بـ ML Kit، يمكنك اكتشاف الوجوهفي الصورة، وتحديد ميزات الوجه الرئيسية، والحصول على ملامح الوجوه المكتشفة. تعمل بشكل جيد جدًا على المعالجة المسبقة للصورة لاكتشاف المنطقة التي سيتم اقتصاصها ثم معالجتها بواسطة نموذج MobileFaceNet.

تثبيت

يتم شرح كيفية تثبيته خطوة بخطوة على صفحة الحزمة الرسمية بشكل جيد للغاية.

تنفيذ Flutter

ملحوظة: سأتخطى الكثير من الأكواد، لأنه إذا قمت بشرح الكود الكامل خطوة بخطوة، فإن هذا البرنامج التعليمي سيكون طويلًا جدًا، على أي حال سأشرح الأجزاء التي أعتبرها الأكثر أهمية.

إذا كنت تريد رؤية الكود الكامل، فراجع هذا Repo.

الوصف

بمجرد إدخال خيار sign up أو sign in، يكتشف نموذج رؤية ML (ML vision) الوجوه البشرية الموجودةفي الإطارات التي تعاينها الكاميرا، ويحتوي فئة الوجه على إحداثيات النقاط التي تشكل الإطار حول الوجه.

```
_cameraService.cameraController.startImageStream((image) async {
    ...
    try {
      /// returns a list of faces, but for our propose we just need the
first one detected
      List<Face> faces = await _mlVisionService.getFacesFromImage(image);
      ...
    } catch (e) {
      ...
    }
}
```

إذا تم الضغط على زر sign up أو sign in(حسب الحالة). يتم التقاط الوجه المكتشف للإطار الأخير واقتصاصه ثم معالجته مسبقًا لتتم معالجته بواسطة نموذج MobileFaceNet.

```
  setCurrentPrediction(CameraImage cameraImage, Face face) {
    /// crops the face from the image and transforms it to an array of data
    List input = _preProcess(cameraImage, face);
```

```
    /// then reshapes input and ouput to model format
    input = input.reshape([1, 112, 112, 3]);
    List output = List(1 * 192).reshape([1, 192]);

    /// runs the interpreter and produces the output
    this._interpreter.run(input, output);
    output = output.reshape([192]);

    this._predictedData = List.from(output);
  }
```

يقوم نموذج MobileFaceNet بإرجاع مخرجات (مجموعة من الأرقام).

- إذا كانت عملية Sign Up، يطلب التطبيق اسمًا وكلمة مرور، ثم يحفظ البيانات الثلاثة لاحقًا (الاسم وكلمة المرور ومخرجات ML).
- إذا كانت عملية Sign In، فسيقوم التطبيق بإجراء بحثفي قاعدة البيانات لمقارنة المسافة الإقليدية بين مخرجات ML للصورة ومخرجات ML المخزنة لكل مستخدم، تلك التي تتطابق أو تكون قريبة بدرجة كافية (دقة أعلى من العتبة) التطبيق يجلب البيانات ويطلب كلمة المرور.

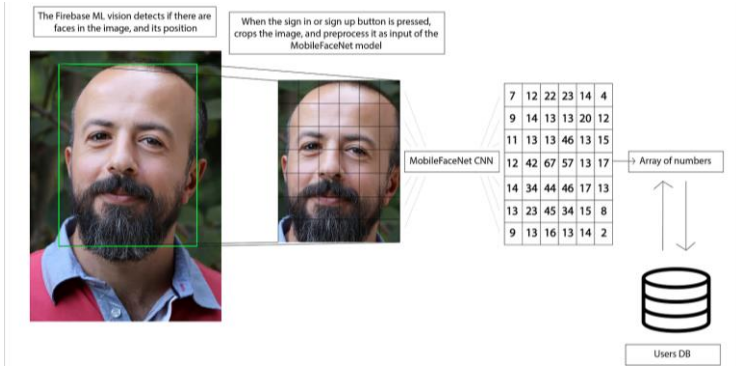
$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

معادلة المسافة الإقليدية.

تعريف صديق للمبرمج

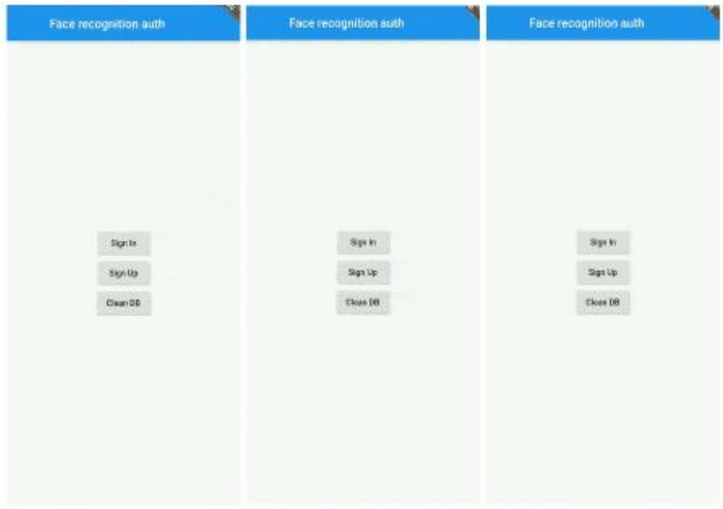
```
  double _euclideanDistance(List e1, List e2) {
    double sum = 0.0;
    for (int i = 0; i < e1.length; i++) {
      sum += pow((e1[i] - e2[i]), 2);
    }
    return sqrt(sum);
  }
```

يقوم المستخدم بإدخال كلمة المرور، ويقوم التطبيق بالتحقق مما إذا كانت كلمة المرور تتوافق مع كلمة مرور المستخدم الذي تم اكتشافه.



يمثل الرسم البياني العملية الموصوفة

مثال



العملية مرتبة من اليسار إلى اليمين:

- الدخول بدون تسجيل.
- عملية التسجيل.
- تسجيل الدخول الناجح بعد التسجيل.

الاستنتاج

في هذا البرنامج التعليمي، لم أشر إلى أمان آلية المصادقة المقدمة، نظرًا لأن فكرة طلب كلمة المرور هي على وجه التحديد لتجنب الهجمات مثل عرض صورة لشخص آخر أمام الكاميرا. ولكن هناك طرق أخرى مثيرة للاهتمام لتحقيق المصادقة الآمنة دون الحاجة إلى كلمة مرور، كما ترونفي الأمثلة التالية:

- مطالبة المستخدم بأداء الوضع والتحقق من صحته باستخدام نموذج PoseNet.
- أمر صوتي مرتبط بالمستخدم من خلال نموذج التعرف على الكلام.

رابط المشروع على GitHub Repo

المصدر:

https://medium.com/analytics-vidhya/face-recognition-authentication-using-flutter-and-tensorflow-lite-2659d941d56e

7) انشاء تطبيق موبايل للكشف المباشر عن الكائنات باستخدام فلاتر وتنسرفلو لايت Build a Live Object Detection Mobile App using Flutter with TensorFlow Lite

دعونا نتعلم كيفية إنشاء تطبيق Flutter يكتشف الكائنات الموجودة على الكاميرا بشكل مباشر.

في هذا البرنامج التعليمي، سنتعلم كيفية إنشاء تطبيق يمكنه اكتشاف الكائنات Object Detection، وباستخدام الذكاء الاصطناعي والتعلم العميق يمكنه تحديد ماهية الكائن. يوفر لنا Tflite إمكانية الوصول إلى TensorFlow Lite. TensorFlow Lite هو إطار عمل مفتوح المصدر للتعلم العميق للاستدلال على الجهاز. لدمج TensorFlow Lite في تطبيق Flutter، نحتاج إلى تثبيت حزمة tflite ونحتاج إلى ملفين model.tflite و labels.txt . model.tflite هو النموذج المدرب والملف labels.txt عبارة عن ملف نصي يحتوي على كافة التصنيفات. توفر لنا العديد من مواقع الويب تسهيلات لتدريب نموذجنا باستخدام مجموعة البيانات الخاصة بنا ونشرها على TensorFlow Lite ويمكننا الحصول على هذين الملفين مباشرة من هناك. يمكنك قراءة البرنامج التعليمي على تطبيق Face Mask Detection App مع Flutter وTensorFlow Lite لتدريب النموذج الخاص بك باستخدام مجموعة البيانات الخاصة بك.

تثبيت الحزم

- Camera/Flutter Package
- tflite/Flutter Package

تكوين أندرويد

قم بتغيير الحد الأدنى لإصدار Android SDK إلى 21 (أو أعلى)في ملف android/app/build.gradle الخاص بك.

```
minSdkVersion 21
```

في android/app/build.gradle، أضف الإعداد التالي في كتلة android.

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

أضف ملفات النماذج والتسمية في مجلد assets، وأضفها أيضًا في pubspec.yaml

تهيئة الكاميرا

داخل طريقة main، قم بتهيئة الكاميرات المتاحة باستخدام availableCameras.

```
List<CameraDescription> cameras;Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras();
  runApp(MyApp());
}
```

camera توفر لنا الحزمة الدعم لبث الصور المباشرة. قم أولاً بإنشاء كائن من CameraController. يأخذ CameraController وسيطتين CameraDescription وResolutionPreset. قم بتهيئة initialize وحدة التحكم بالكاميرا CameraController وبعد ذلك يمكننا بدء تدفق الصور باستخدام طريقة startImageStream. startImageStream توفر لنا الطريقة الصور، وسنعطي هذه الصور إلى CameraImage ثم سنقوم بتشغيل النموذج الخاص بنا.

```
CameraImage cameraImage;
CameraController cameraController;
initCamera() {
cameraController = CameraController(cameras[0], ResolutionPreset.medium);
cameraController.initialize().then((value) {
if (!mounted) return;
setState(() {
cameraController.startImageStream((image) {
cameraImage = image;
runModel();
});
});
});
}
```

تحميل النموذج

يوفر لنا Tflite طريقة LoadModel لتحميل نموذجنا. يستغرق الأمر مسار ملف نموذج قيمتين ومسار ملف التسميات.

```
Future loadModel() async {
Tflite.close();
await Tflite.loadModel(
model: "assets/ssd_mobilenet.tflite",
labels: "assets/ssd_mobilenet.txt");
}
```

تشغيل النموذج

في هذه الطريقة، سنقوم بتشغيل النموذج باستخدام Tflite. نحن هنا نستخدم البث المباشر للصورة لذا سيتعين علينا استخدام طريقة DetectObjectOnFrame لتشغيل نموذجنا.

```
runModel() async {
recognitionsList = await Tflite.detectObjectOnFrame(
bytesList: cameraImage.planes.map((plane) {
return plane.bytes;
}).toList(),
imageHeight: cameraImage.height,
imageWidth: cameraImage.width,
```

```
imageMean: 127.5,
imageStd: 127.5,
numResultsPerClass: 1,
threshold: 0.4,
);

setState(() {
cameraImage;
});
}
```

عرض المربعات حول الكائنات التي تم التعرف عليها

```
List<Widget> displayBoxesAroundRecognizedObjects(Size screen) {
if (recognitionsList == null) return [];

double factorX = screen.width;
double factorY = screen.height;

Color colorPick = Colors.pink;

return recognitionsList.map((result) {
return Positioned(
left: result["rect"]["x"] * factorX,
top: result["rect"]["y"] * factorY,
width: result["rect"]["w"] * factorX,
height: result["rect"]["h"] * factorY,
child: Container(
decoration: BoxDecoration(
borderRadius: BorderRadius.all(Radius.circular(10.0)),
border: Border.all(color: Colors.pink, width: 2.0),
),
child: Text(
"${result['detectedClass']} ${(result['confidenceInClass'] *
100).toStringAsFixed(0)}%",
style: TextStyle(
background: Paint()..color = colorPick,
color: Colors.black,
fontSize: 18.0,
),
),
),
);
}).toList();
}
```

هذا هو المربع الذي سيتم عرضه حول الكائن المكتشف. يحتوي كل عنصر من عناصر recognitionsList على التفاصيل التالية:

```
{
detectedClass: "hot dog",
confidenceInClass: 0.123,
rect: {
x: 0.15,
y: 0.33,
w: 0.80,
h: 0.27
}
}
```

DetectorClass هو اسم الكائن الذي تم اكتشافه. 100* TrustInClass هي نسبة الصحة % of correctness. rect هي أبعاد الكائن. يمكننا استخدام هذه المعلمات لعرض المربعات حول الكائن المحدد.

معاينة الكاميرا

توفر لنا حزمة Tflite أداة CameraPreview لمعاينة الكاميرا على شاشة التطبيق، وتحتاج إلى وحدة cameraController.

```
AspectRatio(
aspectRatio: cameraController.value.aspectRatio,
child: CameraPreview(cameraController)
```

لعرض الصناديق ومعاينة الكاميرا معًا نحتاج إلى المكدس Stack. تقوم DisplayBoxesAroundRecognizedObjects بإرجاع قائمة من المربعات، ونحن بحاجة إلى إضافة هذه القائمة إلى Stack حتى نتمكن من إنشاء متغير list. أضف المربعات الموجودةفي القائمة.

```
List<Widget> list = [];

list.add(
Positioned(
top: 0.0,
left: 0.0,
width: size.width,
height: size.height - 100,
child: Container(
height: size.height - 100,
child: (!cameraController.value.isInitialized)
? new Container()
: AspectRatio(
aspectRatio: cameraController.value.aspectRatio,
child: CameraPreview(cameraController),
),
),
),
);

if (cameraImage != null) {
list.addAll(displayBoxesAroundRecognizedObjects(size));
}
```

الكود الكامل

```
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:tflite/tflite.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras();
  runApp(MyApp());
}

List<CameraDescription> cameras;

class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData.dark(),
      home: HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  CameraController cameraController;
  CameraImage cameraImage;
  List recognitionsList;

  initCamera() {
    cameraController = CameraController(cameras[0],
ResolutionPreset.medium);
    cameraController.initialize().then((value) {
      setState(() {
        cameraController.startImageStream((image) => {
              cameraImage = image,
              runModel(),
            });
      });
    });
  }

  runModel() async {
    recognitionsList = await Tflite.detectObjectOnFrame(
      bytesList: cameraImage.planes.map((plane) {
        return plane.bytes;
      }).toList(),
      imageHeight: cameraImage.height,
      imageWidth: cameraImage.width,
      imageMean: 127.5,
      imageStd: 127.5,
      numResultsPerClass: 1,
      threshold: 0.4,
    );

    setState(() {
      cameraImage;
    });
  }

  Future loadModel() async {
    Tflite.close();
    await Tflite.loadModel(
        model: "assets/ssd_mobilenet.tflite",
        labels: "assets/ssd_mobilenet.txt");
  }

  @override
  void dispose() {
    super.dispose();

    cameraController.stopImageStream();
```

```
    Tflite.close();
  }

  @override
  void initState() {
    super.initState();

    loadModel();
    initCamera();
  }

  List<Widget> displayBoxesAroundRecognizedObjects(Size screen) {
    if (recognitionsList == null) return [];

    double factorX = screen.width;
    double factorY = screen.height;

    Color colorPick = Colors.pink;

    return recognitionsList.map((result) {
      return Positioned(
        left: result["rect"]["x"] * factorX,
        top: result["rect"]["y"] * factorY,
        width: result["rect"]["w"] * factorX,
        height: result["rect"]["h"] * factorY,
        child: Container(
          decoration: BoxDecoration(
            borderRadius: BorderRadius.all(Radius.circular(10.0)),
            border: Border.all(color: Colors.pink, width: 2.0),
          ),
          child: Text(
            "${result['detectedClass']} ${(result['confidenceInClass'] *
100).toStringAsFixed(0)}%",
            style: TextStyle(
              background: Paint()..color = colorPick,
              color: Colors.black,
              fontSize: 18.0,
            ),
          ),
        ),
      );
    }).toList();
  }

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    List<Widget> list = [];

    list.add(
      Positioned(
        top: 0.0,
        left: 0.0,
        width: size.width,
        height: size.height - 100,
        child: Container(
          height: size.height - 100,
          child: (!cameraController.value.isInitialized)
              ? new Container()
              : AspectRatio(
                  aspectRatio: cameraController.value.aspectRatio,
                  child: CameraPreview(cameraController),
```

```
                ),
          ),
        ),
      );

      if (cameraImage != null) {
        list.addAll(displayBoxesAroundRecognizedObjects(size));
      }

      return SafeArea(
        child: Scaffold(
          backgroundColor: Colors.black,
          body: Container(
            margin: EdgeInsets.only(top: 50),
            color: Colors.black,
            child: Stack(
              children: list,
            ),
          ),
        ),
      );
    }
}}
```

المصدر:

https://medium.flutterdevs.com/live-object-detection-app-with-flutter-and-tensorflow-lite-a6e7f7af3b07

8) انشاء تطبيق لتقدير الوضعية في الوقت الفعلي باستخدام فلاتر وتنسرفلو لايت Build a Real-Time Pose Estimation Mobile App using Flutter with TensorFlow Lite

(استخدام كاميرا الهاتف المحمول لتقدير الوضعية في الوقت الفعلي باستخدام Flutter وTensorFlow Lite وPoseNet)

في هذا البرنامج التعليمي سنعمل مع TensorFlow Lite، ونركز هذه المرة على تنفيذ اكتشاف الوضعية Real-Time Pose Detection في الوقت الحقيقي من خلال كاميرا الهاتف المحمول. سيكون التطبيق الذي سنقوم بإنشائه قادرًا على تقدير وضعية الشخص من خلال البث المباشر الذي توفره الكاميرا. ومن خلال إجراء بعض التعديلات وإعادة البناء على قاعدة بيانات الكشف عن الكائنات في الوقت الفعلي، تمكنت من التوصل إلى حل عملي.في هذا البرنامج التعليمي، سأرشدك خلال الحل الخاص بي.

يوفر لنا TensorFlow Lite نماذج مدربة مسبقًا ومحسنة لتحديد مئات فئات الكائنات بمافي ذلك الأشخاص والأنشطة والحيوانات والنباتات والأماكن. باستخدام نموذج PoseNet MobileNet V1 وFlutter Camera Plugin، يمكننا تطوير تطبيق لتقدير الوضعفي الوقت الفعلي.

الحزم المطلوبة

TensorFlow Lite

Camera Plugin

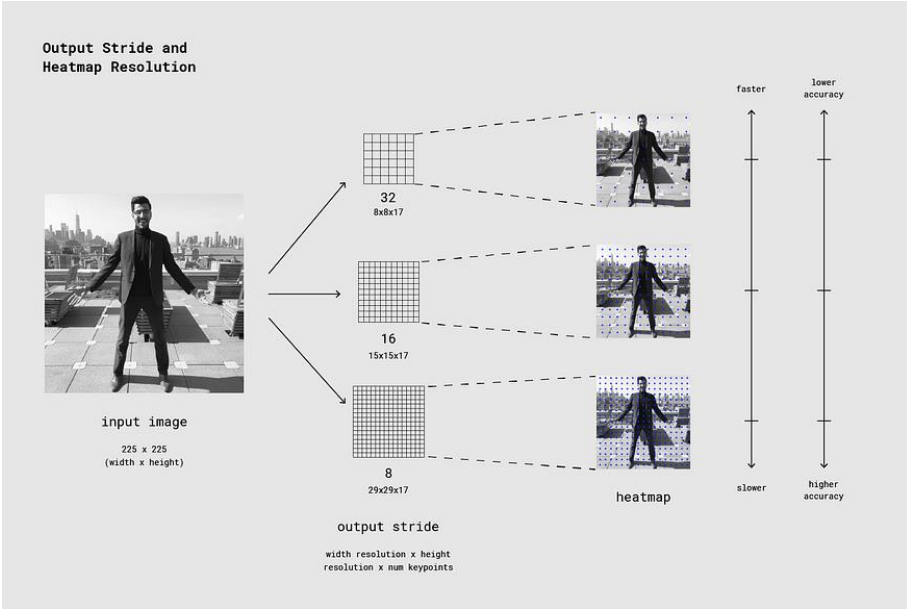
باستخدام هذا البرنامج المساعد، يمكننا التقاط البث المباشر الذي توفره كاميرا الهاتف المحمول إطارًا بإطار. ومن ثم يمكننا تقديم هذه الإطارات كصور لتقدير وضعية الإنسان.

PoseNet MobileNet V1

PoseNet هو نموذج للتعلم العميق يسمح لنا باكتشاف وضعية الشخص بناءً على مكان مفاصله. يأخذ نموذج PoseNet الصورة كمدخل ويخرج مجموعة من المعلومات حول النقاط الرئيسية key points بدرجة ثقة confidence score. يتم جدولة أدناه مثال على مفاصل الجسم المختلفة التي اكتشفها نموذج PoseNet:

Id	Part
0	nose
1	leftEye
2	rightEye
3	leftEar
4	rightEar
5	leftShoulder
6	rightShoulder
7	leftElbow
8	rightElbow
9	leftWrist
10	rightWrist
11	leftHip
12	rightHip
13	leftKnee
14	rightKnee
15	leftAnkle
16	rightAnkle

البُنية عالية المستوى لتقديرات الوضعية هي كما يلي:



للبدء، قم بتنزيل النموذج المطلوب من الرابط أعلاه وانسخ الملفات المستخرجة إلى مجلد assets الخاص بمشروع Flutter.

تطبيق Flutter

الآن بعد أن قمنا بإعداد التكوينات الأولية، نحتاج إلى تثبيت الحزم المطلوبة. ومن ثم يمكننا تطوير تطبيق Flutter الخاص بنا لتحويل كاميرا الهاتف المحمول إلى كاشف للوضعيات.

سيحتوي هذا المشروع على ثلاث فئات رئيسية مختلفة:

- **Camera** : تحتوي هذه الفئة على تنفيذ البرنامج الإضافي للكاميرا لتلقي البث المباشر.
- **BindBox** : تحتوي هذه الفئة على النقاط الرئيسيةفي الجسم مع أسمائها.
- **Home** : تحتوي هذه الفئة على معالجة الأخطاء وتحميل النموذج وتمرير البيانات من خلال الفئات المذكورة أعلاه.

سأقوم الآن بشرح مقتطف الكود المطلوب للفئات المذكورة أعلاه.

أولاً، نحتاج إلى تحميل نموذج PoseNet الذي تم تنزيله نظرًا لأننا نقوم بتشغيل التطبيق دون اتصال بالإنترنت، ويجب تشغيل النموذج الذي تم تحميله من خلال الفئات المذكورة من أجل اكتشاف تقدير وضعية الجسم.

```
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:tflite/tflite.dart';
import 'dart:math' as math;

import 'camera.dart';
import 'bndbox.dart';
import 'models.dart';

class HomePage extends StatefulWidget {
  final List<CameraDescription> cameras;

  HomePage(this.cameras);

  @override
  _HomePageState createState() => new _HomePageState();
}

class _HomePageState extends State<HomePage> {
  List<dynamic> _recognitions;
  int _imageHeight = 0;
  int _imageWidth = 0;
  String  model = "";

  @override
  void initState() {
    super.initState();
  }

  loadModel() async {
    String res;
    switch (_model) {
```

```
      case posenet:
        res = await Tflite.loadModel(
            model: "assets/posenet_mv1_075_float_from_checkpoints.tflite");
        break;
    }
    print(res);
  }

  onSelect(model) {
    setState(() {
      _model = model;
    });
    loadModel();
  }

  setRecognitions(recognitions, imageHeight, imageWidth) {
    setState(() {
      _recognitions = recognitions;
      _imageHeight = imageHeight;
      _imageWidth = imageWidth;
    });
  }

  @override
  Widget build(BuildContext context) {
    Size screen = MediaQuery.of(context).size;
    return Scaffold(
      body: _model == ""
          ? Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  RaisedButton(
                    child: const Text(posenet),
                    onPressed: () => onSelect(posenet),
                  ),
                ],
              ),
            )
          : Stack(
              children: [
                Camera(
                  widget.cameras,
                  _model,
                  setRecognitions,
                ),
                BndBox(
                    _recognitions == null ? [] : _recognitions,
                    math.max(_imageHeight, _imageWidth),
                    math.min(_imageHeight, _imageWidth),
                    screen.height,
                    screen.width,
                    _model),
              ],
            ),
    );
  }
}
```

كما ترون من مقتطف الكود أعلاه، سيتم استدعاء الطريقة loadModel() من خلال زر. بعد تحميل النموذج، سيتم تمريره عبر فئة Camera. ستوفر فئة الكاميرا بثًا مباشرًا إطارًا بإطار لاكتشاف تقدير

الوضعية المعروض في البث. داخل فئة الكاميرا، نتحقق أولاً مما إذا كان قد تم منح إذن استخدام الكاميرا أم لا. إذا تم تقديمه، فسنستخدم الكاميرا لتجميع البث المباشر إطارًا بإطار وتشغيله من خلال Tflite.runPoseNetOnFrame(). لاكتشاف النقاط الرئيسية للوضعية باستخدام النموذج.

```
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:tflite/tflite.dart';
import 'dart:math' as math;

import 'models.dart';

typedef void Callback(List<dynamic> list, int h, int w);

class Camera extends StatefulWidget {
  final List<CameraDescription> cameras;
  final Callback setRecognitions;
  final String model;

  Camera(this.cameras, this.model, this.setRecognitions);

  @override
  _CameraState createState() => new _CameraState();
}

class _CameraState extends State<Camera> {
  CameraController controller;
  bool isDetecting = false;

  @override
  void initState() {
    super.initState();

    if (widget.cameras == null || widget.cameras.length < 1) {
      print('No camera is found');
    } else {
      controller = new CameraController(
        widget.cameras[0],
        ResolutionPreset.high,
      );
      controller.initialize().then((_) {
        if (!mounted) {
          return;
        }
        setState(() {});

        controller.startImageStream((CameraImage img) {
          if (!isDetecting) {
            isDetecting = true;

            int startTime = new DateTime.now().millisecondsSinceEpoch;

           if (widget.model == posenet) {
              Tflite.runPoseNetOnFrame(
                bytesList: img.planes.map((plane) {
                  return plane.bytes;
                }).toList(),
                imageHeight: img.height,
                imageWidth: img.width,
                numResults: 2,
```

```
              ).then((recognitions) {
                int endTime = new DateTime.now().millisecondsSinceEpoch;
                print("Detection took ${endTime - startTime}");

                widget.setRecognitions(recognitions, img.height,
img.width);

                isDetecting = false;
              });
            }
          }
        });
      });
    }
  }

  @override
  void dispose() {
    controller?.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (controller == null || !controller.value.isInitialized) {
      return Container();
    }

    var tmp = MediaQuery.of(context).size;
    var screenH = math.max(tmp.height, tmp.width);
    var screenW = math.min(tmp.height, tmp.width);
    tmp = controller.value.previewSize;
    var previewH = math.max(tmp.height, tmp.width);
    var previewW = math.min(tmp.height, tmp.width);
    var screenRatio = screenH / screenW;
    var previewRatio = previewH / previewW;

    return OverflowBox(
      maxHeight:
          screenRatio > previewRatio ? screenH : screenW / previewW *
previewH,
      maxWidth:
          screenRatio > previewRatio ? screenH / previewH * previewW :
screenW,
      child: CameraPreview(controller),
    );
  }
}
```

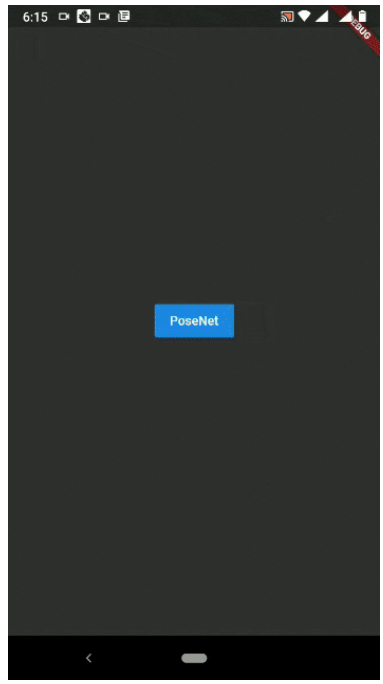
الآن بعد أن شرحت كيفية استخدام البرنامج المساعد للكاميرا مع TensorFlow Lite، دعونا نلقي نظرة على كيفية اكتشاف النقاط الرئيسيةفي الجسم من البث المباشر الذي توفره فئة الكاميرا. باستخدام فئة BindBox، يمكننا تقديم النقاط الرئيسية على النحو التالي.

```
import 'package:flutter/material.dart';
import 'models.dart';

class BndBox extends StatelessWidget {
  final List<dynamic> results;
  final int previewH;
  final int previewW;
```

```
  final double screenH;
  final double screenW;
  final String model;

  BndBox(this.results, this.previewH, this.previewW, this.screenH,
this.screenW,
      this.model);

  @override
  Widget build(BuildContext context) {

    List<Widget> _renderKeypoints() {
      var lists = <Widget>[];
      results.forEach((re) {
        var list = re["keypoints"].values.map<Widget>((k) {
          var _x = k["x"];
          var _y = k["y"];
          var scaleW, scaleH, x, y;

          if (screenH / screenW > previewH / previewW) {
            scaleW = screenH / previewH * previewW;
            scaleH = screenH;
            var difW = (scaleW - screenW) / scaleW;
            x = (_x - difW / 2) * scaleW;
            y = _y * scaleH;
          } else {
            scaleH = screenW / previewW * previewH;
            scaleW = screenW;
            var difH = (scaleH - screenH) / scaleH;
            x = _x * scaleW;
            y = (_y - difH / 2) * scaleH;
          }
          return Positioned(
            left: x - 6,
            top: y - 6,
            width: 100,
            height: 12,
            child: Container(
              child: Text(
                "● ${k["part"]}",
                style: TextStyle(
                  color: Color.fromRGBO(37, 213, 253, 1.0),
                  fontSize: 12.0,
                ),
              ),
            ),
          );
        }).toList();

        lists..addAll(list);
      });

      return lists;
    }

    return Stack(
      children: model == posenet ? _renderKeypoints() : null
    );
  }
}
```

النتائج

الآن بعد أن قمنا بتنفيذ كود تطبيق Flutter، فلنلقِ نظرة على مخرجات التطبيق عندما يكون قيد التشغيل:



المصدر:

https://heartbeat.comet.ml/implementing-real-time-pose-estimation-on-/mobile-using-flutter-af281d3740df

9) انشاء تطبيق للتعرف على الأرقام باستخدام فلاتر وتنسرفلو لايت Build a Digit Recognition Mobile App using Flutter with TensorFlow Lite

(كيفية تطوير تطبيق الهاتف المحمول للتعرف على الأرقام باستخدام Flutter، TensorFlow Lite ومجموعة بيانات MNIST)

يأخذ التعلم الآلي والذكاء الاصطناعي تطوير تطبيقات الهاتف المحمول إلى مستوى جديد. يمكن للتطبيقات التي تستخدم التعلم الآلي التعرف على الكلام والصور والإيماءات.

وهذا يمنحنا طرقًا جديدة ومقنعة للمشاركة والتفاعل مع الأشخاص في العالم من حولنا. ولكن كيف يمكننا دمج التعلم الآلي في تطبيقات الهاتف المحمول لدينا؟

لقد كان تطوير تطبيقات الهاتف المحمول التي تتضمن التعلم الآلي مهمة صعبة منذ فترة طويلة. ولكن بمساعدة المنصات وأدوات التطوير مثل Fritz AI وFirebase's ML وTensorFlow Lite، أصبح القيام بذلك أسهل.

تزودنا هذه الأدوات بنماذج التعلم الآلي المدربة مسبقًا بالإضافة إلى أدوات لتدريب واستيراد نماذجنا المخصصة. ولكن كيف يمكننا بالفعل تطوير تجربة مقنعة فوق نماذج التعلم الآلي تلك؟ وهنا يأتي دور Flutter.

Flutter SDK عبارة عن مجموعة أدوات محمولة لواجهة المستخدم تم إنشاؤها بواسطة Google ومجتمعها مفتوح المصدر لتطوير تطبيقات Android وIOS والويب وسطح المكتب. يجمع Flutter في جوهره بين محرك رسومي عالي الأداء ولغة برمجة Dart.

يوفر Dart أمانًا قويًا للنوع وميزة إعادة التحميل السريع، مما يساعد المطورين على إنشاء تطبيقات موثوقة بسرعة. باستخدام Flutter، يمكننا إنشاء تطبيقات جوال تتمتع بقدرات التعلم الآلي مثل تصنيف الصور واكتشاف الكائنات، لكل من منصات Android وiOS.

في هذا البرنامج التعليمي، سنجمع بين قوة Flutter والتعلم الآلي على الجهاز لتطوير تطبيق Flutter يمكنه التعرف على الأرقام المكتوبة بخط اليد، باستخدام TensorFlow Lite ومجموعة بيانات MNIST الشهيرة.

الحزم المطلوبة

- TensorFlow Lite

في ملف android/app/build.gradle، أضف الإعداد التاليفي كتلة android وقم بتغيير miniSdkVersion إلى 19. تم توضيح خطوات التثبيتفي حزمة Flutter أعلاه.

- مجموعة بيانات MNIST (Modified National Institute of Standards and Technology database): واحدة من أكبر قواعد البيانات للأرقام المكتوبة بخط اليد، وتستخدم عادة لتدريب أنظمة معالجة الصور.

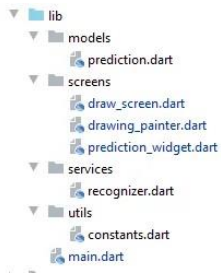
قم بتحميل ملف mnist.tflite من الرابط أعلاه، وضعه داخل مجلد assets الخاص بمشروع Flutter.

قم بإنشاء ملف mnist.txt داخل مجلد assets وأضف تسميات النموذج المناسبة. (0 إلى 9 أرقام)

الآن بعد أن ألقينا نظرة سريعة على كيفية إعداد تطبيق Flutter لتشغيل TensorFlow Lite، فلنلقِ نظرة على كيفية إعداده وتشغيله.

الاعداد

قبل أن نبدأ، نحتاج إلى إنشاء 4 حزم ونماذج وشاشات وخدمات وأدوات مساعدة.



سأعرض مقتطفات التعليمات البرمجية المطلوبة للملفات المذكورة أعلاه.

```
class DrawScreen extends StatefulWidget {
  @override
  _DrawScreenState createState() => _DrawScreenState();
}

class _DrawScreenState extends State<DrawScreen> {
  final _points = List<Offset>();
  final _recognizer = Recognizer();
  List<Prediction> _prediction;
  bool initialize = false;

  @override
  void initState() {
    super.initState();
    _initModel();
  }
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      title: Text('Digit Recognizer'),
    ),
    body: Column(
      children: <Widget>[
        SizedBox(
          height: 10,
        ),
        _drawCanvasWidget(),
        SizedBox(
          height: 10,
        ),
        PredictionWidget(
          predictions: _prediction,
        ),
      ],
    ),
    floatingActionButton: FloatingActionButton(
      child: Text("Clear"),
      onPressed: () {
        setState(() {
          _points.clear();
          _prediction.clear();
        });
      },
    ),
  );
}

Widget _drawCanvasWidget() {
  return Container(
    width: Constants.canvasSize + Constants.borderSize * 2,
    height: Constants.canvasSize + Constants.borderSize * 2,
    decoration: BoxDecoration(
      border: Border.all(
        color: Colors.black,
        width: Constants.borderSize,
      ),
    ),
    child: GestureDetector(
      onPanUpdate: (DragUpdateDetails details) {
        Offset _localPosition = details.localPosition;
        if (_localPosition.dx >= 0 &&
            _localPosition.dx <= Constants.canvasSize &&
            _localPosition.dy >= 0 &&
            _localPosition.dy <= Constants.canvasSize) {
          setState(() {
            _points.add(_localPosition);
          });
        }
```

```
        },
        onPanEnd: (DragEndDetails details) {
          _points.add(null);
          _recognize();
        },
        child: CustomPaint(
          painter: DrawingPainter(_points),
        ),
      ),
    );
  }

  void _initModel() async {
    var res = await _recognizer.loadModel();
  }

  void _recognize() async {
    List<dynamic> pred = await _recognizer.recognize(_points);
    setState(() {
      _prediction = pred.map((json) =>
Prediction.fromJson(json)).toList();
    });
  }
}
```

في مقتطف الكود أعلاه، نستخدم مجموعة بيانات MNIST للأرقام المكتوبة بخط اليد وننشئ تطبيقًا بلوحة Canvas حيث يرسم المستخدم الأرقام. يتم استخدام digits.Widget_drawCanvasWidget() لتحديد حجم اللوحة. بعد تحديد حجم المسودة نحتاج إلى تمرير البيانات إلى استخدام CustomPaint(painter: DrawPainter). من خلال draw_painter.dart يمكننا الحصول على تفاصيل اللوحة التي تحتوي على خصائص اللوحة (حجم الفرشاة، اللون، لون الخلفية، وما إلى ذلك).

```
class DrawingPainter extends CustomPainter {
  final List<Offset> points;

  DrawingPainter(this.points);

  final Paint _paint = Paint()
    ..strokeCap = StrokeCap.round
    ..color = Colors.blue
    ..strokeWidth = Constants.strokeWidth;

  @override
  void paint(Canvas canvas, Size size) {
    for (int i = 0; i < points.length - 1; i++) {
      if (points[i] != null && points[i + 1] != null) {
        canvas.drawLine(points[i], points[i + 1], _paint);
      }
    }
  }
```

```
  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return true;
  }

}
```

بعد الرسم (أي بعد إيقاف الرسم)، تستدعي ميزة onPanEnd فئة ()recognize وتمرير بيانات الصورة. سيتم استخدام هذا داخل recognizer.dart -> service.

```
final _canvasCullRect = Rect.fromPoints(
  Offset(0, 0),
  Offset(Constants.imageSize, Constants.imageSize),
);

final _whitePaint = Paint()
  ..strokeCap = StrokeCap.round
  ..color = Colors.white
  ..strokeWidth = Constants.strokeWidth;

final _bgPaint = Paint()
  ..color = Colors.black;

class Recognizer {
  Future loadModel() {
    Tflite.close();

    return Tflite.loadModel(
      model: "assets/mnist.tflite",
      labels: "assets/mnist.txt",
    );
  }

  dispose() {
    Tflite.close();
  }

  Future<Uint8List> previewImage(List<Offset> points) async {
    final picture = _pointsToPicture(points);
    final image = await
picture.toImage(Constants.mnistImageSize,
Constants.mnistImageSize);
    var pngBytes = await image.toByteData(format:
ImageByteFormat.png);

    return pngBytes.buffer.asUint8List();
  }
   Future recognize(List<Offset> points) async {
     final picture = _pointsToPicture(points);
     Uint8List bytes = await _imageToByteListUint8(
         picture, Constants.mnistImageSize);
     return _predict(bytes);
   }
```

```
  Future _predict(Uint8List bytes) {
    return Tflite.runModelOnBinary(binary: bytes);
  }

  Future<Uint8List> _imageToByteListUint8(Picture pic, int
size) async {
    final img = await pic.toImage(size, size);
    final imgBytes = await img.toByteData();
    final resultBytes = Float32List(size * size);
    final buffer = Float32List.view(resultBytes.buffer);

    int index = 0;

    for (int i = 0; i < imgBytes.lengthInBytes; i += 4) {
      final r = imgBytes.getUint8(i);
      final g = imgBytes.getUint8(i + 1);
      final b = imgBytes.getUint8(i + 2);
      buffer[index++] = (r + g + b) / 3.0 / 255.0;
    }
    return resultBytes.buffer.asUint8List();
  }

  Picture _pointsToPicture(List<Offset> points) {
    final recorder = PictureRecorder();
    final canvas = Canvas(recorder, _canvasCullRect)
      ..scale(Constants.mnistImageSize /
Constants.canvasSize);

    canvas.drawRect(
        Rect.fromLTWH(0, 0, Constants.imageSize,
Constants.imageSize),
        _bgPaint);

    for (int i = 0; i < points.length - 1; i++) {
      if (points[i] != null && points[i + 1] != null) {
        canvas.drawLine(points[i], points[i + 1],
_whitePaint);
      }
    }

    return recorder.endRecording();
  }
}
```

ستقوم فئة أداة Recognizer بتحميل نموذج mnist.tflite وملف التسمية mnist.txt. سيتم بعد ذلك تشغيل ذلك من خلال بيانات الصورة التي نمررها عندما نرسم على اللوحة. سيتم التحقق من تشابه الرقم الذي تحتوي عليه مجموعة البيانات وتمرير الإدخال من خلال فئة Predict.dart لتحويل الفهرس index والثقة confidence والتسمية label إلى تنسيق JSON. ثم نقوم بتمرير هذا إلى التوقع_widget.dart لعرض الرقم الأكثر دقة.

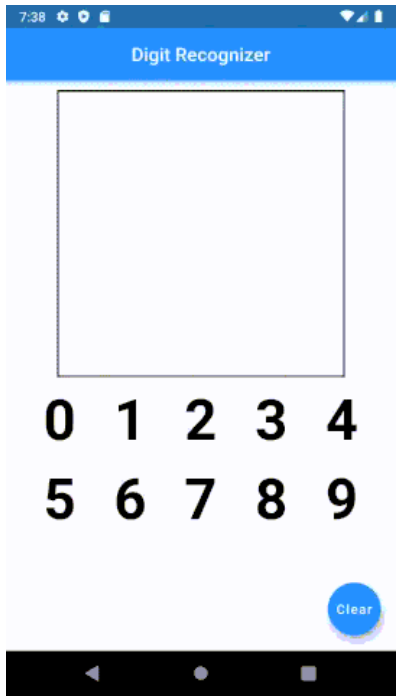
```
class Prediction {
  final double confidence;
  final int index;
  final String label;

  Prediction({this.confidence, this.index, this.label});

  factory Prediction.fromJson(Map<dynamic, dynamic> json) {
    return Prediction(
      confidence: json['confidence'],
      index: json['index'],
      label: json['label'],
    );
  }
}
```

Precision_widget.dart يحتوي على الأنماط والأرقام المطلوبة لإظهار الرقم المتوقع للرقم الذي قمنا بصياغته كمدخل.

```
class PredictionWidget extends StatelessWidget {
  final List<Prediction> predictions;

  const PredictionWidget({Key key, this.predictions}) :
super(key: key);

  Widget _numberWidget(int num, Prediction prediction) {
    return Column(
      children: <Widget>[
        Text(
          '$num',
          style: TextStyle(
            fontSize: 60,
            fontWeight: FontWeight.bold,
            color: prediction == null
                ? Colors.black
                : Colors.blue.withOpacity(
                    (prediction.confidence * 2).clamp(0,
1).toDouble(),
                  ),
          ),
        ),
        Text(
          '${prediction == null ? '' :
prediction.confidence.toStringAsFixed(3)}',
          style: TextStyle(
            fontSize: 12,
```

```
          ),
        )
      ],
    );
  }

  List<dynamic> getPredictionStyles(List<Prediction>
predictions) {
    List<dynamic> data = [
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null
    ];
    predictions?.forEach((prediction) {
      data[prediction.index] = prediction;
    });

    return data;
  }

  @override
  Widget build(BuildContext context) {
    var styles = getPredictionStyles(this.predictions);

    return Column(
      children: <Widget>[
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            for (var i = 0; i < 5; i++) _numberWidget(i,
styles[i])
          ],
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            for (var i = 5; i < 10; i++) _numberWidget(i,
styles[i])
          ],
        )
      ],
    );
  }
}
```

الآن بعد أن ألقينا نظرة على كود تطبيق Flutter، فلنلق نظرة على مخرجات التطبيق عندما يكون قيد التشغيل.



الاستنتاج

بمجرد أن تتقن الأمر، يمكنك معرفة مدى سهولة استخدام TensorFlow Lite مع Flutter لتطوير تطبيقات الهاتف المحمول لإثبات مفهوم التعلم الآلي. لتحسين معرفتك، يمكنك زيارة موقع Kaggle وتنزيل مجموعات بيانات متنوعة لتطوير نماذج تصنيف مختلفة.

المصدر:

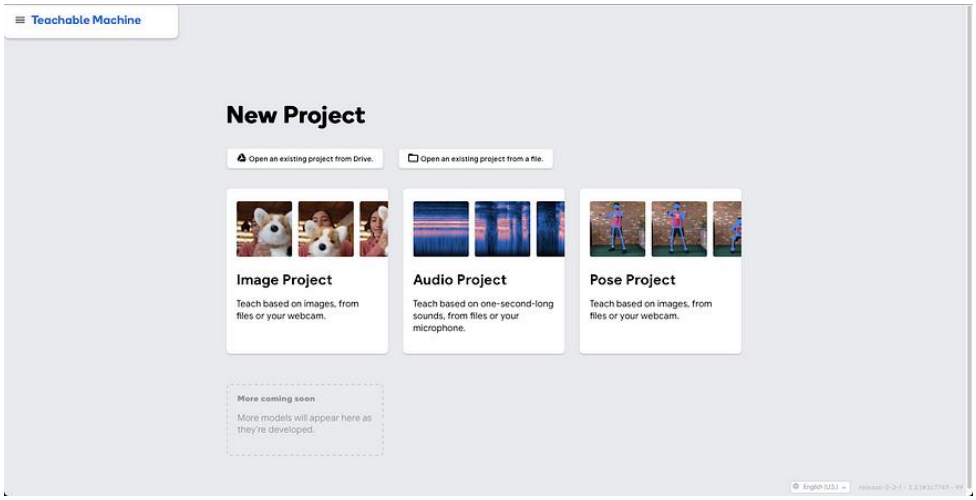
https://fritz.ai/digit-recognizer-with-flutter-and-tensorflow-lite/

10) انشاء تطبيق للتعرف على الاصوات باستخدام فلاتر وتنسرفلو لايت Build an Audio recognition Mobile App using Flutter with TensorFlow Lite

تم تعزيز تطوير تطبيقات الهاتف المحمول من خلال التعلم الآلي (ML) والذكاء الاصطناعي (AI). يتيح دمج نماذج التعلم الآليفي التطبيقات من أجل تصنيف الأحداث أو التنبؤ بها إنشاء تطبيقات قادرة على فهم سلوك المستخدم والتعرف عليه وجعل تجربته أكثر ذكاءً وإثارة للاهتمام. مثال على تسهيل الفهم هو الكلمات التالية المقترحة للمستخدم بناءً على المحتوى السابق المكتوب أثناء إرسال الرسائل النصية. لذلك، ستحلل هذا البرنامج التعليمي دمج نموذج MLفي تطبيق Flutter الهاتف المحمول للتعرف على الصوت audio recognition.

إنشاء نموذج التصنيف

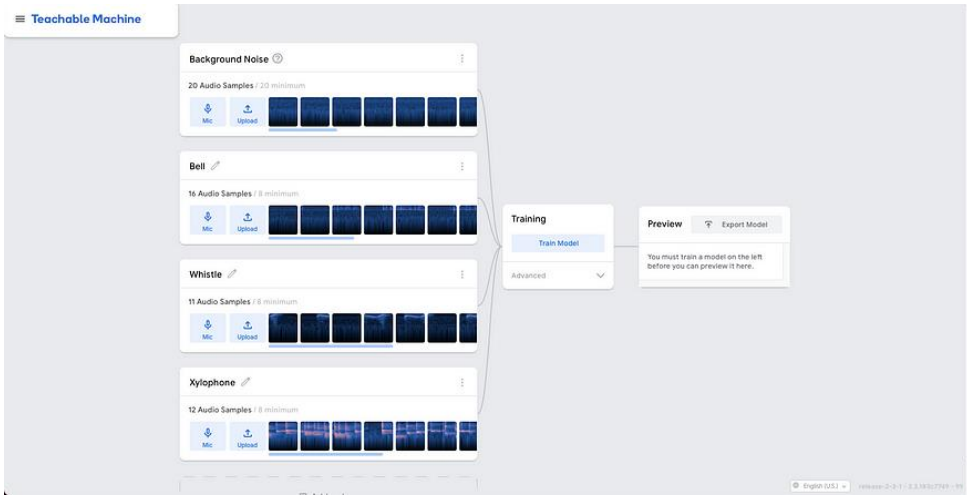
بادئ ذي بدء، من الضروري أن يكون لديك نموذج تصنيف مدرب. على سبيل المثال، إذا كنت مبتدئًا في مفاهيم تعلم الآلة، فإن Google Teachable Machine هي طريقة سريعة وسهلة لإنشاء نموذجك. يوفر هذا الإطار إنشاء ثلاثة أنواع من المشاريع: الصورة والصوت والوضعية.في هذا المثال، سيتم استخدام نموذج Google Teachable Machine (GTM)، وبما أن الهدف الرئيسي هو تكامل نموذج تصنيف الصوت، فيجب اختيار مشروع الصوت.



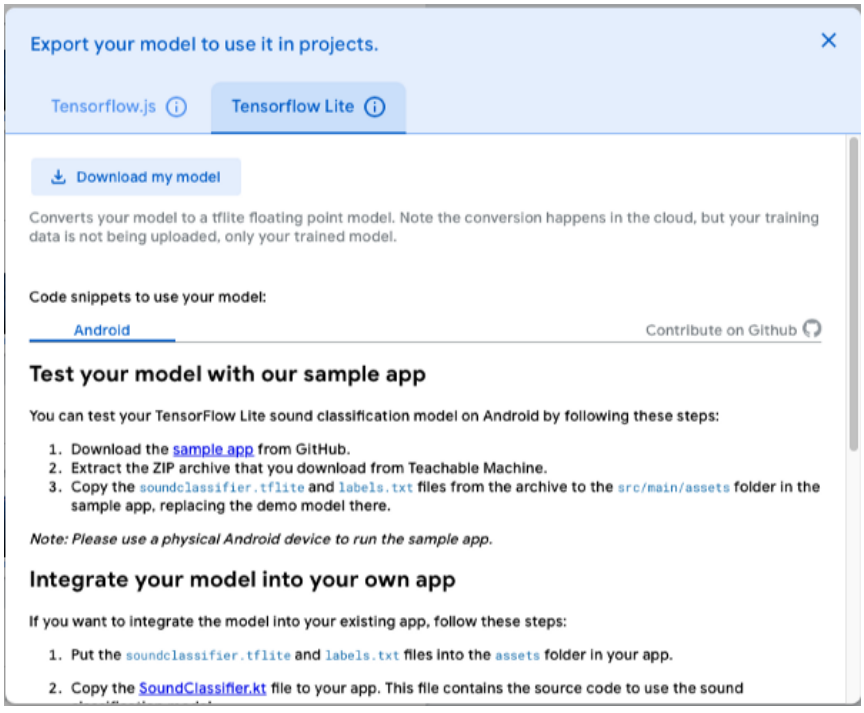
اطار عمل Google Teachable Machine

الخطوة الأولى لإنشاء النموذج هي ترتيب البياناتفي فئات مختلفة. من الممكن تسجيل العيناتفي الوقت الحالي أو تحميل الملفات معها. يحتوي كل فئة على الحد الأدنى المطلوب من العينات الصوتية.

ضجيج الخلفية Background Noise هو فئة افتراضية يجب أن تحتوي على عينات ذات ضجيجفي الخلفية.في هذا المثال، تم إنشاء فئات Bell وWhistle وXylophone مع عينات صوتية من هذه الأصوات.



إنشاء نموذج تصنيف باستخدام Google Teachable Machine



تصدير النموذج المدرب كنموذج Tensorflow Lite

بعد فصل البيانات حسب الفئات، يجب تدريب النموذج وتصديره إلى تنسيق Tensorflow Lite. يحتوي النموذج الذي تم تنزيله على ملفين: labels.txt (يحدد تسميات الفئات) وsoundclassifier.tflite (النموذج). ستتم إضافة هذه الملفات إلى تطبيق الهاتف المحمول حتى يتمكن Tensorflow من قراءتها وتحميل النموذج بمساعدة Tensorflow.

ملاحظة: إذا كنت مرتاحًا لمفاهيم تعلم الآلة، أوصيك بتطوير النموذج الخاص بك وتدريبه لأن استخدام نموذج GTMفي تطبيقك يمكن أن يحسن حجمه. تحتوي وثائق Tensorflow على دليل يساعدفي إنشاء النماذج.

إنشاء تطبيق Flutter

تم إنشاء تطبيق بسيط يحتوي على Text Widgets وMaterialButton الذي سيتم استخدامه لبدء تسجيل الصوت.

لقد تركت هنا الكود الأساسي main.dart:

```
import 'package:flutter/material.dart';
import 'package:tflite_audio/tflite_audio.dart';

void main() {
runApp(MyApp());
}

class MyApp extends StatelessWidget {
// This widget is the root of your application.
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Flutter Demo',
theme: ThemeData.dark(),
home: MyHomePage(),
);
}
}

class MyHomePage extends StatefulWidget {
@override
_MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
String _sound = "Press the button to start";
bool _recording = false;

@override
Widget build(BuildContext context) {
return Scaffold(
body: Container(
decoration: BoxDecoration(
image: DecorationImage(
image: AssetImage("assets/background.jpg"),
fit: BoxFit.cover,
```

```
),
),
child: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.spaceEvenly,
children: <Widget>[
Container(
padding: EdgeInsets.all(20),
child: Text(
"What's this sound?",
style: TextStyle(
color: Colors.white,
fontSize: 60,
fontWeight: FontWeight.w200,
),
),
),
MaterialButton(
onPressed: (){},
color: _recording ? Colors.grey : Colors.pink,
textColor: Colors.white,
child: Icon(Icons.mic, size: 60),
shape: CircleBorder(),
padding: EdgeInsets.all(25),
),
Text(
'$_sound',
style: Theme.of(context).textTheme.headline5,
),
],
),
),
),
);
}
}
```

دمج النموذج في التطبيق

لدمج نماذج Tensorflowفي تطبيقات الهاتف المحمول، يساعد إطار عمل Tensorflow Lite على تشغيل هذه النماذج على الهاتف المحمول. للوصول إلى Tensorflow Lite، من الضروري وجود مكون إضافي لـ Flutter لهذا الغرض: tflite. ومع ذلك، لا يقوم البرنامج المساعد tflite بتحليل الصوت حتى الآن. لذلك، سيتم استخدام مكون إضافي مشابه وحديث لمعالجة الصوت: tflite_audio. إلى جانب أن هذا البرنامج المساعد يدعم نماذج GTM ونماذج مع مدخلات موجة تم فك ترميزها، فإنه لا يزال لديه بعض القيود الموضحةفي الوثائق وهي أنه يعمل فقط عند تشغيله على الأجهزة المحمولة الفعلية. وفي الاستنتاجات، سيتم مناقشة هذا الموضوع بمزيد من التفصيل.

أولاً، أضف التبعية التالية إلى ملف pubspec.yaml الخاص بك:

```
dependencies:
tflite_audio: ^0.1.5+3
```

لتثبيت الحزمة قم بتشغيل:

```
flutter pub get
```

قم بإنشاء دليل assets/ جديد، وأضف ملفات soundclassifier.tflite وlabels.txt التي تم تنزيلها من إطار عمل Google Teachable Machine وقم بتغيير تكوينات pubspec.yaml لتضمين مجلد assets/ في مشروعك.

```
flutter:
# To add assets to your application, add an assets section:
assets:
- assets/labels.txt
- assets/soundclassifier.tflite
```

بعد ذلك، من الضروري تكوين بيئة Android وiOS ويتطلب استخدام نموذج GTM اختيار مشغلي Tensorflow.

تكوينات الاندرويد

1. أضف الأذونات التالية إلى AndroidManifest.xml:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

2. أضف aaptOptions إلى ملف app/build.gradle:

```
android {
   (...)
    aaptOptions {
        noCompress 'tflite'
    }    lintOptions {
        disable 'InvalidPackage'
    }
(...)
}
```

3. تمكين عمليات التحديدفي ملف app/build.gradle:

```
dependencies {
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
implementation "org.tensorflow:tensorflow-lite-select-tf-ops:+"
}
```

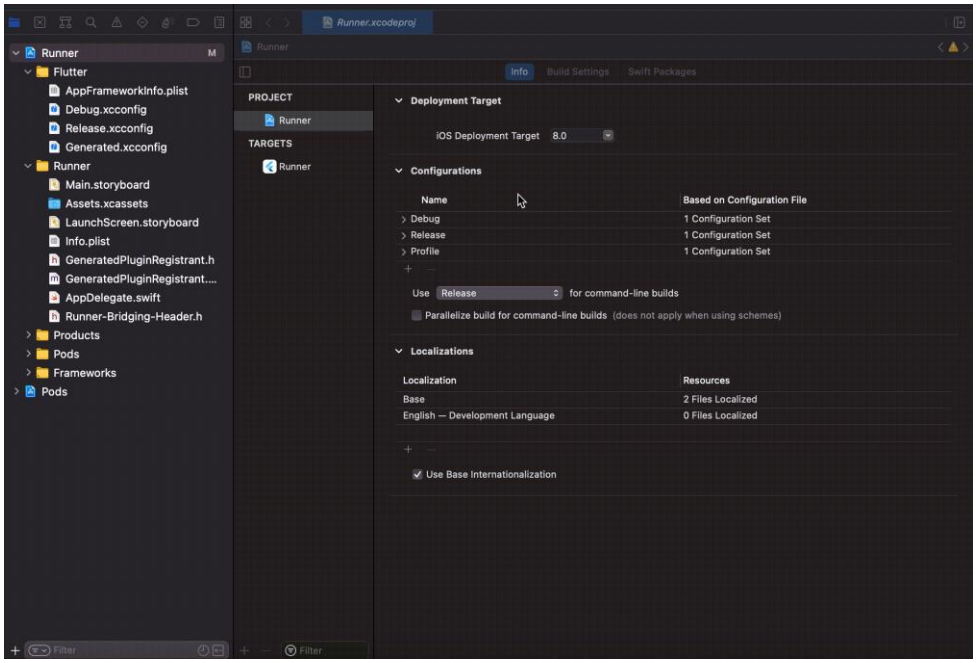
4. قم بتغيير minSdkVersion إلى 21 على الأقلفي ملف app/build.gradle.

تكوينات IOS

أضف الأذونات أدناه toios/Runner/Info.plist :

```
<key>NSMicrophoneUsageDescription</key>
<string>Record audio for playback</string>
```

من الضروري تغيير النشر المستهدف إلى 12.0 على الأقل. على سبيل المثال، افتح iOS/Runner.xcworkspace على XCode، وحدد Runner وقم بتغيير هدف نشر iOSفي علامة تبويب Info:



تغيير هدف نشر iOS إلى 12.0

الآن، أنت بحاجة إلى تحديد نشر هدف iOSفي ios/Runner/Podfile

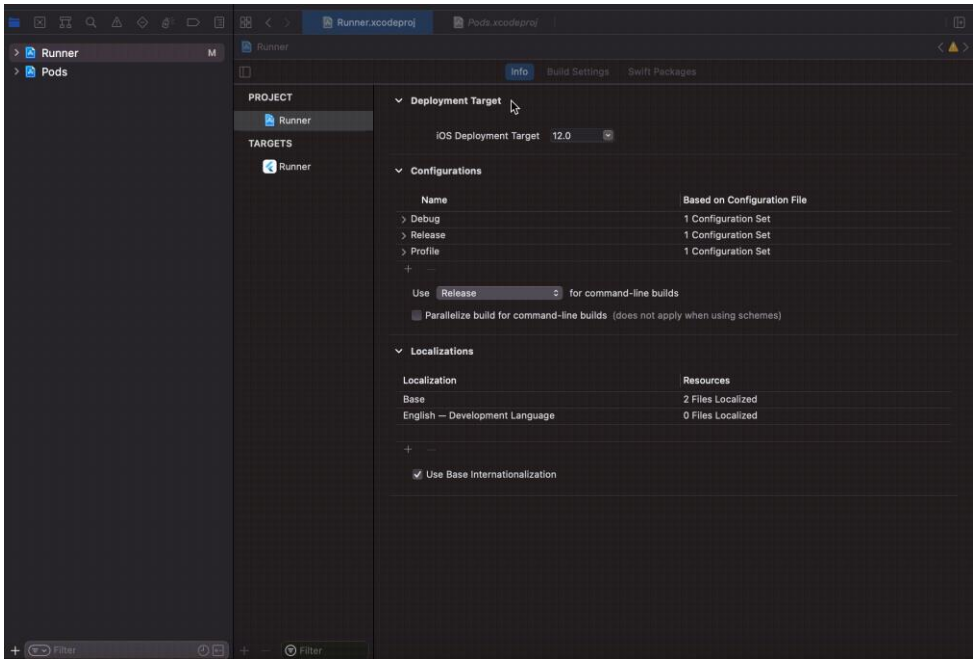
```
platform :ios, '12.0'
```

نظرًا لأن النموذج هو نموذج GTM، أضف سطر "TensorFlowLiteSelectTfOps" لاستهدافه في ios/Runner/Podfile

```
target 'Runner' do
  use_frameworks!
  use_modular_headers!
  pod 'TensorFlowLiteSelectTfOps'  flutter_install_all_ios_pods
File.dirname(File.realpath(__FILE__))
end
```

بعد ذلك، مع فتح المشروع على XCode، حدد Targets > Runner > Build Settings > All وأضف السطر التالي إلى Linking > Other Linker Flag:

```
-force_load
$(SRCROOT)/Pods/TensorFlowLiteSelectTfOps/Frameworks/TensorFlowLiteSelectTf
Ops.framework/TensorFlowLiteSelectTfOps
```



أخيرًا،في دليل ios/ ، قم بتشغيل الأوامر التالية على الوحدة الطرفية:

```
$ flutter pub get
$ pod install
$ flutter clean
```

تأكد من تحديث CocoaPods. أثناء pod install ، من الممكن أن يتم إطلاق تحذير.

```
[!] CocoaPods did not set the base configuration of your project because
your project already has
a custom config set. In order for CocoaPods integration to work at all,
please either set the base
configurations of the target `Runner` to `Target Support Files/Pods-
Runner/Pods-Runner.profile.xcconfig`
or include the `Target Support Files/Pods-Runner/Pods-
Runner.profile.xcconfig` in your build configuration
(`Flutter/Release.xcconfig`).
```

إذا كان لديك هذا التحذير بعد أمر pod install، فأضف السطر التالي إلى ملف ios/Flutter/Release.xcconfig الخاص بك وقم بتشغيل pod install مرة أخرى:

```
#include "Pods/Target Support Files/Pods-Runner/Pods-
Runner.profile.xcconfig"
```

تحميل واستخدام النموذج

في طريقة initState()، يجب عليك تحميل النموذج load the model:

```
TfliteAudio.loadModel(
model: 'assets/soundclassifier.tflite',
label: 'assets/labels.txt',
numThreads: 1,
isAsset: true);
```

والآن أقترح تطبيق طريقة جديدة لتسجيل الصوت وتحليله. تحتوي طريقة startAudioRecognition() لحزمة tflite_audio على المعلمات التالية التي يجب تمريرها عبر الوسائط:

- **numOfInferences** : تحديد عدد الاستدلالات التي سيتم تكرارها.
- **inputType** : تحديد نوع إدخال الصوت.
- **sampleRate**: عدد العيناتفي الثانية.
- **recordingLength**: يحدد حجم مدخلات الموتر.
- **bufferSize** : مقدار الوقت اللازم لمعالجة أي إشارة صوتية واردة.

نظرًا لأن نموذج التصنيف هو نموذج GTM، فيجب عليك تحديد نوع الإدخال للتعرف على الصوت باسم "rawAudio". إذا كنت تستخدم النموذج الخاص بك (نموذج decodedwav) بدلاً من نموذج GTM، فيجب أن يكون نوع الإدخال "decodedWav". بالنسبة لـ SampleRate، القيم الموصى بهافي وثائق الحزمة هي 16000 أو 22050 أو 44100، وسيتم استخدام القيمة الأعلى لتحسين الدقة. يجب أن تكون قيمة RecordLength مساوية لإدخال الموتر tensor input ويجب أن يكون bufferSize نصف قيمة RegistrationLenght.

```
void _recorder() {
String recognition = "";
if (!_recording) {
setState(() => _recording = true);
result = TfliteAudio.startAudioRecognition(
numOfInferences: 1,
inputType: 'rawAudio',
sampleRate: 44100,
recordingLength: 44032,
bufferSize: 22016,
);
result.listen((event) {
recognition = event["recognitionResult"];
}).onDone(() {
setState(() {
_recording = false;
_sound = recognition.split(" ")[1];
});
});
}
}
```

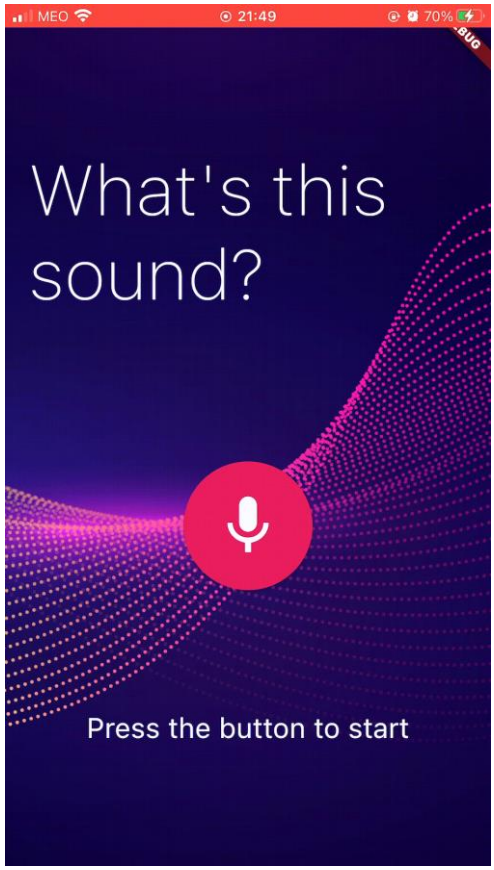
أيضًا، يجب الإعلان عن متغير دفق stream variable النتائج:

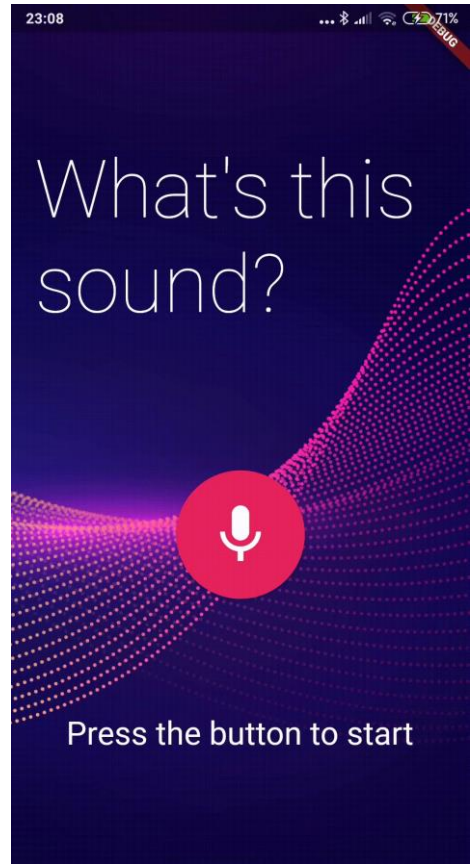
```
class _MyHomePageState extends State<MyHomePage> {
String _sound = "Press the button to start";
bool _recording = false;
Stream<Map<dynamic, dynamic>> result;
```

إذا كنت تريد إيقاف التعرف على الصوت stop audio recognition أثناء التنفيذ، يمكنك استخدام هذه الطريقة:

```
TfliteAudio.stopAudioRecognition();
```

وأخيرا، التطبيق جاهز للاستخدام! لاحظ أن التطبيق سيعمل فقط على الأجهزة المادية التي تعمل بنظامي Android وiOS كما ذكرنا من قبل. إذا كانت لديك بعض المشكلات أثناء عملية النشر على أجهزة iOS، فإنني أوصي بهذه الوثائق. لاحظ أيضًا أنه عند تشغيل التطبيق لأول مرة على جهاز يعمل بنظام iOS، فإنه سيتعطل لأنه يجب عليك الثقة به (التفاصيل موضحة هنا).





الكود

إذا كنت تريد عرض الكود بالكامل:

https://github.com/cmalbuquerque/audio_recognition_app

يرجى التحقق من ملفي الشخصي على GitHub!

الاستنتاجات

هناك بعض القيود المتعلقة بـ Tensorflow Lite:

- لم يعد Tensorflow يدعم المحاكيات ذات البنية x86_64 بسبب مشكلة CocoaPods.
- لا يدعم المكون الإضافي (tflite) Tensorflow Lite التعرف على الصوت حتى الآن.

يمكن تجاوز القيد الأول المعروف باستخدام الأجهزة المادية لاختبار وتصحيح التطبيقات باستخدام هذا البرنامج المساعد أو المكونات الإضافية التي تعتمد عليه. للتغلب على العائق الثاني، يعد tflite_audio مكونًا إضافيًا جيدًا لتحميل الصوت الخام ونماذج صوت wav التي تم فك ترميزها. ومع ذلك، هناك بعض الجوانب المتعلقة باستخدام نماذج Google Teachable Machine والتي من المهم الاحتفاظ بها. يعتمد هذا البرنامج الإضافي أيضًا على Tensorflow مما يعني أن التطبيقات ستتعطل عند استخدام الأجهزة الافتراضية (في Android وiOS).

في هذا السيناريو، يتم استخدام Tensorflowفي وضع عدم الاتصال حيث يتم تخصيص النموذج والتسميات المرتبطة بهفي التطبيق وبهذه الطريقة تعمل على تحسين حجم APK. علاوة على ذلك، يمكن استخدام Tensorflow عبر الإنترنت بدلاً من حفظ النموذج داخل التطبيق. بالإضافة إلى ذلك، كما ذكرنا أعلاه، فإن استخدام نماذج Google Teachable Machine سيؤدي إلى زيادة حجم التطبيقات. لتجنب ذلك، يوصى بإنشاء النماذج الخاصة بك.

المصدر:

https://carolinamalbuquerque.medium.com/audio-recognition-using-tensorflow-lite-in-flutter-application-8a4ad39964ae

11) انشاء تطبيق لتصنيف الرموز التعبيرية باستخدام فلاتر وتنسرفلو لايت Mobile Build an Emoji Classification App using Flutter with TensorFlow Lite

يتم دمج تقنية التعلم الآليفي تطبيقات الهاتف المحمول أكثر فأكثر اليوم. يساعد تكامله على تحسين الأداء العام ويمكنه إضافة وظائف مهمة إلى تجربة التطبيق. وبمرور الوقت، يمكن لنماذج تعلم الآلة أن تتعلم وتتحسن من تفاعلات المستخدم، مما يجعل هذه التجارب بديهية وذكية للغاية.

يهدف هذا البرنامج التعليمي إلى استكشاف استخدام مكتبة التعلم الآلي TensorFlowفي مشروع Flutter لإجراء تصنيف الصور. سنقوم هنا بتصنيف صور الرموز التعبيرية emojis.

للقيام بذلك، سنحتاج إما إلى العمل مع نموذج تم تدريبه مسبقًا أو تدريب النموذج الخاص بنا باستخدام أداة مثل Teachable Machine، وهي خدمة بدون تعليمات برمجية تقدمها Google. ستساعدنا مكتبة TensorFlow Liteفي تحميل النموذج وتطبيقه داخل تطبيقنا.

تتمثل الفكرةفي دمج البرنامج الإضافي TensorFlow lite واستخدامه لتصنيف صورة الرموز التعبيرية من حيث مشاعرها. بالإضافة إلى ذلك، سنحتاج إلى أن نكون قادرين على جلب الصورة من المعرض، ولهذا سنستفيد من مكتبة Image Picker. أصبح كل هذا أكثر بساطة من خلال نموذج ومكتبة TensorFlow Lite، والتي سنستكشفهافي هذا البرنامج التعليمي.

اذا هيا بنا نبدأ!

إنشاء مشروع Flutter جديد

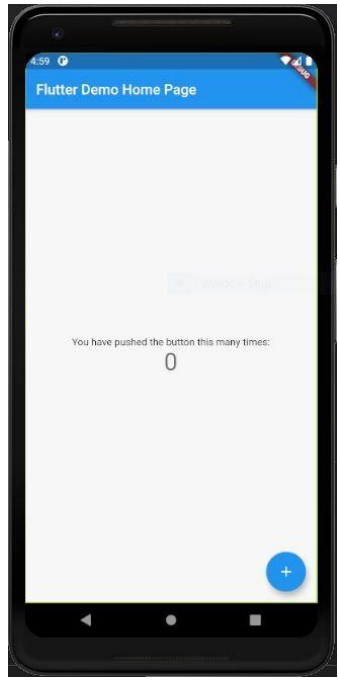
أولاً، نحتاج إلى إنشاء مشروع Flutter جديد. لذلك، تأكد من تثبيت Flutter SDK والمتطلبات الأخرى ذات الصلة بتطوير تطبيق Flutter بشكل صحيح. إذا تم إعداد كل شيء بشكل صحيح، لإنشاء مشروع، يمكننا ببساطة تشغيل الأمر التاليفي الدليل المحلي المطلوب:

```
flutter create productClassifier
```

بعد إعداد المشروع، يمكننا التنقل داخل دليل المشروع وتنفيذ الأمر التاليفي الوحدة الطرفية لتشغيل المشروع إمافي محاكي متاح أو على جهاز فعلي:

```
flutter run
```

بعد نجاح البناء، سنحصل على النتيجة التاليةفي شاشة المحاكي:



إنشاء الشاشة الرئيسية

سنقوم هنا بتنفيذ واجهة مستخدم تتيح للمستخدمين جلب صورة من مكتبة الجهاز وعرضها على شاشة التطبيق. لجلب الصورة من المعرض، سنستخدم مكتبة Image Picker. توفر هذه المكتبة وحدات لجلب مصدر الصورة والفيديو من كاميرا الجهاز والمعرض وما إلى ذلك.

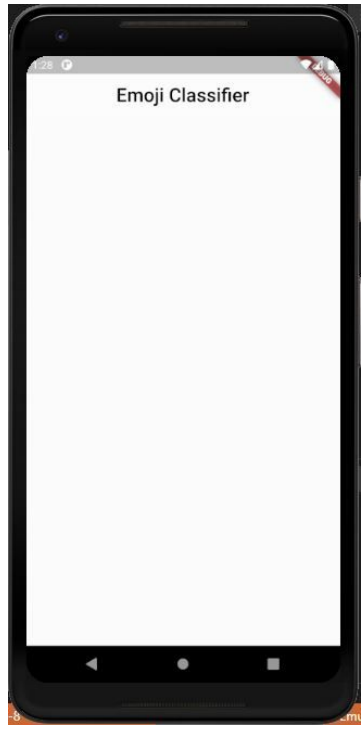
لكن أولاً، سنقوم بإنشاء شاشة رئيسية أساسية.في ملف main.dart، قم بإزالة الكود الافتراضي واستخدم الكود من المقتطف أدناه:

```
import 'package:flutter/material.dart';void main() {
  runApp(MyApp());
}class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: EmojiClassifier(),
    );
  }
}class EmojiClassifier extends StatefulWidget {
  @override
  _EmojiClassifierState createState() => _EmojiClassifierState();
}class _EmojiClassifierState extends State<EmojiClassifier> {
  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: Text(
          "Emoji Classifier",
          style: TextStyle(color: Colors.black, fontSize: 25),
        ),
        backgroundColor: Colors.white,
        elevation: 0,
      ),
      body: Container(
        color: Colors.white70,
      )
    );
  }
}
```

هنا، قمنا بإنشاء كائن فئة جديد EmojiClassifier والذي تمت تهيئته للخيار الرئيسي لعنصر واجهة المستخدم MaterialAppفي كائن فئة MyApp. من أجل البساطة، استخدمنا عنصر واجهة المستخدم Scaffold مع AppBar وحاوية نصية فارغة لشاشة فئة EmojiClassifier.

وبالتالي سنحصل على النتيجة كما هو موضح في الصورة أدناه:



إنشاء واجهة المستخدم لعرض الصور

بعد ذلك، سنقوم بوضع قسم عرض الصور على الشاشة، والذي سيحتوي على حاوية عرض الصور وزر لتحديد الصورة من المعرض.

لاختيار الصورة من معرض الجهاز، سنستخدم مكتبة image_picker. لتثبيته، نحتاج إلى نسخ النص الموجودفي مقتطف التعليمات البرمجية التالي ولصقهفي ملف pubspec.yaml الخاص بمشروعنا:

```
image_picker: ^0.6.7+14
```

الآن، نحن بحاجة إلى استيراد الحزم اللازمةفي ملف main.dart لمشروعنا:

```
import 'dart:io';import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
```

في ملف main.dart، نحتاج إلى تهيئة كائن File الذي يخزن كائن الصورة عند تحديده من معرض الصور الخاص بالجهاز:

```
File _imageFile;
```

الآن، سنقوم ببرمجة واجهة المستخدم، والتي ستمكن المستخدمين من اختيار صورة الرموز التعبيرية وعرضها. ستحتوي واجهة المستخدم على قسم لعرض الصور وزر يسمح للمستخدمين باختيار الصورة من المعرض. يتم توفير قالب واجهة المستخدم الشاملفي مقتطف الكود أدناه:

```
class _EmojiClassifierState extends State<EmojiClassifier> {
File _imageFile;
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
centerTitle: true,
title: Text(
"Emoji Classifier",
style: TextStyle(color: Colors.black, fontSize: 25),
),
backgroundColor: Colors.white,
elevation: 0,
),
body: Container(
color: Colors.white70,
child: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
color: Colors.black12,
offset: Offset(2, 2),
```

```
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
Image.file(_imageFile) :
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
FloatingActionButton(
tooltip: 'Select Image',
onPressed: (){},
child: Icon(Icons.add_a_photo,
size: 25,
color: Colors.blue,
),
backgroundColor: Colors.white,
),
],
),
),
)
);
}
}
```

وبالتالي سنحصل على النتيجة كما هو موضح في الصورة أدناه:

جلب وعرض صورة الرموز التعبيرية

في هذه الخطوة، سنقوم ببرمجة دالة تمكننا من الوصول إلى المعرض، واختيار صورة، ثم عرض الصورة في قسم عرض الصورة على الشاشة. يتم توفير التنفيذ الشامل للدالةفي مقتطف الكود أدناه:

```
Future selectImage() async {
final picker = ImagePicker();
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
setState(() {
if (image != null) {
_imageFile = File(image.path);
} else {
print('No image selected.');
}
});
}
```

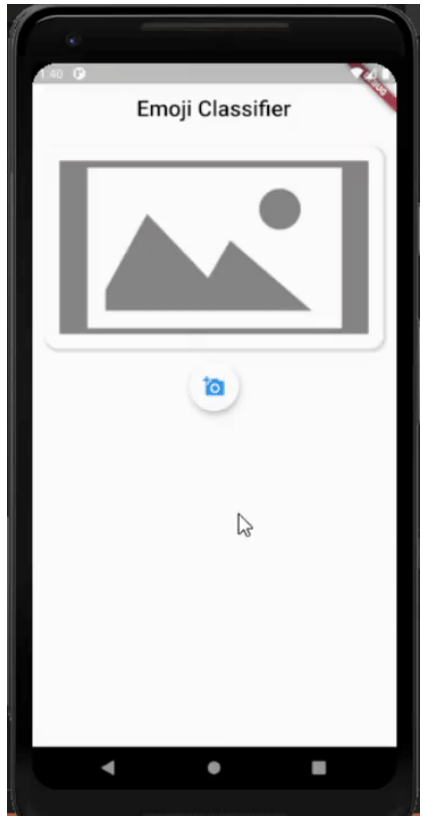
هنا، قمنا بتهيئة مثيل ImagePickerفي متغير picker واستخدمنا طريقة getImage التي يوفرها لجلب الصورة من المعرض إلى متغير الصورة.

بعد ذلك، قمنا بتعيين حالة imageFile_ على نتيجة الصورة التي تم جلبها باستخدام طريقة setState. سيؤدي هذا إلى إعادة عرض طريقة البناء الرئيسية وإظهار الصورة على الشاشة.

الآن، نحن بحاجة إلى استدعاء دالة SelectImageفي الخاصية onPressed لعنصر واجهة المستخدم FloatingActionButton، كما هو موضحفي مقتطف الكود أدناه:

```
FloatingActionButton(
tooltip: 'Select Image',
onPressed: (){
**selectImage();**
},
child: Icon(Icons.add_a_photo,
size: 25,
color: Colors.blue,
),
backgroundColor: Colors.white,
),
```

ومن ثم، يمكننا الآن جلب الصورة من معرضنا وعرضها على الشاشة:



إجراء تحديد الرموز التعبيرية باستخدام TensorFlow

الآن بعد أن قمنا بإعداد واجهة المستخدم، فقد حان الوقت لتكوين نموذج تصنيف صور الرموز التعبيرية الخاص بنا. سنقوم هنا بتصنيف صورة رمز تعبيري معين بناءً على النموذج وتحديد الرمز التعبيري الذي يمثله. من أجل ذلك، سنستخدم نموذج مصنف الرموز التعبيرية المُدرب مسبقًا والذي تم تدريبه باستخدام TensorFlow's Teachable Machine.

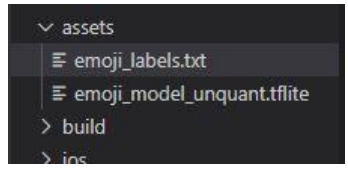
هنا، قدمنا النماذج المدربة بأنفسنا. يمكنك المضي قدمًا وتنزيل ملفات النماذج المدربة من هذا الرابط. يمكنك أيضًا تجربة التدريب على نموذجنا الخاص باستخدام Teachable Machine. يقدم النموذج المرتبط أعلاه الصور المدربة لمختلف الرموز التعبيرية بالإضافة إلى علامات التصنيف. ومن ثم، سنقوم بتحديد المشاعر الموجودةفي الرموز التعبيرية لأغراض الاختبار.

بعد التحميل سنحصل على ملفين:

- emoji_model_unquant.tflite
- emoji_labels.txt

يمكن لملف التصنيفات المتضمن التمييز بين ثلاثة مشاعر تعبيرية فقط، ولكن يمكنك بالتأكيد إنشاء نموذج يصنف المزيد. لقد فعلت ذلك بهذه الطريقة لأغراض الاختبار السريع.

نحتاج إلى نقل الملفين المقدمين إلى المجلد assets/. في دليل المشروع الرئيسي:



ثم نحتاج إلى تمكين الوصول إلى ملفات assets في pubspec.yaml:

```
assets:
- assets/emoji_labels.txt
- assets/emoji_model_unquant.tflite
```

تثبيت TensorFlow lite

بعد ذلك، نحتاج إلى تثبيت حزمة TensorFlow Lite. إنه مكون إضافي لـ Flutter للوصول إلى واجهات برمجة التطبيقات TensorFlow Lite. تدعم هذه المكتبة تصنيف الصور واكتشاف الكائنات وPix2Pix وDeeplab وPoseNet على نظامي iOS وAndroid.

لتثبيت البرنامج الإضافي، نحتاج إلى إضافة السطر التالي إلى ملف pubspec.yaml الخاص بمشروعنا:

```
tflite: ^1.1.1
```

بالنسبة لنظام Android، نحتاج إلى إضافة الإعداد التالي إلى كائن android الخاص بملف android/app/build.gradle/.:

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

هنا، نحتاج إلى التحقق لمعرفة ما إذا كان التطبيق قد تم إنشاؤه بشكل صحيح عن طريق تنفيذ أمر Flutter Run.

في حالة حدوث خطأ، قد نحتاج إلى زيادة الحد الأدنى لإصدار SDK إلى ≤19في ملف android/app/build.gradle/.

```
minSdkVersion 19
```

بمجرد إنشاء التطبيق بشكل صحيح، نكون مستعدين لتنفيذ نموذج TFLite الخاص بنا.

تنفيذ TensorFlow Lite لتصنيف الرموز التعبيرية

أولاً، نحتاج إلى استيراد الحزمة إلى ملف main.dart الخاص بنا، كما هو موضحفي مقتطف الكود أدناه:

```
import 'package:tflite/tflite.dart';
```

تحميل النموذج

الآن، نحن بحاجة إلى تحميل ملفات النموذجفي التطبيق. للقيام بذلك، سنقوم بتكوين دالة تسمى LoadEmojiModel. بعد ذلك، من خلال الاستفادة من طريقة loadModel التي يوفرها مثيل Tflite، سنقوم بتحميل ملفات النماذج الخاصة بنافي مجلد assets إلى تطبيقنا. نحتاج إلى تعيين معلمة النموذج model والتسميات labels داخل طريقة LoadModel، كما هو موضحفي مقتطف الكود أدناه:

```
Future loadEmojiModel() async {
Tflite.close();
String result;
result = await Tflite.loadModel(
model: "assets/emoji_model_unquant.tflite",
labels: "assets/emoji_labels.txt",
);
print(result);
}
```

بعد ذلك، نحتاج إلى استدعاء الدالة داخل طريقة initState بحيث يتم تشغيل الدالة بمجرد دخولنا إلى الشاشة:

```
@override
void initState() {
super.initState();
loadEmojiModel();
}
```

تنفيذ تصنيف الرموز التعبيرية

الآن، سنقوم بكتابة التعليمات البرمجية لتنفيذ تصنيف الرموز التعبيرية فعليًا. أولاً، نحتاج إلى تهيئة متغير لتخزين نتيجة التصنيف، كما هو موضحفي مقتطف الكود أدناه:

```
List _identifiedResult;
```

سيقوم متغير نوع قائمة identifiedResult_ بتخزين نتيجة التصنيف.

بعد ذلك، نحن بحاجة إلى انشاء دالة تسمى identImage التي تأخذ ملف الصورة كمعلمة. يتم توفير التنفيذ الشامل للدالةفي مقتطف الكود أدناه:

```
Future identifyImage(image) async {
_identifiedResult = null;
// Run tensorflowlite image classification model on the image
final List result = await Tflite.runModelOnImage(
```

```
path: image.path,
numResults: 6,
threshold: 0.05,
imageMean: 127.5,
imageStd: 127.5,
);
setState(() {
if (image != null) {
_imageFile = File(image.path);
_identifiedResult = result;
} else {
print('No image selected.');
}
});
}
```

هنا، استخدمنا طريقة runModelOnImage التي يوفرها مثيل Tflite لتصنيف الصورة المحددة. كمعلمات، قمنا بتمرير مسار الصورة وكمية النتيجة وعتبة التصنيف والتكوينات الاختيارية الأخرى لتصنيف أفضل. بعد التصنيف الناجح، قمنا بتعيين النتيجة إلى قائمة _identifiedResult .

الآن، نحن بحاجة إلى استدعاء الدالة داخل دالة SelectImage وتمرير ملف الصورة كمعلمة كما هو موضح في مقتطف الكود أدناه:

```
Future selectImage() async {
final picker = ImagePicker();
var image = await picker.getImage(source: ImageSource.gallery, maxHeight:
300);
identifyImage(image);
}
```

سيسمح لنا ذلك بضبط صورة الرموز التعبيرية على عرض الصورة بالإضافة إلى تصنيف صورة الرموز التعبيرية بمجرد تحديدها من المعرض.

الآن، نحن بحاجة إلى تكوين قالب واجهة المستخدم لعرض نتائج التصنيف. سنعرض نتيجة التصنيف بنمط البطاقة card style كقائمة أسفل عنصر واجهة المستخدم FloatingActionButton مباشرةً.

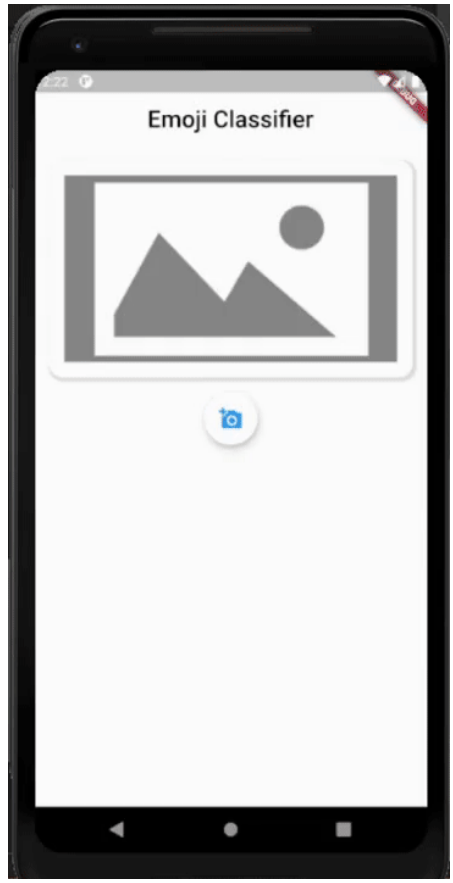
يتم توفير تنفيذ واجهة المستخدم الشاملة للشاشة في مقتطف الكود أدناه:

```
body: Container(
color: Colors.white70,
child: Center(
child: Column(
children: [
Container(
margin: EdgeInsets.all(15),
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(
Radius.circular(15),
),
border: Border.all(color: Colors.white),
boxShadow: [
BoxShadow(
color: Colors.black12,
```

```
offset: Offset(2, 2),
spreadRadius: 2,
blurRadius: 1,
),
],
),
child: (_imageFile != null)?
Image.file(_imageFile) :
Image.network('<https://i.imgur.com/sUFH1Aq.png>')
),
FloatingActionButton(
tooltip: 'Select Image',
onPressed: (){
selectImage();
},
child: Icon(Icons.add_a_photo,
size: 25,
color: Colors.blue,
),
backgroundColor: Colors.white,
),
SizedBox(height: 20),
SingleChildScrollView(
child: Column(
children: _identifiedResult != null ? [
Text(
"Result : ",
style: TextStyle(
color: Colors.black,
fontSize: 18.0,
fontWeight: FontWeight.bold,
fontStyle: FontStyle.italic
),
),
Card(
elevation: 1.0,
color: Colors.white,
child: Container(
width: 100,
margin: EdgeInsets.all(10),
child: Center(
child: Text(
"${_identifiedResult[0]["label"]}",
style: TextStyle(
color: Colors.black,
fontSize: 22.0,
fontWeight: FontWeight.w900),
),
),
),
)
]
:[]
),
),
],
),
),
)
```

هنا، أسفل عنصر واجهة المستخدم FloatingActionButton مباشرةً، قمنا بتطبيق عنصر واجهة المستخدم SingleChildScrollView بحيث يكون المحتوى الموجود بداخله قابلاً للتمرير. بعد ذلك، استخدمنا عنصر واجهة مستخدم Column لسرد عناصر واجهة المستخدم بداخله أفقيًا. داخل عنصر واجهة مستخدم Column، احتفظنا بنتيجة التصنيف داخل عنصر واجهة مستخدم البطاقة Card.

وبالتالي سنحصل على النتيجة كما هو موضح في العرض التوضيحي أدناه:



يمكننا أن نرى أنه بمجرد اختيار الصورة من المعرض، يتم عرض نتيجة المشاعر التعبيرية المصنفة على الشاشة أيضًا. وبالمثل، يمكنك إضافة المزيد من التسميات إلى ملف تسميات النموذج واختبار صور الرموز التعبيرية المختلفة. ويمكننا أيضًا عرض نتيجة نسبة التصنيف classification percentage (أي ثقة النموذج في تنبؤاته). ولكن بالنسبة لهذا النوع من التحليل، يحتاج النموذج إلى التدريب باستخدام المزيد من الصور التعبيرية.

وهذا كل شيء! لقد نجحنافي تنفيذ نموذج أولي لـ Emoji Classifierفي تطبيق Flutter باستخدام TensorFlow Lite.

الاستنتاجات

في هذا البرنامج التعليمي، تعلمنا كيفية إجراء تصنيف لصور الرموز التعبيرية للتعرف على 3 رموز تعبيرية مختلفة وتصنيفها بناءً على المشاعر التي تعبر عنها.

تم توفير ملفات النماذج لهذا البرنامج التعليمي، ولكن يمكنك إنشاء نماذجك المدربة باستخدام خدمات مثل Google's Teachable Machine. جعلت مكتبة TensorFlow Lite التحميل الشامل للصور وتصنيفها أمرًا بسيطًا من خلال توفير طرق ومثيلات مختلفة.

يتمثل التحدي الآنفي تدريب نماذج تصنيف الرموز التعبيرية الخاصة بك باستخدام الرموز التعبيرية المفضلة لديك وتطبيقها على المشروع الحالي. يمكن استخدام مكتبة TensorFlow Lite لتنفيذ مهام تعلم الآلة المعقدة الأخرىفي تطبيقات الهاتف المحمول، مثل اكتشاف الكائنات وتقدير الوضعية واكتشاف الإيماءات والمزيد.

يتم توفير التنفيذ الشامل للكودفي GitHub.

المصدر:

https://heartbeat.comet.ml/emoji-classifier-with-flutter-and-tensorflow-lite-51d026ac2cc

12) انشاء تطبيق للكشف عن الالتهاب الرئوي باستخدام فلاتر وتنسرفلو لايت Build a Pneumonia Detection Mobile App using Flutter with TensorFlow Lite

لقد كان تطوير تطبيقات الهاتف المحمول متعدد المنصات Cross-platform mobile app development دائمًا حلمًا لمطوري تطبيقات iOS وAndroid. حتى مع وجود أدوات مثل Cordova وIonic، ظل هذا الحلم بعيد المنال لفترة طويلة، حيث أدت هذه الأدوات إلى أداء لم يكن قريبًا من الأداء الأصلي. يجادل الكثيرون بأن React Native هو أفضل إطار عمل لنفس الشيء، ولكن منذ إطلاق Flutter، بدأ هذا يتغير.

Flutter عبارة عن مجموعة أدوات واجهة مستخدم مفتوحة المصدر من Google تعتمد على لغة برمجة Dart. لقد تم تصميمه للسماح للمطورين بإنشاء تطبيقات الهاتف المحمول والويب وسطح المكتب المجمعة محليًا من قاعدة تعليمات برمجية واحدة، كما أنه يتمتع بالكثير من المزايا مقارنة بأطر العمل الأخرى عبر الأنظمة الأساسية.

إنه موثق جيداً، ومدعوم بـ IDEs، ويوفر (قريبًا) من الأداء الأصلي، ويحتوي على مجموعة من الميزات مثل إعادة التحميل السريع وإعادة التشغيل السريع التي تجعل عمليات تطوير التطبيقات الأساسية أكثر ملاءمة.

في هذا البرنامج التعليمي، سنقوم بإنشاء تطبيق باستخدام Flutter الذي يعمل على كل من نظامي التشغيل iOS وAndroid (باستخدام نفس قاعدة التعليمات البرمجية بالضبط) لتصنيف صور الأشعة المقطعية على الصدر chest CTs بناءً على ما إذا كانت تظهر عليها علامات الالتهاب الرئوي pneumonia أم لا. سنفعل ذلك باستخدام نموذج TensorFlow Lite.

الحصول على النموذج

في هذا المثال، استخدمت مجموعة البيانات هذه من Kaggle وتدربت عليها باستخدام Teachable Machine. يمكنك تدريبه بنفسك أو تنزيله هنا.

سيحتوي على ملفين: model_unquant.tflite وlabels.txt

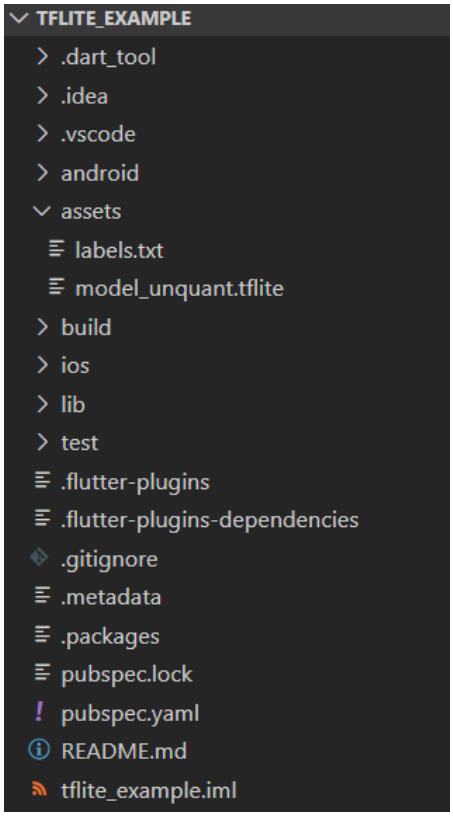
بناء التطبيق

يمكنك إنشاء مشروع Flutter الخاص بك إما من خلال لوحة أوامر VS Code إذا كان لديك ملحق Flutter مثبتًا (مستحسن)، أو قم بتشغيل الأمر التالي في الدليل الذي تريد إنشاء مشروعك فيه:

```
flutter create -i swift -a kotlin tflite_example
```

يؤدي هذا إلى إنشاء مشروع Flutter المسمى tflite_example، باستخدام Swift لنظام iOS وKotlin لنظام Android.

بعد ذلك، نقوم بإنشاء مجلد يسمى "assets" في دليل المشروع ونضيف إليه ملفات نموذج TFLite.



هذا هو الشكل الذي يجب أن تبدو عليه بُنية المجلد

التكوينات والتبعيات

قبل إنشاء التطبيق الفعلي، نحتاج إلى إضافة تبعياتنا إلى ملف pubspec.yaml عن طريق إضافة الأسطر التالية:

```
dependencies:
image_picker:
tflite:
```

نحتاج الآن إلى إضافة ملفات النموذج إلى المشروع عن طريق إضافة ما يلي في نفس الملف pubsbec.yaml:

```
flutter:
assets:
```

```
- assets/labels.txt
- assets/model_unquant.tflite
```

لقد انتهينا تقريبًا من التكوينات الآن، نحتاج فقط إلى إجراء بعض التغييراتفي ملف android/app/build.gradle. ضمن كتلة android أضف ما يلي:

```
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
```

يؤدي هذا إلى منع ضغط النموذج عند تجميعه لمنصة Android.

كتابة الكود

الآن يأتي الجزء المثير للاهتمام، وهو كتابة التطبيق الفعلي. نبدأ بمسح الكود الحاليفي ملف main.dart.

نحتاج إلى استيراد المكتبات والحزم المطلوبة:

```
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';
import 'package:tflite/tflite.dart';
```

نقوم باستيراد image_picker لمساعدتنافي جلب الصور من الجهاز، وtflite لتحميل النموذج وإجراء التصنيف، وتصميم المواد القياسي وحزم الإدخال/الإخراج.

يحتاج كل تطبيق Flutter إلى طريقة رئيسية لتشغيل التطبيق باستخدام فئة من النوع StatelessWidget . وندمج ذلك بإضافة الأسطر التالية:

```
void main() => runApp(MyApp());class MyApp extends StatelessWidget {
@override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
           appBar: AppBar(
             title: Text('TFLite Example'),
             backgroundColor: Colors.transparent
             ),
            body: Center(
              child: MyImagePicker()
              )
            )
          );
  }
}
```

لقد أنشأنا هنا طريقة رئيسية تسمى عند بدء تشغيل التطبيق. تستدعي هذه الطريقة الرئيسية بعد ذلك فئة MyApp، التي تمتد إلى StatelessWidget، مما يجعل التطبيق بأكمله عبارة عن عنصر واجهة مستخدم (كل شيء عبارة عن عنصر واجهة مستخدمفي Flutter).

في فئة MyApp، نقوم ببناء عنصر واجهة مستخدم Scaffold من مكتبة المواد الافتراضية، والتي توفر عنوان شريط التطبيق ولونًا وأيضًا خاصية نص تحتوي على شجرة عناصر واجهة المستخدم للتطبيق.

في حالتنا، تحتوي خاصية الجسم على فئة MyImagePicker، والتي سنحددها بعد قليل.

في Flutter، تكون فئات StatelessWidget غير قابلة للتغيير immutable، مما يعني أنه لا يمكن تغيير أي شيء بداخلها، لذلك لكي يؤدي التطبيق وظائف، هناك حاجة إلى عناصر واجهة مستخدم ذات حالة. يتم إنشاء هذه من خلال فئة StatefulWidget، والتي تقوم أيضًا بإنشاء مثيل لفئة الحالة State class.

نقوم الآن بإنشاء StatefulWidget وفئة الحالة:

```
class MyImagePicker extends StatefulWidget {
@override
MyImagePickerState createState() => MyImagePickerState();
}
```

هذه هي فئة StatefulWidget، والتي تقوم الآن بمثيل فئة MyImagePickerState. سنحدد هذا الآن:

```
class MyImagePickerState extends State {}
```

الآن بعد أن حددنا جميع الفئات المطلوبة، والفئة الرئيسية MyApp، وفئة عناصر واجهة المستخدم ذات الحالة MyImagePicker، وفئة الحالة MyImagePickerState، سننتقل ونكتب الدالة الأساسية لتطبيقنا.

نحن بحاجة إلى الإعلان عن ثلاثة متغيرات:

- **imageURI**: لحفظ الصورة المحملة من الجهاز.
- **path**: لتخزين مسار الصورة المحملة (يستخدم لإجراء التصنيف).
- **result**: لحفظ نتيجة التصنيف من نموذج TFLite.

```
File imageURI;
String result;
String path;
```

الآن بعد أن أصبح لدينا المتغيرات نحتاج إلى دوال الكتابة للحصول على الصورة من الجهاز وإجراء التصنيف مسبقًا. بالنسبة لهذه الدوال، سنستخدم Future وهي أداة برمجة غير متزامنةفي Flutter. يسمح لك Future بتشغيل العمل بشكل غير متزامن لتحرير أي مؤشرات ترابط أخرى.

نكتب دالتنا الأولى عن طريق إضافة ما يلي إلى فئة MyImagePickerState:

```
Future getImageFromCamera() async {
  var image = await ImagePicker.pickImage(source: ImageSource.camera);
setState(() {
    imageURI = image;
```

```
    path = image.path;
  });
}
```

في الكود أعلاه، أعلنا عن متغير var من النوع ImagePicker (الذي حصلنا عليه من تبعيتنا) مع مصدر من النوع camera. لذلك، عند استدعاء هذه الدالة، يلتقط المستخدم صورة بالكاميرا.

نقوم بعد ذلك بتنفيذ دالة setState() ، التي تُعلم إطار العمل بأن الحالة الداخلية للكائن قد تغيرت، وتحفظ الصورة التي تم النقر عليهافي متغير imageURI، ومسار الصورة إلى المتغير المقابل.

وبالمثل، نقوم بتنفيذ دالة الحصول على صورة من المعرض:

```
Future getImageFromGallery() async {
  var image = await ImagePicker.pickImage(source: ImageSource.gallery);
setState(() {
    imageURI = image;
    path = image.path;
  });
}
```

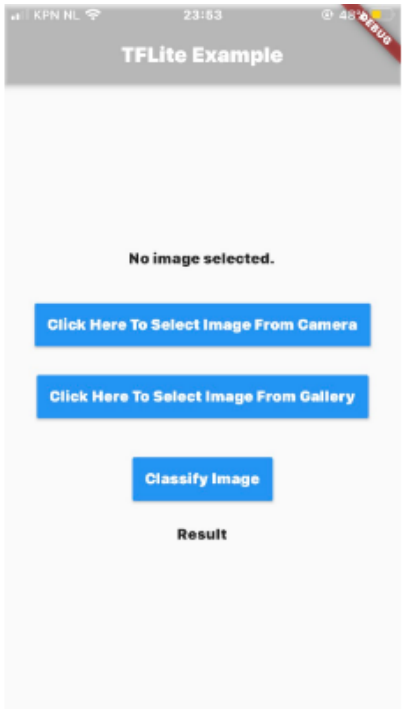
والآن بعد أن أصبح لدينا دوال للحصول على الصورة من الجهاز، نحتاج إلى كتابة دالة أخيرة لإجراء التصنيف على الصورة المحددة.

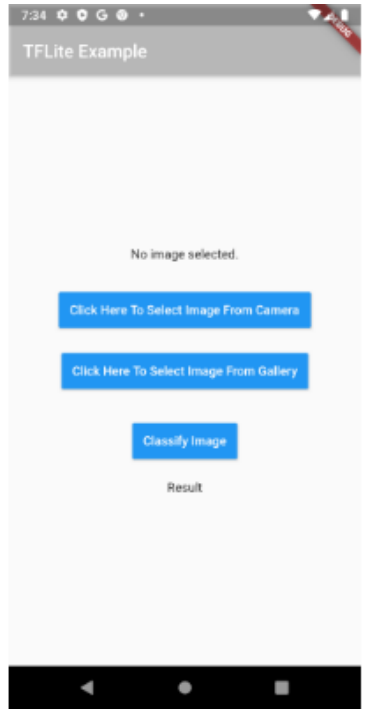
نقوم بذلك عن طريق إضافة ما يلي إلى نفس الفئة:

```
Future classifyImage() async {
  await Tflite.loadModel(model: "assets/model_unquant.tflite",labels:
"assets/labels.txt");
  var output = await Tflite.runModelOnImage(path: path);  setState(() {
    result = output.toString();
  });
}
```

في هذه الدالة، نقوم أولاً بتحميل النموذج باستخدام Tflite.loadModel() ثم نقوم بتشغيل النموذج على الصورة التي حددها المستخدم باستخدام المسار path. ثم نقوم بتخزين النتيجةفي متغير الإخراج output. نستخدم المسار بدلاً من الصورة نفسها، لأن حزمة tflite لا يمكنها تشغيل النموذج على أنواع بيانات الملف File. في setState()، نقوم بتخزين الإخراج كسلسلة نصية string إلى متغير النتيجة.

لدينا أخيرًا جميع الدوال التي نحتاجها ـ والآن حان وقت إنشاء واجهة التصميم layout.





تخطيط التطبيق (ليس جميلًا، لكنه يعمل)

تعد واجهة التصميم واضحة تمامًا ــ صورة يتم تحميلها من الجهاز، وزرين لتحميل الصورة من الكاميرا أو من المعرض، وزر لإجراء التصنيف، ونص نتيجة.

لإنشاء واجهة التصميم ، نضيف دالة بناء ويدجت التاليةفي فئة MyImagePickerState:

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
      imageURI == null
        ? Text('No image selected.')
        : Image.file(imageURI, width: 300, height: 200, fit: BoxFit.cover),
Container(
        margin: EdgeInsets.fromLTRB(0, 30, 0, 20),
        child: RaisedButton(
          onPressed: () => getImageFromCamera(),
          child: Text('Click Here To Select Image From Camera'),
          textColor: Colors.white,
          color: Colors.blue,
          padding: EdgeInsets.fromLTRB(12, 12, 12, 12),
        )),      Container(
        margin: EdgeInsets.fromLTRB(0, 0, 0, 0),
        child: RaisedButton(
          onPressed: () => getImageFromGallery(),
```

```
          child: Text('Click Here To Select Image From Gallery'),
          textColor: Colors.white,
          color: Colors.blue,
          padding: EdgeInsets.fromLTRB(12, 12, 12, 12),
        )),      Container(
        margin: EdgeInsets.fromLTRB(0, 30, 0, 20),
        child: RaisedButton(
          onPressed: () => classifyImage(),
          child: Text('Classify Image'),
          textColor: Colors.white,
          color: Colors.blue,
          padding: EdgeInsets.fromLTRB(12, 12, 12, 12),
        )),      result == null
        ? Text('Result')
        : Text(result)  ]))
  );
}
```

هذا كل شيء، تم الانتهاء من التطبيق لدينا! إنه ليس مصممًا بالكامل، ولكنه يوضح أن تطبيقات متعددة المنصات يمكنها الاستفادة من قوة التعلم الآلي على الجهاز دون أي واجهات برمجة التطبيقات، بينما تعمل بنفس سرعة التطبيقات الأصلية native apps (وأحيانًا أسرع).

يمكنك العثور على كود المصدر الكامل هنا. انطلق واجعل هذا التطبيق جميلًا وجرب ما يقدمه Flutter وTensorFlow Lite.

المصدر:

https://heartbeat.comet.ml/building-a-cross-platform-image-classifier-with-flutter-and-tensorflow-lite-c7789af9b33a

13) انشاء تطبيق للتعرف على ايماءات اليد باستخدام فلاتر وتنسرفلو لايت Build a Hand Gesture Detection Mobile App using Flutter with TensorFlow Lite

قبل أن نبدأ، دعونا نلخص ما تعلمتهفي البرامج التعليمية السابقة.في البرنامج التعليمي الأول إنشاء نموذج ناجح باستخدام Flutter وVGG16: دليل شامل، قمنا بتغطية أساسيات إعداد البيانات والتدريب على شبكة عصبية تلافيفية CNN مخصصة باستخدام إطار عمل Keras.في البرنامج التعليمي الثاني تعزيز نموذج تصنيف الصور الخاص بك باستخدام VGG-16 المُدرب مسبقًا، قمنا بدمج نموذج VGG-16 المُدرب مسبقًافي مهمة الكشف المخصصة لدينا، مما أدى إلى تعزيز كبيرفي الأداء، حتى مع مجموعة بيانات صغيرة.

الآن، دعنا نتعمقفي البرنامج التعليمي النهائي ونرى كيف يمكنك إضفاء الحيوية على نموذج تصنيف الصور الخاص بكفي تطبيق Flutter. سأستخدم إطار عمل Flutter لبناء واجهة مستخدم تسمح للمستخدم باستخدام كاميرا الجهاز لاكتشاف إيماءة يده (حجر rock أو ورق paper أو مقص scissors)، ثم استخدم نموذج تصنيف الصور الخاص بنا للتنبؤ بالإيماءة الصحيحة.

للبدء، ستحتاج إلى إنشاء مشروع Flutter جديد واستيراد التبعيات الضرورية. سنحتاج أيضًا إلى إضافة كود للتعامل مع إدخال الكاميرا وعرض الإخراج على الشاشة.

بعد ذلك، ستحتاج إلى دمج نموذج تصنيف الصور الخاص بكفي تطبيق Flutter. ستقوم بذلك عن طريق تحميل النموذجفي الذاكرة ثم استخدامه للتنبؤ بتدفق الكاميرا.

لذلك لا مزيد من النظرية، مجرد كود.

إعداد النموذج

نظرًا لأنك قمت بإنشاء نموذجك وتدريبه باستخدام إطار عمل Keras، المبني على TensorFlow، يمكنك تحويل نموذجك إلى تنسيق TensorFlow وتشغيله على تطبيق Flutter.

لتحويل نموذج Keras إلى تنسيق TensorFlow، يمكنك استخدام الدالة tf.lite.TFLiteConverter.from_keras_model(). تقوم هذه الدالة بإنشاء كائن محول للنموذج الخاص بك، والذي يمكن استخدامه بعد ذلك لتحويل النموذج إلى تنسيق TensorFlow.

بالإضافة إلى ذلك، يمكنك أيضًا تحويل نموذج TensorFlow إلى تنسيق TensorFlow Lite لتحسين الأداء على الأجهزة المحمولة. TensorFlow Lite هو إصدار خفيف الوزن من

TensorFlow تم تحسينه ليناسب الأجهزة المحمولة والمدمجة. فهو يتيح أوقات استدلال أسرع وأحجام نماذج أصغر، مما يجعله مثاليًا لتطبيقات الهاتف المحمول.

فيما يلي مثال لكيفية تحويل نموذج Keras إلى تنسيق TensorFlow Lite باستخدام الدالة from_keras_model():

```
def convert(model):
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    tflite_model = converter.convert()

    with open('rock_paper_scissors_model.tflite', 'wb') as f:
        f.write(tflite_model)
```

هذا كل شيء، الآن لديك كل شيء جاهز لبدء إنشاء تطبيق Flutter الخاص بك.

جزء Flutter

للبدءفي دمج نموذج تصنيف الصور الخاص بك مع تطبيق Flutter، تحتاج إلى إنشاء مشروع Flutter جديد وإضافة بعض التبعيات إلى ملف pubspec.yaml الخاص بنا. ستحتاج أيضًا إلى إنشاء مجلد assetsفي جذر مشروعنا ووضع ملف rock_paper_scissors_model.tflite الذي تم إنشاؤه من الخطوة السابقة فيه.

فيما يلي الخطوات التي يجب اتباعها:

1. قم بإنشاء مشروع Flutter جديد باستخدام IDE المفضل لديك أو قم بتشغيل Flutter وقم بإنشاء myappفي الجهاز.
2. افتح الملف pubspec.yaml وأضف التبعيات التالية:

```
dependencies:
  flutter:
    sdk: flutter
  tflite_flutter: ^0.9.0
  tflite_flutter_helper: ^0.3.1
  camera: ^0.9.4+5
```

توفر حزمتي tflite_flutter وtflite_flutter_helper الأدوات اللازمة لتشغيل نموذج TensorFlow Lite الخاص بك على الأجهزة المحمولة والكاميرا التي تُستخدم الحزمة للوصول إلى كاميرا الجهاز لالتقاط الإطارات منها.

3. قم بإنشاء مجلد assetsفي جذر مشروعك إذا لم يكن موجودًا بالفعل.
4. ضع ملف rock_paper_scissors_model.tflite الذي قمت بإنشائهفي الخطوة السابقةفي مجلد assets.
5. لا تنس تسجيل assetsفي pubspec.yaml

```
flutter:
  assets:
    - assets/rock_paper_scissors_model.tflite
  uses-material-design: true
```

بعد الانتهاء من ذلك، يمكنك البدءفي العمل على إنشاء واجهة المستخدم وتشغيل النموذج على الكاميرا.

قم بإنشاء عنصر واجهة المستخدم StatefulWidget وقم بتهيئة الكاميرات بطريقة initState

```
class ScannerScreen extends StatefulWidget {
  @override
  _ScannerScreenState createState() => _ScannerScreenState();
}

class _ScannerScreenState extends State<ScannerScreen> {
  late CameraController cameraController;

  bool initialized = false;
  bool isWorking = false;

  @override
  void initState() {
    super.initState();
    initialize();
  }

  Future<void> initialize() async {
    final cameras = await availableCameras();
    // Create a CameraController object
    cameraController = CameraController(
      cameras[0], // Choose the first camera in the list
      ResolutionPreset.medium, // Choose a resolution preset
    );

    // Initialize the CameraController and start the camera preview
    await cameraController.initialize();
    // Listen for image frames
    await cameraController.startImageStream((image) {
      // Make predictions only if not busy
      if (!isWorking) {
        processCameraImage(image);
      }
    });

    setState(() {
      initialized = true;
    });
  }
```

بعد ذلك يجب عليك إنشاء Classifier وهو الكلاس الذي سيكون مسؤولاً عن عملية التصنيف.

```
enum DetectionClasses { rock, paper, scissors, nothing }

class Classifier {
  /// Instance of Interpreter
  late Interpreter _interpreter;

  static const String modelFile = "rock_paper_scissors_model.tflite";

  /// Loads interpreter from asset
  Future<void> loadModel({Interpreter? interpreter}) async {
```

```
    try {
      _interpreter = interpreter ??
          await Interpreter.fromAsset(
            modelFile,
            options: InterpreterOptions()..threads = 4,
          );

      _interpreter.allocateTensors();
    } catch (e) {
      print("Error while creating interpreter: $e");
    }
  }

  /// Gets the interpreter instance
  Interpreter get interpreter => _interpreter;
}
```

في LoadModel، الطريقة التي أقوم بها هي تهيئة Interpreter من حزمة tflite_flutter وتحميل النموذج الخاص بي فيها. يمكنك أيضًا التحقق من معلومات النموذج الخاص بك باستخدام _interpreter . على سبيل المثال، يمكنك التحقق من شكل الإدخال للنموذج الخاص بك باستخدام السطر التالي:

```
_interpreter.getInputTensor(0).shape // Will return [1, 150, 150, 3]
```

بعد ذلك، يمكنك إنشاء مثيل Classifier في _ScannerScreenState واستدعاء LoadModel في طريقة initialize.

```
class _ScannerScreenState extends State<ScannerScreen> {
  final classifier = Classifier();

  ...

  Future<void> initialise() async {
    await classifier.loadModel();
    ...
```

لذلك لا يمكنك إضافة العملية CameraImage الطريقة التي ستلتقط الصور من الكاميرا إطارًا تلو الآخر وتقوم بالتنبؤ بناءً عليها.

```
Future<void> processCameraImage(CameraImage cameraImage) async {
  setState(() {
    isWorking = true;
  });

  DetectionClasses results = await classifier.predict(convertedImage);

  if (detected != result) {
    setState(() {
      detected = results;
    });
  }

  setState(() {
    isWorking = false;
  });
}
```

يجب على Clarifier تحويل CameraImage إلى قائمة قيم البكسلات بالشكل [1، 150، 150، 3] ووضعهافي Interpreter. للقيام بذلك، يجب عليك أولاً تحويل صورة الكاميرا إلى package:image/image.dart example للمثيل Image. للقيام بذلك يمكنك استخدام الكود التالي:

```
import 'package:camera/camera.dart';
import 'package:image/image.dart' as imageLib;

/// ImageUtils
class ImageUtils {
  /// Converts a [CameraImage] in YUV420 format to [imageLib.Image] in RGB
format
  static imageLib.Image convertYUV420ToImage(CameraImage cameraImage) {
    final int width = cameraImage.width;
    final int height = cameraImage.height;

    final int uvRowStride = cameraImage.planes[1].bytesPerRow;
    final int uvPixelStride = cameraImage.planes[1].bytesPerPixel!;

    final image = imageLib.Image(width, height);

    for (int w = 0; w < width; w++) {
      for (int h = 0; h < height; h++) {
        final int uvIndex =
            uvPixelStride * (w / 2).floor() + uvRowStride * (h /
2).floor();
        final int index = h * width + w;

        final y = cameraImage.planes[0].bytes[index];
        final u = cameraImage.planes[1].bytes[uvIndex];
        final v = cameraImage.planes[2].bytes[uvIndex];

        image.data[index] = ImageUtils.yuv2rgb(y, u, v);
      }
    }
    return image;
  }

  /// Convert a single YUV pixel to RGB
  static int yuv2rgb(int y, int u, int v) {
    // Convert yuv pixel to rgb
    int r = (y + v * 1436 / 1024 - 179).round();
    int g = (y - u * 46549 / 131072 + 44 - v * 93604 / 131072 +
91).round();
    int b = (y + u * 1814 / 1024 - 227).round();

    // Clipping RGB values to be inside boundaries [ 0 , 255 ]
    r = r.clamp(0, 255);
    g = g.clamp(0, 255);
    b = b.clamp(0, 255);

    return 0xff000000 |
    ((b << 16) & 0xff0000) |
    ((g << 8) & 0xff00) |
    (r & 0xff);
  }
}
```

بعد إضافة طريقة predict إلى Classifier .

```
import 'dart:typed_data';

import 'package:image/image.dart' as img;
import 'package:rock_paper_scissors_mobile/classes.dart';
import 'package:tflite_flutter/tflite_flutter.dart';

class Classifier {

  ...

  Future<DetectionClasses> predict(img.Image image) async {
    img.Image resizedImage = img.copyResize(image, width: 150, height:
150);

    // Convert the resized image to a 1D Float32List.
    Float32List inputBytes = Float32List(1 * 150 * 150 * 3);
    int pixelIndex = 0;
    for (int y = 0; y < resizedImage.height; y++) {
      for (int x = 0; x < resizedImage.width; x++) {
        int pixel = resizedImage.getPixel(x, y);
        inputBytes[pixelIndex++] = img.getRed(pixel) / 127.5 - 1.0;
        inputBytes[pixelIndex++] = img.getGreen(pixel) / 127.5 - 1.0;
        inputBytes[pixelIndex++] = img.getBlue(pixel) / 127.5 - 1.0;
      }
    }

    // Reshape to input format specific for model. 1 item in list with
pixels 150x150 and 3 layers for RGB
    final input = inputBytes.reshape([1, 150, 150, 3]);

    // Output container
    final output = Float32List(1 * 4).reshape([1, 4]);

    // Run data throught model
    interpreter.run(input, output);

    // Get index of maxumum value from outout data. Remember that models
output means:
    // Index 0 - rock, 1 - paper, 2 - scissor, 3 - nothing.
    final predictionResult = output[0] as List<double>;
    double maxElement = predictionResult.reduce(
      (double maxElement, double element) =>
          element > maxElement ? element : maxElement,
    );
    return DetectionClasses.values[predictionResult.indexOf(maxElement)];
  }
}
```

يمكنك الآن إنشاء واجهة مستخدم لتطبيقك:

```
class _ScannerScreenState extends State<ScannerScreen> {

  ...

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Flutter Camera Demo'),
```

```
      ),
      body: initialized
          ? Column(
              children: [
                SizedBox(
                  height: MediaQuery.of(context).size.width,
                  width: MediaQuery.of(context).size.width,
                  child: CameraPreview(cameraController),
                ),
                Text(
                  "Detected: ${detected.label}",
                  style: const TextStyle(
                    fontSize: 28,
                    color: Colors.blue,
                  ),
                ),
              ],
            )
          : const Center(child: CircularProgressIndicator()),
    );
  }
```

هذا كل شيء. ولكن إذا قمت بتشغيله فمن المحتمل أن تلاحظ أن التطبيق يتأخر. ذلك لأن عملية التشغيل معقدة وتستهلك الكثير من الموارد في UI-Isolate. لكي تعمل بسلاسة وسرعة، عليك أن تفكر في نقل عملية الحساب إلى عزل منفصل.

عزل الحساب

أولاً قم بإنشاء IsolateUtils باستخدام الكود التالي:

```
/// Bundles data to pass between Isolate
class IsolateData {
  CameraImage cameraImage;
  int interpreterAddress;
  SendPort responsePort;

  IsolateData({
    required this.cameraImage,
    required this.interpreterAddress,
    required this.responsePort,
  });
}

class IsolateUtils {
  static const String DEBUG_NAME = "InferenceIsolate";

  late Isolate _isolate;
  final ReceivePort _receivePort = ReceivePort();
  late SendPort _sendPort;

  SendPort get sendPort => _sendPort;

  Future<void> start() async {
    _isolate = await Isolate.spawn<SendPort>(
      entryPoint,
      _receivePort.sendPort,
      debugName: DEBUG_NAME,
    );
```

```
    _sendPort = await _receivePort.first;
  }

  static void entryPoint(SendPort sendPort) async {
    final port = ReceivePort();
    sendPort.send(port.sendPort);

    await for (final IsolateData isolateData in port) {
      Classifier classifier = Classifier();
      // Restore interpreter from main isolate
      await classifier.loadModel(interpreter:
Interpreter.fromAddress(isolateData.interpreterAddress));

      final convertedImage =
ImageUtils.convertYUV420ToImage(isolateData.cameraImage);
      DetectionClasses results = await classifier.predict(convertedImage);
      isolateData.responsePort.send(results);
    }
  }

  void dispose() {
    _isolate.kill();
  }
}
```

يتم الحصول على IsolateData من port وتهيئة Classifier باستخدام Interpreter للعزل الرئيسي باستخدام عنوانه (Interpreter.fromAddress(isolateData.interpreterAddress

بعد الانتهاء من نفس المعالجة المسبقة باستخدام CameraImage كما كان من قبل ووضعهافي Classifier. DetectionClasses ترسل النتيجة مرة أخرى إلى العزلة الرئيسية باستخدام (isolateData.responsePort.send(results; .

بعد الانتهاء من ذلك، يتعين عليك إدخال بعض التحسينات على كود ScannerScreenState_ الخاص بك.

قم بإنشاء مثيل IsolateUtils واستدعاء (isolateUtils.start في طريقة initialize.

إضافة طريقة inference

```
Future<DetectionClasses> inference(CameraImage cameraImage) async {
  ReceivePort responsePort = ReceivePort();
  final isolateData = IsolateData(
    cameraImage: cameraImage,
    interpreterAddress: classifier.interpreter.address,
    responsePort: responsePort.sendPort,
  );

  isolateUtils.sendPort.send(isolateData);
  var result = await responsePort.first;

  return result;
}
```

ونستدعيها بطريقة ProcessCameraImage :

```
Future<void> processCameraImage(CameraImage cameraImage) async {
  setState(() {
    isWorking = true;
  });

  final result = await inference(cameraImage);

  if (detected != result) {
    setState(() {
      detected = result;
    });
  }

  setState(() {
    lastShot = DateTime.now();
    isWorking = false;
  });
}
```

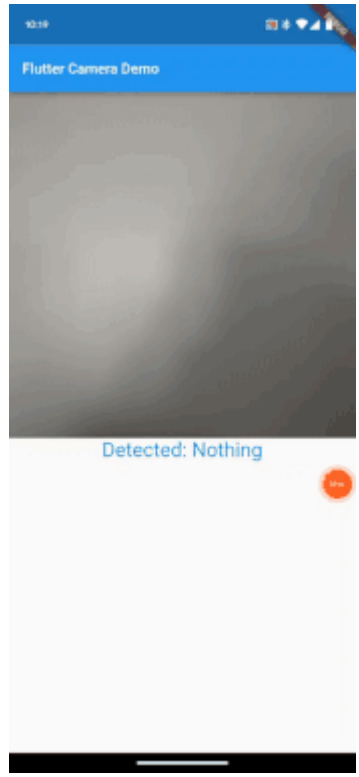
الكود الكامل لـ ScannerScreenState_ هو التالي:

```
class ScannerScreen extends StatefulWidget {
  @override
  _ScannerScreenState createState() => _ScannerScreenState();
}

class _ScannerScreenState extends State<ScannerScreen> {
  late CameraController cameraController;
  late Interpreter interpreter;
  final classifier = Classifier();
  final isolateUtils = IsolateUtils();

  bool initialized = false;
  bool isWorking = false;
  DetectionClasses detected = DetectionClasses.nothing;

  @override
  void initState() {
    super.initState();
    initialize();
  }

  Future<void> initialize() async {
    // Load main isolate Interpreter
    await classifier.loadModel();

    final cameras = await availableCameras();
    // Create a CameraController object
    cameraController = CameraController(
      cameras[0], // Choose the first camera in the list
      ResolutionPreset.medium, // Choose a resolution preset
    );

    // Start Inference isolate
    await isolateUtils.start();

    // Initialize the CameraController and start the camera preview
    await cameraController.initialize();
    // Listen for image frames
```

```
  await cameraController.startImageStream((image) {
    // Make predictions only if not busy
    if (!isWorking) {
      processCameraImage(image);
    }
  });

  setState(() {
    initialized = true;
  });
}

Future<void> processCameraImage(CameraImage cameraImage) async {
  setState(() {
    isWorking = true;
  });

  final result = await inference(cameraImage);

  if (detected != result) {
    detected = result;
  }

  setState(() {
    isWorking = false;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Flutter Camera Demo'),
    ),
    body: initialized
        ? Column(
            children: [
              SizedBox(
                height: MediaQuery.of(context).size.width,
                width: MediaQuery.of(context).size.width,
                child: CameraPreview(cameraController),
              ),
              Text(
                "Detected: ${detected.label}",
                style: const TextStyle(
                  fontSize: 28,
                  color: Colors.blue,
                ),
              ),
            ],
          )
        : const Center(child: CircularProgressIndicator()),
  );
}

Future<DetectionClasses> inference(CameraImage cameraImage) async {
  ReceivePort responsePort = ReceivePort();
  final isolateData = IsolateData(
    cameraImage: cameraImage,
    interpreterAddress: classifier.interpreter.address,
    responsePort: responsePort.sendPort,
  );
```

```
    isolateUtils.sendPort.send(isolateData);
    var result = await responsePort.first;

    return result;
  }

  @override
  void dispose() {
    cameraController.dispose();
    isolateUtils.dispose();
    super.dispose();
  }
}
```

من خلال القيام بهذه الخطوات يجب أن تحصل على النتيجة التالية:



الاستنتاج

في هذه البرنامج التعليمي، لقد نجحتفي دمج نموذج تصنيف الصور المخصص الخاص بكفي تطبيق Flutter. من خلال الجمع بين قوة التعلم الآلي وإطار عمل Flutter سهل الاستخدام، قمت بإنشاء تطبيق للكشف عن إيماءات اليدفي الوقت الفعلي يتسم بالدقة والاستجابة وسهولة الاستخدام.

خلال هذه السلسلة، قمت بتغطية مجموعة من المواضيع، بدءًا من أساسيات بناء نموذج CNN مخصص وحتى التقنيات الأكثر تقدمًا لنقل التعلم وتحسين النموذج. لقد استكشفنا أيضًا تحديات دمج نماذج التعلم الآليفي تطبيقات الهاتف المحمول وقدمنا حلولاً عملية للتغلب عليها.

باتباع هذه السلسلة، لديك الآن المهارات والمعرفة اللازمة لإنشاء نماذج تصنيف الصور المخصصة الخاصة بك، وتحسينها من أجل الأداء، ونشرهافي تطبيق جوال باستخدام Flutter و TensorFlow Lite. أتمنى أن تجد هذه السلسلة مفيدة وغنية بالمعلومات.

للتحقق من الكود الخاص بهذا المشروعفي مستودع GitHub الخاص بي.

المصدر:

https://medium.com/geekculture/bring-your-image-classification-model-to-life-with-flutter-5148efbf3647

14) انشاء تطبيق للكشف عن النص باستخدام فلاتر وتنسرفلو لايت Mobile App Build a Text Classification using Flutter with TensorFlow Lite

إذا كنت ترغبفي وجود طريقة سهلة وفعالة ومرنة لدمج نماذج TensorFlow المدربة مع تطبيقات Flutter الخاصة بك، فيسعدني أن أعلن عن إصدار مكون إضافي جديد tflite_flutter.

النقاط الرئيسية لtflite_flutter:

- فهو يوفر واجهة برمجة تطبيقات Dart مشابهة لواجهات برمجة التطبيقات TFLite Java وSwift، وبالتالي لا يوجد تنازل عن المرونة المتوفرة على تلك الأنظمة الأساسية.
- يرتبط مباشرة بواجهة برمجة تطبيقات TensorFlow Lite C باستخدام dart:ffi، مما يجعله أكثر كفاءة من أساليب تكامل النظام الأساسي.
- لا حاجة لكتابة أي كود خاص بالمنصة.
- يقدم دعم التسريع باستخدام NNAPI ومندوبي GPU على Android ومندوب Metal على iOS.

في هذا البرنامج التعليمي، سأرشدك خلال عملية إنشاء تطبيق Flutter لتصنيف النصوص Text Classification باستخدام tflite_flutter. لنبدأ بإنشاء مشروع Flutter جديد text_classification_app.

الإعداد الأولي

- مستخدمي لينكس وماك

انسخ ملف install.shفي المجلد الجذر لتطبيقك، وقم بتنفيذ الأمر sh install.shفي المجلد الجذر، text_classification_app/في حالتنا.

- مستخدمي ويندوز

انسخ ملف install.batفي المجلد الجذر لتطبيقك، وقم بتنفيذ الأمر، install.batفي المجلد الجذر، text_classification_app/في حالتنا.

سيؤدي هذا تلقائيًا إلى تنزيل أحدث الثنائيات من إصدار assets ووضعهافي المجلدات المناسبة لك.

الحصول على البرنامج المساعد

في pubspec.yaml، قم بتضمين tflite_flutter: ^<latest_version> (التفاصيل هنا).

تحميل النموذج

لاستخدام أي نموذج مدرب على TensorFlow على الهاتف المحمول، نحتاج إلى الحصول عليه بتنسيق tflite. لمزيد من المعلومات حول كيفية تحويل نموذج TensorFlow المدرب إلى تنسيق tflite.، راجع هذا الدليل الرسمي.

سنستخدم نموذج تصنيف النص المُدرب مسبقًا pre-trained Text Classification Model والمتوفر على موقع TensorFlow. إضغط هنا للتحميل.

يتنبأ هذا النموذج المُدرب مسبقًا بما إذا كان اتجاه الفقرة إيجابيًا positive أم سلبيًا negative. تم تدريبه على مجموعة بيانات مراجعة الأفلام الكبيرة الإصدار 1.0 (Large Movie Review Dataset v1.0) من Mass et al، والتي تتكون من مراجعات أفلام IMDB المصنفة على أنها إيجابية أو سلبية. العثور على مزيد من المعلومات هنا.

ضع text_classification.tflite وtext_classification_vocab.txtفي الدليل text_classification_app/assets/.

ضع /assetsفي pubspec.yaml .

```
assets:
- assets/
```

الآن، نحن جاهزون للبدء بالبرمجة.

برمجة المصنف

المعالجة المسبقة

كما هو مذكورفي صفحة نموذج text_classification،

فيما يلي خطوات تصنيف فقرة بالنموذج:

1. قم بترميز الفقرة وتحويلها إلى قائمة معرفات الكلمات باستخدام مفردات محددة مسبقًا.
1. قم بتغذية القائمة إلى نموذج TensorFlow Lite.
2. احصل على احتمال أن تكون الفقرة موجبة أو سالبة من مخرجات النموذج.

سنكتب أولاً طريقة لترميز السلسلة الأولية باستخدام text_classification_vocab.txt كمفردات vocabulary.

قم بإنشاء ملف classifier.dart جديد ضمن المجلد /lib.

لنكتب أولًا التعليمات البرمجية لتحميل text_classification_vocab.txt إلى القاموس.

```
import 'package:flutter/services.dart';

class Classifier {
  final _vocabFile = 'text_classification_vocab.txt';

  Map<String, int> _dict;

  Classifier() {
    _loadDictionary();
  }

  void _loadDictionary() async {
    final vocab = await rootBundle.loadString('assets/$_vocabFile');
    var dict = <String, int>{};
    final vocabList = vocab.split('\n');
    for (var i = 0; i < vocabList.length; i++) {
      var entry = vocabList[i].trim().split(' ');
      dict[entry[0]] = int.parse(entry[1]);
    }
    _dict = dict;
    print('Dictionary loaded successfully');
  }

}
```

الآن، سوف نكتب دالة لترميز tokenize السلسلة الأولية raw string.

```
import 'package:flutter/services.dart';

class Classifier {
  final  vocabFile = 'text classification vocab.txt';

  // Maximum length of sentence
  final int _sentenceLen = 256;

  final String start = '<START>';
  final String pad = '<PAD>';
  final String unk = '<UNKNOWN>';

  Map<String, int> _dict;

  List<List<double>> tokenizeInputText(String text) {

    // Whitespace tokenization
    final toks = text.split(' ');

    // Create a list of length==_sentenceLen filled with the value <pad>
    var vec = List<double>.filled( sentenceLen,  dict[pad].toDouble());

    var index = 0;
    if (_dict.containsKey(start)) {
      vec[index++] = _dict[start].toDouble();
    }

    // For each word in sentence find corresponding index in dict
    for (var tok in toks) {
      if (index > _sentenceLen) {
        break;
      }
      vec[index++] = _dict.containsKey(tok)
          ? _dict[tok].toDouble()
          : _dict[unk].toDouble();
```

```
    }

    // returning List<List<double>> as our interpreter input tensor expects
the shape, [1,256]
    return [vec];
  }
}
```

الاستدلال باستخدام tflite_flutter

هذا هو القسم الرئيسي لهذا البرنامج التعليمي، حيث سنناقش هنا استخدام المكون الإضافي tflite_flutter.

يشير مصطلح الاستدلال inference إلى عملية تنفيذ نموذج TensorFlow Lite على الجهاز من أجل عمل تنبؤات بناءً على بيانات الإدخال. لإجراء الاستدلال باستخدام نموذج TensorFlow Lite، يجب عليك تشغيله من خلال مفسر interpreter. يتعلم أكثر.

إنشاء المفسر، تحميل النموذج

يوفر tflite_flutter طريقة لإنشاء المفسر مباشرة من assets.

```
static Future<Interpreter> fromAsset(String assetName,
{InterpreterOptions options})
```

نظرًا لأن نموذجنا موجودفي الدليل /assets، فسوف نستخدم الطريقة المذكورة أعلاه لإنشاء المفسر. للحصول على معلومات حول InterpreterOptions، راجع هذا.

```
import 'package:flutter/services.dart';

// Import tflite_flutter
import 'package:tflite_flutter/tflite_flutter.dart';

class Classifier {
  // name of the model file
  final _modelFile = 'text_classification.tflite';

  // TensorFlow Lite Interpreter object
  Interpreter _interpreter;

  Classifier() {
    // Load model when the classifier is initialized.
    _loadModel();
  }

  void _loadModel() async {

    // Creating the interpreter using Interpreter.fromAsset
    _interpreter = await Interpreter.fromAsset(_modelFile);
    print('Interpreter loaded successfully');
  }

}
```

إذا كنت لا تريد وضع النموذج الخاص بكفي /assets الدليل، فإن tflite_flutter يوفر منشئي المصنع لإنشاء مفسر أيضًا، راجع الملف هذا.

تنفيذ الاستدلال

سنستخدم هذه الطريقة للاستدلال،

```
void run(Object input, Object output);
```

لاحظ أن هذه الطريقة هي نفس الطريقة التي توفرها Java API.

يجب أن يكون Object input وObject output عبارة عن قوائم متعددة الأبعاد لها نفس شكل موتر الإدخال وموتر الإخراج.

لعرض أشكال وأحجام موترات الإدخال وموترات الإخراج التي يمكنك القيام بها،

```
_interpreter.allocateTensors();// Print list of input tensors
print(_interpreter.getInputTensors());// Print list of output tensors
print(_interpreter.getOutputTensors());
```

في حالة نموذج text_classification الخاص بنا،

```
InputTensorList:
[Tensor{_tensor: Pointer<TfLiteTensor>: address=0xbffcf280, name:
embedding_input, type: TfLiteType.float32, shape: [1, 256], data:
1024]OutputTensorList:
[Tensor{_tensor: Pointer<TfLiteTensor>: address=0xbffcf140, name:
dense_1/Softmax, type: TfLiteType.float32, shape: [1, 2], data:  8]
```

الآن، لنكتب طريقة التصنيف التي تُرجع 1 للإيجاب، و0 للسالب.

```
  int classify(String rawText) {

    // tokenizeInputText returns List<List<double>>
    // of shape [1, 256].
    List<List<double>> input = tokenizeInputText(rawText);

    // output of shape [1,2].
    var output = List<double>(2).reshape([1, 2]);

    // The run method will run inference and
    // store the resulting values in output.
    _interpreter.run(input, output);

    var result = 0;
    // If value of first element in output is greater than second,
    // then sentece is negative

    if ((output[0][0] as double) > (output[0][1] as double)) {
      result = 0;
    } else {
      result = 1;
    }
    return result;
  }
```

هناك بعض الامتدادات المفيدة المحددة ضمن extension ListShape on List في tflite_flutter،

```
// reshapes a given list to shape, provided total number of elements //
remain equal
// Usage: List(400).reshape([2,10,20])
// returns List<dynamic>
```

```
List reshape(List<int> shape)// returns shape of a list
List<int> get shape// return total elements in a list of any shape
int get computeNumElements
```

يجب أن يبدو classifier.dart النهائي هكذا،

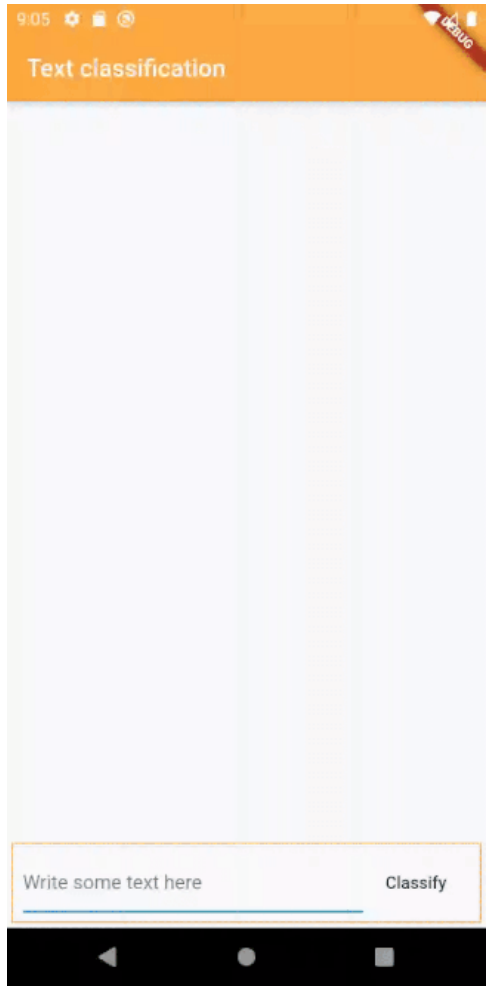
```
import 'package:flutter/services.dart';

// Import tflite_flutter
import 'package:tflite_flutter/tflite_flutter.dart';

class Classifier {
  // name of the model file
  final _modelFile = 'text_classification.tflite';
  final _vocabFile = 'text_classification_vocab.txt';

  // Maximum length of sentence
  final int _sentenceLen = 256;

  final String start = '<START>';
  final String pad = '<PAD>';
  final String unk = '<UNKNOWN>';

  Map<String, int> _dict;

  // TensorFlow Lite Interpreter object
  Interpreter _interpreter;

  Classifier() {
    // Load model when the classifier is initialized.
     loadModel();
    _loadDictionary();
  }

  void _loadModel() async {
    // Creating the interpreter using Interpreter.fromAsset
     interpreter = await Interpreter.fromAsset( modelFile);
    print('Interpreter loaded successfully');
  }

  void _loadDictionary() async {
    final vocab = await rootBundle.loadString('assets/$_vocabFile');
    var dict = <String, int>{};
    final vocabList = vocab.split('\n');
    for (var i = 0; i < vocabList.length; i++) {
      var entry = vocabList[i].trim().split(' ');
      dict[entry[0]] = int.parse(entry[1]);
    }
    _dict = dict;
    print('Dictionary loaded successfully');
  }

  int classify(String rawText) {
    // tokenizeInputText returns List<List<double>>
    // of shape [1, 256].
    List<List<double>> input = tokenizeInputText(rawText);

    // output of shape [1,2].
    var output = List<double>(2).reshape([1, 2]);

    // The run method will run inference and
    // store the resulting values in output.
```

```
    _interpreter.run(input, output);

    var result = 0;
    // If value of first element in output is greater than second,
    // then sentece is negative

    if ((output[0][0] as double) > (output[0][1] as double)) {
      result = 0;
    } else {
      result = 1;
    }
    return result;
  }

  List<List<double>> tokenizeInputText(String text) {
    // Whitespace tokenization
    final toks = text.split(' ');

    // Create a list of length==_sentenceLen filled with the value <pad>
    var vec = List<double>.filled(_sentenceLen, _dict[pad].toDouble());

    var index = 0;
    if (_dict.containsKey(start)) {
      vec[index++] = _dict[start].toDouble();
    }

    // For each word in sentence find corresponding index in dict
    for (var tok in toks) {
      if (index > _sentenceLen) {
        break;
      }
      vec[index++] = _dict.containsKey(tok)
          ? _dict[tok].toDouble()
          : _dict[unk].toDouble();
    }

    // returning List<List<double>> as our interpreter input tensor expects
the shape, [1,256]
    return [vec];
  }
}
```

الآن، الأمر متروك لك لبرمجة واجهة المستخدم المطلوبة لهذا، وسيكون استخدام المصنف بسيطًا،

```
// Create Classifier object
Classifer _classifier = Classifier();// call classify method with sentence
as parameter
 classifier.classify("I liked the movie");
// returns 1 (POSITIVE)_classifier.classify("I didn't liked the movie");
// returns 0 (NEGATIVE)
```

تحقق من التطبيق الكامل لمثال تصنيف النص مع واجهة المستخدم.



تفضل بزيارة المستودع am15h/tflite_flutter_plugin على Github لمعرفة المزيد حول المكون الإضافي tflite_flutter.

المصدر:

https://medium.com/@am15hg/text-classification-using-tensorflow-lite-plugin-for-flutter-3b92f6655982

AI for Mobile

Mobile Apps Created and Programmed using Flutter and TensorFlow Lite

By: Dr. Alaa Taima