

The Power of Python

Enhancing Medical Imaging with AI

قوة البايثون في تحسين
التصوير الطبي بالذكاء الاصطناعي

(الجزء الثاني)

أعداد وتأليف

فلاح كاطع صالح الخفاجي

مطور تطبيقات الذكاء الاصطناعي

2024



ISBN: 9798866146093

الإهداء

إلى والدي ووالدتي الأجلاء رحمة الله عليهما
إلى زوجتي رفيقة الكفاح التي لم تبخل بوقت أو جهد لمساعدتي
إلى أولادي الذين كانوا معي في مسيري العلمية والذين تحملوا مني كل ذلك الانشغال
إلى جموع الأهل والأصدقاء أهدي إليكم هذا الكتاب والعمل المتواضع
شكرا لهم جميعا مع خالص تحياتي لهم

حكمة أحببتها للعالم إسحاق نيوتن

"

إنني جاهل، لا أعرف إلا حقيقة واحدة ، وهي أنني لا أعرف شيئاً

"

الرجاء أرسل رسالة بريد الكتروني أو أترك تعليق في مدونتي بمجرد قراءة هذا الكتاب واستخدامه، واستخدم رأيك غير المتحيز لاتخاذ رأيك في الكتاب واترك اي ملاحظة تراها مناسبة ومفيدة للجميع لغرض تطوير الكتاب نحو الاحسن وإذا واجهت أي خطأ مطبعي او خطأ في الاكواد البرمجية الملحة بهذا الكتاب يرجى التواصل معي على البريد الالكتروني ومدونتي أدناه.

شكراً لك في اختيارك لهذا الكتاب المتواضع والبسيط لغرض تطوير مهاراتك العلمية والعملية

Email: falahgs07@gmail.com

My Blog: <https://iraqprogrammer.wordpress.com>

أو رمز الاستجابة السريع



القراء الأعزاء،

يسعدني أن أقدم الإصدار الثاني من " قوة البائيون في تحسين التصوير الطبي بالذكاء الاصطناعي "

في المجال الطبي: الكشف والتصنيف والتجزئة . تعتمد هذه الطبعة على الأسس الموضوعة في الكتاب الأول، وتغوص بشكل أعمق في التقنيات المتقدمة وتوسيع آفاقنا بمجموعات بيانات جديدة لمجموعة واسعة من الأمراض والسرطانات.

ستجد في هذا الإصدار تغطية شاملة لأحدث المنهجيات والابتكارات في تحليل الصور الطبية باستخدام لغة بائيون. لقد قمنا بدمج التقنيات المتطوره والأمثلة العملية لتعزيز فهمك وتطبيقك لهذه الأدوات القوية في السيناريوهات الطبية في العالم الحقيقي.

أمل أن يكون هذا الكتاب بمثابة مورد قيم في رحلتك للاستفادة من قدرات بائيون لتحقيق تقدم رائد في البحث والممارسات الطبية.

حياتي الحارة،

المؤلف

فلاح كاطع صالح الخفاجي

2024

xii	المقدمة.
xiii	لمن هذا الكتاب؟.
xiv	متطلبات تنفيذ المشاريع والتطبيقات.
xv	مجتمع كاكل وخدمة السحابي كولاب كوكل .
xix	التطبيقات والكودات البرمجية للكتاب.
xx	مجموعة البيانات الطبية.
1	الفصل الاول : مكتبة خدمة الويب وادوات التعليقات التوضيحية.
2	مكتبة Gradio لتطبيقات الويب.
5	ادوات الملصقات والتعليقات التوضيحية.
11	الفصل الثاني : نموذج تجزئة سرطان الثدي باستخدام نموذج UNet.
12	المقدمة.
12	بيانات سرطان الثدي.
12	الكود البرمجي .
19	استدعاء نموذج UNet.
23	كود التدريب النهائي.
24	الملخص .
25	الفصل الثالث : تصنیف الصور ثلاثية الأبعاد من صور الأشعة المقطعيه.
26	المقدمة.
26	تهيئة المكتبات.
26	تهيئة البيانات.
27	معالجة البيانات.
29	بناء مجموعات بيانات التدريب والتحقق من الصحة.
29	زيادة البيانات.
30	تصوير الأشعة المقطعيه المعززة.
32	تحديد شبكة عصبية تلفيفية ثلاثية الأبعاد.
32	تعريف النموذج.
33	تدريب النموذج.
34	معاينة أداء النموذج.
34	الكود البرمجي النهائي.
35	الملخص.
36	الفصل الرابع : Eyes Ocular Disease Classification ResNet-18
37	المقدمة.
38	البيانات.
39	تهيئة النموذج.

- 44 اختبار النموذج النهائي على بيانات الاختبار.
 46 تطبيق ويب.
 47 الكود النهائي.
 48 الملخص.
 49 **الفصل الخامس**: الكشف عن المرض الالتهاب الرئوي.
 50 المقدمة.
 50 التعلم العميق والكشف عن المرض الالتهاب الرئوي.
 51 بيانات التدريب.
 54 نموذج الشبكة العصبية ResNet18
 57 نتائج التدريب.
 59 استدلال النموذج على بيانات الاختبار.
 60 عمل تطبيق ويب.
 62 الكود البرمجي النهائي.
 62 الملخص.
الفصل السادس
 63 Using MobileNetV2 Eyes Ocular Disease Classification:
 64 المقدمة.
 64 ما هو نموذج ?MobileNetV2.
 64 تدريب النموذج.
 68 استدعاء نموذج MobileNetV2.
 73 استدلال النموذج على بيانات الاختبار.
 76 تطبيق الويب.
 77 الكود البرمجي النهائي.
 78 الملخص.
الفصل السابع : Skin Cancer Malignant vs. Benign
 79 المقدمة.
 80 نقل التعلم ونموذج MobilenetV2
 81 تدريب النموذج.
 89 الاستدلال النهائي للنموذج.
 90 خدمة تطبيق الويب.
 92 الكود النهائي.
 92 الملخص.
الفصل الثامن : تصنيف 8 فئات لسرطان القولون والمستقيم.
 93 المقدمة.
 94 ما هي مجموعة البيانات؟
 96 نموذج الشبكة العصبية MobileNetV2.
 103 استدلال النموذج على صور الاختبار.
 105 حلول أخرى في حالة فشل التدريب.

- 106 الكود النهائي.
106 الملخص.
107 **الفصل التاسع**: سرطان الرئة باستخدام نموذج ResNet-50.
108 المقدمة.
108 نموذج ResNet-50 والكشف عن سرطان الرئة.
109 تطبيق النموذج على بيانات سرطان الرئة.
110 تدريب النموذج.
119 الاستدلال النهائي للنموذج على بيانات الاختبار.
121 خدمة تطبيق الويب.
123 الكود البرمجي النهائي.
123 الملخص.
124 **الفصل العاشر**: مرض سرطان الثدي تصنيف سرطان الأقنية الغازية.
125 المقدمة.
125 ما هو نقل التعلم؟.
125 تطبيق نموذج ResNet50 في تصنيف سرطان الثدي.
126 البدء في تدريب النموذج.
139 تطبيق ويب باستخدام مكتبة Gradio
142 الكود البرمجي النهائي.
143 الملخص.
144 **الفصل الحادي عشر**: الكشف عن سرطان العظام بتقنية Yolov8.
145 المقدمة.
145 ما هو نموذج Yolov8؟.
147 تطبيق على كشف سرطان العظام.
151 الكود البرمجي للتدريب والتحقق.
153 الاستدلال على صور الاختبار.
155 تطبيق ويب باستخدام مكتبة Gradio
157 التطبيقات والأفكار المستقبلية.
158 الكود النهائي.
159 الملخص.
160 **الفصل الثاني عشر**: تقنية تجزئة مرض سرطان الدماغ باستخدام نموذج Yolov8-Seg.
161 المقدمة.
163 ما هو نموذج التجزئة Yolov8-Seg.؟.
164 تطبيق على كشف سرطان الدماغ .
167 نموذج التدريب.
169 الاستدلال على صور الاختبار.
173 تطبيق ويب باستخدام مكتبة Gradio
174 الكود البرمجي النهائي.

174	التطبيقات والأفكار المستقبلية.
177	الملخص.
179	الفصل الثالث عشر: نموذج Yolov8 لكشف مرض سرطان الجلد
180	المقدمة.
180	ما هو نموذج YOLOv8x؟.
181	تطبيق على تصنیف مرض سرطان الجلد.
185	نموذج التدريب.
188	الاستدلال على صور الاختبار.
189	تطبيق ويب باستخدام مكتبة Gradio.
191	التطبيقات والأفكار المستقبلية.
192	الكود النهائي.
193	الملخص.
194	الفصل الرابع عشر: تصنیف مرض هشاشة مفاصل الرکبة ودرجة الخطورة
195	المقدمة.
196	نقل التعلم ونموذج ResNet50
197	تدريب النموذج.
212	خدمة تطبيق الويب.
213	الكود البرمجي النهائي.
214	الملخص.
215	الخاتمة.
217	السيرة الذاتية للمؤلف.
220	المصادر.

برزت Python، وهي لغة برمجة قوية ومتعددة الاستخدامات، كأداة مهيمنة في مختلف المجالات، بما في ذلك مجال التصوير الطبي، حيث عززت بشكل كبير قدرات الذكاء الاصطناعي (AI). بفضل بساطتها وقابليتها للقراءة ودعمها الشامل للمكتبة، أصبحت لغة Python هي اللغة المفضلة لتطوير خوارزميات وتطبيقات الذكاء الاصطناعي. يوفر النظام البيئي الشامل للمكتبات في Python، مثل TensorFlow وKeras وPyTorch، أطر عمل قوية لتنفيذ نماذج التعلم العميق في التصوير الطبي. تتيح هذه المكتبات للباحثين والمطوريين الاستفادة من النماذج المدربة مسبقاً، وإجراء تصنيف الصور، والتجزئة، ومهام الكشف، وتحسين دقة التشخيص. أحدث الجمع بين Python وAI ثورة في التصوير الطبي من خلال تمكين تقنيات تحليل الصور المتقدمة. يمكن لخوارزميات الذكاء الاصطناعي الآن استخراج معلومات مفيدة من الصور الطبية، والمساعدة في الكشف المبكر عن الأمراض، وتجزئة الورم بدقة، وتحديد التشوّهات. عززت قدرة Python على التعامل مع هياكل البيانات المعقدة ودعمها للحوسبة العلمية دقة وكفاءة أنظمة التصوير الطبي التي تعمل بالذكاء الاصطناعي. علاوة على ذلك، سهلت سهولة تكامل Python مع التقنيات والأطر الأخرى نشر حلول الذكاء الاصطناعي في إعدادات الرعاية الصحية. يسمح بالتكامل السلس مع الأنظمة الحالية، مما يجعل من الممكن تطوير واجهات سهلة الاستخدام، والتواصل مع السجلات الصحية الإلكترونية، وتمكين التحليل في الوقت الفعلي ودعم القرار.

في الختام، تكمن قوّة Python في قدرتها على الدمج مع تقنيات الذكاء الاصطناعي، وتوفير التصوير الطبي بقدرات متقدمة. من خلال الاستفادة من بساطة Python والمكتبات الشاملة وقدرات التكامل، شهد المجال الطبي تقدماً ملحوظاً في تحسين التشخيص وتعزيز تخطيط العلاج وتحسين نتائج المرضى في نهاية المطاف.

العنوان: "قوة لغة بايثون: تحسين التصوير الطبي باستخدام الذكاء الاصطناعي، الجزء الثاني - توسيع نماذج التعلم العميق"

تقدم هذه النسخة المحدثة من الكتاب استكشافاً موسعاً لدمج برمجة Python والذكاء الاصطناعي في مجال التصوير الطبي. إنه مصمم خصيصاً لجمهور متعدد، ويوفر رؤى لا تقدر بثمن للمحترفين والطلاب على حد سواء. سواء كنت أخصائني أشعة متعرساً، أو مهندساً متخصصاً، أو باحثاً فضولياً، أو طالباً متحمساً، فإن هذا الكتاب يزودك بالأدوات اللازمة للتعقب في تعقيدات خوارزميات الذكاء الاصطناعي المطبقة على تحليل الصور الطبية.

تضمن الإضافات الجديدة إلى هذه الطبعة مجموعة غنية من نماذج التعلم العميق، مما يعزز فهم القارئ والتنفيذ العملي لتقنيات الذكاء الاصطناعي المتقدمة. من خلال برمجة بايثون، يمكنك القراء خبرة عملية في تطوير واختبار وتحسين الخوارزميات الخاصة بهم. يعد هذا المورد الشامل بمثابة رفيق حيوي لأي شخص متحمس للاستفادة من قوة Python والذكاء الاصطناعي لإحداث ثورة في التصوير الطبي.

أصبحت Python أداة قوية في مجال الطب، لا سيما في تحليل ومعالجة الصور الطبية. يتطلب تنفيذ المشاريع المتعلقة بتحليل الصور الطبية متطلبات وتطبيقات معينة. يتضمن ذلك تثبيت وحدات ومكتبات Python ذات الصلة، بالإضافة إلى استخدام نماذج الشبكة العصبية التلaffيفية (CNN) ومكتبات الذكاء الاصطناعي.

- سنحتاج إلى تثبيت (Python 3.5+) عاملًا على نظامك المحلي. او استخدام الخدمة السحابية كولاب لشركة كوكل. وهي أفضل طريقة لتجنب مشاكل انظمة التشغيل والاعدادات.
- سنحتاج غالباً إلى (Tensor Flow 1.x) أو x.2 طوال الوقت. وطبعا هي موجودة ضمن اعدادات كولاب كوكل السحابي.
- حساب على الخادم السحابي كولاب كوكل. Google Colab Cloud
- حساب لمجتمع كاكل (Kaggle) (لأستيراد بعض الداتاسيس لغرض التدريب.
- حاليا فقط لاستيراد البيانات الضخمة والمهمة في مشاريعنا.

جميع أدوات البرامج التي ستحتاجها في هذا الكتاب متاحة مجاناً. وتم شرحها في الكتاب مع روابطها طبعاً ننصح بكتابة الكود بنفسك أو قم بالوصول إلى مصدر الأكواد البرمجية عبر مستودع (GitHub). (الرابط متاح في نهاية كل فصل او في قسم المصادر). وايضاً تم رفعها على كوكل درايف (Google drive). طبعاً سوف يساعدك القيام بذلك على تجنب أي أخطاء محتملة ذات صلة بالمشروع والتطبيق. كل هذه الأدوات سوف نشرحها في هذا الكتاب المتواضع والبسيط بعيد عن التعقيد والسرد الأكاديمي لجعل هذا الكتاب متداول لجميع الفئات والاختصاصات عملياً وواقعاً.

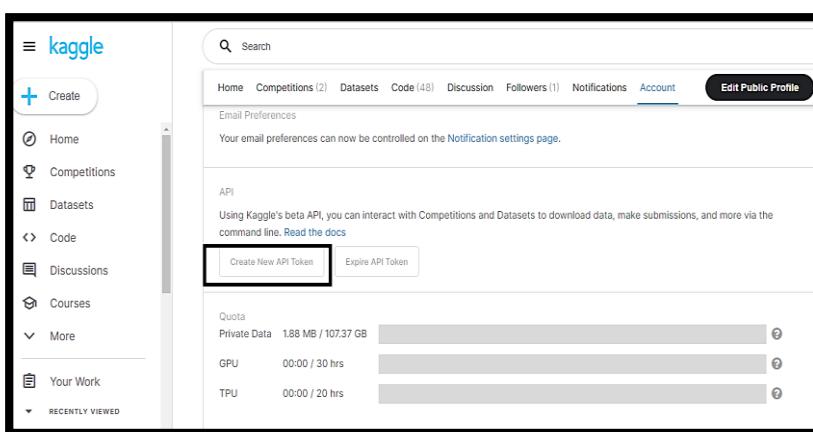
وأيضاً سوف يحتاج المطوروون والباحثون إلى تثبيت وحدات ومكتبات خاصة بتحليل الصور الطبية في Python. توفر هذه الوحدات، مثل NumPy و SciPy و OpenCV، وظائف لمعالجة الصور ومعالجتها وتحليلها. بالإضافة إلى ذلك، تتيح المكتبات المتخصصة مثل Simplest PyDICOM للمستخدمين العمل مع بيانات التصوير الطبي بتنسيقات مختلفة، مثل DICOM.

تلعب نماذج CNN دوراً مهماً في تحليل الصور الطبية. يمكن تدريب نماذج التعلم العميق هذه على اكتشاف وتصنيف التشوّهات المختلفة، مثل الأورام أو الآفات، في الصور الطبية. توفر Python العديد من الأطر، بما في ذلك (Libs. Pytorch, TensorFlow, Keras)، التي تبسيط تنفيذ وتدريب نماذج CNN لتحليل الصور الطبية. علاوة على ذلك، تقدم مكتبات الذكاء الاصطناعي مثل scikit-learn و PyTorch Learn مجموعة واسعة من الأدوات والخوارزميات لمعالجة المسابقة للبيانات واستخراج الميزات وتقييم النماذج في سياق تحليل الصور الطبية. لا يفيد تطبيق Python في صور الطب الباحثين والمبرمجين المحترفين فحسب، بل يفيد أيضاً ممارسي الرعاية الصحية. من خلال الاستفادة من قدرات Python، يمكن للمطوروين إنشاء تطبيقات تساعد في التشخيص والتبيؤ والتخطيط العلاجي لمختلف الحالات الطبية. يمكن لهذه التطبيقات تحليل الصور الطبية واستخراج الميزات ذات الصلة وتقديم رؤى وتوصيات لمتخصصي الرعاية الصحية.

مجتمع كاكل وخدمة السحابي كولاب كوكل . Google Colab Cloud and Kaggle community

• ماهو مجتمع كاكل ...؟

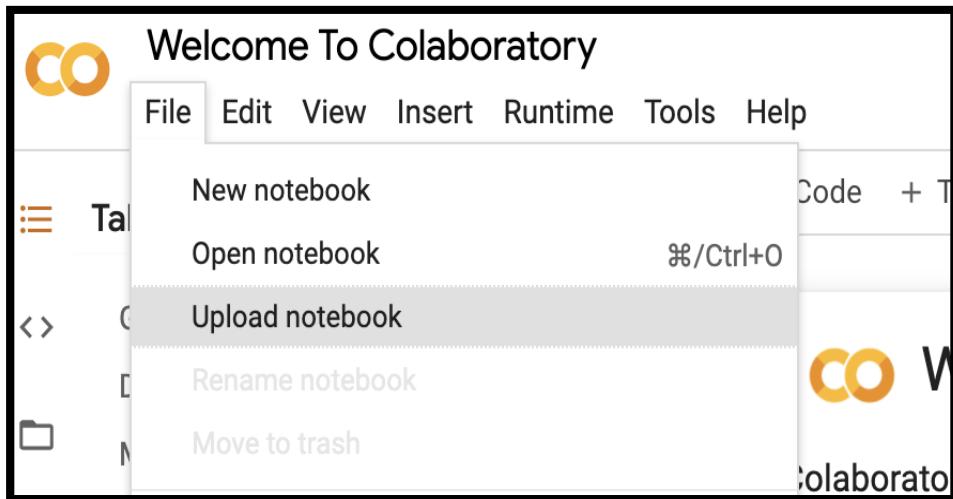
هي إحدى الشركات التابعة لشركة جوجل، عبارة عن مجتمع عبر الإنترن特 لعلماء البيانات ومهندسي التعلم الآلي. يسمح كاكل للمستخدمين بالعثور على مجموعات البيانات ونشرها، واستكشاف نماذج الذكاء الاصطناعي وإنشائها في بيئة علم البيانات المستندة إلى الويب، والعمل مع علماء البيانات الآخرين ومهندسي التعلم الآلي، والدخول في مسابقات لحل تحديات علوم البيانات. كانت بداية كاكل في عام 2010 من خلال تقديم مسابقات التعلم الآلي وعلوم البيانات، وتقدم الآن أيضاً منصة بيانات عامة ومنصة عمل قائمة على السحابة لعلوم البيانات وتعليم الذكاء الاصطناعي. رابط الموقع من هنا <https://www.kaggle.com> علينا عمل حساب الكتروني ... وحصلنا على مفتاح التخويل Kaggle json API لغرض الاستفادة منه في مشاريعنا القادمة في الفصول القادمة .



صورة رقم (1)

• ماهو الخادم السحابي كوكل كولاب؟ ... Google Colab Cloud [الرابط من هنا](https://colab.research.google.com)

هو مشروع بحثي من Google مصمم للمساعدة في نشر تدريب التعلم الآلي ونتائج البحث. إنها بيئة تطوير لكتابة كود برمجي بلغة الباليتون أو مايسمي دفتر ملاحظات Jupyter يمكن استخدامها بدون أي إعدادات ، وتعمل بالكامل في السحابة. يتم تخزين دفاتر الملاحظات التعاونية في Google Drive ويمكن مشاركتها تماماً مثلما تستخدم مستندات Google أو جداول البيانات Colaboratory مجاني للاستخدام.

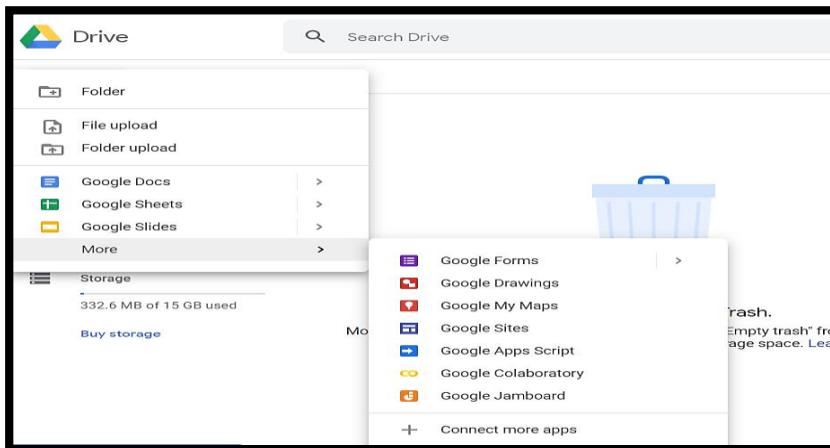


صورة رقم (2)

باستخدام Colaboratory يمكنك بسهولة استخدام Keras و TensorFlow و PyTorch وأطر أخرى لتطوير تطبيقات التعلم العميق. بالمقارنة مع الخدمات السحابية الأخرى ، فإن الميزة الأكثر أهمية هي يوفر Colab GPU وهو مجاني تماماً.

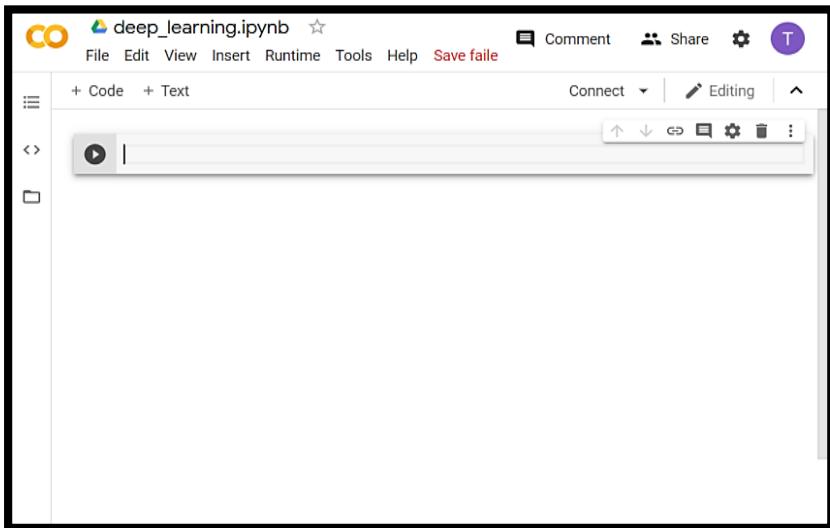
- كيفية استخدام Google Colab ؟

- قم بإنشاء دفتر ملاحظات colab على Google Cloud Disk
 - a- قم أولاً بربط colab بـ Google Cloud Disk
 - انقر فوق جديد > توصيل المزيد من التطبيقات ، وال Thur على Google Colaboratory.
 - b- قم بإنشاء دفتر ملاحظات جديد من colab
- انقر فوق جديد > المزيد Google Colaboratory ->



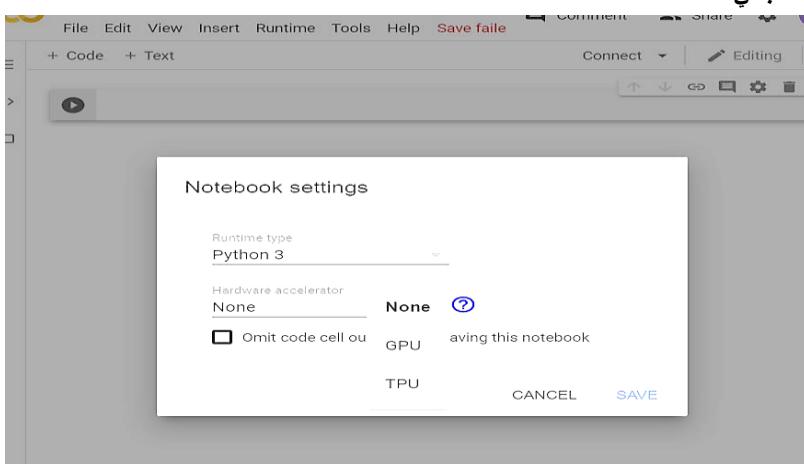
صورة رقم (3)

يظهر بعد الإنشاء



صورة رقم (4)

استخدم GPU المجاني



صورة رقم (5)

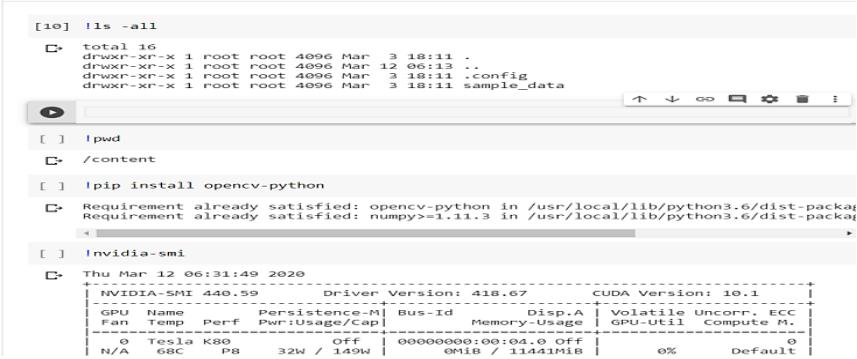
نفذ الأمر (اضغط على alt + enter لتنفيذ بسرعة . يمكن تشغيل كود بايثون مباشرة ، مثل

```
import os
import numpy as np
x = 'ok'
print(x)
```

The code cell contains four lines of Python code: importing os and numpy, defining a variable x as 'ok', and printing x. Below the code cell, the output cell shows the result 'ok'.

صورة رقم (6)

يمكن لهذا الكمبيوتر الدفتري أيضًا تنفيذ بعض الأوامر تحت نظام لينكس ، لأن هذا في الواقع جهاز لينكس ظاهري ولكن عند تنفيذ أوامر لينكس ، أضاف! أمامه ، مثل ls ::! pwd.



```
[10] ls -all
total 16
drwxr-xr-x 1 root root 4096 Mar  3 18:11 .
drwxr-xr-x 1 root root 4096 Mar 12 06:13 ..
drwxr-xr-x 1 root root 4096 Mar  3 18:11 .config
drwxr-xr-x 1 root root 4096 Mar  3 18:11 sample_data

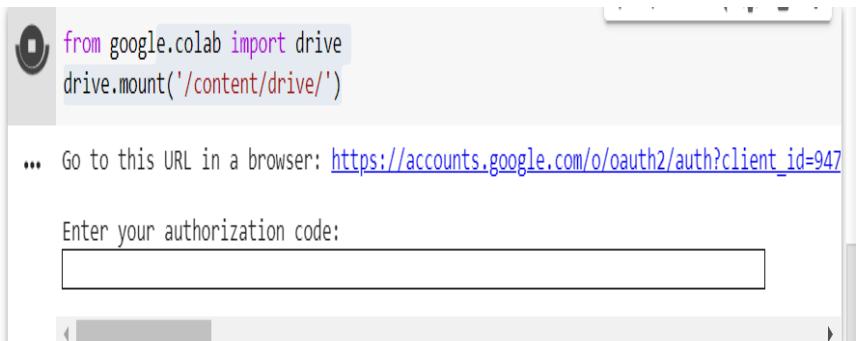
[ ] !pwd
[ ] /content

[ ] !pip install opencv-python
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages

[ ] !nvidia-smi
Thu Mar 12 06:31:49 2020
+-----+-----+-----+
| NVIDIA-SMI 440.59 | Driver Version: 418.67 | CUDA Version: 10.1 |
+-----+-----+-----+
| GPU Name Persistence-M| Bus-Id Disp.A Memory-Usage | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr|Usage/Cap| GPU-Util Compute M. |
| 0 Tesla K80 Off 00000000:00:04.0 Off 0MiB / 11441MiB | 0% Default |
| N/A   68C P8 32W / 149W |                         |
+-----+-----+-----+
```

صورة رقم (7)

لتحميل والتعامل مع جوجل درايف لغرض الاستيراد وتصدير البيانات منه وإليه . في الواقع ، يمكن لهذا الكمبيوتر الدفتري الوصول إلى Google Cloud Disk ، وتشغيل الكود التالي ، وسيظهر رابط التفويض.



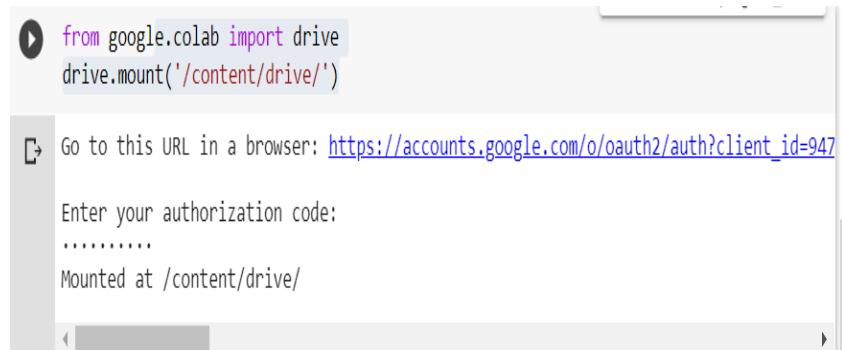
```
from google.colab import drive
drive.mount('/content/drive/')

... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=947

Enter your authorization code:
```

صورة رقم (8)

انقر فوق الارتباط ، وانسخ رمز التحقق إلى المربع ، واضغط على Enter لإكمال ترخيص التحميل



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

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=947

Enter your authorization code:
.....
Mounted at /content/drive/
```

صورة رقم (9)

•بعض مميزات كولاب:-

1. كولاب هو جهاز Linux الظاهري مع GPU ، تحتاج إلى إضافة "!" قبل تنفيذ أمر لينكس، يمكنك كتابة وتنفيذ كود ببايثون مباشرة.
2. في كل مرة تقوم فيها بتسجيل الدخول مرة أخرى ، تحتاج إلى تحميل Google Drive ، وسيتم إنشاء مجلد محرك أفراد (ie / content / drive /) في الجهاز الظاهري ، ثم تحتاج إلى تنفيذ أمر لتبديل المسار الحالي إلى / content / drive / My Drive / لرؤية الملف على Google Cloud Disk.
3. يمكن استخدام Colab بشكل مستمر لمدة تصل إلى 12 ساعة ، وبمرور الوقت ، سيقوم النظام بمقاطعة برنامج التشغيل بالقوة واستعادة الجهاز الظاهري المشغول. وسيتم تعين أولوية أقل لمستخدمي colab الذين يستخدمون GPU بشكل متكرر لاستخدام GPU ، وبإمكانك عمل جلسة تشغيل مرة أخرى لمهمة أخرى .

في الختام، توفر Python، جنباً إلى جنب مع الوحدات النمطية والمكتبات ونماذج CNN وأدوات الذكاء الاصطناعي، منصة شاملة لتطوير التطبيقات المتعلقة بتحليل الصور الطبية. هذا يمكن الباحثين والمبرمجين المحترفين وممارسي الرعاية الصحية من تعزيز فهتمهم واتخاذهم للقرار في مجال الطب.

التطبيقات وال kodas البرمجية للكتاب

نرجو من قرائنا الكرام تنزيل جميع التطبيقات وال kodas البرمجية من خلال رمز الاستجابة السريع قبل البدء بالتطبيق العملي لجميع تطبيقات الكتاب ويرجى خزنها في الحاسبة الشخصية للمستخدم ومتابعة الخطوات مع الكتاب. مع جزيل الشكر والتقدير



نقدم مجموعة بيانات طبية شاملة يتجاوز حجمها أكثر من 50 جيجابايت، مصدرها مراكز طبية وموقع إلكترونية عالمية. تم تنظيم وتدقيق مجموعة البيانات هذه بدقة، وهي متاحة لأبحاث الذكاء الاصطناعي في الطب. في مجال التعلم العميق ضمن تطبيقات التصنيف والكشف والتجزئة. إنه مصرح للاستخدام الإنساني، ويلبي احتياجات المبرمجين والمطورين والباحثين وطلبة الحاسوب ومراكز الحاسوب في المؤسسات والمراكز الطبية وعشاق الطب.

التفاصيل الرئيسية:-

1. **المحتوى:** تغطي مجموعة البيانات أكثر من 18 نوعاً من الأمراض البشرية الخطيرة.
2. **التنسيق:** يتم تقديم البعض منها بطريقة منظمة عبر 13 فصلاً، مكتملة بأكواد البرمجة للتطبيقات الخاصة بالأمراض.
3. **الوصول:** يمكن للمستخدمين الوصول إلى مجموعة البيانات على الآلات الحاسوبية الشخصية أو الخوادم السحابية المجانية عبر Google Lab.
4. **الاستخدام:** يمكن للمطورين والباحثين الاستفادة من مجموعة البيانات للتدريب وبناء نماذج ذكاء اصطناعي متخصصة لأمراض معينة.
5. **الوصية:** يوصى بفهم التطبيقات المقدمة واتباع المنهجية الصحيحة واكتساب الخبرة العملية لتطوير النماذج بنجاح. بعد فهمك لامثلة وتطبيقات الكتاب الذي غطي بعض الامراض المذكورة في بعض فصول الكتاب.

قائمة مجموعة البيانات:-

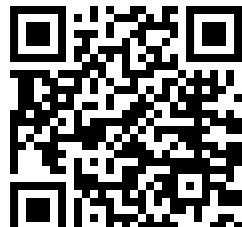
1. مجموعة بيانات ورم الدماغ.
2. فئات ومجاميع الدم.
3. مجموعة بيانات أمراض العيون.
4. سرطان قولون المستقيم.
5. مجموعة بيانات سرطان الكبد.
6. مجموعة بيانات سرطان الرئة.
7. سرطان الثدي-UltraSonicImages.
8. سرطان الثدي.
9. سرطان البروستاتا.
10. سرطان الأقنية في الثدي.
11. حجم التصوير المقطعي للركبة.
12. صور الأشعة المقطعة للجمجمة.
13. صور القلب المقطعة.
14. صور الأشعة السينية للصدر (الالتهاب الرئوي).

15. التعرف على أمراض العين.
16. سرطان الجلد.
17. صور الصدر-الأشعة السينية-بمعيار DICOM
18. مرض الرئة فايروس كوفيد-19.
19. مرض سرطان الدم اللوكيمياء للأطفال.
20. مرض المفاصل وهشاشة العظام.
21. سرطان العظام.
22. سرطان البنكرياس.
23. سرطان الكلى.
24. مرض مفاصل الركبة.
25. مرض الاسنان والتسوس للأطفال.
26. الامراض الجلدية.
27. سرطان الرحم.
28. امراض القلب.
29. سرطان الامعاء.
30. كسور العظام.
31. تسوس الاسنان.
32. سرطان الكبد.
33. هشاشة العظام.
34. سرطان العظام.
35. سرطان الدم اللوكيمياء.

وكثير من مجموعة بيانات الامراض تجدها في روابط رمز الاستجابة السريع .

المجموعة الثانية من مجموعة البيانات الطبية

المجموعة الاولى من مجموعة البيانات الطبية



الفصل الاول: مكتبة خدمة الويب وادوات التعليقات التوضيحية.

الفصل الاول

مكتبة خدمة الويب وادوات التعليقات التوضيحية.

مكتبة Gradio لتطبيقات الويب.



ادوات الملصقات والتعليقات التوضيحية.



مكتبة Gradio لتطبيقات الويب.

Gradio هي مكتبة Python تتيح لك إنشاء مكونات واجهة مستخدم قابلة للتخصيص بسرعة لنموذج التعلم الآلي لديك. إنه مفيد بشكل خاص لإنشاء واجهات ويب للتفاعل مع النماذج الخاصة بك دون الحاجة إلى كتابة كود أمامي (Front End Page Code) واسع النطاق. فيما يلي تفصيل لميزاته الرئيسية وكيف يمكن أن يكون مفيداً لتطبيقات الويب للتعلم العميق في مشاريعك :

1. **إنشاء واجهة بسيطة:** يوفر Gradio واجهة برمجة تطبيقات بسيطة لإنشاء مكونات واجهة المستخدم مثل حقول الإدخال وشرائح التمرير والأزرار. وهذا يجعل من السهل تصميم واجهات للتفاعل مع نماذج التعلم العميق الخاصة بك.

2. **دعم أنواع الإدخال المختلفة:** يدعم Gradio أنواع الإدخال المختلفة بما في ذلك إدخالات النص، وشرائح التمرير، ومربيعات الاختيار، وتحميل الملفات، والمزيد. يتيح لك ذلك إنشاء واجهات تقبل أنواعاً مختلفة من بيانات الإدخال لنموذجك.

3. **التكامل مع أطر التعلم العميق:** يتكامل Gradio بسلاسة مع أطر التعلم العميق الشائعة مثل Keras وTensorFlow وPyTorch. يمكنك بسهولة توصيل نماذجك المدربة وإنشاء واجهات التفاعل معها.

4. **تحديثات النماذج المباشرة:** يوفر Gradio تحديثات مباشرة عند التفاعل مع نماذجك، مما يسمح للمستخدمين برؤية النتائج في الوقت الفعلي أثناء قيامهم بضبط معلمات الإدخال.

5. **قابلية التخصيص:** يقدم Gradio خيارات تخصيص لمكونات واجهة المستخدم، مما يسمح لك بتخصيص الواجهة وفقاً لمتطلباتك المحددة وتفضيلات التصميم.

6. **خيارات النشر:** يدعم Gradio النشر على منصات مختلفة بما في ذلك النشر المحلي والنشر السحابي والتكميل مع تطبيقات الويب الحالية.

في مشروعك، يمكنك استخدام Gradio لإنشاء واجهات ويب للتفاعل مع نماذج التعلم العميق المتعلقة بتصنيف الكتب وأنظمة التوصيات وتحليل المشاعر وتصنيف الامراض وكتشاف وتجزئة صور الامراض وكثير من تطبيقات التعلم الآلي والتعلم العميق لمشاريع الذكاء الاصطناعي التفاعلية مع المستخدم والمزيد من التطبيقات . على سبيل المثال، يمكنك إنشاء واجهة حيث يمكن للمستخدمين إدخال (رفع) صورة لأشعة مرض معين ويمكن لنموذج التعلم العميق الخاص بك التنبؤ بنوع المرض وهذا ما سوف نستخدمه في بعض التطبيقات المذكورة في فصول الكتاب . من خلال الاستفادة من Gradio، يمكنك إنشاء نموذج أولي لهذه الواجهات ونشرها بسرعة دون الحاجة إلى قضاء الكثير من الوقت في تطوير الواجهة الأمامية، مما يسمح لك بالتركيز بشكل أكبر على جوانب التعلم العميق لمشروعك.

لا يتطلب الأمر سوى بضعة أسطر من لغة بايثون لإنشاء عرض توضيحي جميل ، فلنبدأ

• التثبيت .

المطلب السابق: يتطلب Gradio إصدار Python 3.8 أو أعلى. نوصي بتنصيب Gradio باستخدام الامر (pip)، والتي يتم تضمينها افتراضياً في Python. قم بتشغيل هذا في المحطة الطرفية أو موجه الأوامر.

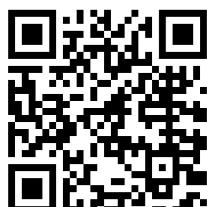
```
pip install gradio  
#or in google colab using !pip install gradio
```

• بناء العرض التجاري الأول الخاص بك.

يمكنك تشغيل Gradio في محرر التعليمات البرمجية المفضل لديك، أو دفتر Jupyter، أو Google Colab، أو في أي مكان آخر تكتب فيه لغة Python. لنكتب تطبيق Gradio الأول الخاص بك:

```
import gradio as gr  
def greet(name, intensity):  
    return "Hello, " + name + "!" * int(intensity)  
demo = gr.Interface(  
    fn=greet,  
    inputs=["text", "slider"],  
    outputs=["text"],)  
demo.launch(share=True)  
#save this code app.py  
#to run in command line using  
#python app.py
```

في كتابنا هذا لانتطرق ولانتعق كثيرا الى هذه المكتبة بالتفصيل لأنها تحتاج الى كتاب مستقل لتعلم هذه المكتبة الرائعة في بناء تطبيقات الويب لجميع مشاريع وتطبيقات التعلم الآلي والتعلم العميق. للاطلاع على هذه المكتبة ومشاريعها اليك هذا الرابط .



الختام

في الختام، تقدم Gradio حلًّا مباشراً لإنشاء واجهات مستخدم تفاعلية لنماذج التعلم العميق، وتبسيط عملية التطوير وتعزيز مشاركة المستخدم. تشمل تحسيناتها المستقبلية المحتملة زيادة التخصيص، ودعم إطار العمل الأوسع، والميزات التعاونية، وتحسين الأداء، وتحسينات النشر، وتعزيز النظام البيئي المجتمعي النابض بالحياة. مع Gradio، يبدو مستقبل بناء واجهات التعلم الآلي البديهية واعداً ومتاحاً للجميع.

ادوات الملصقات والتعليقات التوضيحية (Annotation Tools).

1. اداة LabelMe Tool

2. الاداة Labelme2yolo

اداة Labelme Tool

عبارة عن اداة تعليقات توضيحية للصور قوية ومتعددة الاستخدامات، وُتُستخدم بشكل أساسى في مهام التعرف على الكائنات وتجزئة الصور في التعلم الآلي :

- رسم المربعات المحيطة: قم بتضمين المناطق محل الاهتمام في الصورة لتحديد الكائنات.
- إنشاء مجموعات: تحديد شكل الكائنات المعقدة بدقة.
- تصنيف الكائنات: قم بتعيين أسماء أو فئات ذات معنى للمناطق المنشورة.
- تحميلمجموعات البيانات وإدارتها: إنشاءمجموعات من الصور المصنفة وتنظيمها.
- التعاون مع الآخرين: شارك بياناتك المصنفة مع مستخدمين آخرين لأغراض البحث أو التدريب.

• الميزات الرئيسية لبرنامج LabelMe :

- واجهة سهلة الاستخدام: تم تصميم الواجهة المستندة إلى الويب لسهولة الاستخدام، مما يجعلها في متناول الأشخاص الذين ليس لديهم خبرة سابقة في التعليقات التوضيحية للصور.
- أدوات التعليقات التوضيحية المتنوعة: يوفر LabelMe مجموعة من الأدوات لتلبية احتياجات التعليقات التوضيحية المختلفة، بما في ذلك المربعات المحيطة والمجموعات والنقط والخطوط.
- تصنيف الكائنات المتعددة: يمكنك تصنيف كائنات متعددة داخل صورة واحدة، مما يسمح بمجموعات بيانات معقدة.
- مفتوحة المصدر وتعاونية: LabelMe هي أداة مفتوحة المصدر تعمل على تعزيز التعاون ومشاركةمجموعات البيانات المصنفة داخل مجتمع التعلم الآلي.
- دعم تنسيقات الصور المختلفة: يتعامل مع تنسيقات الصور المختلفة، بما في ذلك JPEG، GIF، وPNG.

• حالات استخدام LabelMe :

- اكتشاف الكائنات: تدريب نماذج التعلم الآلي لتحديد الكائنات داخل الصور، مثل السيارات أو المشاة أو منتجات معينة.
- تجزئة الصور: تحديد الخطوط العريضة الدقيقة للأشياء، مثل الأعضاء في الصور الطبية أو الأنواع المختلفة من النباتات في صور القمر الصناعي.
- أبحاث رؤية الكمبيوتر: إنشاءمجموعات بيانات لمشاريع بحثية متعددة في مجال رؤية الكمبيوتر، بما في ذلك تصنيف الصور وتتبع الكائنات وفهم المشهد.

الفصل الاول: مكتبة خدمة الويب وادوات التعليقات التوضيحية.

4- **تعزيز البيانات**: إنشاء صور ذات علامات جديدة من خلال تطبيق التحويلات على البيانات المنشورة الموجودة، مما يزيد من تنوع وحجم مجموعة التدريب الخاصة بك.

• من أين تبدأ مع :LabelMe

1- **المنصة عبر الإنترن트**: قم بزيارة موقع LabelMe الرسمي (<http://labelme.csail.mit.edu/Release3.0/>) لبدء استخدام الأداة مباشرةً على الويب الخاص بك .browser

2- **الوثائق**: يوفر موقع LabelMe الإلكتروني وثائق وبرامج تعليمية شاملة لمساعدتك على بدء استخدام الأداة.

3- **منتديات المجتمع**: يمكنك العثور على موارد مفيدة والتواصل مع مستخدمي LabelMe الآخرين في المنتديات والمجتمعات عبر الإنترن特.

• بدائل LabelMe

على الرغم من أن LabelMe يعد خياراً شائعاً، إلا أن هناك أدوات أخرى متاحة للتعليق على الصور،

مثل:

1- **VGG Image Annotator (VIA)**: أداة تعتمد على الويب مع التركيز على سهولة الاستخدام ومهام التعليقات التوضيحية البسيطة.

2- **LabelImg**: أداة بسيطة وفعالة لإنشاء التعليقات التوضيحية للمربي المحيط، وذلك بشكل أساسي لاكتشاف الكائنات.

3- **CVAT**: نظام أساسي أكثر تقدماً يدعم أنواع التعليقات التوضيحية المتنوعة والميزات التعاونية والتكامل مع إطار عمل التعلم الآلي.

اختيار الأداة المناسبة:

ستعتمد أفضل أداة للتعليق التوضيحي على الصور لتلبية احتياجاتك على عوامل مثل مدى تعقيد مهام التعليقات التوضيحية وحجم مجموعة البيانات وميزانيتك. فكر في تجربة أدوات مختلفة للعثور على الأداة التي تناسب احتياجاتك.

• التنصيب.

```
pip install labelme
```

بعد تكملة التنصيب عند نافذة سطر الأوامر اكتب الامر التالي : **labelme** وانتظر قليلا لنظهر لك نافذة واجهة المستخدم الرسومية (GUI) . في مقدمة بنية شركة Labelme توجد واجهة المستخدم الرسومية البديهية، والتي تعمل كواجهة أساسية للمستخدمين للتعليق على الصور بشكل تفاعلي. توفر واجهة المستخدم الرسومية مجموعة متنوعة من أساسيات التعليقات التوضيحية، مما يسهل وضع العلامات الدقيقة للكائنات داخل الصور.

الفصل الاول: مكتبة خدمة الويب وادوات التعليقات التوضيحية.

على سبيل المثال، يمكن للمستخدمين إضافة تعليقات توضيحية إلى الكائنات في مثل صورة الحافلة أدناه، مع تمثيل كل تسمية باللون وأشكال مميزة من أجل الوضوح والتنظيم. تتميز الحافلات والسيارات بأنها مميزة ومصنفة بشكل صحيح.

تعمل واجهة المستخدم الرسومية على تحسين تجربة المستخدم من خلال السماح بالمرنة في طرق الإدخال، ودعم التعليقات التوضيحية للملفات الفردية أو مجلدات الصور بأكملها. بالإضافة إلى ذلك، يتتوفر للمستخدمين خيار تحديد العلامات، مما يتيح تنسيقات الإخراج المخصصة ووظائف حفظ البيانات تلقائياً. يمكن هذا المستوى من التخصيص للمستخدمين من تخصيص عملية التعليق التوضيحي وفقاً لمتطلباتهم المحددة، مما يعزز الكفاءة والراحة. في الصورة أدناه مثال للتعليق باستخدام المربعات المحيطة وتسميات الحفظ التلقائي في ملف الإخراج المحدد.



صورة تظهر تحديد باص النقل بمضلعتها او بمربعات مع مسمياتها

الاداة الثانية:-

وهي أداة تعمل على سد الفجوة بين أداتين شائعتين في عالم التعليقات التوضيحية للصور واكتشاف الكائنات: YOLO و LabelMe (أنت تنظر مرة واحدة فقط). إليك ما تحتاج إلى معرفته حول أداة :LabelMe2YOLO

• ما هي **LabelMe2YOLO**؟

LabelMe2YOLO هي أداة معايدة مفيدة مصممة لتحويل التعليقات التوضيحية للصور التي تم إنشاؤها بتنسيق LabelMe إلى تنسيق YOLO.

وهذا أمر ضروري لأنه:

- **تنسيق LabelMe:** يستخدم LabelMe بنية البيانات الخاصة به للتعليقات التوضيحية، والتي يتم تخزينها عادةً في ملفات JSON.

- **تنسيق YOLO:** YOLO، وهو إطار عمل قوي لاكتشاف الكائنات، يفضل التعليقات التوضيحية بتنسيق نصي مختلف لتدريب نماذجه.

• لماذا يعد التحويل ضروريًا؟

إذا قمت بتسمية مجموعة بيانات بدقة في LabelMe وترغب في تدريب نموذج اكتشاف كائن YOLO، فأنك بحاجة إلى تحويل تلك التعليقات التوضيحية إلى تنسيق يمكن له YOLO فهمه. وهذا يأتي دور LabelMe2YOLO.

• كيف تعمل؟

تأخذ أداة LabelMe2YOLO ملفات LabelMe JSON كملفات إدخال وإخراج نصية متواقة مع إطار عمل YOLO. يقوم بشكل أساسى بما يلى:

1. يقرأ بيانات **LabelMe JSON**: يوزع معلومات المربع المحيط بالكائن، والتسميات، ومسارات الصور المخزنة بتنسيق LabelMe JSON.
2. **التحويل إلى تنسيق YOLO**: يترجم البيانات إلى بنية ملف نصي محدد يتطلبه YOLO للتدريب. يتضمن هذا غالباً تحويل الإحداثيات وتغيير حجم المعلومات وإضافة تسميات بطريقة تتوقعها YOLO.

البحث عن **LabelMe2YOLO** واستخدامه:

* **مصدر مفتوح:** يمكنك عادةً العثور على تطبيقات مفتوحة المصدر له LabelMe2YOLO على منصات مثل GitHub.

* **التثبيت:** غالباً ما تتطلب الأداة التثبيت كحزمة أو برنامج نصي له Python.

* **تشغيل التحويل:** بمجرد التثبيت، ستستخدم أمراً بسيطاً لتوجيه الأداة إلى دليل ملفات LabelMe JSON لديك. سيقوم بعد ذلك بإنشاء ملفات YOLO النصية المقابلة في دليل الإخراج المطلوب.

ملاحظات هامة:

* **إصدار YOLO:** تأكد من أن أداة LabelMe2YOLO التي تستخدمها متواقة مع إصدار YOLO المحدد (YOLOv3، YOLOv4، YOLOv5).

* **التخصيصات:** قد توفر بعض تطبيقات LabelMe2YOLO خيارات للتخصيص، مثل تغيير حجم الصور أو تعديل تنسيق الإخراج قليلاً ليناسب إعداد YOLO المحدد لديك.

الفصل الاول: مكتبة خدمة الويب وادوات التعليقات التوضيحية.

تعد أداة **LabelMe2YOLO** بمثابة جسر حيوي للمستخدمين الذين يرغبون في الاستفادة من قوة **YOLO** لاكتشاف الكائنات مع الاستفادة من راحة **LabelMe** للتعليقات التوضيحية. إذا كنت تعمل مع مجموعات بيانات الصور وتخطط لاستخدام **YOLO**، فإن أداة التحويل هذه تعد من الأصول القيمة.

- **تنصيب الاداة:**

مع تنفيذ تحويل ملفات الجيسون من الخطوة السابقة الى ملفات الملصقات بنسق (txt)

```
pip install labelme2yolo  
labelme2yolo --json_dir /path/to/labelme_json_dir
```

بامكانك الاطلاع عليها في الموقع التالي.

[/https://pypi.org/project/labelme2yolo](https://pypi.org/project/labelme2yolo)

وهناك كثير من المواقع المجانية او المدفوعة مسبقا باجور رمزية معينة تقوم مهام تنظيم الملصقات وتحويل الناتج النهائي الى بيانات التدريب والتحقق والاختبار مع ملف التكوين . بامكانك البحث عن الادوات () Annotation label for detection tools (). في كوكيل

- **الناتج النهائي لملف الملصقات.**

- **تنسيق الملصقات YOLO**

تدعم معظم منصات وادوات التعليقات التوضيحية اعلاه التصدير بتنسيق **YOLO** ، مما يوفر ملفاً نصياً واحداً للتعليقات التوضيحية لكل صورة. يحتوي كل ملف نصي على تعليق توضيحي واحد للمربيع المحيط (BBox) لكل كائن في الصورة. يتم تسوية التعليقات التوضيحية وفقاً لحجم الصورة، وتقع في النطاق من 0 إلى 1. ويتم تمثيلها بالتنسيق التالي:

<object-class-ID> <عرض الصندوق> <Y مركز> <X center> <Box height>

0	0.383	0.439	0.183	0.628
0	0.507	0.454	0.191	0.713

صورة شكل الملف النصي لعناصر وابعاد الصندوق المحاط بعنصر معين ذات الفئة صفر

الختام

نختم الجزء هذا والذي تحدث عن اهم ادوات التعليقات التوضيحية ببعض الخيارات المهمة:-

• الخيار الأول: التأكيد على أهمية الاختيار

يعد اختيار أداة التعليقات التوضيحية خطوة حاسمة في أي مشروع للتعلم الآلي القائم على الصور. لا توجد أداة "أفضل" واحدة؛ ويعتمد الحل المثالي على الطبيعة المحددة لبياناتك، وسهولة الاستخدام وأنواع التعليقات التوضيحية المدعومة والتوافق مع إطار التعلم الآلي المختارة تلعب جميعها دوراً، فمن خلال التقييم الدقيق للخيارات المتاحة والنظر في هذه العوامل، يمكن للباحثين والمطورين اختيار أداة التعليقات التوضيحية التي تدعم أهدافك بشكل أفضل وتسهل عملية أكثر كفاءة وفعالية.¹¹

• الخيار الثاني: تسليط الضوء على التنوع والتعاون.

توفر أدوات التعليقات التوضيحية للصور الحديثة مجموعة من الميزات التي تمكّن الباحثين والمطورين من إنشاء مجموعات بيانات عالية الجودة لمجموعة متنوعة من مهام رؤية الكمبيوتر. بدءاً من التعليقات التوضيحية الأساسية للمربي المحيط وحتى تجزئة المضلعات المعقدة وحتى ميزات التعاون، توفر هذه الأدوات المرونة والكافحة ومن خلال احتضان الطبيعة التعاونية لهذه المنصات والاستفادة من وظائفها المتنوعة، يمكننا تسريع عملية تطوير نماذج قوية ودقيقة للتعلم الآلي.

• الخيار 3: التركيز على الاتجاهات المستقبلية.

إن مشهد أدوات التعليقات التوضيحية للصور يتظاهر باستمرار، متقدماً بالتقىم في التعلم الآلي وتصميم واجهة المستخدم وتشمل الاتجاهات الناشئة للتعليقات التوضيحية بمساعدة الذكاء الاصطناعي، والتي تعزز التعلم الآلي لأنّه أجزاء من العملية وتقليل وقت وضع العلامات. بالإضافة إلى ذلك، فإن التطوير تعمل المنصات السحابية على تعزيز قابلية التوسيع والتعاون مع استمرار هذه التقنيات في النضج، يمكننا أن نتوقع ظهور أدوات أكثر تطواراً وسهولة الاستخدام، مما يزيد من تسهيل إنشاء بيانات تدريب عالية الجودة لتطبيقات الرؤية الحاسوبية.

الفصل الثاني

نموذج تجزئة سرطان الثدي باستخدام نموذج UNet

Breast Cancer by using UNet Segmentation Model

المقدمة.

بيانات سرطان الثدي.

ال코드 البرمجي.

استدعاء نموذج UNet

كود التدريب النهائي.

الملخص.

المقدمة

يعد سرطان الثدي أحد أكثر أسباب الوفاة شيوعاً بين النساء في جميع أنحاء العالم. يساعد الاكتشاف المبكر في تقليل عدد الوفيات المبكرة. تستعرض البيانات الصور الطبية لسرطان الثدي باستخدام الفحص بالموجات فوق الصوتية. يتم تصنيف مجموعة بيانات الموجات فوق الصوتية للثدي إلى ثلاثة فئات: الصور العاديّة والحميدة والخبيثة. يمكن أن تنتج صور الموجات فوق الصوتية للثدي نتائج رائعة في تصنيف سرطان الثدي واكتشافه وتجزئته عندما يقترن بالتعلم الآلي.

في هذا الفصل من الكتاب سأحاول تطوير مجموعات من تقنيات تصنيف الصور باستخدام Deep PyTorch Learning لتحليل صور أنسجة الثدي الخاصة بسرطان. وتتجذر الإشارة إلى أن مهمة التحديد الدقيق للأنواع الفرعية لسرطان الثدي وتصنيفها هي مهمة سريرية حاسمة يمكن أن تستغرق ساعات من اختصاصي علم الأمراض المدربين لإكمالها. سأحاول أتمتها في تطبيق أساليب التعلم العميق على الأنسجة وهي تقنية تجزئة أنسجة سرطان الثدي ، كما تم التقاطها في صور الشريان بأكملها. تم استخدام نماذج التشغيل وفك التشغيل باستخدام نموذج UNet والتي قدمت تنبؤاً ناجحاً للتجزئة يتفق كمياً جيداً مع الحقائق الأساسية.

بيانات سرطان الثدي.

تتضمن البيانات التي تم جمعها في الأساس صوراً بالموجات فوق الصوتية للثدي بين النساء اللاتي تتراوح أعمارهن بين 25 و 75 عاماً. تم جمع هذه البيانات في عام 2018. وبلغ عدد المرضى 600 مريض. تكون مجموعة البيانات من 780 صورة بمتوسط حجم صورة 500 * 500 بكسل. الصور بصيغة PNG يتم تقديم صور الحقيقة الأساسية مع الصور الأصلية. يتم تصنيف الصور إلى ثلاثة فئات ، وهي طبيعية وحميدة وخبيثة.

• الكود البرمجي .

```
# Colab's file access feature
from google.colab import files
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
```

ملاحظة:-

طبعاً بعد عمل حساب على خدمة كاكل وتم شرحه في الفصول السابقة للحصول على مفتاح التخويل .
<https://www.kaggle.com/settings>

```
!kaggle datasets download -d falahgatea/breast-cancer-ultrasonicinages
!unzip /content/breast-cancer-ultrasonicinages.zip
!rm -r /content/breast-cancer-ultrasonicinages.zip
```

• استدعاء المكتبات المهمة .

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
from glob import glob
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow import keras
from tensorflow.keras.layers import Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import backend as K
from tensorflow.keras import layers
from tensorflow.keras import models
from PIL import Image
import cv2
```

• شرح الكود.

يستورد الكود مكتبات ووحدات أساسية لمعالجة البيانات ، والتصور ، وبناء نموذج التعلم العميق. وهي تشمل NumPy و Pandas لمعالجة البيانات ، و Matplotlib للخطيط ، و TensorFlow للتعلم العميق ، و PIL و OpenCV لمعالجة الصور ، و scikit-Learn لتقسيم البيانات. تستخدم وحدة Keras داخل TensorFlow لبناء الشبكات العصبية وتدريبيها. يستخدم الكود أيضًا فئة ImageDataGenerator لزيادة البيانات أثناء تدريب النموذج. بشكل عام ، تتيح هذه المكتبات عمليات مختلفة على البيانات ومعالجة الصور وإنشاء نماذج التعلم العميق.

```
def create_data(data_dir, categories):
    images = []
    masks = []
    labels = []
    for i, category in enumerate(categories):
        category_dir = os.path.join(data_dir, category)
        for file_name in os.listdir(category_dir):
            if "_mask" in file_name:
                continue
            image_file_path = os.path.join(category_dir, file_name)
            mask_file_name = file_name.replace(".", "_mask.")
            mask_file_path = os.path.join(category_dir, mask_file_name)
            if not os.path.exists(mask_file_path):
```

```

    continue
    images.append(image_file_path)
    masks.append(mask_file_path)
    labels.append(category)
print(f"Found {len(images)} image/mask pairs in {len(categories)} categories")
return images, masks, labels

```

• شرح الكود.

تقوم وظيفة "create_data" بتجمیع أزواج الصور والقیاع مع التسمیات المقابلة لها من دلیل محدد. يقوم بالتکرار عبر الفنات المتوفرة ومسح الدلیل لملفات الصور والقیاع. يتخطی الملفات التي تحتوي على "mask_" في أسمائها وتلك التي لا تحتوي على ألقعة مقابلة. تقوم الوظيفة بتخزین مسارات الملفات والتسمیات في قوائم منفصلة وإرجاعها. هذه الوظيفة مفيدة لتجمیع بيانات الصورة لمهم مثل تجزئة الصورة أو تصنیفها.

```

data_dir = "/content/BreastCancer"
categories = ['benign', 'malignant', 'normal']
images, masks, labels = create_data(data_dir, categories)

```

• شرح الكود.

يجمع الكود المعطى أزواج الصور والقیاع جنبا إلى جنب مع الملصقات المقابلة لها من الدلیل " / BreastCancer ". الفنات المعترفة هي "حميدة" و "خبیثة" و "طبيعيه". تستدیع الوظيفة `create_data` بالدلیل والفنات المحددة ، وتعید ثلاثة قوائم: "الصور" ، التي تحتوي على مسارات ملفات الصور ، "الاقعه" ، التي تحتوي على مسارات ملفات الألقعة المقابلة ، و "labels" ، والتي تحتوي على تسمیات الفنات المرتبطة بكل زوج. يتيح هذا الرمز استرداد بيانات الصورة والتسمیات المرتبطة بها لمزيد من التحلیل أو التدربی على النموذج.

Found 780 image/mask pairs in 3 categories

• الكود البرمجي.

```

df = pd.DataFrame({'image':images,'mask':masks,'label':labels})
df.head()

```

• شرح الكود.

يقوم الكود المعطى بإنشاء DataFrame ي تكون DataFrame من ثلاثة أعمدة: "صورة" و "قیاع" و "تسمیة". يتم ملء عمود "الصورة" بالقيم من قائمة "الصور" ، ويتم ملء عمود "القیاع" بالقيم من قائمة "الاقعه" ، ويتم ملء عمود "التصنیف" بالقيم من "التصنیفات" قائمة. بعد إنشاء DataFrame ، يتم استدعاء الأسلوب `head ()` في DataFrame ، والذي يعرض الصحفوف القليلة الأولى من DataFrame. يوفر هذا معاینة سریعة للبيانات ، ويظهر مسارات ملفات الصور ومسارات ملفات القیاع والتسمیات المقابلة.

• الكود البرمجي.

```
df_train, df_test = train_test_split(df, test_size=0.1)
```

يقسم الكود المعطى `df` إلى إطارين منفصلین من `DataFrames` `df_train` و `df_test` ، باستخدام الوظيفة `train_test_split` من مکتبة `scikit-Learn` . يتم إجراء التقسيم بناءً على

الفصل الثاني: نموذج تجزئة سلطان الثدي باستخدام نموذج UNet
 حجم اختبار محدد قدره 0.1، مما يعني أنه سيتم تخصيص 10٪ من البيانات لمجموعة الاختبار ، وسيتم تخصيص نسبة 90٪ المتبقية لمجموعة التدريب.
 تقوم الوظيفة "train_test_split" بترتيب البيانات عشوائياً وتقسيمها إلى مجموعتين، مما يضمن أن كل المجموعتين لها تمثيل نسبي للفنات أو التسميات المختلفة الموجودة في DataFrame الأصلي. يساعد هذا التوزيع العشوائي على تجنب التحيز في المجموعات الناتجة.
 الغرض من هذا الرمز هو إنشاءمجموعات بيانات تدريب واختبار منفصلة، وهي ممارسة شائعة في التعلم الآلي.
 تُستخدم مجموعة التدريب لتدريب نموذج، بينما تُستخدم مجموعة الاختبار لتقييم أداء النموذج على البيانات غير المرئية. يسمح هذا الانقسام بتقدير مدى نجاح النموذج في التعميم على الأمثلية الجديدة غير المرئية ويساعد على تقييم فعاليته في سيناريوهات العالم الحقيقي.

- **ال코드 البرمجي .**

```
df_benign=df_train[df_train['label']=='benign']
df_malignant=df_train[df_train['label']=='malignant']
df_normal=df_train[df_train['label']=='normal']
```

- **شرح الكود .**

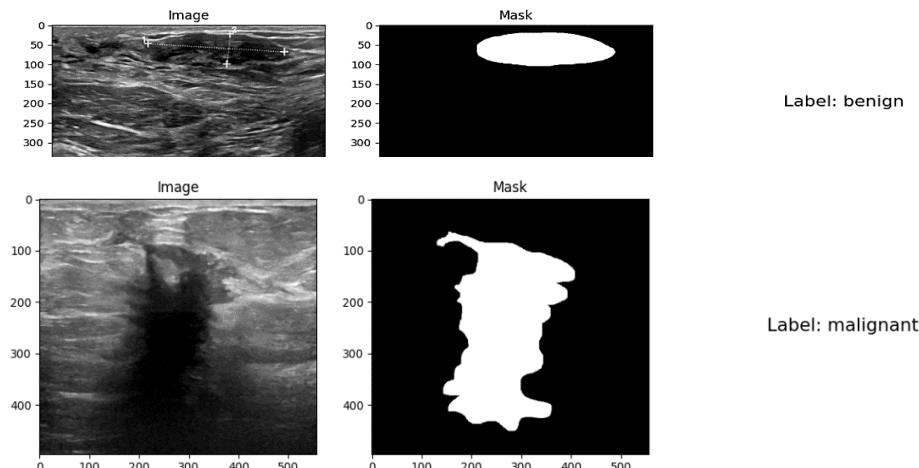
ينشئ الكود المحدد ثلاثة إطارات بيانات منفصلة ، "df_benign" ، "df_malignant" ، و "df_normal" ، عن طريق تصفية "df_train" DataFrame استناداً إلى قيم تسمية مختلفة. يحتوي كل إطار DataFrame على صفوف من "df_train" تتوافق مع فئة تصنيف معينة ، مثل "حميد" أو "خبيث" أو "عادي". يسمح هذا بالتحليل والتلاعب ببيانات التدريب الخاصة بكل فئة. من خلال تقسيم البيانات إلى هذه المجموعات الفرعية ، يصبح من الأسهل تطبيق العمليات أو النماذج الخاصة بالفن ، ودراسة الخصائص والأنماط الفريدة داخل كل مجموعة فرعية على حدة.

- **ال코드 البرمجي .**

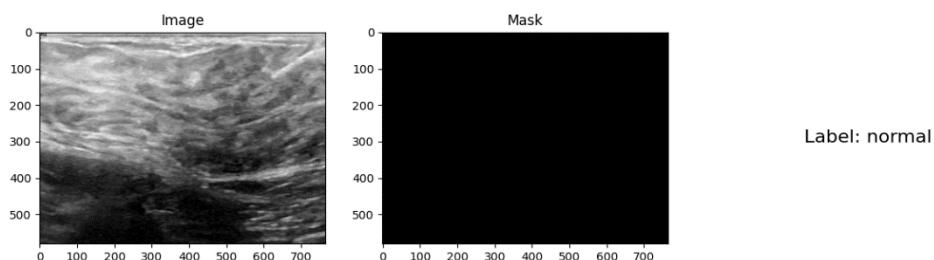
```
def plot_data(data,index):
    row = data.iloc[index]
    img = cv2.imread(row['image'])
    msk = cv2.imread(row['mask'])
    label = row['label']
    fig, axs = plt.subplots(1, 3, figsize=(15, 5))
    axs[0].imshow(img)
    axs[0].set_title('Image')
    axs[1].imshow(msk)
    axs[1].set_title('Mask')
    axs[2].text(0.5, 0.5, f'Label: {label}', ha='center', va='center', fontsize=16)
    axs[2].axis('off')
    plt.show()
```

• شرح الكود .

تعمل وظيفة `plot_data` على تصور الصورة والقائمة المقابل لها والتسمية المرتبطة بها. يأخذ DataFrame وفهرس كمدخلات. باستخدام OpenCV ، يقرأ الصور وملفات القائمة المحددة في صف DataFrame. يتم عرض الصورة والقائمة في مخططات فرعية منفصلة ، بينما يتم عرض التسمية كنص. تتيح هذه الوظيفة طريقة سريعة ومرحة لفحص بيانات الصورة بصرياً جنباً إلى جنب مع الألقنة والتسميات.



صورة رقم (2)



صورة رقم (3)

• الكود البرمجي .

```
def train_val_gen(df):
    data_train,data_val=train_test_split(df,test_size=0.2)
    datagen = ImageDataGenerator(rotation_range=45,
                                 width_shift_range=0.2,
                                 height_shift_range=0.2,
                                 shear_range=0.2,
                                 zoom_range=0.2,
```

```

horizontal_flip=True,
fill_mode='reflect',
rescale=1./255)

image_train=datagen.flow_from_dataframe(data_train,
                                         target_size=(256,256),
                                         color_mode='rgb',
                                         shuffle=True,
                                         seed=42,
                                         x_col ="image",
                                         batch_size=32,
                                         class_mode=None)

mask_train=datagen.flow_from_dataframe(data_train,
                                         target_size=(256,256),
                                         color_mode='grayscale',
                                         shuffle=True,
                                         seed=42,
                                         x_col ="mask",
                                         batch_size=32,
                                         class_mode=None)

image_validation=datagen.flow_from_dataframe(data_val,
                                              target_size=(256,256),
                                              color_mode='rgb',
                                              shuffle=True,
                                              seed=42,
                                              x_col ="image",
                                              batch_size=32,
                                              class_mode=None)

mask_validation=datagen.flow_from_dataframe(data_val,
                                              target_size=(256,256),
                                              color_mode='grayscale',
                                              shuffle=True,
                                              seed=42,
                                              x_col ="mask",
                                              batch_size=32,
                                              class_mode=None)

train_gen=(image_train,mask_train)
valid_gen=(image_validation,mask_validation)
return train_gen,valid_gen

```

• شرح الكود .

تقوم وظيفة "train_val_gen" بإعداد وإنشاء بيانات التدريب والتحقق من الصحة لنموذج التعلم العميق. يقسم الإدخال إلى مجموعات بيانات التدريب والتحقق من الصحة. يتم تطبيق تقنيات زيادة البيانات مثل التدوير والتغيير والقص والتكبير / التقليب وإعادة القياس على الصور باستخدام "ImageDataGenerator". يتم إنشاء مولدات بيانات منفصلة للصورة وبيانات القناع في كل من مجموعات التدريب والتحقق من الصحة. تقوم الدالة بارجاع المجموعات التي تحتوي على مولدات بيانات التدريب والتحقق من الصحة. تتيح هذه الوظيفة التوليد الفعال لبيانات المعززة للتدريب والتحقق من الصحة، وهو أمر ضروري لتدريب نماذج التعلم العميق القوية.

• الكود البرمجي .

```
train_gen_benign,valid_gen_benign=train_val_gen(df_benign)
train_gen_malignant,valid_gen_malignant=train_val_gen(df_malignant)
train_gen_normal,valid_gen_normal=train_val_gen(df_normal)
```

• شرح الكود .

يعد الكود المعطى بأنه يقوم بتوليد بيانات التدريب والتحقق من الصحة لثلاث فئات: "حميدة" و "خبيثة" و "طبيعية". يتم إنشاء مولدات البيانات المنفصلة لكل فئة عن طريق استدعاء الوظيفة `train_val_gen` بالفترة المقابلة DataFrame. تُمكّن مولدات البيانات هذه من توليد البيانات المعززة بكفاءة لأغراض التدريب والتحقق من الصحة، خاصة بكل فئة. يضمن الكود أن لكل فئة مجموعة خاصة من بيانات التدريب والتحقق، مما يسمح بالتدريب المستهدف وتقييم النماذج المصممة لكل فئة. بشكل عام، يسهل هذا الكود إعداد البيانات الخاصة بالفئة لنماذج التعلم العميق.

• الكود البرمجي .

```
train_gen1 = zip(train_gen_benign[0], train_gen_benign[1])
train_gen2 = zip(train_gen_malignant[0], train_gen_malignant[1])
train_gen3 = zip(train_gen_normal[0], train_gen_normal[1])
valid_gen1 = zip(valid_gen_benign[0], valid_gen_benign[1])
valid_gen2 = zip(valid_gen_malignant[0], valid_gen_malignant[1])
valid_gen3 = zip(valid_gen_normal[0], valid_gen_normal[1])
```

• شرح الكود .

ينشئ الكود المحدد كائنات مضغوطه تجمع بين الدفعات من مولدات بيانات التدريب والتحقق من الصحة لكل فئة. تتيح هذه الكائنات المضغوطه إمكانية التكرار الفعال على الدفعات المجمعة أثناء عملية تدريب النموذج. من خلال ضغط مجموعات الصورة والقناع معًا، يسمح الرمز بالتجزئة المريحة لبيانات الإدخال والبيانات المستهدفة في وقت واحد لكل فئة أثناء التدريب والتحقق من الصحة. يضمن هذا النهج أن النموذج يتلقى الصورة المقابلة وبيانات القناع لكل دفعه، مما يسهل التدريب والتقييم الخاصين بالفترة. بشكل عام، يبسط الكود معالجة مجموعات البيانات ويعزز عملية التدريب للنماذج الخاصة بالفترة.

استدعاء نموذج UNet

```

def unet(input_size=(256,256,3)):
    inputs = layers.Input(input_size)
    conv1 = layers.Conv2D(64, (3, 3), padding='same')(inputs)
    bn1 = layers.Activation('relu')(conv1)
    conv1 = layers.Conv2D(64, (3, 3), padding='same')(bn1)
    bn1 = layers.BatchNormalization(axis=3)(conv1)
    bn1 = layers.Activation('relu')(bn1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(bn1)
    conv2 = layers.Conv2D(128, (3, 3), padding='same')(pool1)
    bn2 = layers.Activation('relu')(conv2)
    conv2 = layers.Conv2D(128, (3, 3), padding='same')(bn2)
    bn2 = layers.BatchNormalization(axis=3)(conv2)
    bn2 = layers.Activation('relu')(bn2)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(bn2)
    conv3 = layers.Conv2D(256, (3, 3), padding='same')(pool2)
    bn3 = layers.Activation('relu')(conv3)
    conv3 = layers.Conv2D(256, (3, 3), padding='same')(bn3)
    bn3 = layers.BatchNormalization(axis=3)(conv3)
    bn3 = layers.Activation('relu')(bn3)
    pool3 = layers.MaxPooling2D(pool_size=(2, 2))(bn3)
    conv4 = layers.Conv2D(512, (3, 3), padding='same')(pool3)
    bn4 = layers.Activation('relu')(conv4)
    conv4 = layers.Conv2D(512, (3, 3), padding='same')(bn4)
    bn4 = layers.BatchNormalization(axis=3)(conv4)
    bn4 = layers.Activation('relu')(bn4)
    pool4 = layers.MaxPooling2D(pool_size=(2, 2))(bn4)
    conv5 = layers.Conv2D(1024, (3, 3), padding='same')(pool4)
    bn5 = layers.Activation('relu')(conv5)
    conv5 = layers.Conv2D(1024, (3, 3), padding='same')(bn5)
    bn5 = layers.BatchNormalization(axis=3)(conv5)
    bn5 = layers.Activation('relu')(bn5)
    up6 = layers.concatenate([layers.Conv2DTranspose(512, (2, 2), strides=(2, 2),
padding='same')(bn5), conv4], axis=3)
    conv6 = layers.Conv2D(512, (3, 3), padding='same')(up6)
    bn6 = layers.Activation('relu')(conv6)
    conv6 = layers.Conv2D(512, (3, 3), padding='same')(bn6)
    bn6 = layers.BatchNormalization(axis=3)(conv6)

```

```

bn6 = layers.Activation('relu')(bn6)
up7 = layers.concatenate([layers.Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(bn6), conv3], axis=3)
conv7 = layers.Conv2D(256, (3, 3), padding='same')(up7)
bn7 = layers.Activation('relu')(conv7)
conv7 = layers.Conv2D(256, (3, 3), padding='same')(bn7)
bn7 = layers.BatchNormalization(axis=3)(conv7)
bn7 = layers.Activation('relu')(bn7)
up8 = layers.concatenate([layers.Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(bn7), conv2], axis=3)
conv8 = layers.Conv2D(128, (3, 3), padding='same')(up8)
bn8 = layers.Activation('relu')(conv8)
conv8 = layers.Conv2D(128, (3, 3), padding='same')(bn8)
bn8 = layers.BatchNormalization(axis=3)(conv8)
bn8 = layers.Activation('relu')(bn8)
up9 = layers.concatenate([layers.Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(bn8), conv1], axis=3)
conv9 = layers.Conv2D(64, (3, 3), padding='same')(up9)
bn9 = layers.Activation('relu')(conv9)
conv9 = layers.Conv2D(64, (3, 3), padding='same')(bn9)
bn9 = layers.BatchNormalization(axis=3)(conv9)
bn9 = layers.Activation('relu')(bn9)
conv10 = layers.Conv2D(1, (1, 1), activation='sigmoid')(bn9)
return models.Model(inputs=[inputs], outputs=[conv10])

```

• شرح الكود.

يحدد الكود المعطى بنية U-Net لتجزئة الصورة. فيما يلي وصف موجز: تتكون بنية U-Net من شبكة التشفير وفك التشفير، حيث تُستخدم وصلات التخطي للحفاظ على المعلومات المكانية أثناء عملية (downsampling) و (upsampling). المدخل للشبكة عبارة عن صورة بحجم 256×256 بثلاث قواعد (RGB).

يتكون جزء المشفر من سلسلة من الطبقات التلافيفية مع عدد متزايد من المرشحات، متبعاً بتنبييع الدفعات وتنشيط ReLU. يتم إجراء التجميع الأقصى بعد كل كتلة تلافيفية لتقليل الأبعاد المكانية. تحتوي طبقة عنق الزجاجة على طبقتين تلافيفيتين مع 1024 مرشحاً، متبعاً بتنبييع الدفعات وتنشيط ReLU.

يبدأ جزء مفكك الشفرة بنقل الطبقات التلافيفية لجمع خرائط المعلم. يتم بعد ذلك ربط خرائط المعلم المضاعفة مع خرائط المعلم المقابلة من جزء المشفر باستخدام اتصالات التخطي. تكون كل كتلة وحدة فك ترميز من طبقتين تلافيفيتين مع تناقص عدد المرشحات وتنبييع الدفعات وتنشيط ReLU.

الطبقة النهائية هي طبقة تلافيفية 1×1 بوظيفة تنشيط سيني، تنتج قناع إخراج بقيم بين 0 و 1. تم إنشاء النموذج باستخدام واجهة برمجة التطبيقات الوظيفية لإطار عمل Keras.

الفصل الثاني: نموذج تجزئة سلطان الثدي باستخدام نموذج UNet بشكل عام ، تعد بنية U-Net خياراً شائعاً لمهام تجزئة الصور ، والمعروفة بقدرتها على التقاط التفاصيل الدقيقة مع الحفاظ على السياق المكاني.

• الكود البرمجي .

```
smooth=1
def dice_coef(y_true, y_pred):
    y_true = K.flatten(y_true)
    y_pred = K.flatten(y_pred)
    intersection = K.sum(y_true * y_pred)
    union = K.sum(y_true) + K.sum(y_pred)
    return (2.0 * intersection + smooth) / (union + smooth)
def dice_coef_loss(y_true, y_pred):
    return 1 - dice_coef(y_true, y_pred)
def bce_dice_loss(y_true, y_pred):
    bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)
    return dice_coef_loss(y_true, y_pred) + bce(y_true, y_pred)
```

• شرح الكود.

يحدد الكود وظائف لتقدير التشابه بين أقنعة التجزئة الحقيقية والمتواعدة باستخدام معامل الترد. تحسب الدالة "dice_coef" معامل الترد من خلال مقارنة تقاطع اتحاد الأقنعة. دالة "dice_coef_loss" تحسب الخسارة على أساس معامل الترد. تجمع الدالة `bce_dice_loss` بين خسارة معامل الترد وخسارة Binary Crossentropy. هذه الوظائف مفيدة لنماذج التدريب في مهام تجزئة الصور ، وتحسينها لتعظيم معامل الترد وتقليل الخسارة الثانية المقاطعة.

• الكود البرمجي.

```
model = unet(input_size=(256, 256, 3))
```

• شرح الكود.

يقوم الكود بإنشاء نموذج UNet لتجزئة الصور بحجم إدخال يبلغ 256×256 و 3 قوّات. وهو يتّألف من بنية وحدة فك التشفير مع وصلات التخطي. يستخرج جزء المشفر الميزات من خلال الطبقات التلaffيفية والتجميع. يقوم جزء وحدة فك التشفير بتجميع الميزات باستخدام الطبقات التلaffيفية للتبديل ويجمعها مع ميزات من المشفر من خلال اتصالات التخطي. تنتج الطبقة النهائية أقنعة تجزئة لكل بكسل باستخدام وظيفة التنشيط السيني.

• الكود البرمجي.

```
model.compile(
    optimizer='adam',
    loss=bce_dice_loss,
    metrics=[dice_coef,'accuracy'])
```

• شرح الكود.

يقوم الكود بتجميع نموذج U-Net للتدريب عن طريق تكوين "adam" ، ووظيفة خسارة مركبة من معامل الترد و (bce_dice_loss) ، مقاييس التقييم بما في ذلك معامل الترد والدقة. يعد هذا النموذج للتدريب والتحسين باستخدام المحسن المحدد لتقليل وظيفة الخسارة. يقيس معامل الترد التشابه بين الأقتعة المتوقعة والحقيقة، بينما تقييم الدقة الدقة الإجمالية لتنبؤات النموذج. من خلال تجميع النموذج مع هذه الإعدادات، يكون جاهزاً للخضوع للتدريب بهدف تقليل الخسارة وتحسين أداء التجزئة.

• الكود البرمجي.

```
history = model.fit_generator(  
    train_gen,  
    validation_data=valid_gen,  
    epochs=30,  
    validation_steps=32,  
    steps_per_epoch=100)
```

• شرح الكود.

يقوم الكود بإجراء تدريب على نموذج U-Net باستخدام تغذية البيانات القائمة على المولد. يقوم بتدريب النموذج على أكثر من 30 حقبة باستخدام دفعات من البيانات التي تم إنشاؤها من "train_gen" ، مع التحقق من صحة أداء النموذج على دفعات من "valid_gen". مع 100 خطوة لكل فترة أثناء التدريب و 32 خطوة للتحقق من الصحة ، يتم تدريب النموذج باستخدام "adam" ووظيفة الخسارة المحددة مسبقاً. يتم تسجيل تقييم التدريب ومقاييس التقييم في كائن "history". يسمح هذا النهج بالتدريب الفعال مع مجموعات البيانات الكبيرة والقدرة على دمج تقييمات زيادة البيانات.

• الكود البرمجي .

```
eval_results = model.evaluate(valid_gen, steps=32, verbose=1)
```

• شرح الكود.

يقوم الكود بتقييم نموذج U-Net المدرب على بيانات التحقق من الصحة باستخدام مولد ("valid_gen"). يحسب مقاييس الأداء مثل الخسارة والدقة. يتم إجراء التقييم على دفعات ، مع 32 خطوة. يتم تخزين النتائج في كائن "EVAL_results". يتيح لنا ذلك تقييم مدى جودة أداء النموذج على بيانات التحقق غير المرئية ، وتوفير رؤى حول فعاليته وتوجيه التحسينات المحتملة.

• الكود البرمجي .

```
for i in range(10):  
    img = cv2.imread(df_test['image'].iloc[i])  
    mask=cv2.imread(df_test['mask'].iloc[i])  
    mask = cv2.resize(mask ,(256, 256))  
    img = cv2.resize(img ,(256, 256))  
    img = img / 255
```

```
img = img[np.newaxis, :, :, :]
pred=model.predict(img)
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(np.squeeze(img))
axs[0].set_title('Original Image')
axs[1].imshow(np.squeeze(mask))
axs[1].set_title('Original Mask')
axs[2].imshow(np.squeeze(pred[0]))
axs[2].set_title('Prediction')
plt.show()
```

• **شرح الكود.**

ينفذ الكود تنبؤات باستخدام نموذج U-Net مدرب على مجموعة من صور الاختبار. يقوم بتغيير حجم الصور والأقنعة، وتنبيه الصور، واستخدام النموذج للتنبؤ بأقنعة الصور. يتم بعد ذلك عرض الصورة الأصلية والقانع الأصلي والقانع المتوقع لكل صورة. يتيح ذلك الفحص البصري لأداء النموذج في مجموعة بيانات الاختبار.

كود التدريب النهائي . 

من رمز الاستجابة السريع



الملخص

نموذج تجزئة سرطان الثدي من PyTorch UNet المقدم في هذا الكود هو خط أنابيب شامل لتقسيم صور سرطان الثدي باستخدام بنية U-Net. يستخدم الكود مكتبات ووظائف مختلفة لتحميل النموذج ومعالجته وتدريبه وتقييمه.

يبدأ الكود باستيراد المكتبات الضرورية وتحديد الوظائف. ثم ينتقل إلى إنشاء وظيفة تجمع أزواج الصور والأقعة للفئات المختلفة من سرطان الثدي. يتم تقسيم البيانات إلى مجموعات تدريب واختبار، ويتم إنشاء إطار بيانات الباندا لتنظيم البيانات.

يتم تعريف نموذج U-Net باستخدام PyTorch ، ويتألف من طبقات تلافيفية وتجميعية مع وصلات تخطي لسلسل المعلم. هذه البنية مناسبة تماماً لمهام تجزئة الصور.

يتم تجميع النموذج باستخدام مُحسّن ووظيفة الخسارة ومقاييس التقييم. يتم استخدام مُحسّن آدم، جنباً إلى جنب مع مزيج من فقدان الانتروبيا الثنائي وخسارة معامل الترد. يتم حساب معامل الترد والدقة كمقاييس تقييم لتقدير أداء النموذج.

تتضمن عملية التدريب استخدام مولدات البيانات لتطبيق تقنيات زيادة البيانات وإنشاء مجموعات من الصور والأقعة للتدريب والتحقق من الصحة. يتم تدريب النموذج باستخدام وظيفة fit_generator ، مع تحديد عدد الحقبات والخطوات لكل حقبة.

بمجرد تدريب النموذج، يتم تقييمه على بيانات التحقق من الصحة باستخدام وظيفة التقييم، مما يوفر رؤى حول أدائه بالإضافة إلى ذلك، يتم إجراء تنبؤات على مجموعة فرعية من صور الاختبار، ويتم تصور الصور الأصلية والأقعة الأصلية والأقعة المتوقعة لمزيد من التحليل.

يوفر هذا الكود الشامل خريطة وخط أنابيب كامل لتجزئة صورة سرطان الثدي باستخدام نموذج U-Net في PyTorch. ويوضح أهمية المعالجة المسبقة للبيانات، وهندسة النماذج، وتقنيات التدريب في تحقيق تقسيمات دقيقة. ويسمح تصور التنبؤات بإجراء تقييم نوعي ورؤى محتملة حول نقاط القوة والضعف في النموذج.

بشكل عام، يعد هذا الكود البرمجي بمثابة أداة قيمة للباحثين والممارسين العاملين في مجال تحليل صور سرطان الثدي. إنه يوفر نهجاً فعالاً لتجزئة صور سرطان الثدي، مما قد يساهم في التقدم في التشخيص وتحفيظ العلاج.

الفصل الثالث

3D Image Classification From CT Scans

المقدمة.

تهيئة المكتبات.

تهيئة البيانات.

معالجة البيانات.

بناء مجموعات بيانات التدريب والتحقق من الصحة.

زيادة البيانات.

تصوير الأشعة المقطعيه المعززة.

تحديد شبكة عصبية تلافية ثلاثية الأبعاد.

تعريف النموذج.

تدريب النموذج.

تصور أداء النموذج.

الكود البرمجي النهائي.

الملخص .

المقدمة.

سيوضح هذا الفصل من الكتاب الخطوات الازمة لبناء شبكة عصبية ثلاثية الأبعاد (CNN) للتنبؤ بوجود التهاب رئوي فيروسي في فحوصات التصوير المقطعي المحوسب (CT). تُستخدم شبكات CNN ثنائية الأبعاد بشكل شائع لمعالجة صور RGB (3 قنوات). أما شبكات (3DCNN) هي ببساطة المكافئ ثلاثي الأبعاد: فهو يأخذ كمدخل حجم ثلاثي الأبعاد أو سلسلة من الإطارات ثنائية الأبعاد (مثل الشرائح في التصوير المقطعي المحوسب) ، تعد شبكات CNN ثلاثية الأبعاد نموذجاً قوياً لتعلم تمثيلات البيانات الحجمية.

تهيئة المكتبات.

```
import os
import zipfile
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

تهيئة البيانات.

في هذا الفصل سوف نستخدم مجموعة فرعية من MosMedData: Chest CT Scans مع نتائج COVID-19 ذات الصلة. تكون مجموعة البيانات هذه من فحوصات التصوير المقطعي المحوسب للرئة مع النتائج ذات الصلة بـ COVID-19 ، وكذلك بدون مثل هذه النتائج. سنستخدم النتائج الإشعاعية المرتبطة بالأشعة المقطعة كسميات لبناء مصنف للتنبؤ بوجود التهاب الرئوي الفيروسي. ومن ثم ، فإن المهمة هي مشكلة تصنيف ثانوي.

```
# Download url of normal CT scans.
```

```
url = "https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-0.zip"
filename = os.path.join(os.getcwd(), "CT-0.zip")
keras.utils.get_file(filename, url)
# Download url of abnormal CT scans.
url = "https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-23.zip"
filename = os.path.join(os.getcwd(), "CT-23.zip")
keras.utils.get_file(filename, url)
# Make a directory to store the data.
os.makedirs("MosMedData")
# Unzip data in the newly created directory.
with zipfile.ZipFile("CT-0.zip", "r") as z_fp:
```

```

z_fp.extractall("./MosMedData/")
with zipfile.ZipFile("CT-23.zip", "r") as z_fp:
    z_fp.extractall("./MosMedData/")

```

معالجة البيانات:

يتم توفير الملفات بتنسيق Nifti .nii. لقراءة عمليات المسح ، نستخدم حزمة nibabel . يمكن تثبيت الحزمة عبر pip install nibabel . تخزن الأشعة المقطعيية كثافة فوكسل الخام في وحدات Hounsfield (HU) . وهي تتراوح من -1024 إلى أكثر من 2000 في مجموعة البيانات هذه. فوق 400 عظمة ذات كثافة إشعاعية مختلفة ، لذلك يتم استخدام هذا كحد أعلى. يتم استخدام عتبة بين 1000 و 400 بشكل شائع لتطبيع التصوير المقطعي المحوسب.

لمعالجة البيانات ، نقوم بما يلي:

نقوم أولاً بتدوير الأحجام بمقدار 90 درجة ، لذلك يكون الاتجاه ثابتاً

نقوم بقياس قيم HU لتكون بين 0 و 1 .

نقوم بتغيير حجم العرض والارتفاع والعمق.

نحدد هنا عدة وظائف مساعدة لمعالجة البيانات. سيتم استخدام هذه الوظائف عند إنشاءمجموعات بيانات التدريب والتحقق من الصحة.

```

import nibabel as nib
from scipy import ndimage
def read_nifti_file(filepath):
    """Read and load volume"""
    # Read file
    scan = nib.load(filepath)
    # Get raw data
    scan = scan.get_fdata()
    return scan
def normalize(volume):
    """Normalize the volume"""
    min = -1000
    max = 400
    volume[volume < min] = min
    volume[volume > max] = max
    volume = (volume - min) / (max - min)
    volume = volume.astype("float32")
    return volume
def resize_volume(img):
    """Resize across z-axis"""
    # Set the desired depth
    desired_depth = 64
    desired_width = 128
    desired_height = 128
    # Get current depth

```

```

current_depth = img.shape[-1]
current_width = img.shape[0]
current_height = img.shape[1]
# Compute depth factor
depth = current_depth / desired_depth
width = current_width / desired_width
height = current_height / desired_height
depth_factor = 1 / depth
width_factor = 1 / width
height_factor = 1 / height
# Rotate
img = ndimage.rotate(img, 90, reshape=False)
# Resize across z-axis
img = ndimage.zoom(img, (width_factor, height_factor, depth_factor), order=1)
return img

def process_scan(path):
    """Read and resize volume"""
    # Read scan
    volume = read_nifti_file(path)
    # Normalize
    volume = normalize(volume)
    # Resize width, height and depth
    volume = resize_volume(volume)
    return volume

```

دعنا نقرأ مسارات التصوير المقطعي المحوسب من أدلة الفنات والاصناف

```

# Folder "CT-0" consist of CT scans having normal lung tissue,
# no CT-signs of viral pneumonia.
normal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-0", x)
    for x in os.listdir("MosMedData/CT-0")]
# Folder "CT-23" consist of CT scans having several ground-glass opacifications,
# involvement of lung parenchyma.
abnormal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-23", x)
    for x in os.listdir("MosMedData/CT-23")]
print("CT scans with normal lung tissue: " + str(len(normal_scan_paths)))
print("CT scans with abnormal lung tissue: " + str(len(abnormal_scan_paths)))

```

- بناء مجموعات بيانات التدريب والتحقق من الصحة.

اقرأ عمليات المسح من أدلة الفصل وقم بتعيين ملصقات. اخترل عمليات المسح للحصول على شكل $128 \times 128 \times 64$. أعد قياس قيم HU الأولية إلى النطاق من 0 إلى 1. وأخيراً ، قسم مجموعة البيانات إلى مجموعات فرعية للفطار والتحقق من الصحة.

```
# Folder "CT-0" consist of CT scans having normal lung tissue,
# no CT-signs of viral pneumonia.
normal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-0", x)
    for x in os.listdir("MosMedData/CT-0")]
]
# Folder "CT-23" consist of CT scans having several ground-glass opacifications,
# involvement of lung parenchyma.
abnormal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-23", x)
    for x in os.listdir("MosMedData/CT-23")]
print("CT scans with normal lung tissue: " + str(len(normal_scan_paths)))
print("CT scans with abnormal lung tissue: " + str(len(abnormal_scan_paths)))
```

زيادة البيانات.

تم زيادة عمليات التصوير المقطعي المحوسب أيضاً بالتناوب بزوايا عشوائية أثناء التدريب. نظراً لأنه يتم تخزين البيانات في موتر من الدرجة 3 للشكل (العينات ، الارتفاع ، العرض ، العمق) ، نضيف بعدها بالحجم 1 في المحور 4 حتى نتمكن من إجراء التفاوتات ثلاثية الأبعاد على البيانات. وهكذا يكون الشكل الجديد (عينات ، ارتفاع ، عرض ، عمق ، 1). هناك أنواع مختلفة من تقنيات المعالجة المسبقة والتعزيز الموجودة هناك ، يوضح هذا المثال عدداً قليلاً من الأساليب البسيطة للبدء.

```
import random
from scipy import ndimage
@tf.function
def rotate(volume):
    """Rotate the volume by a few degrees"""
    def scipy_rotate(volume):
        # define some rotation angles
        angles = [-20, -10, -5, 5, 10, 20]
        # pick angles at random
        angle = random.choice(angles)
        # rotate volume
        volume = ndimage.rotate(volume, angle, reshape=False)
        volume[volume < 0] = 0
        volume[volume > 1] = 1
        return volume
    return tf.numpy_function(scipy_rotate, [volume])
```

```

augmented_volume = tf.numpy_function(scipy_rotate, [volume], tf.float32)
return augmented_volume
def train_preprocessing(volume, label):
    """Process training data by rotating and adding a channel."""
    # Rotate volume
    volume = rotate(volume)
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label

```

أثناء تحديد محمّل بيانات التدريب والتحقق من الصحة ، يتم تمرير بيانات التدريب ووظيفة التعزيز التي تقوم بتدوير الحجم بشكل عشوائي بزوايا مختلفة. لاحظ أنه قد تم بالفعل إعادة قياس بيانات التدريب والتحقق من الصحة بحيث تحتوي على قيم بين 0 و 1.

```

# Define data loaders.
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))
batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2))
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2))

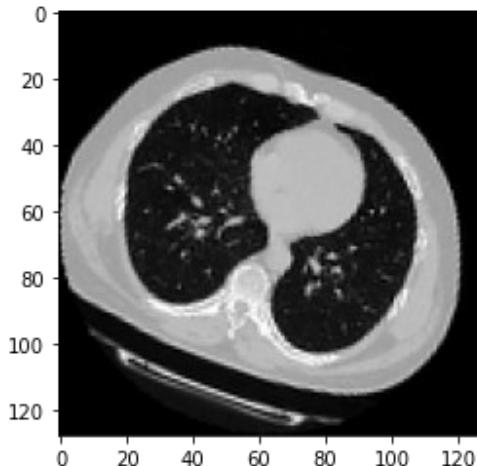
```

• تصوير الأشعة المقطعة المعززة.

```

import matplotlib.pyplot as plt
data = train_dataset.take(1)
images, labels = list(data)[0]
images = images.numpy()
image = images[0]
print("Dimension of the CT scan is:", image.shape)
plt.imshow(np.squeeze(image[:, :, 30]), cmap="gray")

```



صورة رقم (1)

نظرًا لأن الفحص بالأشعة المقطعيه يحتوي على العديد من الشرائح ، اذا بإمكاننا ان نصور اغلب الشائج كمقطع واحد .

```
def plot_slices(num_rows, num_columns, width, height, data):
    """Plot a montage of 20 CT slices"""
    data = np.rot90(np.array(data))
    data = np.transpose(data)
    data = np.reshape(data, (num_rows, num_columns, width, height))
    rows_data, columns_data = data.shape[0], data.shape[1]
    heights = [slc[0].shape[0] for slc in data]
    widths = [slc.shape[1] for slc in data[0]]
    fig_width = 12.0
    fig_height = fig_width * sum(heights) / sum(widths)
    f, axarr = plt.subplots(
        rows_data,
        columns_data,
        figsize=(fig_width, fig_height),
        gridspec_kw={"height_ratios": heights}, )
    for i in range(rows_data):
        for j in range(columns_data):
            axarr[i, j].imshow(data[i][j], cmap="gray")
            axarr[i, j].axis("off")
    plt.subplots_adjust(wspace=0, hspace=0, left=0, right=1, bottom=0, top=1)
    plt.show()
```

```
# Visualize montage of slices.  
# 4 rows and 10 columns for 100 slices of the CT scan.  
plot_slices(4, 10, 128, 128, image[:, :, :40])
```



صورة رقم(2)

٤ تحديد شبكة عصبية تلaffيفية ثلاثة الأبعاد.

لتسهيل فهم النموذج، نقوم ببنائه في كتل. تعتمد بنية شبكة CNN ثلاثة الأبعاد المستخدمة في هذا المثال على هذه الورقة.

<https://arxiv.org/abs/2007.13224>

تعريف النموذج:

```
def get_model(width=128, height=128, depth=64):  
    """Build a 3D convolutional neural network model."""  
    inputs = keras.Input((width, height, depth, 1))  
    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)  
    x = layers.MaxPool3D(pool_size=2)(x)  
    x = layers.BatchNormalization()(x)  
    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(x)  
    x = layers.MaxPool3D(pool_size=2)(x)  
    x = layers.BatchNormalization()(x)  
    x = layers.Conv3D(filters=128, kernel_size=3, activation="relu")(x)  
    x = layers.MaxPool3D(pool_size=2)(x)  
    x = layers.BatchNormalization()(x)  
    x = layers.Conv3D(filters=256, kernel_size=3, activation="relu")(x)  
    x = layers.MaxPool3D(pool_size=2)(x)  
    x = layers.BatchNormalization()(x)  
    x = layers.GlobalAveragePooling3D()(x)
```

```

x = layers.Dense(units=512, activation="relu")(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(units=1, activation="sigmoid")(x)
# Define the model.
model = keras.Model(inputs, outputs, name="3dcnn")
return model
# Build model.
model = get_model(width=128, height=128, depth=64)
model.summary()

```

تدريب النموذج :

```

# Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True)
model.compile(
    loss="binary_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    metrics=["acc"])
# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    "3d_image_classification.h5", save_best_only=True)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)
# Train the model, doing validation at the end of each epoch
epochs = 100
hist=model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
    shuffle=True,
    verbose=2,
    callbacks=[checkpoint_cb, early_stopping_cb])

```

من المهم ملاحظة أن عدد العينات صغير جداً (200 عينة فقط) ولا نحدد بذرة عشوائية. على هذا النحو ، يمكنك توقع تباين كبير في النتائج. يمكن العثور هنا على مجموعة البيانات الكاملة التي تتكون من أكثر من 1000 فحص بالتصوير المقطعي المحوسب. باستخدام مجموعة البيانات الكاملة ، تم تحقيق دقة تبلغ 83٪. لوحظ تباين بنسبة 6-7٪ في أداء التصنيف في كلتا الحالتين.

● معاينة أداء النموذج.

هنا يتم رسم دقة النموذج وخسارة التدريب ومجموعات التحقق من الصحة. نظراً لأن مجموعة التحقق من الصحة متوازنة مع الفئة ، فإن الدقة توفر تمثيلاً غير متحيز لأداء النموذج.

```
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()
for i, metric in enumerate(["acc", "loss"]):
    ax[i].plot(model.history.history[metric])
    ax[i].plot(model.history.history["val_" + metric])
    ax[i].set_title("Model {}".format(metric))
    ax[i].set_xlabel("epochs")
    ax[i].set_ylabel(metric)
    ax[i].legend(["train", "val"])
```

قم بعمل تنبؤات على فحص واحد بالأشعة المقطعيّة

```
# Load best weights.
model.load_weights("3d_image_classification.h5")
prediction = model.predict(np.expand_dims(x_val[0], axis=0))[0]
scores = [1 - prediction[0], prediction[0]]
class_names = ["normal", "abnormal"]
for score, name in zip(scores, class_names):
    print(
        "This model is {:.2f} percent confident that CT scan is {}".format(
            ((100 * score), name)))
```

● الكود البرمجي النهائي.

بإمكانك تنزيل الكود وتقوم بترتيب قيم التدريب لغرض الحصول على أعلى دقة باقل خسارة لبيانات التحقق بعد تدريب النموذج لعدة حقبات التكرار على بيانات التدريب.



الملخص

قدم هذا الفصل من الكتاب نظرة شاملة عن الخطوات المتبعة في بناء شبكة عصبية تلأفيافية ثلاثة الأبعاد (3DCNN) لغرض التنبؤ بوجود الالتهاب الرئوي الفيروسي في عمليات التصوير المقطعي المحوسب (CT). في حين تم استخدام شبكات CNN ثنائية الأبعاد على نطاق واسع لمعالجة صور RGB بثلاث قنوات، فقد فتح ظهور شبكات CNN ثلاثة الأبعاد فرصاً مثيرة للتعامل مع البيانات الحجمية، مثل عمليات المسح المقطعي المحوسب، والتي تتكون من سلسلة من الإطارات أو الشرائح ثنائية الأبعاد.

يمثل تطوير شبكات CNN ثلاثة الأبعاد تقدماً كبيراً في مجال تحليل الصور الطبية ورؤيه الكمبيوتر. ومن خلال توسيع قدرات شبكات CNN التقليدية ثنائية الأبعاد في عالم البيانات ثلاثة الأبعاد، فإننا مجهزون بشكل أفضل لاستخراج المعلومات القيمة من طرق التصوير الطبي الحجمي. وقد أوجز هذا الفصل الخطوات الرئيسية المتبعة في تسخير إمكانات الشبكات العصبية الاصطناعية ثلاثة الأبعاد للكشف عن الالتهاب الرئوي الفيروسي، وهي مهمة حاسمة في مجال التشخيص الطبي.

كما تم تسلیط الضوء على إجراءات التحقق من الصحة والاختبار، مع التأکید على ضرورة تقسیم مجموعة البيانات إلى مجموعات تدريب وتحقیق واختبار لتقيیم قدرات تعیین النموذج بدقة. تم تقديم التحقق من الصحة واستراتیجیات التحقق الأخرى کأدوات قيمة لضمان متانة النموذج المدرب.

باختصار، يمثل تطوير شبكات CNN ثلاثة الأبعاد للتنبؤ بالالتهاب الرئوي الفيروسي في الأشعة المقطعيه وسيلة واعدة في مجال تحليل الصور الطبية. وقد قدم هذا الفصل دليلاً شاملاً لبناء مثل هذه الشبكات وتدريبها، بدءاً من المعالجة المسبقة للبيانات حتى تقييم النماذج. ومن خلال تسخير قوة الشبكات CNN ثلاثة الأبعاد، يمكننا تحسين دقة وكفاءة تشخيص الالتهاب الرئوي، مما يساهم في نهاية المطاف في تحسين رعاية المرضى والنتائج في مجال التصوير الطبي والتشخيص. مع استمرار تقدم التكنولوجيا، فإن تطبيق شبكات CNN ثلاثة الأبعاد في الرعاية الصحية يستعد لتحقيق المزيد من الخطوات، مما قد يحدث ثورة في مجال تحليل الصور الطبية.

الفصل الرابع

Eyes Ocular Disease Classification ResNet-18 By PyTorch API

المقدمة.

البيانات.

تهيئة النموذج.

تطبيق الويب.

ال코드 النهائي.

الملخص.

المقدمة

قد يكون من الصعب على الأطباء تحديد اضطرابات العين مبكراً بما يكفي باستخدام صور قاع العين. إن تشخيص أمراض العيون يدوياً يستغرق وقتاً طويلاً وعرضة للخطأ ومعقد. لذلك، من الضروري وجود نظام آلي للكشف عن أمراض العين باستخدام أدوات بمساعدة الكمبيوتر للكشف عن اضطرابات العين المختلفة باستخدام صور قاع العين. أصبح مثل هذا النظام ممكناً الآن نتيجة لخوارزميات التعلم العميق التي حسنت قدرات تصنيف الصور. يتم تقديم نهج قائم على التعلم العميق للكشف عن العين المستهدفة في هذه الدراسة. في هذه الدراسة، استخدمنا خوارزميات تصنيف الصور الحديثة، يهدف تصنيف أمراض العين باستخدام PyTorch API ونموذج ResNet18 PyTorch إلى تصنیف أمراض العین إلى خمس فئات متمیزة: العيون المتفاکرة واعتام عدسة العین والعيون المتقاطعة والزرق والتھاب القرھیة.

1. انتفاخ العينين: تميز هذه الحالة ببروز غير طبيعي لإحدى العينين أو كليهما من مجرهما. يمكن أن يكون ناتجاً عن عوامل مختلفة، بما في ذلك مشاكل الغدة الدرقية أو الصدمات أو الحالات الطبية الأساسية.

2. اعتام عدسة العين: يشير اعتام عدسة العين إلى غشاوة العدسة داخل العين، مما يؤدي إلى تشوش الرؤية وانخفاض حدة البصر. غالباً ما ترتبط هذه الحالة بالشيخوخة ولكن يمكن أن تنتج أيضاً عن عوامل وراثية أو صدمة أو بعض الحالات الطبية.

3. Crossed_Eyes: المعروف أيضاً باسم الحول، Crossed_Eyes هو حالة تكون فيها العيون منحرفة ولا تركز بشكل صحيح على نفس الكائن. يمكن أن يسبب ازدواج الرؤية ويؤثر على إدراك العمق. يمكن أن تكون العيون المتقاطعة موجودة منذ الولادة أو تتطور لاحقاً في الحياة بسبب مشاكل في عضلات العين أو التحكم في الأعصاب.

4. الجلوکوما: الجلوکوما هي مجموعة من أمراض العيون تميز بتأذف العصب البصري، وغالباً ما يرتبط بزيادة الضغط داخل العين. يمكن أن يؤدي إلى فقدان البصر التدريجي وقد يؤدي في النهاية إلى العمى إذا ترك دون علاج. عادةً ما يتتطور الجلوکوما ببطء ويمكن أن يكون ناتجاً عن عوامل متعددة، بما في ذلك العمر والوراثة وبنية العين.

5. التھاب القرھیة: يشير التھاب القرھیة إلى التھاب الطبقة الوسطى من العین. يمكن أن يسبب احمرار العین، والآلم، وحساسية الضوء، وعدم وضوح الرؤية. يمكن أن يحدث التھاب القرھیة بسبب الالتهابات أو اضطرابات المناعة الذاتية أو حالات كامنة أخرى.

من خلال استخدام نموذج ResNet18، وهو هيكل تعليمي عميق شائع، يمكن للنظام تعلم تحليل وتصنيف صور العين ، مما يتيح تحديداً دقيقاً لأمراض العين الخمسة هذه. يمكن أن يساعد هذا التصنیف في الكشف المبكر والعلاج الفوري وتحسين إدارة حالات العین، مما يساهم في النهاية في تحسين نتائج المرضى وصحة العین.

بيانات

بعد عمل حساب عند خدمة كاكل وكما تم شرحه سابقاً نقوم بتحميل البيانات

```
# Colab's file access feature
from google.colab import files
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600
~/.kaggle/kaggle.json
```

`!kaggle datasets download -d falahgatea/eye-disease-dataset`

فتح الضغط عنها

```
!unzip /content/eye-disease-dataset.zip
!rm -r /content/eye-disease-dataset.zip
```

تهيئة البيانات وتجزئتها إلى بيانات التدريب والاختبار التدقیق.

```
import os
import shutil
from sklearn.model_selection import train_test_split
# Specify the path to your dataset
dataset_path = "/content/Eye_diseases/"
# Specify the paths for train, test, and validation sets
train_path = "/content/dataset/train/"
test_path = "/content/dataset/test/"
val_path = "/content/dataset/validation/"
# Split the dataset into train, test, and validation sets
class_names = os.listdir(dataset_path)
for class_name in class_names:
    class_path = os.path.join(dataset_path, class_name)
    files = os.listdir(class_path)
    if len(files) < 2:
        print(f"Skipping class '{class_name}' due to insufficient samples for train/test split.")
        continue
    train_files, test_val_files = train_test_split(files, test_size=0.5, random_state=42)
    if len(train_files) < 1:
        print(f"Skipping class '{class_name}' due to insufficient samples for train set.")
        continue
```

```

test_files, val_files = train_test_split(test_val_files, test_size=0.5, random_state=42)
# Move the files to their respective folders
for file in train_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(train_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)

for file in test_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(test_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)

for file in val_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(val_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)

```

• تهيئة النموذج.

```

import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import matplotlib.pyplot as plt
# Load pre-trained transformer model
model = models.resnet18(pretrained=True)

```

• شرح الكود.

يستورد الكود المكتبات / الوحدات الضرورية لمهام التعلم العميق في PyTorch. يستورد على وجه التحديد وحدات للشبكات العصبية وخوارزميات التحسين وتحولات الصور ومجموعات البيانات ونماذج المدربة مسبقاً والتصور.

يقوم الكود بعد ذلك بتحميل نموذج ResNet-18 مدرب مسبقاً من مكتبة torchvision. يستخدم هذا النموذج على نطاق واسع في مهام تصنيف الصور نظراً لأناته الممتاز. ضمن الوسيطة "Prerained = True" تنزيل واستخدام الأوزان المدربة مسبقاً.

بشكل عام ، يقوم مقتطف الكود هذا بإعداد المكونات الضرورية للعمل مع نماذج التعلم العميق وتهيئة نموذج ResNet-18 المدربين مسبقاً لمزيد من الاستخدام.

• الكود البرمجي.

```
# Modify the last layer to match the number of classes in your dataset
```

```
num_classes = 5 # Replace with the actual number of classes in your dataset
model.fc = nn.Linear(model.fc.in_features, num_classes)
```

• شرح الكود.

يعدل مقتطف الكود المحدد الطبقة الأخيرة من نموذج ResNet-18 الذي تم تحميله مسبقاً لمطابقة عدد الفئات في مجموعة بيانات معينة. فيما يلي تفصيل للكود:

1. "num_classes = 5": يعيّن هذا السطر المتغير "num_classes" إلى عدد الفئات المطلوب في مجموعة البيانات. تحتاج إلى استبدال "5" بالعدد الفعلي للفئات في مجموعة البيانات الخاصة بك.

2. `model.fc = nn.Linear (model.fc.in_features)` : يستبدل هذا السطر آخر طبقة من النموذج (`model.fc`) بطبقة خطية جديدة (`nn.Linear`) يضبط عدد ميزات الإخراج لمطابقة قيمة "عدد_الفئات". يمثل "model.fc.in_features" عدد معلم الإدخال في الطبقة الأخيرة، ويمثل "num_classes" عدد فئات الإخراج في مجموعة البيانات.

من خلال تعديل الطبقة الأخيرة من النموذج، يمكنك تكييفها لأداء مهام التصنيف على مجموعة بيانات محددة بعد مختلف من الفئات.

• الكود البرمجي.

```
# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

• شرح الكود.

يعين مقتطف الكود المقدم الجهاز للحساب بناءً على توفر وحدة معالجة الرسومات. فيما يلي تفصيل للكود:

1. `device = torch.device (" cuda " if torch.cuda.is_available () else " cpu ")` : يتحقق هذا الخط من توفر وحدة معالجة الرسومات (`torch.cuda.is_available ()`). إذا كانت وحدة معالجة الرسومات متاحة ، فإنها تقوم بتعيين متغير "الجهاز" على "cuda" لاستخدام وحدة معالجة الرسومات للحساب. وإلا ، فإنه يضبط متغير "الجهاز" على "" وحدة المعالجة المركزية "" لاستخدام وحدة المعالجة المركزية.

2. `model = model.to (device)` : هذا الخط ينقل النموذج ("model") إلى الجهاز المحدد ("device"). إذا كان الجهاز عبارة عن GPU ، فإنه يمكن النموذج من استخدام GPU لإجراء العمليات الحسابية. إذا كان الجهاز عبارة عن وحدة معالجة مركزية ، فإنه ببساطة يحتفظ بالنموذج على وحدة المعالجة المركزية.

من خلال ضبط الجهاز ونقل النموذج وفقاً لذلك ، يمكنك التأكد من إجراء العمليات الحسابية على الأجهزة المتاحة (وحدة معالجة الرسومات أو وحدة المعالجة المركزية) لتحقيق الأداء الأمثل واستخدام الموارد.

• الكود البرمجي.

```
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

l)

• شرح الكود.

1. يحدد مقتطف الكود المقدم سلسلة من التحويلات التي سيتم تطبيقها على كل صورة. فيما يلي شرح لكل تحول:
 1. ``Transforms.Resize(256)``: يؤدي هذا التحويل إلى تغيير حجم الصورة بحيث يكون الجانب الأقصر بطول 256 مع الحفاظ على نسبة العرض إلى الارتفاع. سيتم تعديل الجانب الأطول وفقاً لذلك.
 2. ``Transforms.CenterCrop(224)``: هذا التحول يقطع المنطقة المركزية من الصورة، ويبقيها مربعة بطول ضلع يبلغ 224 بكسل.
 3. ``Transforms.ToTensor()``: يحول هذا التحويل الصورة من كائن PIL Image إلى موتور PyTorch. موتورات PyTorch هي بنية البيانات الأولية المستخدمة للحسابات في PyTorch.
 4. تحويلات. تُستخدم قيم الاتحراف المتوسط والمعياري بشكل شائع للنماذج المدرية مسبقاً مثل ResNet. من خلال تطبيق هذه التحويلات ، ستم معالجة الصور المدخلة مسبقاً وتحويلها إلى موتورات مناسبة للإدخال في نموذج ResNet-18. تساعد خطوة المعالجة المسبقة هذه في مواهمة الصور مع متطلبات النموذج وتحسين أداء النموذج.

• الكود البرمجي.

```
train_dataset = datasets.ImageFolder("/content/dataset/train", transform=transform)
val_dataset = datasets.ImageFolder("/content/dataset/validation", transform=transform)
# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
```

• شرح الكود.

- يقوم مقتطف الكود المحدد بإعداد البيانات للتدريب والتحقق من الصحة من خلال إنشاء برامج تحميل البيانات. فيما يلي تفصيل للكود:

1. `train_dataset = datasets.ImageFolder (" / content / dataset / train ")` : يُشَرِّعُ هذَا السُّطْرَ مُثِيلًا لفَتَةً مُجَمَّوِعَةِ الْبَيَانَاتِ `ImageFolder` مِنَ الْوَحْدَةِ النَّمَطِيَّةِ `مُجَمَّوِعَاتِ الْبَيَانَاتِ` . يَأْخُذُ الْمَسَارُ إِلَى دَلِيلِ مُجَمَّوِعَةِ بَيَانَاتِ التَّدْرِيْبِ (` / content / dataset / train `) وَيَطْبِقُ "التحويل" المحدد عَلَى كُلِّ صُورَةٍ فِي مُجَمَّوِعَةِ الْبَيَانَاتِ.

2. `val_dataset = datasets.ImageFolder (" / content / dataset / validation ")` : يُشَرِّعُ هذَا السُّطْرَ مُثِيلًا لفَتَةً مُجَمَّوِعَةِ بَيَانَاتِ `ImageFolder` لِمُجَمَّوِعَةِ بَيَانَاتِ التَّحْقِيقِ . يَأْخُذُ الْمَسَارُ إِلَى دَلِيلِ مُجَمَّوِعَةِ بَيَانَاتِ التَّحْقِيقِ (` / content / dataset / validation `) وَيَطْبِقُ نَفْسَ "التحويل" مُثِيلًا لِمُجَمَّوِعَةِ بَيَانَاتِ التَّدْرِيْبِ .

3. "حجم_الدفعة = 64": يُعَنِّفُ هذَا السُّطْرَ حِجْمَ الدَّفْعَةِ ، وَالَّذِي يَحْدُدُ عَدْدَ الْعَيْنَاتِ الَّتِي تَتَمَّعِلُ بَعْدَهَا فِي كُلِّ تَكَارُ أَثْنَاءِ التَّدْرِيْبِ .

`batch_size = train_loader = torch.utils.data.DataLoader (train_dataset" .4`: ينشئ هذا السطر أداة تحميل بيانات لمجموعة بيانات التدريب. يأخذ `"shuffle = True)` ، `batch_size` "train_dataset" ويغادر "train_dataset" لتحميل البيانات ومعالجتها على دفعات. تقوم الوسيطة `shuffle = True` بترتيب مجموعة بيانات التدريب قبل كل فترة لإدخال العشوائية أثناء التدريب.`

`batch_size = val_loader = torch.utils.data.DataLoader (val_dataset" .5`: ينشئ هذا السطر محمل بيانات لمجموعة بيانات التحقق. يأخذ `"shuffle = False)` ، `batch_size` "val_dataset" ويغادر "val_dataset" عدم تبديل مجموعة بيانات التتحقق عشوائياً ، مما يسمح بالتقدير المتعدد.

باستخدام الطريقة `DataLoader` ، يتم تحميلمجموعات بيانات التدريب والتحقق من الصحة بكفاءة على دفعات ، وهو أمر بالغ الأهمية لتدريب نماذج التعلم العميق. تتعامل برامج تحميل البيانات مع تحميل البيانات وخلطها وتجميعها، مما يسهل التكرار علىمجموعات البيانات أثناء التدريب والتقدير.

• الكود البرمجي.

```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• شرح الكود.

يحدد مقتطف الكود المحدد وظيفة الخسارة والمحسن لتدريب النموذج. فيما يلي شرح لكل سطر:

1. `criterion = nn.CrossEntropyLoss()`: ينشئ هذا السطر مثيلاً للفئة `CrossEntropyLoss` من الوحدة النمطية `nn`. يستخدم فقدان الانتروبيا بشكل شائع في مهام التصنيف متعددة الفئات. فهو يجمع بين وظيفة تنشيط `softmax` وخسارة احتمالية السجل السلبي، مما يجعله مناسباً للنماذج التي تنتج احتمالات فئة الإخراج.

2. `optimizer = optim.Adam (model.parameters(), lr = 0.001)`: ينشئ هذا السطر مثيلاً `optimizer = optim.Adam (model.parameters()` من الوحدة النمطية `optim`. يستخدم المحسن لتحديث معلمات النموذج أثناء التدريب. يأخذ وسيطتين: معلمات النموذج `(model.parameters())` ومعدل التعلم `(lr)`. يحدد معدل التعلم حجم الخطوة في كل خطوة من خطوات التحسين.

من خلال تحديد وظيفة الخسارة والمحسن، فإنك تحدد كيفية تقييم أداء النموذج وكيف سيتم تحديث معلماته أثناء التدريب. يتم استخدام `CrossEntropyLoss` لقياس أداء النموذج، ويتم استخدام `Adam` لتحسين معلمات النموذج.

• الكود البرمجي.

```
# Training loop
num_epochs = 10
```

```

train_losses = []
val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    train_losses.append(epoch_loss)
    model.eval() # Set the model to evaluation mode
    correct_predictions = 0
    total_predictions = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images = images.to(device)
            labels = labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total_predictions += labels.size(0)
            correct_predictions += (predicted == labels).sum().item()
    val_accuracy = correct_predictions / total_predictions
    val_accuracies.append(val_accuracy)
    val_loss = criterion(outputs, labels)
    val_losses.append(val_loss.item())
print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}")

```

شرح الكود البرمجي.

يمثل مقتطف الكود المقدم حلقة تدريب لنموذج التعلم العميق باستخدام بنية ResNet-18. تتكرر الحلقة على عدد محدد من الحقب وتؤدي خطوات التدريب والتحقق من الصحة في كل حقبة.

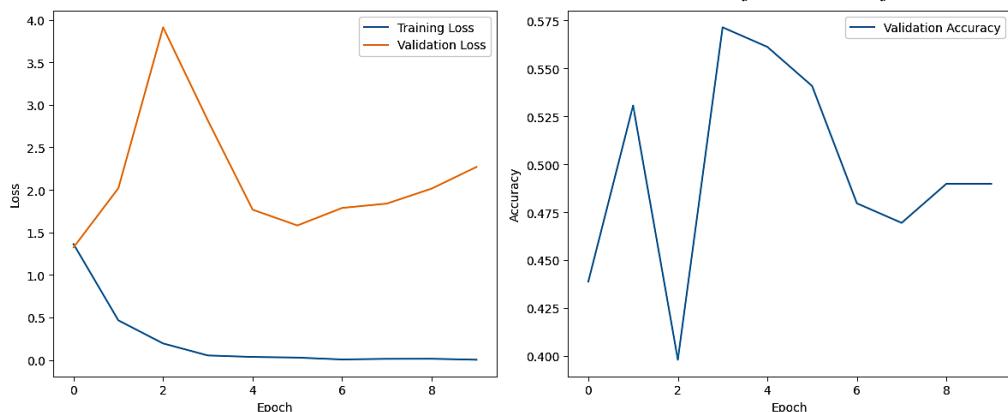
أثناء التدريب، تتكرر الحلقة من خلال مجموعة بيانات التدريب على دفعات. لكل دفعة، يتم نقل صور الإدخال والتسميات إلى الجهاز المناسب (وحدة معالجة الرسومات أو وحدة المعالجة المركزية). يتم مسح التدرجات اللونية للمحسن، ويتم إجراء تمريرة للأمام من خلال النموذج. تتم مقارنة تنبؤات المخرجات مع تسميات الحقيقة الأرضية باستخدام وظيفة فقدان الانتروبيا المتقطعة، ويتم حساب التدرجات واستخدامها لتحديث معلمات النموذج. يتم تجميع خسارة التشغيل لمراقبة تقدم التدريب.

بعد كل فتره تدريب، يتم تحويل النموذج إلى وضع التقييم. ثم تتكرر الحلقة من خلال مجموعة بيانات التحقق لتقدير أداء النموذج على البيانات غير المرئية. يتم نقل صور الإدخال والتسميات إلى الجهاز، ويتم الحصول على تنبؤات النموذج. يتم تتبع العدد الإجمالي للتنبؤات وعد التنبؤات الصحيحة لحساب دقة التتحقق من الصحة. يتم حساب خسارة التتحقق أيضاً باستخدام دالة خسارة الانتروبيا المتقاطعة.

تخرج حلقة التدريب رقم الفترة، وخشارة التدريب، ودقة التتحقق من الصحة لكل حقبة. خسارة التدريب هي متوسط الخسارة على مجموعة بيانات التدريب بأكملها، بينما تمثل دقة التتحقق النسبة المئوية للتسميات المتوقعة بشكل صحيح في مجموعة بيانات التتحقق من الصحة.

تساعد حلقة التدريب هذه في تدريب النموذج ومراقبة أدائه على مدى حقب متعددة. إنه يمكن النموذج من التعلم من مجموعة بيانات التدريب ويوفر نظرة ثاقبة لقدرته على التعليم من خلال دقة التتحقق من الصحة والخسارة.

وكم نرى في المخطط التالي:-



صورة رقم(1)

حيث نرى بان دقة بيانات التتحقق مع الحقبات غير منتظمة بامكانك اعادة تدريب البيانات ضمن عدة حقبات لغرض الحصول على اعلى دقة وتمت مناقشة الطرق والتقنيات لتحسين الدقة سابقاً.

- اختبار النموذج النهائي على بيانات الاختبار.

```
test_dataset = datasets.ImageFolder("/content/dataset/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,
                                         shuffle=False)
# Save the trained model
torch.save(model.state_dict(), "model_eyes_ocular.pth")
```

- **شرح الكود.**

يقوم مقتطف الكود الإضافي بإعداد مجموعة بيانات الاختبار وتحميل البيانات لتقدير النموذج المدرب. يقوم بإنشاء مثيل لفئة مجموعة البيانات "ImageFolder" لمجموعة بيانات الاختبار وإعداد حمل بيانات. يتم حفظ قاموس الحالة الخاص بالنموذج المدرب في ملف لاستخدامه في المستقبل. يسمح ذلك بتقدير أداء النموذج على البيانات غير المرئية ويتيح إعادة تحميل النموذج دون إعادة التدريب للنشر أو مزيد من التدريب.

- **ال kod البرمجي.**

```
# Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
```

```
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("model_eyes_ocular.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()
```

• شرح الكود.

يقوم مقتطف الكود المقدم بتحميل النموذج المحفوظ لل الاستدلال. يقوم بإنشاء مثيل لنموذج ResNet-18 ، ويعدل الطبقة الأخيرة لمطابقة عدد الفئات في مجموعة البيانات ، ويحمل معلمات النموذج المحفوظة. ثم يتم نقل النموذج الذي تم تحميله إلى الجهاز المحدد (CPU أو GPU) لل استدلال وضبطه على وضع التقييم. هذا يسمح باستخدام النموذج المدرب لعمل تنبؤات بشأن البيانات الجديدة.

• الكود البرمجي.

```
# Perform inference on test images and display the results
```

```
class_names = train_dataset.classes
```

```
with torch.no_grad():
```

```
    for images, labels in test_loader:
```

```
        images = images.to(device)
```

```
        outputs = loaded_model(images)
```

```
        _, predicted = torch.max(outputs.data, 1)
```

```
        for i in range(images.size(0)):
```

```
            image = images[i].permute(1, 2, 0).cpu().numpy()
```

```
            label = predicted[i].item()
```

```
            class_name = class_names[label]
```

```
            plt.imshow(image)
```

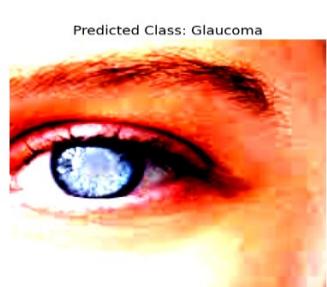
```
            plt.title(f"Predicted Class: {class_name}")
```

```
            plt.axis("off")
```

```
            plt.show()
```

• شرح الكود البرمجي.

ينفذ مقتطف الكود المقدم استناداً على صور الاختبار باستخدام النموذج المحمّل ويعرض النتائج. يتكرر من خلال مجموعة بيانات الاختبار، ويمرر الصور عبر النموذج للحصول على تنبؤات ، ويصور كل صورة جنباً إلى جنب مع فنتها المتوقعة باستخدام matplotlib. يتم الحصول على أسماء الفئات من مجموعة بيانات التدريب. يسمح ذلك بتقييم أداء النموذج على البيانات غير المرئية وفحص تنبؤاته بصرياً.



صورة رقم(2)

تطبيقات الويب.

بعد حصولنا على النموذج النهائي حان الاومن ان نستخدمه في تطبيقات الويب لغرض تجربته والاستدلال عن الكشف لبعض صور الاختبار الغير مدربة مسبقاً لغرض الاستفادة من قبل المستخدمين كتطبيق نهائى للاستدلال والكشف عن امراض العيون .

- الكود البرمجي.

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import gradio as gr
from PIL import Image
# Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
num_classes = 5 # Replace with the actual number of classes in your dataset
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("/content/model_eyes_ocular.pth"))
loaded_model = loaded_model.eval()
```

- شرح الكود.

يدمج مقتطف الكود المحدث النموذج المحمول مع مكتبة Gradio لإنشاء خدمة ويب لتصنيف الصور. يسمح Gradio بإنشاء تطبيق ويب للمستخدمين حيث نعمل بتحميل الصور لمعرفة التنبؤات من النموذج من خلال واجهة سهلة الاستخدام. يقوم الكود بتحميل النموذج المحفوظ، وضبطه على وضع التقييم، ويببدأ تعريف خدمة ويب Gradio. ومع ذلك، فإن التنفيذ الكامل لخدمة الويب، بما في ذلك واجهات الإدخال والإخراج ،

- الكود البرمجي.

```
# Define function for performing inference on the input image
def classify_image(img):
    img = Image.fromarray(img)
    img_tensor = transform(img).unsqueeze(0)
    outputs = loaded_model(img_tensor)
    _, predicted = torch.max(outputs.data, 1)
    label = predicted.item()
    class_names = ['Bulging_Eyes', 'Cataracts', 'Crossed_Eyes', 'Glaucoma', 'Uveitis']
    class_name = class_names[label]
    return class_name
```

- شرح الكود البرمجي.

يحدد مقتطف الكود المقدم وظيفة تسمى "classify_image" تأخذ صورة كمدخلات وتتلقى استنتاجاً باستخدام النموذج المحمول. يقوم بتحويل الصورة إلى موتر ، ويمررها عبر النموذج ، ويسترجع التسمية المتوقعة ، ويضعها في اسم فئة ، ويعيد اسم الفئة كنتيجة. تسمح هذه الوظيفة بتصنيف صورة الإدخال بسهولة باستخدام النموذج المحمول.

• الكود البرمجي.

```
# Create a Gradio interface
image_input = gr.inputs.Image()
label_output = gr.outputs.Textbox()
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output)
iface.launch(debug=True)
```

• شرح الكود.

يتضمن مقتطف الكود المقدم واجهة Gradio لوظيفة "classify_image". يحدد إدخال الصورة وإخراج مربع النص. تسمح الواجهة للمستخدمين بتحميل صورة وتلقي تسمية الفتة المتوقعة كإخراج. من خلال تشغيل الواجهة، يمكن للمستخدمين التفاعل مع نموذج تصنيف الصور بطريقة سهلة الاستخدام.

• الكود النهائي.

بإمكانك تجربة الكود النهائي من هنا

من رمز الاستجابة السريع



الملخص

في الختام، يوفر نموذج ResNet18 الذي تم تنفيذه باستخدام API PyTorch حلًا فعالًا لتصنيف أمراض العيون . بفضل بنيتها العميقه ووصلات التخطي، يمكن لـ ResNet18 أن تتعلم بشكل فعال الأنماط والميزات المعقّدة من صور العين، مما يؤدي إلى تصنّيف دقيق للأمراض.

من خلال نشر نموذج ResNet18 كخدمة ويب باستخدام حزمة Gradio، يصبح سهل الوصول إليه وسهل الاستخدام. يتيح Gradio للمستخدمين التفاعل مع النموذج من خلالواجهة ويب، مما يجعله مناسباً لمختصي الرعاية الصحية أو الأفراد لتحميل صور للعين والحصول على نتائج تصنّيف فورية للأمراض. يتّبّع الجمع بين ResNet18 و PyTorch و Gradio للمهنيين الطبيين أداة فعالة لتشخيص أمراض العين، مما قد يساعد في الكشف المبكر والعلاج. يعزّز نشر خدمة الويب من إمكانية الوصول إلى النموذج، مما يسمح له بالوصول إلى جمهور أوسع وربما المساعدة في سيناريوهات الطب عن بعد أو عن بعد. بشكل عام، فإن نموذج ResNet18 المزود بواجهة برمجة تطبيقات PyTorch ، عند نشره كخدمة ويب باستخدام حزمة Gradio ، يقوم حلًا قيّماً لتصنيف أمراض العيون ، مما يفيد كل من المهنيين الطبيين والأفراد الذين يسعون إلى تشخيص دقيق في الوقت المناسب.

الفصل الخامس الكشف عن مرض الالتهاب الرئوي

المقدمة.

التعلم العميق والكشف عن المرض الالتهاب الرئوي.

بيانات التدريب.

نموذج الشبكة العصبية ResNet18.

نتائج التدريب.

أستدلال النموذج على بيانات الاختبار.

عمل تطبيق الويب.

الكود البرمجي النهائي.

الملخص.

المقدمة

تُستخدم صور الصدر بالأشعة السينية المتعلقة بالالتهاب الرئوي على نطاق واسع في المستشفيات ومرافق الأبحاث في جميع أنحاء العالم للمساعدة في تشخيص أمراض الجهاز التنفسى وعلاجهما. الالتهاب الرئوي هو حالة شائعة وخطيرة تتميز بالتهاب في الرئتين، غالباً ما ينبع عن عدوى بكتيرية أو فيروسية. في المستشفيات، يتم إجراء صور الصدر بالأشعة السينية بشكل روتيني كأداة تشخيصية للكشف عن الالتهاب الرئوي وتقييمه. توفر هذه الصور رؤى قيمة عن حالة الرئتين، مما يسمح لأخصائي الرعاية الصحية بتقييم وجود الالتهاب الرئوي ومدى انتشاره. يقوم أخصائيو الأشعة وأخصائي أمراض الرئة بتحليل صور الأشعة السينية لتحديد السمات المميزة مثل ارتشاح الرئة أو الاندماج أو تراكم السوائل، والتي تشير إلى الالتهاب الرئوي. تلعب مراكز الأبحاث دوراً مهماً في تعزيز فهم الالتهاب الرئوي وعلاجه. يجرون الدراسات والتجارب السريرية للتحقيق في جوانب مختلفة من المرض، بما في ذلك أسبابه وعوامل الخطر وطرق التشخيص وخيارات العلاج. تعد صور الصدر بالأشعة السينية مفيدة في هذه المساعي البحثية، حيث تساعد الباحثين على تحليل الأنماط وتحديدها كمياً، وتطوير خوارزميات أو نماذج جديدة للتشخيص الآلي، وتحسين إدارة الالتهاب الرئوي بشكل عام.

التعلم العميق والكشف عن المرض الالتهاب الرئوي.

لقد ثبت أن تنفيذ نموذج ResNet18 لتصنيف صور الصدر بالأشعة السينية المتعلقة بالالتهاب الرئوي باستخدام تقنيات التعلم العميق فعال للغاية. تسمح بنية ResNet18، المعروفة بوصلاتها المتبقية، بتدريب شبكات أعمق دون مشكلة تلاشي التدرجات. يتبع ذلك للنموذج التقاط الميزات والأنماط المعقّدة في صور الأشعة السينية، مما يؤدي إلى تصنيف دقيق للالتهاب الرئوي.

من خلال الاستفادة من إطار التعلم العميق مثل PyTorch، يمكن تدريب نموذج ResNet18 على مجموعة بيانات كبيرة لصور الصدر بالأشعة السينية المشروحة بملصقات الالتهاب الرئوي. من خلال عملية التدريب التكراري، يتعلم النموذج التفريقي بين الرئتين السليمة والمصابة بالالتهاب الرئوي، ويكتسب القدرة على عمل تنبؤات دقيقة بشأن الصور غير المرئية.

بمجرد تدريب نموذج ResNet18 وتحقيق أداء مرض، يمكن نشره لإنشاء حل عملي لتصنيف الالتهاب الرئوي. من خلال استخدام إطار عمل خدمة ويب مثل Gradio ، يمكن تغليف النموذج بواجهة سهلة الاستخدام ، مما يسمح لأخصائي الرعاية الصحية أو الأفراد بتحميل صور Chest X-Ray والحصول على تنبؤات في الوقت الفعلي حول ما إذا كان الالتهاب الرئوي موجوداً أم لا.

يوفر الجمع بين نموذج ResNet18 وتقنيات التعلم العميق ونشر خدمة الويب أداة قوية لتشخيص الالتهاب الرئوي. يمكن أن يساعد المهنيين الطبيين في اتخاذ القرارات في الوقت المناسب، مما يقلل من الوقت والجهد اللازمين للتفسير اليدوي لأنشطة الصدر السينية. بالإضافة إلى ذلك، قد يساعد في تحسين دقة التشخيص، لا سيما في الحالات التي قد لا يتتوفر فيها خبراء الأشعة بسهولة.

بختصار، فإن نموذج ResNet18، عند تدريبه على صور الصدر بالأشعة السينية لتصنيف الالتهاب الرئوي باستخدام تقنيات التعلم العميق، يوفر حلاً قوياً. عند نشرها كخدمة ويب مع إطار عمل مثل Gradio، تصبح أداة سهلة الوصول وقيمة لمختصي الرعاية الصحية، مما يحتمل أن يعزز كفاءة ودقة تشخيص الالتهاب الرئوي بصورة أسرع.

بيانات التدريب.

بإمكاننا الحصول على البيانات باستخدام الكود

!kaggle datasets download -d falahgatea/chest-x-ray-images-pneumonia

وهي صور عددها اكثراً من 5000 صورة

```

import os
def count_images(folder_path):
    image_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp'] # Add more extensions if
needed
    count = 0
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            _, ext = os.path.splitext(file)
            if ext.lower() in image_extensions:
                count += 1
    return count
# Provide the path to the folder you want to count images in
folder_path = '/content/chest_xray'
num_images = count_images(folder_path)
print(f"Total number of images: {num_images}")

```

بإمكانك عرض عينة من الصور قبل التدريب

```

import os
import random
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
def plot_images(images, labels, num_rows, num_cols):
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 10))
    for i, ax in enumerate(axes.flat):
        ax.imshow(images[i])
        ax.axis('off')
        ax.set_title(labels[i])
    plt.tight_layout()
    plt.show()
def select_random_images(folder_path, label1, label2, num_images):
    label1_images = []
    label2_images = []
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            image_path = os.path.join(root, file)
            if label1 in image_path:

```

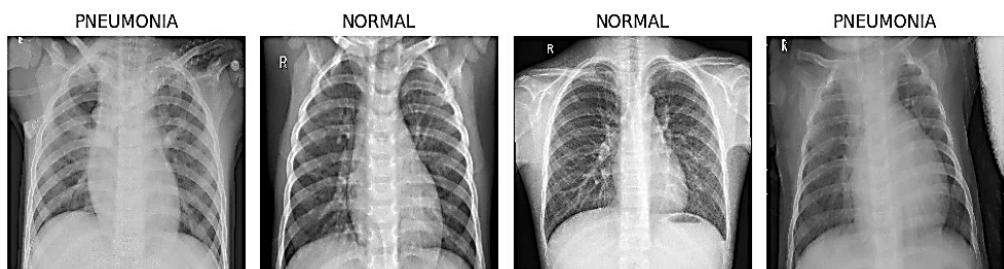
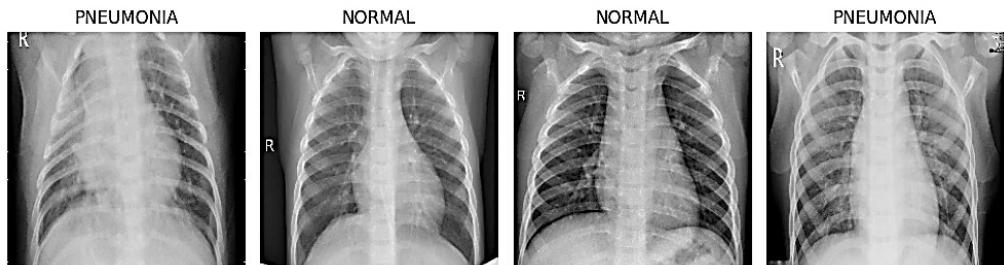
```

label1_images.append(image_path)
elif label2 in image_path:
    label2_images.append(image_path)
selected_images = random.sample(label1_images, num_images // 2) +
random.sample(label2_images, num_images // 2)
random.shuffle(selected_images)
images = []
image_labels = []
for path in selected_images:
    img = load_img(path, target_size=(224, 224)) # Adjust the target_size as per your
requirements
    img = np.array(img)
    images.append(img)

if label1 in path:
    image_labels.append(label1)
elif label2 in path:
    image_labels.append(label2)
return images, image_labels

# Define the path to the folder containing the images
folder_path = '/content/chest_xray/test'
# Define the two class labels
label1 = 'NORMAL'
label2 = 'PNEUMONIA'
# Define the number of images to select from each class
num_images_per_class = 4
# Select random images from the two classes
selected_images, image_labels = select_random_images(folder_path, label1, label2,
num_images_per_class * 2)
# Define the desired number of rows and columns in the grid plot
num_rows = 2
num_cols = num_images_per_class
# Display the grid plot of random images with labels
plot_images(selected_images, image_labels, num_rows, num_cols)

```



صورة رقم(1)

• استدعاء المكتبات المهمة واطر التعلم العميق

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import matplotlib.pyplot as plt
```

• شرح الكود

يستورد مقتطف الكود الوحدات الضرورية من مكتبات PyTorch و torchvision لبناء الشبكات العصبية وتدريبها على مهام رؤية الكمبيوتر. يتضمن وحدات لتعريف معماريات الشبكة العصبية (torch.nn) ، وخوارزميات التحسين (torch.optim) ، وتحويلات الصور (torchvision.transforms) ، ومجموعات البيانات المحددة مسبقاً (torchvision.datasets) ، التمادج المدربة (torchvision.models) ، والتصور (matplotlib.pyplot). بالإضافة إلى ذلك ، يتم تحميل نموذج ResNet-18 .torchvision

نموذج الشبكة العصبية .ResNet18

```
# Load pre-trained transformer model
model = models.resnet18(pretrained=True)
```

يقوم سطر الكود `model = Models.resnet18 (pretrained = True)` بتحميل نموذج ResNet-18 المدرب مسبقاً من وحدة نماذج `torchvision` ويخصمه للمتغير `Model`. ترجع الدالة `resnet18` بأوزان مُدرَّبة مسبقاً. من خلال تعين "تم اختباره مسبقاً" = "True" ، يتم تحميل النموذج بأوزان تم تدريبيها مسبقاً على مجموعة بيانات `ImageNet`. يتيح لك ذلك استخدام نموذج ResNet-18 المدرب مسبقاً للقيام بمهام مثل تصنيف الصور أو استخراج الميزات دون الحاجة إلى تدريبيها من البداية.

- **الكود البرمجي.**

```
# Modify the last layer to match the number of classes in your dataset
num_classes = 2 # Replace with the actual number of classes in your dataset
model.fc = nn.Linear(model.fc.in_features, num_classes)
```

يعدل الكود الطبقة الأخيرة من نموذج ResNet-18 ("النموذج") لمطابقة عدد الفئات في مجموعة البيانات الخاصة بك. يمثل المتغير "num_classes" عدد فئات الإخراج في مجموعة البيانات الخاصة بك. من خلال تعين "model.fc" = "num_classes" (`model.fc = nn.Linear (model.fc.in_features, num_classes)`) ، يتم استبدال آخر طبقة متصلة بالكامل بطبقة خطية جديدة تحتوي على العدد المناسب من ميزات الإخراج ، مهمتها التصنيف المحددة الخاصة بك. يسمح هذا الضبط للنموذج بانتاج تنبؤات لعدد الفئات المطلوب.

- **الكود البرمجي.**

```
# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

- **شرح الكود.**

يضبط الكود الجهاز للحساب إما على وحدة معالجة الرسومات أو وحدة المعالجة المركزية بناءً على التوفير. يتحقق مما إذا كانت وحدة معالجة الرسومات ممتلكة ويعين متغير الجهاز (device) إلى "cuda" إذا كان صحيحاً ، مما يشير إلى أنه سيتم إجراء العمليات الحسابية على وحدة معالجة الرسومات. إذا كانت وحدة معالجة الرسومات غير متوفرة ، فإنها تقوم بتعيين "الجهاز" إلى "وحدة المعالجة المركزية" للحساب على وحدة المعالجة المركزية. ثم يتم نقل النموذج إلى الجهاز المحدد باستخدام `model.to (device)` ، مما يتيح إجراء الحسابات على الجهاز المختار (CPU أو GPU). يتيح لك ذلك الاستفادة من تسريع وحدة معالجة الرسومات (GPU) إذا كان متاحاً ، أو الرجوع إلى تنفيذ وحدة المعالجة المركزية بخلاف ذلك.

• الكود البرمجي.

```
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]))
```

• شرح الكود.

يحدد الكود خط أنابيب التحويل ("التحويل") ليتم تطبيقه على كل صورة في مجموعة البيانات. تتضمن التحولات تغيير حجم الصورة إلى 256×256 بكسل ، واقتاصاص المنطقة المركزية إلى 224×224 بكسل ، وتحويل الصورة إلى موتور ، وتطبيع قيم البكسل باستخدام المتوسط المحدد وقيمة الانحراف المعياري. يقوم خط أنابيب التحويل هذا بإعداد الصور للإدخال في نموذج ResNet-18 ، مما يضمن الحصول على الحجم والشكل المناسبين.

• الكود البرمجي.

```
# Load your dataset
train_dataset = datasets.ImageFolder("/content/chest_xray/train",
                                     transform=transform)
val_dataset = datasets.ImageFolder("/content/chest_xray/val",
                                     transform=transform)
```

• شرح الكود.

يقوم الكود بتحميلمجموعات بيانات التدريب والتحقق من الصحة باستخدام فئة `datasets.ImageFolder`. يفترض أنه تم تنظيم الصور في مجلدات خاصة بالفصل ضمن دلائل الجنس المحددة (/content / chest_xray / val و / content / chest_xray / train). يتم تطبيق خط أنابيب التحويل ("التحويل") المحدد مسبقاً على كل صورة في مجموعة البيانات ، مما يضمن تغيير حجمها واقتاصاصها وتحويلها إلى موترات وتوحيدتها. يعمل هذا على تبسيط عملية تحميل مجموعة البيانات ومعالجتها مسبقاً للتدريب والتحقق من الصحة.

• الكود البرمجي.

```
# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
                                           shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
                                         shuffle=False)
```

• شرح الكود.

يحدد الكود برنامج تحميل البيانات لمجموعات بيانات التدريب والتحقق من الصحة. يتم تعين حجم الدفعات على 64 ، وهو ما يحدد عدد العينات التي تمت معالجتها في كل تكرار. يتم إنشاء "train_loader" باستخدام فئة "DataLoader" ، التي تحمل مجموعة بيانات التدريب ("مجموعة بيانات التدريب") على دفعات صغيرة ، مع تبديل البيانات عشوائياً قبل كل فترة. تم إنشاء "val_loader" بشكل مشابه لمجموعة بيانات التحقق ("val_dataset") ، ولكن بدون تبديل البيانات. تجعل برنامج تحميل البيانات من المريح تكرار مجموعة البيانات على دفعات أثناء التدريب والتحقق من الصحة.

• الكود البرمجي.

```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• شرح الكود.

يحدد الكود وظيفة الخسارة والمحسن لتدريب النموذج. يتم تعين وظيفة الخسارة على "CrossEntropyLoss" ، والتي تُستخدم بشكل شائع لمهام التصنيف متعددة الفئات. تم تعين المحسن على Adam ، ومعدل التعلم هو 0.001. تحسب وظيفة الخسارة الخسارة بين تنبؤات النموذج والتسميات المستهدفة ، بينما يقوم المحسن بتحديث معلمات النموذج بناءً على الخسارة المحسوبة ومعدل التعلم.

• الكود البرمجي.

```
# Training loop
num_epochs = 10
train_losses = []
val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    train_losses.append(epoch_loss)
    model.eval() # Set the model to evaluation mode
    correct_predictions = 0
```

```

total_predictions = 0
with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total_predictions += labels.size(0)
        correct_predictions += (predicted == labels).sum().item()
    val_accuracy = correct_predictions / total_predictions
    val_accuracies.append(val_accuracy)
    val_loss = criterion(outputs, labels)
    val_losses.append(val_loss.item())
print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}")

```

• شرح الكود.

الكود عبارة عن حلقة تدريبية تتكرر على عدد محدد من الصور لتدريب نموذج. يحسب الخسارة ويحدث معلمات النموذج باستخدام المُحسّن في كل فترة. بعد كل فترة ، يقوم بتقدير أداء النموذج في مجموعة بيانات التحقق من الصحة ويسجل فقدان التدريب وفقدان التحقق من الصحة ودقة التتحقق من الصحة. تطبع الحلقة أيضاً خسارة التدريب ودقة التتحقق من الصحة لكل حقبة. تسمح حلقة التدريب هذه للنموذج بالتعلم وتحسين أدائه عبر تكرارات متعددة.

• نتائج التدريب.

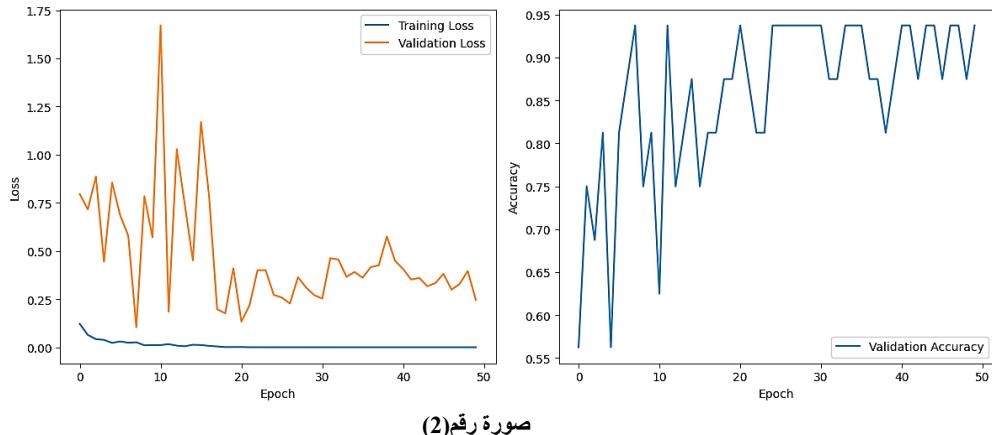
• الكود البرمجي.

```

# Plotting the results
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

```

```
plt.tight_layout()
plt.show()
```



صورة رقم(2)

• شرح الكود البرمجي.

يستخدم الكود **matplotlib** لرسم نتائج التدريب والتحقق من الصحة. يخلق شكلًا مع مجموعتين فرعيتين:

1. توضح المجموعة الفرعية الأولى خسائر التدريب والتحقق من الصحة على مر العصور.

2. يوضح الحركة الفرعية الثانية دقة التحقق من الصحة عبر الحقبات .

يتم رسم خسائر التدريب بعنوان "خسارة التدريب" ، ويتم رسم خسائر التتحقق من الصحة بعنوان "خسارة التتحقق من الصحة". يطلق على المحور السيني اسم "الحقبة" ، ويتم تسمية المحور الصادي باسم "الخسارة". تمت إضافة وسيلة إيضاح إلى هذه المجموعة الفرعية .

وبالمثل ، فإن المجموعة الفرعية الثانية ترسم دقة التتحقق مع تسمية "دقة التتحقق من الصحة". يطلق على المحور السيني اسم "الحقيقة" ، ويتم تصنيف المحور الصادي على أنه "الدقة". تمت إضافة وسيلة إيضاح إلى هذه المجموعة الفرعية أيضًا.

يضبط `plt.tight_layout()` التباعد بين المجموعات الفرعية ، ويعرض `plt.show()` المؤامرة. يسمح لك هذا التصور بمراقبة تقدم التدريب والتحقق من الصحة من خلال فحص اتجاهات الخسارة والدقة على مر الحقبات .

• الكود البرمجي.

```
test_dataset = datasets.ImageFolder("/content/chest_xray/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,
shuffle=False)
```

• شرح الكود البرمجي.

ينشئ الكود مجموعة بيانات اختبار ومحمل بيانات اختبار. يتم استخدامها لتقدير أداء النموذج المدرب على بيانات جديدة غير مرئية.

• الكود البرمجي.

```
# Save the trained model
torch.save(model.state_dict(), "model.pth")
```

• شرح الكود البرمجي.

يحفظ الكود قاموس الحالة للنموذج المدرب في ملف يسمى "model.pth". يتيح لك ذلك تخزين معلمات النموذج وبنيته للاستخدام المستقبلي ، مثل تحميل النموذج وعمل تنبؤات دون إعادة التدريب.

• أستدلال النموذج على بيانات الاختبار.

• الكود البرمجي.

```
# Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("model.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()

# Perform inference on test images and display the results
class_names = train_dataset.classes
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
        _, predicted = torch.max(outputs.data, 1)
        for i in range(images.size(0)):
            image = images[i].permute(1, 2, 0).cpu().numpy()
            label = predicted[i].item()
            class_name = class_names[label]
            plt.imshow(image)
            plt.title(f"Predicted Class: {class_name}")
            plt.axis("off")
            plt.show()
```

• شرح الكود.

يقوم الكود بتحميل نموذج محفوظ ويقوم بالاستدلال على صور الاختبار. يتباين فئة كل صورة ويعرض الصورة مع اسم الفئة المتوقعة. يتم تحميل النموذج بناته ومعلماته، وتتم معالجة صور الاختبار وتغذيتها من خلال النموذج للحصول على تنبؤات.

يتم عرض أسماء الفئات المتوقعة بجانب الصور. يسمح ذلك بتقييم أداء النموذج على البيانات غير المرئية وفحص تنبؤاته بصرياً.

Predicted Class: PNEUMONIA



Predicted Class: NORMAL



صورة رقم (3)

حيث نرى كيف تنبأ النموذج بصور غير مرئية في كلا الحالتين للرئة.

عمل تطبيق الويب.

لتنشيط مكتبة Gradio لبناء تطبيق خدمة ويب لمتعدد التصنيفات نموذج الذكاء الاصطناعي ، يمكنك استخدام الأمر التالي:

```
!pip install gradio
```

سيقوم هذا الأمر بتنشيط مكتبة Gradio ، والتي توفر واجهة بسيطة وبديهية لإنشاء تطبيقات تفاعلية قائمة على الويب باستخدام نماذج التعلم الآلي. باستخدام Gradio ، يمكنك بسهولة إنشاء واجهات ويب لعرض تنبؤات نموذجك والسماح للمستخدمين بالتفاعل مع النموذج في الوقت الفعلي.

• الكود البرمجي.

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import gradio as gr
from PIL import Image
```

```

# Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
num_classes = 2 # Replace with the actual number of classes in your dataset
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("model.pth"))
loaded_model = loaded_model.eval()
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
# Define function for performing inference on the input image
def classify_image(img):
    img = Image.fromarray(img)
    img_tensor = transform(img).unsqueeze(0)
    outputs = loaded_model(img_tensor)
    _, predicted = torch.max(outputs.data, 1)
    label = predicted.item()
    class_names = ['NORMAL', 'PNEUMONIA']
    class_name = class_names[label]
    return class_name
# Create a Gradio interface
image_input = gr.inputs.Image()
label_output = gr.outputs.Textbox()
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output)
iface.launch(debug=True)

```

• شرح الكود.

يقوم الكود بإعداد واجهة Gradio لخدمة ويب متعددة التصنيفات نموذج AI. يقوم بتحميل نموذج محفوظ، ويحدد تحويل الصورة، وينشئ وظيفة لأداء الاستدلال على صورة الإدخال. تأخذ الوظيفة "classify_image" صورة، وتطبق التحويل المحدد، وتتوقع تسمية الفئة باستخدام النموذج المحمول. تقوم بارجاع اسم الفئة المتوقعة.

باستخدام Gradio، يتم إنشاء واجهة تأخذ صورة كمدخلات وتعرض تسمية الفصل المتوقعة كأخرج. ستستخدم وظيفة gr.Interface لتعريف أنواع المدخلات والمخرجات، ويتم تحديد وظيفة classify_image كوظيفة رئيسية للاستدلال.

من خلال تشغيل الواجهة، يمكن للمستخدمين تحميل الصور وتلقي تنبؤات في الوقت الفعلي لتسميات الفصل من النموذج المدرب. توفر الواجهة واجهة سهلة الاستخدام قائمة على الويب للتفاعل مع النموذج.

يبسط **Gradio** عملية إنشاء تطبيقات ويب تفاعلية لنماذج التعلم الآلي والتعلم العميق، مما يسمح للمستخدمين بالوصول إلى توقعات النموذج واستخدامها من خلال واجهة ويب دون الحاجة إلى أي كود أو خبرة تقنية.

الكود البرمجي النهائي.



الملخص

في هذا الفصل تم تدريب نموذج تصنيف التعلم العميق المستند إلى بنية ResNet18 لتصنيف صور الأشعة السينية للصدر على أنها إما "عادية" أو "ذات الرئة". تم تدريب النموذج على مجموعة بيانات مصنفة وحقق أداءً جيداً في التمييز بين الحالات الطبيعية وحالات الالتهاب الرئوي. النموذج المدرب قادر على التنبؤ بدقة بسميات صور الصدر بالأشعة السينية. تم حفظه ويمكن تحميله للاستدلال على صور جديدة غير مرئية. يستخدم النموذج تحويلات الصور والتقطيع لمعالجة الصور المدخلة مسبقاً قبل إجراء التنبؤات.

لجعل النموذج سهل الوصول إليه وسهل الاستخدام، تم تطوير خدمة ويب باستخدام مكتبة Gradio. تتيح خدمة الويب للمستخدمين تحميل صور Chest X-Ray من خلال واجهة ويب وتتلقى تنبؤات في الوقت الفعلي من النموذج المدرب. تعرض الواجهة تسمية الفئة المتوقعة، مما يشير إلى ما إذا كانت الصورة تظهر حالة طبيعية أو التهاب رئوي.

يوفر نموذج ResNet18 مع خدمة الويب طريقة ملائمة للمهنيين الطبيين والمستخدمين لتصنيف صور الأشعة السينية للصدر للكشف عن الالتهاب الرئوي. تلغى خدمة الويب حاجة المستخدمين إلى امتلاك خبرة في الترميز أو الخبرة الفنية، حيث يمكنهم ببساطة تحميل الصور وتلتقي التنبؤات من خلال الواجهة البديهية. بشكل عام، يقوم نموذج تصنيف التعلم العميق وخدمة الويب أداة قيمة للتحليل الفعال والدقيق لصور الصدر بالأشعة السينية لتصنيف الالتهاب الرئوي. لديه القدرة على مساعدة المهنيين الطبيين في تشخيص حالات الالتهاب الرئوي وتحسين كفاءة عمليات الرعاية الصحية.

الفصل السادس

Eyes Ocular Disease Classification PyTorch 4 classes Using MobileNetV2

مقدمة.

نموذج MobileNetV2.

الكود البرمجي.

استدعاء نموذج التدريب.

أستدلال النموذج على بيانات الاختبار.

تطبيق الويب.

الكود البرمجي النهائي.

الملخص.

مقدمة.

يهدف مشروع تصنیف وتعريف أربعة أنواع من امراض العيون إلى تطوير نموذج تعليمي عميق باستخدام بنية MobileNet الإصدار 2 لتصنیف أمراض العین إلى أربع فئات مختلفة. يركز المشروع على الاستفادة من تقنيات الرؤية الحاسوبية للمساعدة في الكشف المبكر عن أمراض العین وتشخیصها، مما يساعد في النهاية في العلاج والوقاية من فقدان البصر في الوقت المناسب.

نماذج MobileNetV2

تصميم النموذج المختار لهذا المشروع هو MobileNet الإصدار 2. MobileNet عبارة عن بنية شبكة عصبية عميقة خفيفة الوزن مصممة خصيصاً للأجهزة المحمولة والمدمجة. إنه يحقق توازناً بين الدقة والكفاءة من خلال استخدام التقنيات قبلة للفصل على العمق، مما يقلل من التعقيد الحسابي مع الحفاظ على دقة معقولة.

يتم تدريب النموذج على مجموعة بيانات كبيرة من صور العین، والتي تحتوي على أمثلة من أربع فئات متميزة من أمراض العین. يمكن أن تشمل هذه الفئات حالات مثل اعتلال الشبكية السكري، والزرق، والتنكس البقعي المرتبط بالعمر (AMD)، وإعتام عدسة العین. تم تصنیف مجموعة البيانات بغاية من قبل خبراء المجال لضمان التصنيف الدقيق أثناء التدريب.

تم اختبار نموذج MobileNet الإصدار 2 مسبقاً على مجموعة بيانات صور واسعة النطاق مثل ImageNet ، والتي توفر تهيئة جيدة لمهمة تصنیف أمراض العین. يتم استخدام التعلم بالنقل، حيث يتم ضبط النموذج قبل التدريب على مجموعة بيانات أمراض العین المحددة لتعلم الميزات الخاصة بالمرض. تتضمن عملية التدريب تحسين النموذج باستخدام وظيفة خسارة مناسبة، مثل الانتروبيا الفنوية، واستخدام محسن مثل أصل التدرج العشوائي (SGD) أو آدم. يتم ضبط الباراميترات الفانقة للنموذج، مثل معدل التعلم وحجم الدفع، لتحقيق الأداء الأمثل.

لتقييم أداء النموذج، يتم استخدام مجموعة بيانات اختبار منفصلة، تحتوي على صور للعين غير مرئية. يتم حساب مقاييس التقييم المختلفة، مثل الدقة والتذكر ودرجة F1 ، لتقييم أداء تصنیف النموذج عبر فئات المرض الأربع.

يوضح النموذج نتائج واحدة في التصنيف الدقيق لصور العین إلى أربع فئات مرضية متميزة. هذه التكنولوجيا لديها القدرة على مساعدة المتخصصين في الرعاية الصحية في الكشف المبكر عن أمراض العین وتشخیصها، وبالتالي تمكين التدخل في الوقت المناسب وتحسين نتائج المرض. مزيد من التحسين والتقييم للنموذج على مجموعات البيانات الأكبر والمتنوعة يمكن أن يعزز أداءه وقابلیته للتعیین في الكشف السريري المبكر في العالم الحقيقي.

تدريب النموذج.

الکود البرمجي.

طبعاً بعد عمل حساب في خدمة كاكل وكما تم شرحه سابقاً. بإمكاننا تحميل البيانات منه وهي من ضمن بيانات حسابي الشخصي والتي تم تجميع أكثر من 40 كباً لجميع بيانات المشاريع الطيبة في الكتاب وطبعاً ان شاء الله مستقبلاً قابلة للتتوسيع.

Colab's file access feature

```

from google.colab import files
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json

```

- تحميل البيانات.

`!kaggle datasets download -d falahgatea/eye-diseases-classification`

فك الضغط عنها

```

!unzip /content/eye-diseases-classification.zip
!rm -r /content/eye-diseases-classification.zip

```

حيث تحتوي على أربع فئات مصنفة كالتالي

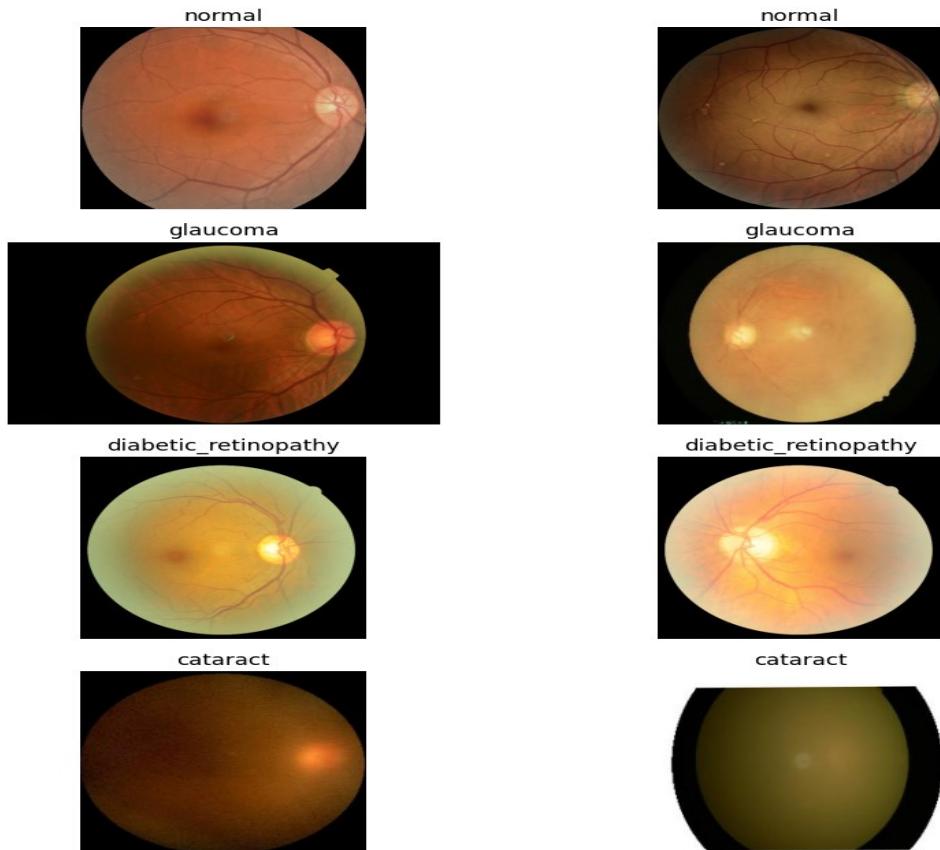
[cataract ,diabetic retinopathy,glaucoma,normal]

إعتمام عدسة العين- اعتلال الشبكية السكري - الجلوكوما - الحالة الطبيعية.
وهي حالات مألوفة ومعروفة لدى طبيب العيون والمرضى. بإمكانك عرض عينة من الحالات قبل بدأ التدريب

```

import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
# Path to the main folder containing subfolders with images
main_folder = "/content/eye_diseases_classification"
# List of subfolders (image classes)
subfolders = os.listdir(main_folder)
# Number of sample images to display from each class
num_samples = 2
# Plotting the sample images
fig, axes = plt.subplots(len(subfolders), num_samples, figsize=(10, 10))
for i, subfolder in enumerate(subfolders):
    subfolder_path = os.path.join(main_folder, subfolder)
    image_files = os.listdir(subfolder_path)[:num_samples]
    for j, image_file in enumerate(image_files):
        image_path = os.path.join(subfolder_path, image_file)
        image = mpimg.imread(image_path)
        axes[i, j].imshow(image)
        axes[i, j].axis("off")
        axes[i, j].set_title(subfolder)
plt.tight_layout()
plt.show()

```



صورة رقم (1)

البدء بتقسيم البيانات الى بيانات التدريب وبيانات التحقق وبيانات الاختبار.

```
import os
import shutil
from sklearn.model_selection import train_test_split
# Specify the path to your dataset
dataset_path = "/content/eye_diseases_classification/"
# Specify the paths for train, test, and validation sets
train_path = "/content/dataset/train/"
test_path = "/content/dataset/test/"
val_path = "/content/dataset/validation/"
# Split the dataset into train, test, and validation sets
class_names = os.listdir(dataset_path)
for class_name in class_names:
    class_path = os.path.join(dataset_path, class_name)
```

```

files = os.listdir(class_path)
if len(files) < 2:
    print(f"Skipping class '{class_name}' due to insufficient samples for train/test split.")
    continue
train_files, test_val_files = train_test_split(files, test_size=0.5, random_state=42)
if len(train_files) < 1:
    print(f"Skipping class '{class_name}' due to insufficient samples for train set.")
    continue
test_files, val_files = train_test_split(test_val_files, test_size=0.5, random_state=42)
# Move the files to their respective folders
for file in train_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(train_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)
for file in test_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(test_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)
for file in val_files:
    src_path = os.path.join(class_path, file)
    dst_path = os.path.join(val_path, class_name, file)
    os.makedirs(os.path.dirname(dst_path), exist_ok=True)
    shutil.copy(src_path, dst_path)

```

يعد تقسيم مجموعة البيانات إلى مجموعات تدريب واختبار والتحقق من الصحة وهي ممارسة شائعة في التعلم العميق لتقدير أداء النموذج وتحسينه. إليك سبب استخدامنا لهذه المجموعات المنفصلة:

- مجموعة التدريب :** تستخدم مجموعة التدريب لتدريب النموذج. يحتوي على جزء كبير من مجموعة البيانات ويستخدم لتحسين معلمات النموذج (الأوزان والتحيزات) من خلال **backpropagation**. يتعلم النموذج من هذه البيانات عن طريق تعديل أوزانه لتقليل وظيفة الخسارة المختارة.
- مجموعة التحقق :** تُستخدم مجموعة التتحقق لضبط الباراميترات الفائقة وتقييم أداء النموذج أثناء التدريب. يساعد في اختيار أفضل بنية نموذجية، وتحسين الباراميترات الفائقة (على سبيل المثال، معدل التعلم، وقوف التنظيم)، ومنع الإفراط في التجهيز. يوجه أداء النموذج في مجموعة التتحقق من صحة التعديلات التي تم إجراؤها على بنية النموذج أو الباراميترات الفائقة.
- مجموعة الاختبار :** تُستخدم مجموعة الاختبار لتقدير الأداء النهائي وقدرة التعميم للنموذج المدرب. هذه المجموعة منفصلة عن مجموعات التدريب والتحقق، مما يضمن تقييم النموذج على بيانات غير مرئية. تقدم مجموعة الاختبار تقديرًا غير متحيز لأداء النموذج، مما يساعد على قياس قدرته على التعميم على الأمثلة الجديدة غير المرئية.

من خلال تقسيم مجموعة البيانات إلى هذه المجموعات المنفصلة، يمكننا تقدير مدى جودة أداء النموذج على البيانات غير المرئية، وكذلك اتخاذ قرارات مستنيرة بشأن ضبط الأوزان الفائقة واختيار النموذج. يساعد في

الفصل السادس: Eyes Ocular Disease Classification

منع فرط التخصيص، حيث يعمل النموذج جيداً على بيانات التدريب ولكنه يفشل في التعميم على أمثلة جديدة. علاوة على ذلك، فإنه يتبع إجراء مقارنات عادلة بين النماذج المختلفة أو إعدادات المعامل الفائق. ملاحظة:-

أحياناً عند تحميل البيانات لغرض التدريب سوف تصادفنا بعض المشاكل وخاصة سوف تكون في مجموعة الصور صور شاذة وتكون ربما غير مرئية المستخدم لكنها تؤثر على سير وخط التدريب للتأكد من ان جميع الصور مستوفية الشروط عليك استخدام الكود البرمجي لغرض التدقيق والفحص وإزالة الصور الشاذة من التدريب لم يتم شرحه في الكتاب لكنه موجود ضمن الكود النهائي لغرض التنوية فقط.

• استدعاء النموذج.

Steps for Training

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import matplotlib.pyplot as plt
# Load pre-trained MobileNet model
model = models.mobilenet_v2(pretrained=True)
```

• شرح الكود البرمجي.

باختصار، ينفذ جزء الكود البرمجي الذي قدمته بالخطوات التالية:-

1. استيراد المكتبات اللازمة.
2. قم بتحميل نموذج MobileNet الإصدار 2 الذي تم تدريبيه مسبقاً.
3. تعين عدد الفئات لتصنيف أمراض العين.
4. قم بتعديل آخر طبقة متصلة بالكامل من النموذج لمطابقة عدد الفئات.
5. تحديد وظيفة الخسارة (CrossEntropyLoss).
6. اختر محسّناً (آدم) لتحديث أوزان النموذج أثناء التدريب.
7. تحديد وحدة المعالج (GPU أو CPU) الذي سيتم استخدامه للتدريب.
8. انقل النموذج إلى وحدة المعالج المحدد.
9. تحميل ومعالجةمجموعات بيانات التدريب والاختبار والتحقق من الصحة.
10. تحديد برنامج تحميل البيانات للتكرار علىمجموعات البيانات أثناء التدريب.
11. أداء حلقة التدريب لعدد محدد من الحقبات (epoch).
12. حساب خسائر التدريب والتحقق من الصحة.
13. تصوّر قيم الخسارة باستخدام matplotlib.pyplot.

ملاحظة:-

لا يتضمن جزء الكود البرمجي بعض الخطوات الإضافية مثل حفظ النموذج المدرب ، وتقدير النموذج في مجموعة الاختبار ، وضبط الباراميترات التشعبية استناداً إلى أداء مجموعة التحقق من الصحة. قد يلزم إضافة هذه الخطوات لإكمال عملية التدريب.

• الكود البرمجي.

```
# Modify the last layer to match the number of classes in your dataset
num_classes = 4 # Replace with the actual number of classes in your dataset
model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)
# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

• شرح الكود.

باختصار ، يقوم جزء الكود البرمجي بتعديل الطبقة الأخيرة من النموذج لمطابقة عدد الفئات في مجموعة البيانات الخاصة بك وتعيين وحدة المعالج للتدريب .1. تعديل الطبقة الأخيرة : يستبدل الكود الطبقة الأخيرة من نموذج MobileNet بطبقة خطية جديدة ، مع ضبط حجم الإدخال لمطابقة حجم الإدخال السابق للطبقة الأخيرة وتعيين حجم الإخراج على العدد المطلوب من الفئات في مجموعة البيانات الخاصة بك.

2. ضبط وحدة المعالج : يتحقق من توفر وحدة معالجة الرسومات ويضبط وحدة المعالج على "cuda" إذا كان كذلك ، وإلا فإنه يضبط وحدة المعالج على "cpu". ثم يتم نقل النموذج إلى وحدة المعالج المحدد لضمان كفاءة الحساب على وحدة معالجة الرسومات أو وحدة المعالجة المركزية .
تتضمن هذه الخطوات تكوين النموذج بشكل صحيح لمهمة التصنيف المحددة الخاصة بك وأن العمليات الحسابية يتم إجراؤها على وحدة المعالج المناسب.

• الكود البرمجي.

```
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]))
```

• شرح الكود.

من خلال تطبيق هذه التحولات، يضمن الكود أن صور الإدخال يتم معالجتها مسبقاً بشكل متson قبل إدخالها في نموذج التعلم العميق. تساعد هذه التحولات في توحيد بيانات الإدخال وتحسين قدرة النموذج على تعلم ميزات مفيدة من الصور.

• الكود البرمجي:

```
# Load your dataset
train_dataset = datasets.ImageFolder("/content/dataset/train", transform=transform)
val_dataset = datasets.ImageFolder("/content/dataset/validation", transform=transform)
# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
```

• شرح الكود.

باختصار، ينفذ جزء الكود البرمجي الخطوات التالية لتحميل مجموعة البيانات وتحديد برنامج تحميل البيانات:
1. تحميل Dataset : يستخدم الكود فئة **ImageFolder** لتحميل مجموعة البيانات. تحدد المسارات إلىمجموعات بيانات التدريب والتحقق من الصحة وتطبق التحويلات المحددة على الصور أثناء التحميل.

2. Define Data Loaders : يقوم الكود بإنشاء برنامج تحميل البيانات باستخدام فئة **torch.utils.data.DataLoader**. يقوم بتعيين حجم الدفعة إلى 64 لكل من بيانات التدريب والتحقق من الصحة. يقوم مُحمل بيانات التدريب بترتيب البيانات عشوائياً قبل كل دفعة ، بينما لا يقوم مُحمل بيانات التحقق من الصحة بتبديل البيانات.

تضمن هذه الخطوات تحميل مجموعة البيانات بالتحولات المرغوبة وأنه تم إعداد برنامج تحميل البيانات للتكرار عبر مجموعة البيانات على دفعات أثناء التدريب والتحقق من الصحة.

• الكود البرمجي.

```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• شرح الكود.

باختصار، يحدد جزء الكود البرمجي المكونات التالية للتدريب:

1. وظيفة الخسارة : يتم استخدام **nn.CrossEntropyLoss()** كدالة خسارة. يحسب الخسارة عبر الانتروبيا بين احتمالات الطبقة المتوقعة وسميات الحقيقة الأرضية.

2. Optimizer : يتم استخدام **optim.Adam (model.parameters(), lr = 0.001)** كمحسن. يطبق خوارزمية آدم الأمثل لتحديث معلمات النموذج أثناء التدريب، بمعدل تعلم 0.001. هذه المكونات ضرورية لتدريب النموذج. تقيس وظيفة الخسارة أداء النموذج، ويقوم المحسن بضبط معلمات النموذج لتقليل الخسارة وتحسين الدقة.

• الكود البرمجي.

```
# Training loop
num_epochs = 10
train_losses = []
val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    train_losses.append(running_loss / len(train_loader))
    if epoch % 5 == 0:
        val_accuracy = evaluate(model, val_loader)
        val_accuracies.append(val_accuracy)
        print(f'Epoch {epoch+1}: Train Loss: {train_losses[-1]:.4f}, Val Accuracy: {val_accuracies[-1]:.4f}
```

```

running_loss += loss.item() * images.size(0)
epoch_loss = running_loss / len(train_dataset)
train_losses.append(epoch_loss)

model.eval() # Set the model to evaluation mode
correct_predictions = 0
total_predictions = 0
with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total_predictions += labels.size(0)
        correct_predictions += (predicted == labels).sum().item()
    val_accuracy = correct_predictions / total_predictions
    val_accuracies.append(val_accuracy)
    val_loss = criterion(outputs, labels)
    val_losses.append(val_loss.item())
print(f'Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}')

```

• شرح الكود البرمجي.

- باختصار، يمثل جزء الكود البرمجي حلقة تدريب بالخطوات الرئيسية التالية:
1. خطوة التدريب : يتم تعين النموذج على وضع التدريب `model.train()` ويتكرر على بيانات التدريب.
 2. خطوة التقييم : يتم تعين النموذج على وضع التقييم `model.eval()` ويقيم أدائه على بيانات التحقق من الصحة. يحسب الدقة والخسارة دون إجراء انتشار عكسي.
 3. نتائج الطباعة : يقوم الكود بطباعة رقم الحقبة وخسارة التدريب ودقة التحقق.
- من خلال تنفيذ هذا الكود، يتم تدريب النموذج على عدد الحقبات (epoch) المحدد ، ويتم مراقبة فقدان التدريب ودقة التحقق من الصحة.

• الكود البرمجي.

```

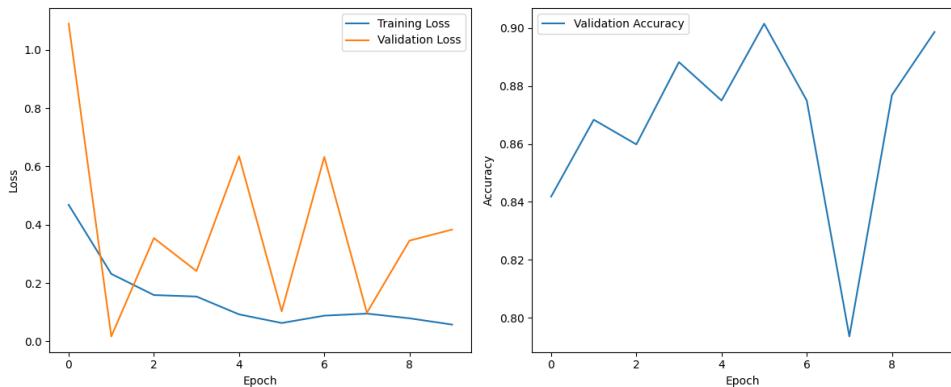
# Plotting the results
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(val_accuracies, label='Validation Accuracy')

```

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.tight_layout()
plt.show()
```

• شرح الكود.

باختصار، يرسم جزء الكود البرمجي نتائج التدريب والتحقق من الصحة باستخدام **Matplotlib**. يقوم بإنشاء رقم مع حدين فرعرين: أحدهما لعرض التدريب وفقدان التحقق من الصحة ، والآخر لإظهار دقة التتحقق من الصحة. توفر المخططات تمثيلاً مرمياً لأداء النموذج على مدار الحقبات (epoch) ، مما يسمح بسهولة تفسير وتحليل تقدم التدريب.



صورة رقم(2)

سوف نرى بعض المشاكل في عدم الدقة في نتائج التدريب والتحقق من الصحة من حيث دقة النموذج ودقة الخسارة الغير منتظمة أثناء تكرار حقبات التدريب من خلال صورة النتائج وذلك قد يكون عدم انتظام فقدان التدريب والدقة أثناء التدريب بسبب عدة عوامل:

1. **Overfitting**: يحدث Overfitting عندما يتعلم النموذج الأداء الجيد على بيانات التدريب لكنه يفشل في التعلم على البيانات غير المرئية. في مثل هذه الحالات، قد ينخفض فقدان التدريب بينما يزداد فقدان التتحقق، مما يشير إلى أن النموذج أصبح متخصصاً جداً في بيانات التدريب. يمكن أن تساعد تقنيات التنظيم مثل الترسّب أو التوقف المبكر في معالجة فرط التجهيز.

2. **معدل التعلم**: يحدد معدل التعلم حجم الخطوة أثناء تحديثات الباراميترات. إذا كان معدل التعلم مرتفعاً جداً، فقد لا يتقارب النموذج بشكل صحيح، مما يؤدي إلى تقلبات في الخسارة والدقة. يمكن أن يساعد تحسين معدل التعلم باستخدام جداول معدل التعلم أو خوارزميات معدل التعلم التكيفي أو الضبط اليدوي في استقرار عملية التدريب.

3. **عدم توازن البيانات**: إذا كانت مجموعة البيانات غير متوازنة، مما يعني أن بعض الفصول تحتوي على عينات أقل بكثير من غيرها، فقد يؤدي ذلك إلى تدريب متحيز. قد يركز النموذج بشكل أكبر على فئات الأغلبية، مما يؤدي إلى دقة تدريب غير منتظمة. يمكن لتقنيات مثل زيادة البيانات أو ترجيح الفصل أو استراتيجيات أخرى العينات أن تعالج عدم توازن البيانات.

• لمعالجة هذه المشكلات، يمكنك التفكير في العلاجات التالية:-

1. **تقنيات التنظيم** : تطبيق تقنيات تنظيم مثل التسرب من السمنة أو تناقص الوزن أو التوقف المبكر لمنع فرط الملاعة وتحسين التعميم.
2. **تحسين معدل التعلم** : جرب معدلات التعلم المختلفة وجداول معدلات التعلم للعثور على القيمة المثلثى لنموذجك. يمكن لتقنيات مثل تسوس معدل التعلم أو خوارزميات معدل التعلم التكيفي (على سبيل المثال، adm) أن تساعد في استقرار عملية التدريب.
3. **زيادة البيانات** : تطبيق تقنيات زيادة البيانات مثل الاقتصاص العشوائى أو التقليل أو التدوير لزيادة تنوع بيانات التدريب بشكل مصطنع والتخفيض من فرط التجهيز.
4. **وزن الفئة** : قم بتعيين أوزان أعلى لفئات الأقيات أثناء عملية التدريب لمعالجة مشكلات عدم توازن البيانات ومنع التحيز تجاه فئات الأغذية.
5. **زيادة بيانات التدريب** : إذا كان ذلك ممكناً، ففكر في زيادة حجم مجموعة بيانات التدريب الخاصة بك لتقييم المزيد من الأمثلة المتنوعة للنموذج للتعلم منها، مما قد يساعد في تحسين التعميم.

• الكود البرمجي:

```
# Save the trained model
torch.save(model.state_dict(), "model.pth")
```

بتنفيذ جزء الكود البرمجي هذا، سيتم حفظ قاموس حالة النموذج المدرب في مسار الملف المحدد. يتيح لك ذلك تحميل النموذج المدرب وإعادة استخدامه في وقت لاحق للاستدلال أو الضبط الدقيق أو النشر.

 أستدلال النموذج على بيانات الاختبار.

• الكود البرمجي:

```
# Load the saved model for inference
loaded_model = models.mobilenet_v2(pretrained=False)
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features,
num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()
# Load your test dataset
test_dataset = datasets.ImageFolder("/content/dataset/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1, shuffle=True)
```

• شرح الكود.

- باختصار، ينفذ جزء الكود البرمجي الخطوات التالية لتحميل النموذج المحفوظ وتنفيذ الاستدلال على مجموعة بيانات اختبار:
- تحميل النموذج :** يُحمل الكود نموذجًا محفوظًا عن طريق إنشاء مثيل جديد من بنية النموذج المطلوبة ("النموذج_المحمول") وتعديل الطبقة الأخيرة لمطابقة عدد الفئات في مجموعة البيانات. يتم تحميل قاموس حالة النموذج من الملف المحفوظ باستخدام `torch.load()`.
 - ربط وحدة المعالج ووضع التقييم :** يتم نقل "النموذج_المحمول" إلى وحدة المعالج المحدد وضبطه على وضع التقييم. هذا يضمن أن النموذج يستخدم موارد الأجهزة المتاحة على النحو الأمثل ويتصرف بشكل صحيح أثناء الاستدلال.
 - تحميل مجموعة بيانات الاختبار :** يقوم الكود بتحميل مجموعة بيانات الاختبار باستخدام "مجموعات البيانات. مجلد الصور" ويطبق التحولات المرغوبة على الصور.
 - تحديد أداة تحميل بيانات الاختبار :** يتم تطبيق مجموعة بيانات الاختبار في محمل بيانات باستخدام `torch.utils.data.DataLoader` تبديل البيانات عشوائياً إذا رغبت في ذلك. بتتنفيذ جزء الكود البرمجي هذا، يتم تحميل النموذج المحفوظ، ويمكنك استخدام النموذج المحمول لعمل تنبؤات على مجموعة بيانات الاختبار. يسهل محمل البيانات التكرار عبر مجموعة بيانات الاختبار، مما يتيح تقييم وتحليل أداء النموذج على البيانات غير المرئية.

• الكود البرمجي.

```
# Select 5 images for testing
test_images = []
test_labels = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
        _, predicted = torch.max(outputs.data, 1)
        for i in range(images.size(0)):
            image = images[i].permute(1, 2, 0).cpu().numpy()
            label = predicted[i].item()
            class_name = test_dataset.classes[label]
            test_images.append(image)
            test_labels.append(class_name)
            if len(test_images) >= 5:
                break
    if len(test_images) >= 5:
        break
```

• شرح الكود.

- باختصار، ينفذ جزء الكود البرمجي الخطوات التالية لتحديد 5 صور من مجموعة بيانات الاختبار للاختبار:

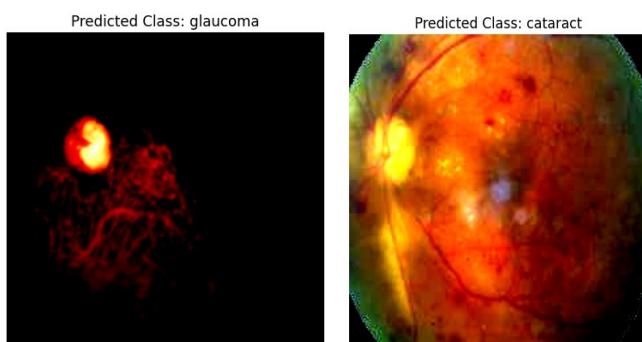
الفصل السادس: Eyes Ocular Disease Classification

1. **حدد اختبار الصور**: يتكرر الرمز عبر أداة تحميل بيانات الاختبار ويقوم بالاستدلال على كل دفعه من الصور باستخدام النموذج الذي تم تحميله. يسترجع التسميات المتوقعة لكل صورة.
 2. **تخزين الصور التجريبية**: لكل صورة، يقوم بتحويل المولت إلى مصفوفة NumPy ويخزنها في قائمة "صور_الاختبار". كما يقوم أيضًا باسترداد التسمية المتوقعة باسم الفئة المطابق، المخزنين في قائمة "تصنيفات_الاختبار".
 3. **الحد من 5 صور**: يتحقق الكود مما إذا كان عدد صور الاختبار التي تم جمعها قد وصل إلى 5. إذا كان الأمر كذلك، فإنه ينفصل عن الحلقة لإيقاف جمع المزيد من الصور. من خلال تنفيذ جزء الكود البرمجي هذا، يمكنك الحصول على 5 صور اختبار مع تسمياتها المتوقعة. يمكن تحليل هذه الصور أو تصوّرها أو استخدامها لأي أغراض اختبار مطلوبة.
- **الكود البرمجي.**

```
# Display the selected test images
for i in range(len(test_images)):
    image = test_images[i]
    label = test_labels[i]
    plt.imshow(image)
    plt.title(f"Predicted Class: {label}")
    plt.axis("off")
    plt.show()
```

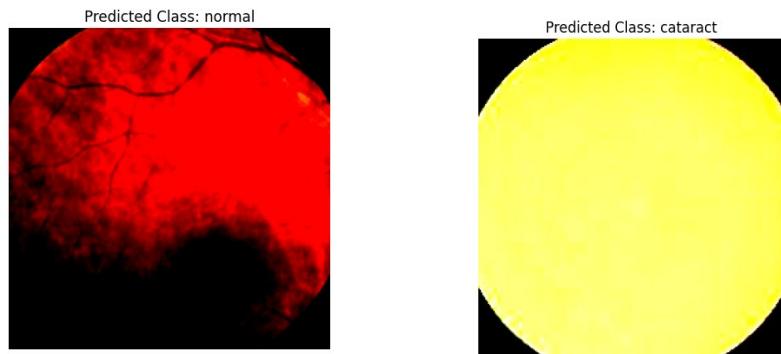
• شرح الكود البرمجي.

- باختصار، ينفذ جزء الكود البرمجي الخطوات التالية لعرض صور الاختبار المحددة مع تسمياتها المتوقعة:
1. **عرض صور الاختبار**: الكود يتكرر على صور الاختبار المخزنة في قائمة "صور_الاختبار" والتسميات المتوقعة المقابلة لها في قائمة "علامات_الاختبار".
 2. **التصور**: لكل صورة اختبار، يتم استرداد الصورة والملصق. يستخدم plt.imshow() لعرض الصورة و plt.title() لتعيين العنوان كتسمية للفئة المتوقعة. يتم استخدام محور plt.axis ("off") لإزالة تسميات المحور وعلامات التجزئة.
 3. **إظهار الصورة**: أخيراً، يتم استدعاء plt.show() لعرض الصورة مع تسمية الفئة المتوقعة. من خلال تنفيذ جزء الكود البرمجي هذا ، يمكنك تصوّر صور الاختبار المحددة ومراقبة تسمياتها المتوقعة. يمكن أن يساعدك هذا في التحقق من أداء النموذج الذي تم تحميله في مجموعة بيانات الاختبار واكتساب رؤى حول دقتها.



صورة رقم(3)

او مشاهدة الصورة الحقيقة قبل التنبؤ وبعد التنبؤ لمعرفة دقة النموذج



صورة رقم(4)

• تطبيق الويب.

قمنا بدمج النموذج المدرب كخدمة ويب باستخدام حزمة **Gradio**. يمكن للمستخدمين تحميل الصور من خلال واجهة الويب، ويقوم النموذج بإجراء استدلال لتوفير تنبؤات لتصنيف أمراض العين. يجعل واجهة **Gradio** التطبيق سهل الاستخدام ويمكن الوصول إليه، مما يسمح للأفراد بالتفاعل مع النموذج دون الحاجة إلى خبرة في البرمجة.

• الكود البرمجي.

```
!pip install gradio
```

```
# Load the saved model for inference
loaded_model = models.mobilenet_v2(pretrained=False)
num_classes = 4 # Replace with the actual number of classes in your dataset
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features,
num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
loaded_model = loaded_model.eval()
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
# Define function for performing inference on the input image
def classify_image(img):
    img = Image.fromarray(img)
```

```

img_tensor = transform(img).unsqueeze(0)
outputs = loaded_model(img_tensor)
_, predicted = torch.max(outputs.data, 1)
label = predicted.item()
class_names = ['normal', 'diabetic_retinopathy', 'glaucoma', 'cataract']
class_name = class_names[label]
return class_name

# Create a Gradio interface
image_input = gr.inputs.Image()
label_output = gr.outputs.Textbox()
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output)
iface.launch(debug=True)

```

ولا اريد اعادة كيفية تشغيل واجهة المستخدم التفاعلية مع مكتبة (Gradio). لأن تم شرحه سابقا.

باتباع هذه الخطوات، تتمكن من إنشاء خدمة ويب حيث يمكن للمستخدمين التفاعل مع النموذج المدرب. يمكن للمستخدمين تحميل صورهم من خلال الواجهة، وسيقوم النموذج بإجراء الاستدلال وت تقديم فحنة المرض المتوقعة كمخرجات. هذا يجعل الأمر مناسباً وسهل الاستخدام للأفراد للاستفادة من قدرات النموذج دون الحاجة إلى أي معرفة برمجية.

ملاحظة:-

عند العمل هذا التطبيق في الحاسبة الشخصية يرجى التأكد من المسارات والأدلة وموقع المجلدات وхран النموذج النهائي. وأيضاً الغاء علامة التعبّر من الامر (pip) لتنصيب اي مكتبة بلغة بايثون.

الكود البرمجي النهائي.

من رمز الاستجابة السريع



الملخص

في هذا المشروع، استخدمنا نموذج MobileNetV2 لتصنيف الصور وقمنا بتدريبه على مجموعة بيانات تتكون من أربع فئات من أمراض العين. ثم اتباع الخطوات التالية:-

1. إعداد النموذج:

- تحميل نموذج MobileNetV2 المدرب مسبقاً من [torchvision](#).
- تم تعديل الطبقة الأخيرة من النموذج لمطابقة عدد الفئات في مجموعة البيانات.

2. إعداد مجموعة البيانات:

- تحويلات الصور المحددة بما في ذلك تغيير الحجم، والاقتصاص المركزي ، والتحويل إلى موتر ، والتطبيع.

▪ تم تحميلمجموعات بيانات التدريب والتحقق من الصحة باستخدام فئة `ImageFolder` من [torchvision](#)

▪ تم إنشاء برامج تحميل البيانات من أجل معالجة البيانات بكفاءة أثناء التدريب والتحقق من الصحة.

3. التدريب:

- تم تعريف وظيفة الخسارة على أنها `CrossEntropyLoss` والمحسن مثل `Adam`.
- تم تنفيذ حلقة التدريب على عدد محدد من الحقب.

▪ درب النموذج من خلال التكرار علىمجموعات بيانات التدريب، وحساب الخسارة ، وتحديث أوزان النموذج.

▪ تقييم النموذج في مجموعة بيانات التحقق لتبني تقدم التدريب.

4. التقييم:

▪ تحميل النموذج المدرب المحفوظ للاستدلال.

▪ تحميل مجموعة بيانات الاختبار باستخدام فئة `ImageFolder` وإنشاء أداة تحميل بيانات للاختبار.

▪ اختيار مجموعة فرعية من صور الاختبار للتقييم.

▪ تم إجراء الاستدلال على صور الاختبار باستخدام النموذج المحمّل والحصول على الملصقات المتوقعة.

5. التصور:

▪ رسم صور الاختبار المحددة وتسليط الضوء الأساسية الخاصة بها والتسميات المتوقعة لها.

6. تكامل خدمة الويب:

▪ تم تثبيت حزمة `Gradio` لإنشاء واجهة ويب.

▪ إنشاء وظيفة لأداء الاستدلال على إدخال الصور باستخدام النموذج المحمّل.

▪ تم تطوير واجهة `Gradio` تقبل الصور التي تم تحميلها من قبل المستخدم وتعرض فئة المرض المتوقع.

ينتهي المشروع بخدمة ويب مفيدة تتيح للمستخدمين تحميل الصور وتلقي تنبؤات لتصنيف أمراض العين. حقق النموذج المدرب أداءً مرضياً في مجموعة بيانات الاختبار، مما يدل على فعاليته في تصنيف أمراض العين.

الفصل السابع

تصنيف مرض سرطان الجلد

Skin Cancer Malignant vs. Benign

المقدمة.

ـ نقل التعلم ونموذج MobilenetV2

ـ تدريب النموذج.

ـ الاستدلال النهائي للنموذج.

ـ خدمة تطبيق الويب.

ـ الكود النهائي.

ـ الملخص.

المقدمة

يعتبر سرطان الجلد مصدر قلق صحي كبير في جميع أنحاء العالم، مع زيادة معدلات الإصابة به في السنوات الأخيرة. يعد الاكتشاف المبكر والتصنيف الدقيق للآفات الجلدية أمراً ضرورياً للتشخيص والعلاج الفعال. أظهرت تقنيات التعلم العميق، مثل الشبكات العصبية التاليفية العميقة (CNN)، نتائج واعدة في التصنيف الآلي لسرطان الجلد. في هذا التطبيق المميز، نستفيد من نموذج MobileNet الذي تم تدريبه مسبقاً، وهو بنية CNN شائعة الاستخدام ، للتمييز بين الأورام الجلدية الخبيثة والحميدة.



صورة رقم(1)

نقل التعلم ونموذج MobileNetV2

هي بنية CNN خفيفة الوزن مصممة للتطبيقات المحمولة والمدمجة. يوفر حلّاً فعالاً ودقيقاً لمهام تصنيف الصور. من خلال الاستفادة من تقنية نقل التعلم ، يمكننا الاستفادة من المعرفة المكتسبة من تدريب نموذج MobileNet على مجموعة بيانات واسعة النطاق لتصنيف اورام سرطان الجلد بدقة. هدفنا في هذا التطبيق هو تطوير نظام تصنیف قائم على التعلم العميق يمكن أن يساعد أطباء الأمراض الجلدية في التمييز بين الأورام الجلدية الخبيثة والحميدة. من خلال استخدام نموذج MobileNet المدرب مسبقاً، نهدف إلى تحقيق دقة عالية وحساسية وخصوصية في تحديد الأورام الجلدية السرطانية.

إن تطبيق تصنيف التعلم العميق باستخدام نموذج MobileNet مدرب مسبقاً يحمل إمكانات كبيرة في الكشف الدقيق والتمييز بين الأورام الجلدية الخبيثة والحميدة. من خلال استخدام نقل التعلم، يمكننا الاستفادة من الميزات المكتسبة من مجموعة بيانات واسعة النطاق، مما يمكن نموذجنا من التعلم بشكل جيد على الحالات الجديدة. يمكن أن يساعد تطوير نظام التصنیف هذا بشكل كبير أطباء الأمراض الجلدية في عملية التشخيص، مما يحتمل أن يحسن الدقة الكلية وكفاءة تشخيص سرطان الجلد. من خلال توفير أداة آلية للتقدير الأولي، يمكن أن يساعد هذا النهج أيضاً في فرز المرضى، وتحديد أولويات الحالات العاجلة، وضمان التدخل في الوقت المناسب لأولئك المعرضين للخطر.

ومع ذلك، من المهم ملاحظة أنه على الرغم من أن أنظمة التصنیف القائمة على التعلم العميق تبشر بالخير ، فلا ينبغي اعتبارها بديلاً للحكم الطبي الخبر. يجب أن يشارك أطباء الأمراض الجلدية دائمًا في التشخيص النهائي

الفصل السابع: Skin Cancer: Malignant vs. Benign

وقدرات العلاج. علاوة على ذلك، فإن البحث والتطوير المستمر ضروريان لتحسين دقة وتعزيز نماذج التعلم العميق لتصنيف سرطان الجلد.

فإن الجمع بين تقنيات التعلم العميق، مثل استخدام النماذج المدربة مسبقاً مثل MobileNet، مع الخبرة الطبية، لديه القدرة على التأثير بشكل كبير في مجال الأمراض الجلدية، مما يؤدي إلى تحسين النتائج لمرضى سرطان الجلد.

تدريب النموذج.

• مجموعة البيانات.

سرطان الجلد: الأورام الخبيثة مقابل الأورام الحميدة (Malignant vs. Benign) تحتوي مجموعة البيانات هذه على مجموعة بيانات متوازنة لصور الشامات الجلدية الحميدة والشامات الجلدية الخبيثة. تتكون البيانات من مجلدين مع كل 1800 صورة (224×244) لنوعين من الشامات.

• الكود البرمجي.

```
# Colab's file access feature
from google.colab import files
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d falahgatea/skin-cancer
```

• شرح الكود.

تحميل بيانات سرطان الجلد من موقع Kaggle بعد عمل حساب والحصول على مفتاح التخويل وعدها أكثر من 6000 صورة مصنفة بين سرطان حميد وخبيث في ثلاثة مجلدات بيانات التدريب والتحقق والاختبار لتدريب ضمن نموذج

MobileNetV2.

بإمكانك عرض عينة من الصور قبل التدريب



صورة رقم(1)

• الكود البرمجي.

```
# Load pre-trained MobileNet model
```

```
model = models.mobilenet_v2(pretrained=True)
```

```
# Modify the last layer to match the number of classes in your dataset
```

```
num_classes = 2 # Replace with the actual number of classes in your dataset
```

```
model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)
```

```
# Set the device (GPU if available, else CPU)
```

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

```
model = model.to(device)
```

```
# Define the transformation applied to each image
```

```
transform = transforms.Compose([
```

```
    transforms.Resize(256),
```

```
    transforms.CenterCrop(224),
```

```
    transforms.ToTensor(),
```

```
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]))])
```

• شرح الكود.

يوضح الكود البرمجي المقدم عملية تحميل نموذج MobileNet V2 مدرب مسبقاً، وتعديل الطبقة الأخيرة لمطابقة عدد الفئات في مجموعة البيانات المخصصة، وإعداد الجهاز (CPU أو GPU)، وتحديد تحويلات الصورة. يعتبر نموذج MobileNet V2 هي بنية شبكة عصبية تلافيفية تم تدريبيها مسبقاً على مجموعة بيانات ImageNet ، والتي تحتوي على عدد كبير من الصور المصنفة من فئات مختلفة. من خلال تحميل هذا النموذج المدرب مسبقاً، يمكننا الاستفادة من المعرفة التي اكتسبها أثناء تدريبيه على كمية هائلة من الصور المتنوعة. يعدل الكود الطبقة الأخيرة من النموذج لتكيفها مع مجموعة البيانات المحددة بعدد مختلف من الفئات. هذه الخطوة ضرورية لأن نموذج MobileNet V2 الأصلي تم تدريبيه للتنبؤ ب 1000 فئة مختلفة، ونحن بحاجة إلى تعديله لمطابقة عدد الفئات في مجموعة البيانات الخاصة بنا.

يتحقق الرمز أيضاً مما إذا كانت وحدة معالجة الرسومات متاحة ويعين الجهاز وفقاً لذلك. يمكن أن يؤدي استخدام وحدة معالجة الرسومات إلى تسريع العمليات الحسابية المتضمنة في عمليات التدريب والاستدلال بشكل كبير. علاوة على ذلك، يحدد الكود سلسلة من تحويلات الصور باستخدام الوحدة النمطية "torchvision.transforms". تضمن هذه التحويلات أن صور الإدخال يتم معالجتها مسبقاً بشكل متson قبل إدخالها في النموذج. تتضمن التحويلات تغيير حجم الصور إلى حجم معين، واقتراض المركز، والتحويل إلى موترات، وتسوية قيم البكسل بناءً على قيم الانحراف المعياري والمتوسط المحسوب مسبقاً.

بشكل عام، يقوم الكود المقدم بإعداد نموذج MobileNet V2 مدرب مسبقاً للضبط الدقيق على مجموعة بيانات مخصصة. بعد الانتهاء من خطوات الإعداد هذه، يمكن للمرء المضي قدماً في تدريب النموذج على مجموعة البيانات، والذي يتضمن تغذية الصور المحولة في النموذج، وفقدان الحوسبة، وتحديث أوزان النموذج لتقليل الخسارة من خلال خوارزميات التوسيع العكسي والتحسين مثل نزول التدرج العشوائي (SGD).

• الكود البرمجي.

```
train_dataset = datasets.ImageFolder("/content/skin-cancer/data", transform=transform)
```

```
# Define the percentage of data for training, validation, and test sets
```

```
train_split = 0.8 # 80% for training
```

```
val_split = 0.1 # 10% for validation
```

```

test_split = 0.1 # 10% for testing
# Calculate the sizes of each split
train_size = int(train_split * len(train_dataset))
val_size = int(val_split * len(train_dataset))
test_size = len(train_dataset) - train_size - val_size
# Split the dataset
train_dataset, val_dataset, test_dataset = random_split(train_dataset, [train_size, val_size,
test_size])

```

• شرح الكود.

ستحتوي مجموعة بيانات التدريب على 80٪ من مجموعة بيانات `train_dataset` الأصلية ، وستحتوي مجموعة بيانات التحقق من الصحة على 10٪ ، وستحتوي مجموعة بيانات الاختبار على نسبة 10٪ المتبقية. تتضمن عملية التقسيم هذه تقسيم البيانات إلى مجموعات منفصلة للتدريب والتحقق من الصحة والاختبار ، وهو ما يتم عادةً في التعلم الآلي لتقييم أداء النموذج ومنع التجاوز الزائد.

Train set size: 2637

Validation set size: 329

Test set size: 331

• الكود البرمجي.

```
# Define the data loaders
```

```
batch_size = 64
```

```
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
```

```
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
```

• شرح الكود.

ينشئ الكود البرمجي المقدم برنامج تحميل بيانات لمجموعات بيانات التدريب والتحقق من الصحة. تُعد برامج تحميل البيانات في PyTorch فنادق أدوات مساعدة تتيح تحميل البيانات بكفاءة وسهولة أثناء عمليات التدريب والتحقق من الصحة.

تأخذ فئة "DataLoader" كائن مجموعة البيانات ("train_dataset" أو "val_dataset") والمعلمات الإضافية مثل "batch_size" و "shuffle" لتصنيص سلوك تحميل البيانات.

- "حجم دفعه": يحدد عدد العينات لكل دفعه. يحدد عدد العينات التي ستتم معالجتها معاً في كل تكرار أثناء التدريب أو التتحقق من الصحة.

- `shuffle = True` : يشير إلى وجوب تبديل البيانات عشوائياً لكل فترة أثناء التدريب. يساعد الخلط في إدخال العشوائية ويعنِّي التأكيد من عدم ترتيب البيانات، مما قد يكون مفيداً لعمليات التعلم أفضل.

- `shuffle = False` : تحديد عدم تبديل البيانات عشوائياً أثناء التتحقق من الصحة. يضمن الاحتفاظ ببيانات التحقق بترتيب ثابت تقييماً ومقارنة متسقين للنتائج.

توفر برنامج تحميل البيانات عنصراً متكرراً يمكن استخدامه في حلقات التدريب أو التتحقق من الصحة. إنهم يتعاملون تلقائياً مع عملية جلب البيانات وتجميعها، مما يسمح بمعالجة وتدريب فعالين لنماذج التعلم الآلي.

• الكود البرمجي.

```
# Define the loss function and optimizer
```

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• شرح الكود.

من خلال تحديد وظيفة الخسارة والمحسن، تكون قد حددت كيفية تقييم أداء النموذج وكيف سيتم تحديث معلمات النموذج أثناء عملية التدريب. يحسب **CrossEntropyLoss** الخسارة بناءً على احتمالات الفئة المتوقعة للنموذج واسميات الحقيقة الأساسية. يطبق **Adam** نزولاً متدرجاً لتحسين معلمات النموذج عن طريق تقليل دالة الخسارة. يحدد معدل التعلم حجم الخطوة التي يضبط بها المحسن معلمات النموذج أثناء النشر العكسي. يمكن أن يؤدي معدل التعلم الصغير إلى تقارب بطيء، في حين أن معدل التعلم الكبير يمكن أن يتسبب في تجاوز الحد وعدم الاستقرار.

• الكود البرمجي.

```
# Training loop
num_epochs = 10
train_losses = []
val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    train_losses.append(epoch_loss)
    model.eval() # Set the model to evaluation mode
    correct_predictions = 0
    total_predictions = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images = images.to(device)
            labels = labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total_predictions += labels.size(0)
            correct_predictions += (predicted == labels).sum().item()
    val_accuracy = correct_predictions / total_predictions
```

```

val_accuracies.append(val_accuracy)
val_loss = criterion(outputs, labels)
val_losses.append(val_loss.item())
print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation
Accuracy: {val_accuracy:.4f}")

```

• شرح الكود.

يوضح الكود البرمجي المقدم حلقة تدريب كاملة لتدريب نموذج. إنه يتكرر على عدد محدد من الفترات، ويقوم بإجراء تمريرات للأمام والخلف لكل دفعه من بيانات التدريب. يتم حساب خسارة التدريب وتخزينها لكل حقبة. بالإضافة إلى ذلك، يتم تقييم النموذج على مجموعة التحقق من الصحة ، ويتم حساب وتتبع فقدان التحقق من الصحة والدقة. يعرض الكود استخدام نموذج التدريب والتقييم ومراقبة المقاييس مثل الفقد والدقة.

أثناء التدريب سوف تظهر عملية المعالجة وسير عملية التدريب

```

Epoch 1/10, Training Loss: 0.5680, Validation Accuracy: 0.7872
Epoch 2/10, Training Loss: 0.5230, Validation Accuracy: 0.7933
Epoch 3/10, Training Loss: 0.5181, Validation Accuracy: 0.7933
Epoch 4/10, Training Loss: 0.5113, Validation Accuracy: 0.7933
Epoch 5/10, Training Loss: 0.4965, Validation Accuracy: 0.7933
Epoch 6/10, Training Loss: 0.4953, Validation Accuracy: 0.7933
Epoch 7/10, Training Loss: 0.4788, Validation Accuracy: 0.7842
Epoch 8/10, Training Loss: 0.4720, Validation Accuracy: 0.7933
Epoch 9/10, Training Loss: 0.4648, Validation Accuracy: 0.7751
Epoch 10/10, Training Loss: 0.4187, Validation Accuracy: 0.7781

```

تشير هذه التفاصيل إلى تقدم النموذج أثناء التدريب، مع انخفاض خسارة التدريب تدريجياً وتقليل دقة التحقق. تسمح مراقبة هذه القيم بتقييم أداء النموذج والتركيب المحتمل أو عدم الملائمة.

• عرض ومعاينة النتائج.

```

# Plotting the results
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.tight_layout()
plt.show()

```

• شرح الكود.

من خلال التخطيط لخسائر التدريب والتحقق من الصحة بالإضافة إلى دقة التحقق، يمكنك تحليل تقدم تدريب النموذج بصرياً وملحوظة كيف تتغير هذه المقاييس على مر العصور. يساعد هذا التصور في تقييم أداء النموذج، وتحديد أي مشاكل تتعلق بالملاءمة أو النقص، واتخاذ قرارات مستنيرة لمزيد من التحسينات في النموذج.



صورة رقم(2)

• كود البرمجي.

```
# Save the trained model
```

```
torch.save(model.state_dict(), "model_skin-cancer.pth")
```

• شرح الكود.

خزن النموذج النهائي لغرض الاستدلال والتطبيق النهائي على بيانات اختبارية غير مدربة مسبقاً.

• الكود البرمجي.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
# Initialize variables for storing the predictions and true labels
all_predictions = []
all_labels = []
# Set the model to evaluation mode
model.eval()
# Disable gradient calculation for efficiency
with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        # Append the predicted labels and true labels to the respective lists
        all_predictions.extend(predicted.tolist())
        all_labels.extend(labels.tolist())
# Plot the confusion matrix
confusion_matrix(all_labels, all_predictions)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_matrix(all_labels, all_predictions), annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

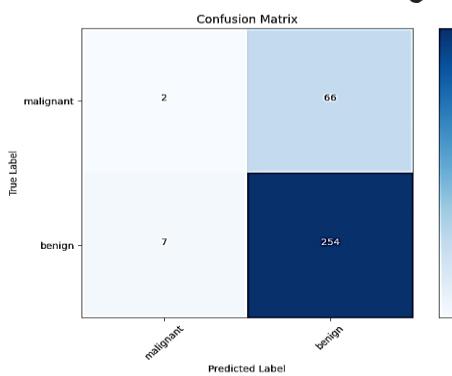
```

all_labels.extend(labels.tolist())
# Calculate the confusion matrix
cm = confusion_matrix(all_labels, all_predictions)
# Plot the confusion matrix
classes = ['malignant', 'benign'] # Replace with the actual class labels
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
fmt = 'd'
thresh = cm.max() / 2.
for i, j in np.ndindex(cm.shape):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()
# Calculate precision, recall, F1-score, and support for each class
class_report = classification_report(all_labels, all_predictions, target_names=classes)
print("Classification Report:")
print(class_report)

```

• شرح الكود.

يحسب مقتطف الكود المقدم مصفوفة الارتكاك ويطبع تقرير التصنيف لنتائج النموذج في مجموعة التحقق من الصحة. تصور مصفوفة الارتكاك توزيع التسميات الحقيقية والمتواعدة، بينما يوفر تقرير التصنيف مقاييس مثل الدقة والاستدعاء ودرجة F1 والدعم لكل فئة. تساعد هذه التقييمات في تقييم أداء النموذج وفهم قرته على تصنيف الفئات المختلفة بشكل صحيح.



صورة رقم(3)

Classification Report:

	precision	recall	f1-score	support
malignant	0.22	0.03	0.05	68
benign	0.79	0.97	0.87	261
accuracy			0.78	329
macro avg	0.51	0.50	0.46	329
weighted avg	0.68	0.78	0.70	329

الدقة الإجمالية للنموذج في مجموعة التحقق من الصحة هي 0.78. متوسط درجة F1 الكلية، الذي يحسب متوسط درجة F1 عبر جميع الفئات، هو 0.46. متوسط درجة F1 ، الذي يعتبر عدم توازن الفئة، هو 0.70. توفر هذه المقاييس نظرة ثاقبة على أداء النموذج لكل فئة، مما يشير إلى أنه يتمتع بدقة أعلى، واسترجاع، ودرجة F1 للفئة "الحميدة" مقارنة بالفئة "الخبيثة". ظهرت الدقة الكلية ودرجات F1 الأداء العام للنموذج في مجموعة التحقق من الصحة.

بناءً على تقرير التصنيف المقدم، يبدو أن النموذج يعمل بشكل جيد من حيث الدقة للفئة "الحميدة" بدقة 0.79، والتذكر والاسترجاع 0.97، ودرجة F1 من 0.87. ومع ذلك، فإن أداء فئة "الخبيثة" منخفض نسبياً بدقة 0.22، واسترجاع 0.03، ودرجة F1 تبلغ 0.05.

تحسين الدقة للفئة "الخبيثة"، يمكنك مراعاة الخطوات التالية:

- زيادة البيانات :** زيادة تنوع وكمية عينات الفئة "الخبيثة" من خلال تطبيق تحويلات مثل التدوير العشوائي أو التقليبات أو تعديلات السطوع على الصور أثناء التدريب. يمكن أن يساعد هذا النموذج في تعلم المزيد من الميزات القوية لتصنيف الحالات الخبيثة.
 - معالجة عدم توازن الفئة :** إذا كان هناك خلل كبير في الفئة بين الفئتين "الخبيثة" و "الحميدة" ، يمكنك معالجة هذه المشكلة من خلال استخدام تقنيات مثل الإفراط فيأخذ عينة من فئة الأقلية (الخبيثة) أو تقليل العينات من فئة الأغلبية (حميدة). هذا يضمن أن النموذج يتعرض لعدد متساوٍ من العينات من كل فصل أثناء التدريب.
 - تعديلات معمارية النموذج :** ضع في اعتبارك تعديل بنية النموذج عن طريق إضافة المزيد من الطبقات، أو زيادة عدد المرشحات، أو تجربة بنى شبكة مختلفة تماماً. يمكن أن يساعد هذا في التقاط ميزات أكثر تعقيداً خاصة بالفئة "الخبيثة".
 - ضبط Hyperparameter :** جرب معدلات التعلم المختلفة، وخيارات المحسّن، وقيم تناقص الوزن، وأحجام الدفعات للعثور على التركيبة المثلث لتحسين أداء النموذج في فئة "الخبيثة".
 - جمع المزيد من البيانات :** إذا كان ذلك ممكناً، فإن جمع عينات تمثيلية أكثر تنوعاً للفئة "الخبيثة" يمكن أن يساعد في تحسين قدرة النموذج على تعميم وتصنيف مثل هذه الحالات بدقة.
- من خلال تنفيذ هذه الاستراتيجيات، يمكنك أن تهدف إلى تحسين الدقة ومقاييس الأداء الأخرى لفئة "الخبيثة"، وفي النهاية تحقيق نموذج أكثر توازناً ودقة لكلا الفئتين.

الفصل السابع: Skin Cancer: Malignant vs. Benign:
لنفرض تم نجاح التدريب بعد اتخاذ الإجراءات أعلاه الان نقوم باستدعاء النموذج واختباره على صور اختبارية لمعرفة وكشف السرطان الخبيث من السرطان الحميد.

الاستدلال النهائي للنموذج.

- الكود البرمجي.

```
# Load the saved model for inference
loaded_model = models.mobilenet_v2(pretrained=False)
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features, num_classes)
loaded_model.load_state_dict(torch.load("/content/model_skin-cancer.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()

# Load your test dataset
test_dataset = datasets.ImageFolder("/content/skin-cancer/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1, shuffle=True)

# Select 5 images for testing
test_images = []
test_true_labels = []
test_predicted_labels = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
        _, predicted = torch.max(outputs.data, 1)
        for i in range(images.size(0)):
            image = images[i].permute(1, 2, 0).cpu().numpy()
            true_label = test_dataset.classes[labels[i].item()]
            predicted_label = test_dataset.classes[predicted[i].item()]

            test_images.append(image)
            test_true_labels.append(true_label)
            test_predicted_labels.append(predicted_label)
        if len(test_images) >= 5:
            break
    if len(test_images) >= 5:
        break

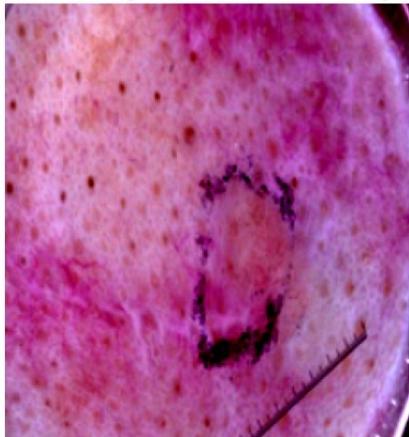
# Display the selected test images with true and predicted class labels
for i in range(len(test_images)):
    image = test_images[i]
    true_label = test_true_labels[i]
    predicted_label = test_predicted_labels[i]
    plt.imshow(image)
    plt.title(f"True Class: {true_label}, Predicted Class: {predicted_label}")
```

```
plt.axis("off")
plt.show()
```

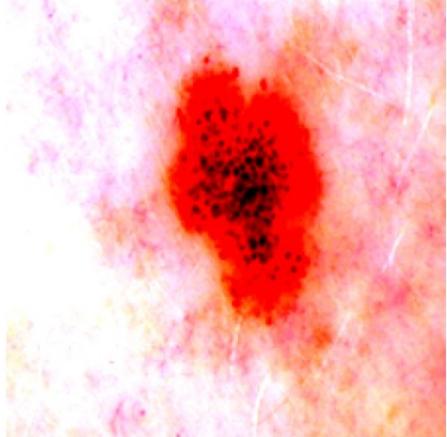
• شرح الكود البرمجي.

إليك الكود الذي قدمته، والذي يقوم بتحميل النموذج المحفوظ ، ويحمل مجموعة بيانات الاختبار ، ويختار 5 صور للاختبار ، ويعرض كل صورة اختبار جنبا إلى جنب مع تسمية الفئة الحقيقة وتسمية الفئة المتوقعة.

True Class: malignant, Predicted Class: malignant



True Class: benign, Predicted Class: malignant



صورة رقم(4)

خدمة تطبيق الويب.

يمكن إنشاء خدمة ويب للكشف عن سرطان الجلد باستخدام حزمة Gradio للاستدلال على النموذج النهائي. تتيح خدمة الويب هذه للمستخدمين تحميل صور بشرتهم ، والتي تتم معالجتها بعد ذلك بواسطة النموذج المدرب للتنبؤ باحتمالية الإصابة بسرطان الجلد. يتم عرض النتائج على المستخدم وتزويده بمعلومات قيمة عن وجود أو عدم وجود سرطان الجلد.

• الكود البرمجي.

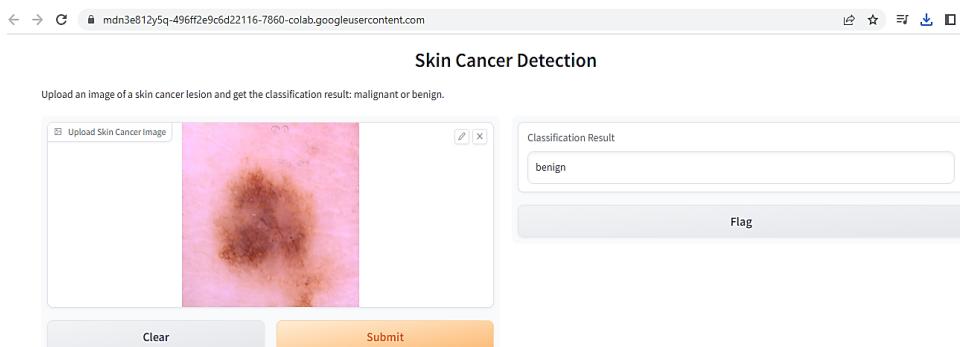
```
!pip install gradio
import torch
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import gradio as gr
from PIL import Image
# Load the saved model for inference
loaded_model = models.mobilenet_v2(pretrained=False)
num_classes = 2 # Replace with the actual number of classes in your dataset
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features, num_classes)
loaded_model.load_state_dict(torch.load("/content/model_skin-cancer.pth"))
```

```

loaded_model = loaded_model.eval()
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]))
# Define function for performing inference on the input image
def classify_image(img):
    img = Image.fromarray(img)
    img_tensor = transform(img).unsqueeze(0)
    outputs = loaded_model(img_tensor)
    _, predicted = torch.max(outputs.data, 1)
    label = predicted.item()
    class_names = ['malignant', 'benign']
    class_name = class_names[label]
    return class_name
# Create a Gradio interface
image_input = gr.inputs.Image(label="Upload Skin Cancer Image")
label_output = gr.outputs.Textbox(label="Classification Result")
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output,
title="Skin Cancer Detection", description="Upload an image of a skin cancer lesion and get the classification result: malignant or benign.")
iface.launch(debug=True)

```

لتظهر لنا واجه الويب بامكانك رفع أي صورة للتحقق من نوع الورم



صورة رقم(5)

الكود النهائي.

بإمكانك تنزيل الكود البرمجي من رمز الاستجابة السريع



الملخص

في الخاتم، يوفر نموذج **MobileNet V2** المدرب على تصنيف سرطان الجلد حلاً موثوقًا وفعالاً للكشف عن الآفات الجلدية الخبيثة والحميدة والتمييز بينها. من خلال الاستفادة من تقنيات التعلم العميق، يحقق النموذج دقة عالية في التنبؤ بتصنيفات صور سرطان الجلد. تسمح هندسته خفيفة الوزن بالاستدلال السريع، مما يجعله مناسباً للتطبيقات في الوقت الفعلي. يمكن نشر النموذج كخدمة ويب، مما يوفر واجهة سهلة الاستخدام للمستخدمين لتحميل صور سرطان الجلد والحصول على نتائج تصنيف فورية. بشكل عام، يوفر نموذج **MobileNet V2** أداة قيمة للمساعدة في الكشف المبكر عن سرطان الجلد وتشخيصه، مما يساهم في تحسين نتائج المرضى.

الفصل الثامن

تصنيف 8 فئات لسرطان القولون والمستقيم

المقدمة.

ما هي مجموعة البيانات؟

نموذج الشبكة العصبية .MobileNetV2

استدلال النموذج على صور الاختبار.

حلول أخرى في حالة فشل التدريب.

ال코드 النهائي.

الملخص.

المقدمة :-

ما هو سرطان القولون والمستقيم؟ سرطان القولون والمستقيم هو مرض شائع، وربما مميت. الكشف المبكر أمر بالغ الأهمية. تشمل طرق الفحص تنظير القولون، والتنظير السيني، واختبارات البراز، مثل اختبار الدم الخفي في البراز (FOBT) والاختبار الكيميائي المناعي للبراز (FIT). يلعب التاريخ الوراثي والعائلي دوراً في تقييم المخاطر. قد تشمل الأعراض الدم في البراز، والتغيرات في عادات الأمعاء، وفقدان الوزن غير المبرر. أدوات التخخيص مثل الأشعة المقطعة والخز عات توّك الإصابة بالسرطان. يعد الفحص المنتظم للأفراد الذين تزيد أعمارهم عن 45 عاماً أو المعرضين لخطر أعلى على أمراً ضروريّاً للإصابة بالمرض في مراحله المبكرة والقابلة للعلاج. الاكتشاف المبكر يزيد من معدلاتبقاء على قيد الحياة وخيارات العلاج.

مع تطوير العلاجات المستهدفة، تعتمد العديد من العلاجات على الدراسات الجزيئية، والتي تتطلب أخذ عينات من أنسجة الورم من كتل البارافين للسلسل. يمكن أن يقلل الحل الآلي من عباء العمل على اختصاصي علم الأمراض من خلال العمل كجهاز فحص وقد يقلل من الذاتية في التخخيص.

في التخخيص المعتمد على الأنسجة، لا يزال يتبعن القيام بمعظم العمل يدوياً بواسطة أخصائي علم الأمراض باستخدام المجهر لفحص الشريان الملطخة بالهياماتوكسيلين والأيوزين (H&E). لتحديد مرحلة PN في المرض المنتشر أو محتوى الورم لتحليل الجينوم المصاب، من الضروري تحديد حجم وأعداد مناطق الورم. أساس هذه المهام هو التمييز الدقيق بين الخلايا السرطانية / الخبيثة والخلايا الطبيعية / الحميدة. ومع ذلك، فإن تحديد محتوى الورم ضعيف التكثير مع تباين كبير بين الأمراض. اعتماداً على كيان المرض المحدد، وخبرة أخصائي علم الأمراض، وحالة أخصائي علم الأمراض، قد تختلف النتائج بشكل كبير. بالإضافة إلى ذلك، فإن تحديد محتوى الورم ودوران مناطق الورم على شرائح H&E لتحديد المناطق التي يجب أخذ عينات منها للتحليل الجيني المصاب هو خطوة تحليلية مطلوبة من قبل هيئات الاعتماد من أجل إثراء محتوى الورم وضمان التحديد الدقيق للمتغيرات الجينية. نظراً لأن حجم مناطق الورم يمكن أن يكون صغيراً جداً، غالباً ما يُطلب من أخصائي علم الأمراض استخدام تكبير عالي للكشف عن الخلايا السرطانية. هذا الشرط يزيد بشكل كبير من عباء العمل على أخصائي علم الأمراض. تم تطبيق التعلم العميق بنجاح على العديد من المهام بما في ذلك معالجة الصور، معالجة الصوت، وترجمة اللغة. يظهر التقدم الحديث أنه يمكن أيضاً تطبيق التعلم العميق على معالجة الصور الطبية، مثل التصوير بالرنين المغناطيسي التصوير المقطعي المحوسب ، والتنظير الداخلي. في الآونة الأخيرة، أصبحت مجموعات بيانات علم الأمراض الرقمية متاحة للجمهور والباحثين وفتحت إمكانية تقييم جدوى تطبيق تقنيات التعلم العميق لتحسين كفاءة وجودة التخخيص النسجي.

ما هي مجموعة البيانات؟

تم الحصول على مجموعة البيانات بمعدل 5000 شريحة رقمية متعلقة بسرطان القولون.
بإمكانك الحصول عليها بكتابة الأمر البرمجي التالي:

```
# Colab's file access feature
from google.colab import files
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
```

الفصل الثامن: Colorectal Cancer Detection

```
# Then move kaggle.json into the folder where the API expects to find it.
```

```
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600
```

```
~/.kaggle/kaggle.json
```

وبعد حصولك على مفتاح التخوين من حساب كاكل بامكانك تنزيل الداتا سبيت

```
!kaggle datasets download -d falahgatea/colorectal-cancer
```

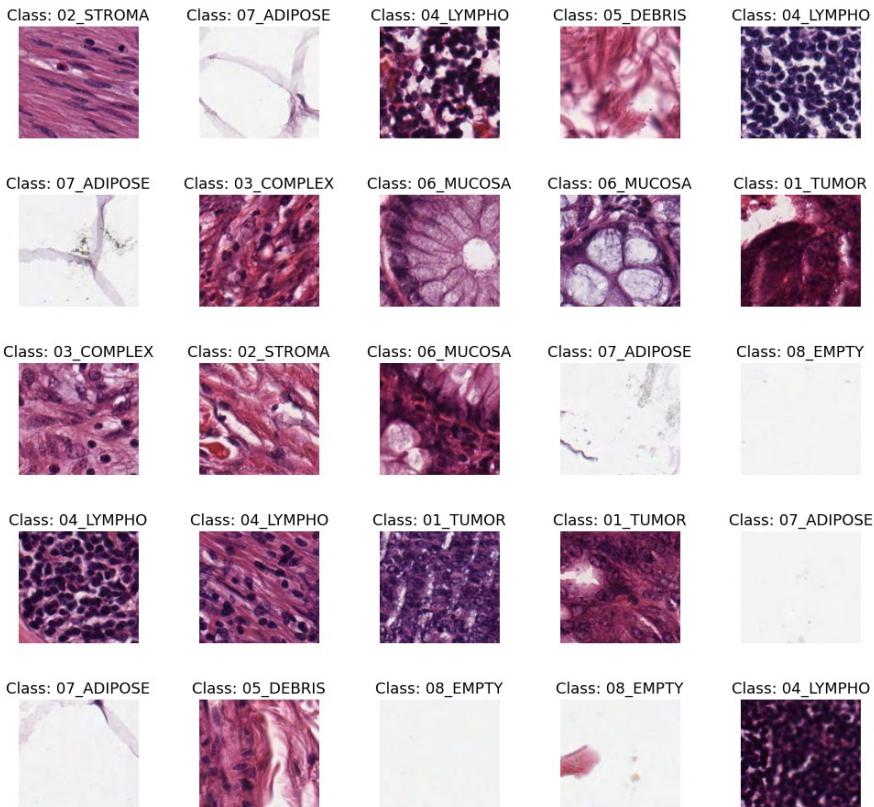
وفك الضغط عنها

```
!unzip /content/colorectal-cancer.zip
```

```
!rm -r /content/colorectal-cancer.zip
```

سوف نحصل على 8 فئات لسرطان القولون

[DEBRIS',COMPLEX',TUMOR',EMPTY',STROMA',MUCOSA',LYMPHO',ADIPOSE']



صورة رقم(1)

بإمكانك الاطلاع عليها من الموقع التالي المكتبة الوطنية للطب

[/https://www.ncbi.nlm.nih.gov](https://www.ncbi.nlm.nih.gov)

نحوٌ الشبكة العصبية .MobileNetV2

سوف نستخدم نموذج MobileNetV2

```
# Load pre-trained MobileNet model
model = models.mobilenet_v2(pretrained=True)
# Modify the last layer to match the number of classes in your dataset
num_classes = 8 # Replace with the actual number of classes in your dataset
model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)
# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
```

- البدء بالكود البرمجي الشامل.
- تقسيم مجموعات البيانات.

```
import torchvision.datasets as datasets
from torchvision.transforms import transforms
import os
from PIL import Image
# Define the transformations for the dataset
transform = transforms.Compose([
    transforms.Resize((224, 224)), # Resize the images to a specific size
    transforms.ToTensor(), # Convert images to tensors
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize the images
])

# Load the original dataset
dataset_path = "/content/Colorectal_Cancer/Kather_texture_2016_image_tiles_5000"
train_dataset = datasets.ImageFolder(dataset_path, transform)

# Get the class names
class_names = train_dataset.classes
print("Class Names:", class_names)
# Define the percentage of data for training, validation, and test sets
train_split = 0.8 # 80% for training
val_split = 0.1 # 10% for validation
test_split = 0.1 # 10% for testing
```

```

# Calculate the sizes of each split
dataset_size = len(train_dataset)
train_size = int(train_split * dataset_size)
val_size = int(val_split * dataset_size)
test_size = dataset_size - train_size - val_size
# Split the dataset
train_dataset, val_dataset, test_dataset = random_split(train_dataset, [train_size,
val_size, test_size])
# Save the test dataset to the "test" folder
test_folder = "/content/Colorectal_Cancer/test"
os.makedirs(test_folder, exist_ok=True) # Create the folder if it doesn't exist
for i, (image, label) in enumerate(test_dataset):
    class_name = class_names[label]
    class_folder = os.path.join(test_folder, class_name)
    os.makedirs(class_folder, exist_ok=True) # Create class folder if it doesn't exist
    filename = f'image_{i}.tif' # Generate a unique filename with the .tif extension
    image = transforms.ToPILImage()(image) # Convert tensor to PIL Image
    image.save(os.path.join(class_folder, filename))
# Print the sizes of each split
print(f"Train set size: {len(train_dataset)}")
print(f"Validation set size: {len(val_dataset)}")
print(f"Test set size: {len(test_dataset)}")

```

والحصول على مجموعة منها ضمن العدد التالي

Train set size: 4000
Validation set size: 500
Test set size: 500

• الكود البرمجي.

```

# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)

```

• شرح الكود.

يحدد الكود المقدم برامح تحميل البيانات للتدريب وبيانات التحقق من الصحة. يتم إنشاء برامح تحميل البيانات باستخدام `torch.utils.data.DataLoader`.

- تدريب "DataLoader": "Train_loader" يستخدم للتدريب ويتم إنشاؤه باستخدام مجموعة بيانات التدريب ("train_dataset"). يتم خلط البيانات (عشوائي = صحيح) أثناء التدريب لإدخال العشوائية ومنع التحيز.

الفصل الثامن: Colorectal Cancer Detection

- التحقق من الصحة: يستخدم "val_loader" للتتحقق و يتم إنشاؤه باستخدام مجموعة بيانات التحقق ("val_dataset"). لا يتم تبديل البيانات عشوائياً ("عشوائي = خطأ") أثناء التتحقق من الصحة، حيث يتم الاحتفاظ بالطلب الأصلي لأغراض التقييم.

تسمح برامج تحميل البيانات هذه بالتكرار الفعال لبيانات التدريب والتحقق من الصحة على دفعات، وهو أمر مفيد للتدريب وتقدير نماذج التعلم الآلي.

• الكود البرمجي.

```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• شرح الكود.

تنيس وظيفة الخسارة أداء النموذج، ويقوم المحسن بتحديث معلمات النموذج. وظيفة الخسارة المستخدمة هي الانتروبيا المتقطعة، ومناسبة لمهام التصنيف. المحسن المستخدم هو Adam ، الذي يضبط معدل التعلم لكل معلمة على حدة. تم ضبط معدل التعلم على 0.001.

• التدريب.

```
# Training loop
num_epochs = 50
train_losses = []
val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    train_losses.append(epoch_loss)
    model.eval() # Set the model to evaluation mode
    correct_predictions = 0
    total_predictions = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images = images.to(device)
            labels = labels.to(device)
```

```

outputs = model(images)
_, predicted = torch.max(outputs.data, 1)
total_predictions += labels.size(0)
correct_predictions += (predicted == labels).sum().item()
val_accuracy = correct_predictions / total_predictions
val_accuracies.append(val_accuracy)
val_loss = criterion(outputs, labels)
val_losses.append(val_loss.item())
print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation
Accuracy: {val_accuracy:.4f}")

```

• شرح الكود.

في حلقة التدريب، يتم تدريب النموذج لعدد محدد من الحقب. تتكون الحلقة من الخطوات التالية:

1. التدريب:

- تم ضبط النموذج على وضع التدريب.
- يتم تكرار بيانات التدريب على دفعات.
- تدرجات المحسن .
- يتم حساب تنبؤات النموذج للصور المدخلة.
- يتم حساب الخسارة بين التوقعات والتسميات الفعلية.
- يتم تنفيذ Backpropagation ، ويقوم المحسن بتحديث معلمات النموذج.
- خسارة التدريب متراکمة.

2. التحقق من الصحة:

- تم ضبط النموذج على وضع التقييم.
- يتم تكرار بيانات التتحقق على دفعات.
- يتم حساب تنبؤات النموذج للصور المدخلة.
- تُحسب الدقة بمقارنة التوقعات مع الملصقات الفعلية.
- يتم تسجيل دقة التتحقق من الصحة والخسارة.

3. الطباعة:

- تتم طباعة رقم الحقبة وحسارة التدريب ودقة التتحقق.

تتعقب حلقة التدريب خسائر التدريب والتحقق من الصحة بالإضافة إلى دقة التتحقق من الصحة لكل فترة. تكرر الحلقة هذه العملية لعدد محدد من الحقبات .

تساعد هذه الحلقة في تدريب النموذج من خلال ضبط وزانه بناءً على التدرجات المحسوبة وتقييم أدائه على مجموعة التتحقق من الصحة لمراقبة التقدم.

• خزن النموذج النهائي.

لفرض الاستدلال مستقبلاً لبيانات الاختبار والتطبيقات الأخرى.

```

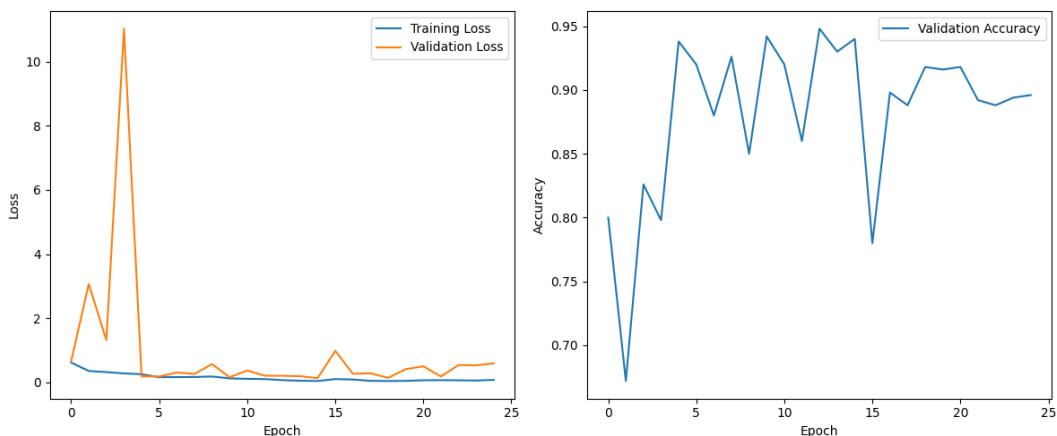
# Save the trained model
torch.save(model.state_dict(), "model.pth")

```

• نتائج التدريب.

Plotting the results

```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.tight_layout()
plt.show()
```



صورة رقم (2)

• مصفوفة الاربак للمقاييس.

#confusion matrix

```
from sklearn.metrics import confusion_matrix
# Initialize variables for storing the predictions and true labels
all_predictions = []
all_labels = []
# Set the model to evaluation mode
model.eval()
# Disable gradient calculation for efficiency
with torch.no_grad():
    for images, labels in val_loader:
```

```

images = images.to(device)
labels = labels.to(device)
outputs = model(images)
_, predicted = torch.max(outputs.data, 1)
# Append the predicted labels and true labels to the respective lists
all_predictions.extend(predicted.tolist())
all_labels.extend(labels.tolist())
# Calculate the confusion matrix
cm = confusion_matrix(all_labels, all_predictions)
print("Confusion Matrix:")
print(cm)

```

• شرح الكود.

يحسب الكود المتوفر مصفوفة الارباك عن طريق تقييم تنبؤات النموذج مقابل التسميات الحقيقة. يقوم بجمع التنبؤات والتسميات ، ويحسب مصفوفة الارباك باستخدام وظيفة "confusion_matrix" في sklearn ، ثم يطبع المصفوفة الناتجة. توفر مصفوفة الارباك معلومات حول أداء النموذج لكل فئة ، بما في ذلك التنبؤات الإيجابية الحقيقة والسلبية الحقيقة والإيجابية الخاطئة والتنبؤات السلبية الخاطئة.

• رسم مصفوفة الارباك بين الدقة والصور الحقيقة.

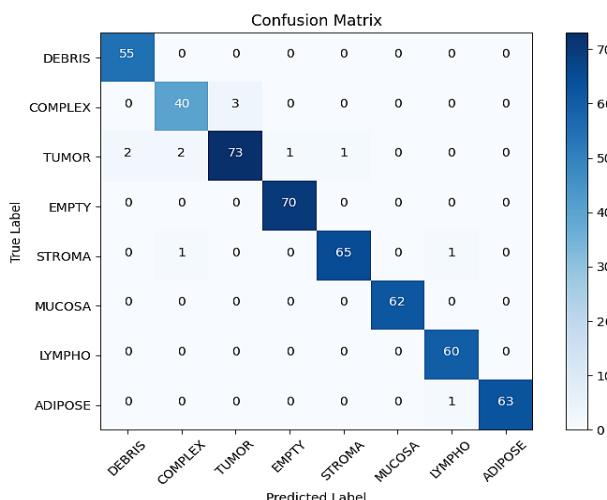
```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
# Initialize variables for storing the predictions and true labels
all_predictions = []
all_labels = []
# Set the model to evaluation mode
model.eval()
# Disable gradient calculation for efficiency
with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        # Append the predicted labels and true labels to the respective lists
        all_predictions.extend(predicted.tolist())
        all_labels.extend(labels.tolist())
# Calculate the confusion matrix
cm = confusion_matrix(all_labels, all_predictions)
# Plot the confusion matrix
classes = ['DEBRIS',
           'COMPLEX',

```

```
'TUMOR',
'EMPTY',
'STROMA',
'MUCOSA',
'LYMPHO',
'ADIPOSE'] # Replace with the actual class labels
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
fmt = 'd'
thresh = cm.max() / 2.
for i, j in np.ndindex(cm.shape):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()

# Calculate precision, recall, F1-score, and support for each class
class_report = classification_report(all_labels, all_predictions, target_names=classes)
print("Classification Report:")
print(class_report)
```



صورة رقم(3)

Classification Report:

	precision	recall	f1-score	support
DEBRIS	0.96	1.00	0.98	55
COMPLEX	0.93	0.93	0.93	43
TUMOR	0.96	0.92	0.94	79
EMPTY	0.99	1.00	0.99	70
STROMA	0.98	0.97	0.98	67
MUCOSA	1.00	1.00	1.00	62
LYMPHO	0.97	1.00	0.98	60
ADIPOSE	1.00	0.98	0.99	64
accuracy		0.98		500
macro avg	0.97	0.98	0.98	500
weighted avg	0.98	0.98	0.98	500

• شرح الكود.

- مصفوفة الارباك: توفر مصفوفة الارباك تمثيلاً مرتباً لتوقعات النموذج. يعرض عدد التنبؤات الإيجابية الحقيقية، السلبية الحقيقية، الإيجابية الخاطئة، والتنبؤات السلبية الخاطئة لكل فئة.
- تقرير التصنيف: يوفر تقرير التصنيف مقاييس إضافية لتقدير أداء النموذج. يتضمن الدقة (نسبة الإيجابيات الحقيقة إلى مجموع الإيجابيات الحقيقة والإيجابيات الزائفة)، الاسترجاع (نسبة الإيجابيات الحقيقة إلى مجموع الإيجابيات الحقيقة والسلبيات الخاطئة) ، درجة F1 (مقياس متوازن للدقة والذكر) ، والدعم (عدد تكرارات كل فئة في الملخصات الحقيقة).
- يعطي كل من مصفوفة الارباك وتقرير التصنيف نظرة ثاقبة لأداء النموذج على أساس الفصل ، مما يساعد على تقييم قدرته على التنبؤ بشكل صحيح بكل فئة.

أستدلال النموذج على صور الاختبار.

```
# Load the saved model for inference
loaded_model = models.mobilenet_v2(pretrained=False)
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features,
num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()

# Load your test dataset
test_dataset = datasets.ImageFolder("/content/Colorectal_Cancer/test",
transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1,
shuffle=True)

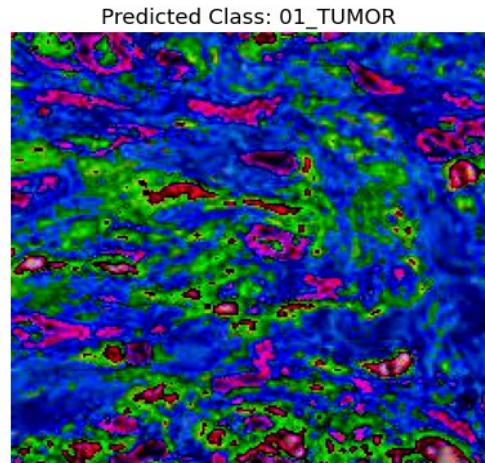
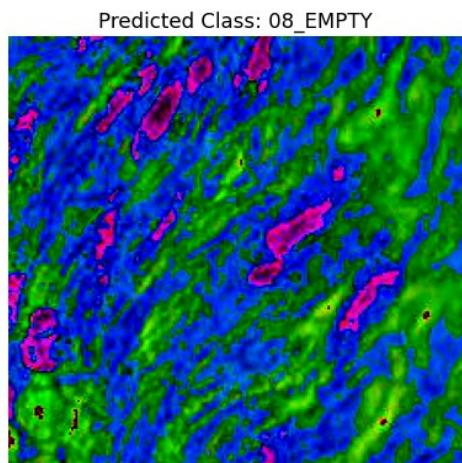
# Select 5 images for testing
test_images = []
```

```

test_labels = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
        _, predicted = torch.max(outputs.data, 1)
        for i in range(images.size(0)):
            image = images[i].permute(1, 2, 0).cpu().numpy()
            label = predicted[i].item()
            class_name = test_dataset.classes[label]
            test_images.append(image)
            test_labels.append(class_name)
            if len(test_images) >= 5:
                break
        if len(test_images) >= 5:
            break

# Display the selected test images
for i in range(len(test_images)):
    image = test_images[i]
    label = test_labels[i]
    plt.imshow(image)
    plt.title(f"Predicted Class: {label}")
    plt.axis("off")
    plt.show()

```



صورة رقم (4)

حلول أخرى في حالة فشل التدريب.

إذا فشل تدريب النموذج أو إذا كانت دقة النموذج غير مرضية ، فهناك عدة طرق لتحسين أداء النموذج. فيما يلي بعض الاستراتيجيات الشائعة:

1. زيادة قدرة النموذج: إذا لم يكن النموذج معقداً بما يكفي لانتقاد الأنماط الأساسية في البيانات ، فقد يكون غير ملائم. يمكن تحقيق زيادة سعة النموذج عن طريق إضافة المزيد من الطبقات ، أو زيادة عدد الوحدات المخفية ، أو استخدام نموذج أكبر مدعوماً مسبقاً.

2. جمع المزيد من بيانات التدريب: يمكن أن توزدي بيانات التدريب غير الكافية إلى التجهيز الزائد ، حيث يحفظ النموذج أمثلة التدريب بدلاً من التعلم الجيد للبيانات غير المرئية. يمكن أن يساعد جمع بيانات تدريب أكثر تنوعاً وتمثيلاً في تحسين قدرة النموذج على التعلم.

3. زيادة البيانات: يمكن لتقييات زيادة البيانات مثل الدوران والقياس والتقليل وإضافة الضوضاء أن تزيد بشكل مصطنع من حجم مجموعة بيانات التدريب وتساعد النموذج على تعلم ميزات أكثر قوة وثباتة.

4. تقييات التنظيم: يمكن استخدام طرق التنظيم مثل تنظيم L1 أو L2 لمنع الإفراط في التجهيز عن طريق إضافة مصطلح جزائي إلى وظيفة الخسارة. يمكن أن يؤدي التسرب، الذي يعطي بعض الخلايا العصبية بشكل عشوائي أثناء التدريب، إلى تنظيم النموذج ومنع الاعتماد المفرط على ميزات محددة.

5. ضبط المعلمات (Hyperparameters) الفائقة: يمكن للمعلمات التشغيلية مثل معدل التعلم وحجم الدفعه والمحسن أن تؤثر بشكل كبير على عملية التدريب. يمكن أن تساعد تجربة قيم مختلفة للمعلمات التشغيلية في العثور على التكوينات التي تعمل على تحسين دقة النموذج.

6. نقل التعلم: إذا كان لديك مجموعة بيانات صغيرة، فإن استخدام النماذج المدربة مسبقاً كنقطة بداية والضبط الدقيق لها في مهمتك المحددة يمكن أن يؤدي غالباً إلى نتائج أفضل. لقد تعلمت النماذج المدربة مسبقاً ميزات مفيدة منمجموعات البيانات واسعة النطاق ويمكن الاستفادة منها لتحسين الأداء في مجموعات البيانات الأصغر.

7. التحقق من مشكلات جودة البيانات: تأكد من تسمية بيانات التدريب بشكل صحيح وعدم وجود تناقضات أو أخطاء في مجموعة البيانات. يمكن أيضاً لخطوات المعالجة المسبقة للبيانات، مثل معالجة القيم المفقودة أو القيم المتطرفة، تحسين أداء النموذج.

8. استكشاف البنى المختلفة: جرب معماريات نموذجية مختلفة أو أشكال مختلفة من الهندسة المعمارية الحالية لإيجاد حل أفضل للمشكلة المطروحة. قد يتضمن ذلك تغيير عدد الطبقات، أو استخدام وظائف تنشيط مختلفة، أو دمج طبقات متخصصة مثل الطبقات التلافيية أو المترددة.

9. وقت تدريب أطول: في بعض الأحيان، قد يتطلب النموذج مزيداً من وقت التدريب للوصول إلى الحل الأمثل. قد تكون زيادة عدد الحقب أو التكرارات التدريبية ضرورية لتحقيق دقة أفضل.

10. التحليل والتصحيح: قم بتحليل أداء النموذج، وفحص منحنيات الخسارة، وتحقق من الأمثلة التي تم تصنيفها بشكل خاطئ. يمكن أن يوفر هذا نظرة ثاقبة لنقاط ضعف النموذج ويووجه المزيد من التحسينات.

تنظر أن تحسين دقة النموذج عملية تكرارية، وقد تتطلب تجربة استراتيجية متعددة معًا. التجربة والصبر والتحليل الدقيق هي المفتاح لتحسين أداء النموذج بشكل متكرر.

الكود النهائي.

رمز الاستجابة السريع



الملخص

في الختام، يمكن أن يكون PyTorch MobileNetV2 أداة قيمة في مجال تشخيص سرطان القولون والمستقيم. من خلال الاستفادة من قدرات نموذج التعلم العميق هذا، يمكن للأطباء الاستفادة من نهج فعال ودقيق لتحديد الفئات المختلفة المرتبطة بسرطان القولون والمستقيم، مثل TUMOR وCOMPLEX وDEBRIS وADPOSE وLYMPHO وMUCOSA وEMPTY وSTROMA.

توفر بنية MobileNetV2 توافرًا جيداً بين حجم النموذج ودقته، مما يجعله مناسباً لتطبيقات العالم الحقيقي، بما في ذلك التشخيص المبكر لسرطان القولون والمستقيم. يتيح تصميمه خفيف الوزن الاستدلال السريع على الأجهزة المحمولة أو في البيانات محدودة الموارد، مما يجعله في متناول الأطباء وذوي الاختصاص في علم الامراض.

من خلال تدريب النموذج بمجموعة بيانات متنوعة ومنسقة جيداً، يمكنه التعرف على السمات المختلفة المرتبطة بسرطان القولون والمستقيم وتصنيفها، مما يساعد الأطباء في إجراء تشخيصات دقيقة وفي الوقت المناسب. يمكن أن توفر قدرة النموذج على اكتشاف أنماط وهياكل معينة رؤى وتساعد في الكشف المبكر عن سرطان القولون والمستقيم، مما قد يؤدي إلى تحسين نتائج المرضى.

بشكل عام، يمكن استخدام PyTorch MobileNetV2 كأداة مفيدة للأطباء في تشخيص سرطان القولون والمستقيم. يمكن أن يتيح تطبيقه الكشف المبكر والمساعدة في قرارات العلاج الشخصية، مما يساهم في النهاية في تحسين رعاية المرضى ونتائجهم في مكافحة سرطان القولون والمستقيم.

الفصل التاسع

سرطان الرئة باستخدام نموذج ResNet-50

المقدمة.

نموذج ResNet-50 والكشف عن سرطان الرئة.

تطبيق النموذج على بيانات سرطان الرئة.

تدريب النموذج.

الاستدلال النهائي للنموذج على بيانات الاختبار.

خدمة تطبيق الويب.

الكود البرمجي النهائي.

الملخص.

مقدمة

ما هو سرطان الرئة ؟ سرطان الرئة هو نوع من السرطان يبدأ في الرئتين. يحدث عندما تنمو الخلايا غير الطبيعية في الرئتين وتتشكل لا يمكن السيطرة عليه، مكونة الأورام. يمكن أن تتدخل هذه الأورام مع الأداء الطبيعي للرئتين وقد تنتشر إلى أجزاء أخرى من الجسم من خلال عملية تسمى ورم خبيث.

• هناك نوعان رئيسيان من سرطان الرئة:

1. سرطان الرئة ذو الخلايا غير الصغيرة (NSCLC): هذا هو النوع الأكثر شيوعاً، ويمثل حوالي 80-85% من جميع حالات سرطان الرئة. يتضمن عدة أنواع فرعية، مثل السرطان الغدية وسرطان الخلايا الحرشفية وسرطان الخلايا الكبيرة.

2. سرطان الرئة ذو الخلايا الصغيرة (SCLC): هذا النوع من سرطان الرئة أقل شيوعاً ولكنه يميل إلى النمو والانتشار بسرعة أكبر من سرطان الرئة NSCLC. يرتبط بشدة بتدخين التبغ.

ينتاج سرطان الرئة في المقام الأول عن التعرض للمواد الضارة، وخاصة دخان التبغ. التدخين هو السبب الرئيسي لسرطان الرئة ويمثل غالبية الحالات. تشمل عوامل الخطير الأخرى التعرض للتدخين السلبي، والتعرض المهني لبعض المواد الكيميائية (مثل الأسبستوس، والرادون، والزرنين)، والتاريخ العائلي لسرطان الرئة، وأمراض الرئة الكامنة. يمكن أن تختلف أعراض سرطان الرئة ولكنها قد تشمل السعال المستمر، وألم الصدر، وضيق التنفس، وسعال الدم، والتعب، وفقدان الوزن، والتهابات device التنفسية المتكررة. ومع ذلك، في المراحل المبكرة، قد لا يسبب سرطان الرئة أعراضًا ملحوظة، مما يجعل من الصعب اكتشافه وتشخيصه.

التخيص المبكر أمر بالغ الأهمية لتحسين فرص نجاح العلاج. تشمل طرق التخيص لسرطان الرئة اختبارات التصوير (مثل الأشعة السينية للصدر والأشعة المقطعة)، وعلم خلايا البلغم ، والخزعة. تعتمد خيارات العلاج على نوع ومرحلة سرطان الرئة ولكنها قد تشمل الجراحة والعلاج الإشعاعي والعلاج الكيميائي والعلاج الموجه والعلاج المناعي. تتضمن الوقاية من سرطان الرئة تجنب أو تقليل التعرض لعوامل الخطير، وخاصة التدخين والتدخين السلبي. يعد الإقلاع عن التدخين والمشاركة في برامج الإقلاع عن التدخين أمراً ضرورياً لتقليل خطر الإصابة بسرطان الرئة. يعد سرطان الرئة مصدر قلق كبير للصحة العامة، ويلعب الاكتشاف المبكر وخيارات العلاج المحسنة والتدابير الوقائية دوراً حاسماً في الحد من تأثيره على الأفراد والمجتمع.

نموذج ResNet-50 والكشف عن سرطان الرئة

أظهرت تقنيات الذكاء الاصطناعي (AI)، وخاصة خوارزميات التعلم العميق ، واعدة في الكشف المبكر عن سرطان الرئة وتشخيصه. تم استخدام ResNet-50 ، وهي بنية التعلم العميق ، في العديد من الدراسات لهذا الغرض. فيما يلي بعض التقنيات والتطبيقات الرئيسية:

1. **الاكتشاف الآلي للعقيدات الرئوية:** عقائد الرئة عبارة عن نمو صغير غير طبيعي في الرئتين يمكن أن يكون مؤشرات مبكرة لسرطان الرئة. تم استخدام خوارزميات الذكاء الاصطناعي، بما في ذلك ResNet-50

الفصل التاسع : سرطان الرئة باستخدام نموذج ResNet-50
للكشف التلقائي عن عقيدات الرئة وتصنيفها في التصوير الطبي ، مثل التصوير المقطعي المحوسب للصدر. تقوم هذه الخوارزميات بتحليل الصور وتحديد العقيدات المحتملة بناءً على شكلها وملمسها وميزاتها الأخرى.

2. التشخيص بمساعدة الكمبيوتر : يمكن أن تساعد أنظمة التشخيص بمساعدة الكمبيوتر (CAD) القائمة على الذكاء الاصطناعي أطباء الأشعة في تفسير الصور الطبية من خلال توفير التحليل الآلي وإبراز المناطق المشبوهة. تم استخدام ResNet-50، جنباً إلى جنب مع نماذج التعلم العميق الأخرى ، لتطوير أنظمة CAD لتشخيص سرطان الرئة. يمكن أن تساعد هذه الأنظمة في تحسين الدقة والكفاءة في التعرف على سرطان الرئة في مرحلة مبكرة.

3. تقييم المخاطر والتنبؤ بها: تم استخدام نماذج التعلم العميق، بما في ذلك ResNet-50 ، لتحليل بيانات المريض وتحديد عوامل الخطير المرتبطة بتطور سرطان الرئة. من خلال تحليل عوامل مثل تاريخ التدخين والمعلومات الوراثية والسجلات الطبية، يمكن لخوارزميات الذكاء الاصطناعي التنبؤ باحتمالية إصابة الفرد بسرطان الرئة. يمكن أن يساعد ذلك في تحديد الأفراد المعرضين لمخاطر عالية والذين قد يستفيدون من الفحص والتدخل المبكر.

4. التكامل مع البيانات السريرية: يمكن دمج تقنيات الذكاء الاصطناعي الحديثة بما في ذلك نموذج (ResNet-50) مع البيانات السريرية ، مثل التركيبة السكانية للمريض ، والتاريخ الطبي ، والنتائج المختبرية ، لتطوير نماذج شاملة للكشف عن سرطان الرئة والتنبؤ به. من خلال الجمع بين بيانات التصوير والمعلومات الأخرى الخاصة بالمريض، يمكن أن توفر هذه النماذج نهجاً أكثر شمولية للكشف المبكر وتخطيط العلاج المخصص.

5. تعلم زيادة البيانات ونقلها: يمكن أن يستفيد نموذج ResNet-50 بفضل بنيتها العميقة وعدها الكبير من المعلومات من تقنيات زيادة البيانات لتحسين أداء نماذج الكشف عن سرطان الرئة. تتضمن زيادة البيانات تطبيق تحويلات مختلفة على مجموعة البيانات الحالية لإنشاء عينات تدريب إضافية، مما يساعد على تحسين متانة النموذج. بالإضافة إلى ذلك، يمكن استخدام تقنيات نقل التعلم حيث يتم ضبط نموذج ResNet-50 المدرب مسبقاً والمدرب على مجموعة بيانات كبيرة (على سبيل المثال ، ImageNet) على مجموعة بيانات أصغر لسرطان الرئة ، مما يسمح بتعزيز دقة أفضل.

توضح هذه التقنيات إمكانات الذكاء الاصطناعي، وخاصة باستخدام ResNet-50 ، في تحسين الكشف المبكر عن سرطان الرئة وتشخيصه. من خلال الاستفادة من خوارزميات التعلم العميق ومجموعات البيانات واسعة النطاق، تهدف هذه الأساليب إلى مساعدة المتخصصين في الرعاية الصحية في اتخاذ قرارات دقيقة وفي الوقت المناسب، مما يؤدي في النهاية إلى تحسين نتائج المرضى في إدارة سرطان الرئة. ومع ذلك، من المهم ملاحظة أن تقنيات الذكاء الاصطناعي هذه لا تزال قيد البحث والتحقق من صحتها، وأن دمجها في الممارسة السريرية يتطلب مزيداً من دراسات التقييم والتحقق من الصحة.

تطبيق النموذج على بيانات سرطان الرئة.

هناك ثلاثة فئات في مجموعة البيانات، كل منها تحتوي على 5000 صورة، وهي:

1. Lung benign tissue
2. Lung adenocarcinoma
3. Lung squamous cell carcinoma

1. أنسجة الرئة الحميد.
2. سرطان الغدة الرئوية.

3. سرطان الخلايا الحرشفية في الرئة.

تم إنشاء الصور من عينة أصلية من المصادر المتفاقة مع HIPAA والتي تم التحقق من صحتها، وتتألف من 15000 صورة إجمالية لأنسجة الرئة (500 أنسجة رئوية حميدة، و500 سرطان رئوية، و500 سرطان خلايا رئوية حرشفية).

تدريب النموذج.

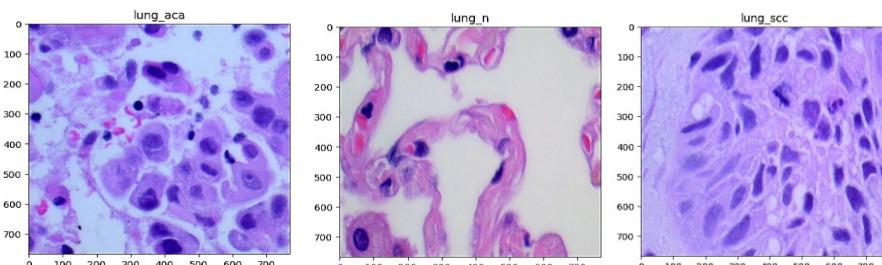
- تنزيل البيانات.**

```
# Colab's file access feature
from google.colab import files
#retrieve uploaded file
uploaded = files.upload
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
    # Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d falahgatea/lung-cancer-dataset
!unzip /content/lung-cancer-dataset.zip
!rm -r /content/lung-cancer-dataset.zip
```

بإمكانك معرفة عدد الصورة باستخدام الكود التالي :

```
#calculat number images
import os
def count_files(folder_path):
    file_count = 0
    for dirpath, dirnames, filenames in os.walk(folder_path):
        file_count += len(filenames)
    return file_count
# Specify the path to the folder you want to count the files in
folder_path = '/content/lung_image_sets'
total_files = count_files(folder_path)
print(f"Total number of files: {total_files}")
```

فتبين أن عددها 15000 صورة للرئة المذكورة أعلاه مصنفة لثلاثة فئات.



• تحميل نموذج الشبكة العصبية .ResNet50

```
# Load pre-trained ResNet-50 model
model = models.resnet50(pretrained=True)

# Modify the last layer to match the number of classes in your dataset
num_classes = 3 # Replace with the actual number of classes in your dataset
model.fc = nn.Linear(model.fc.in_features, num_classes)

# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

• شرح الكود.

دعنا ننتقل إلى الكود خطوة بخطوة لفهم ما يفعله كل جزء:
الخطوة 1: قم بتحميل نموذج ResNet-50 المدرب مسبقاً

```
model = models.resnet50(pretrained=True)
```

في هذه الخطوة، نقوم بتحميل نموذج ResNet-50 المدرب مسبقاً من وحدة `torchvision.models`. يقوم `Models.resnet50 (prerained = True)` بتحميل نموذج ResNet-50 `model.fc` بأوزان مدربة مسبقاً على مجموعة بيانات `ImageNet`.

الخطوة 2: قم بتعديل الطبقة الأخيرة لمطابقة عدد الفئات في مجموعة البيانات الخاصة بك

```
num_classes = 3 # Replace with the actual number of classes in your dataset
```

```
model.fc = nn.Linear(model.fc.in_features, num_classes)
```

في هذه الخطوة، نقوم بتعديل آخر طبقة متصلة بالكامل (`model.fc`) من نموذج ResNet-50 `model.fc.in_features` لمطابقة عدد الفئات في مجموعة البيانات الخاصة بك. هنا، يعطينا "model.fc.in_features" عدد ميزات الإدخال للطبقة الأخيرة ، والتي تُستخدم بعد ذلك لإنشاء طبقة خطية جديدة بوحدات الإخراج "عدد_الفئات".

الخطوة 3: اضبط `device` (GPU) إذا كان متاحاً، وإلا وحدة المعالجة المركزية (CPU)

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

```
model = model.to(device)
```

في هذه الخطوة، قمنا بتعيين `device` إما على وحدة معالجة الرسومات (ان وجد) أو وحدة المعالجة المركزية.

تحتفظ وظيفة `torch.cuda.is_available` من توفر وحدة معالجة الرسومات (GPU) التي تدعم CUDA.

إذا كان الأمر كذلك، فلن نضبط `device` على "cuda" (GPU) ، وإلا على "cpu". ثم نقوم بنقل النموذج

إلى `device` المختار باستخدام طريقة ".to (device)"

الخطوة 4: حدد التحويل المطبق على كل صورة

```
transform = transforms.Compose()
```

```
transforms.Resize,(256)
```

```
transforms.CenterCrop,(224)
```

```
transforms.ToTensor,
```

الفصل التاسع : سرطان الرئة باستخدام نموذج ResNet-50

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]))

في هذه الخطوة، نحدد تسلسل تحويلات الصور باستخدام فئة "transforms.Compose" من `torchvision`. التحولات المطبقة على كل صورة هي كما يلي:

-transforms.Resize(256): Resizes the image to a square of size 256x256 pixels.

-transforms.CenterCrop(224): Takes a center crop of size 224x224 pixels from the image.

-transforms.ToTensor: Converts the image to a PyTorch tensor.

-transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]):

تطبيع موتر الصورة بطرح المتوسط والقسمة على الانحراف المعياري. تستخدم قيم الانحراف المعياري والمتوسط بشكل شائع للنماذج المدربة مسبقاً المدربة على ImageNet.

يتم تطبيق هذه التحويلات عادةً على صور الإدخال قبل إدخالها في النموذج للاستدلال.

هذا كل شيء! لديك الآن نموذج ResNet-50 مدرب مسبقاً مع آخر طبقة تم تعديلها لمهمة التصنيف المحددة الخاصة بك، وتعينتها على `device` المناسب، وبتحويل محدد ليتم تطبيقه على كل صورة إدخال.

• تقسيم البيانات.

```
import torchvision.datasets as datasets
from torchvision.transforms import transforms
import os
from PIL import Image
# Define the transformations for the dataset
transform = transforms.Compose([
    transforms.Resize((224, 224)), # Resize the images to a specific size
    transforms.ToTensor(), # Convert images to tensors
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize the images
])
# Load the original dataset
dataset_path = "/content/lung_image_sets"
train_dataset = datasets.ImageFolder(dataset_path, transform=transform)
# Get the class names
class_names = train_dataset.classes
print("Class Names:", class_names)
# Define the percentage of data for training, validation, and test sets
train_split = 0.8 # 80% for training
val_split = 0.1 # 10% for validation
test_split = 0.1 # 10% for testing
# Calculate the sizes of each split
dataset_size = len(train_dataset)
train_size = int(train_split * dataset_size)
val_size = int(val_split * dataset_size)
test_size = dataset_size - train_size - val_size
# Split the dataset
train_dataset, val_dataset, test_dataset = torch.utils.data.random_split(train_dataset,
[train_size, val_size, test_size])
```

```
# Save the test dataset to the "test" folder
test_folder = "/content/dataset/test"
os.makedirs(test_folder, exist_ok=True) # Create the folder if it doesn't exist
for i, (image, label) in enumerate(test_dataset):
    class_name = class_names[label]
    class_folder = os.path.join(test_folder, class_name)
    os.makedirs(class_folder, exist_ok=True) # Create class folder if it doesn't exist
    filename = f'image_{i}.tif' # Generate a unique filename with the .tif extension
    image = transforms.ToPILImage(image) # Convert tensor to PIL Image
    image.save(os.path.join(class_folder, filename))
# Print the sizes of each split
print(f"Train set size: {len(train_dataset)}")
print(f"Validation set size: {len(val_dataset)}")
print(f"Test set size: {len(test_dataset)}")
```

لتظهر النتيجة كالتالي:

```
Class Names: ['lung_aca', 'lung_n', 'lung_scc']
Train set size: 12000
Validation set size: 1500
Test set size: 1500
```

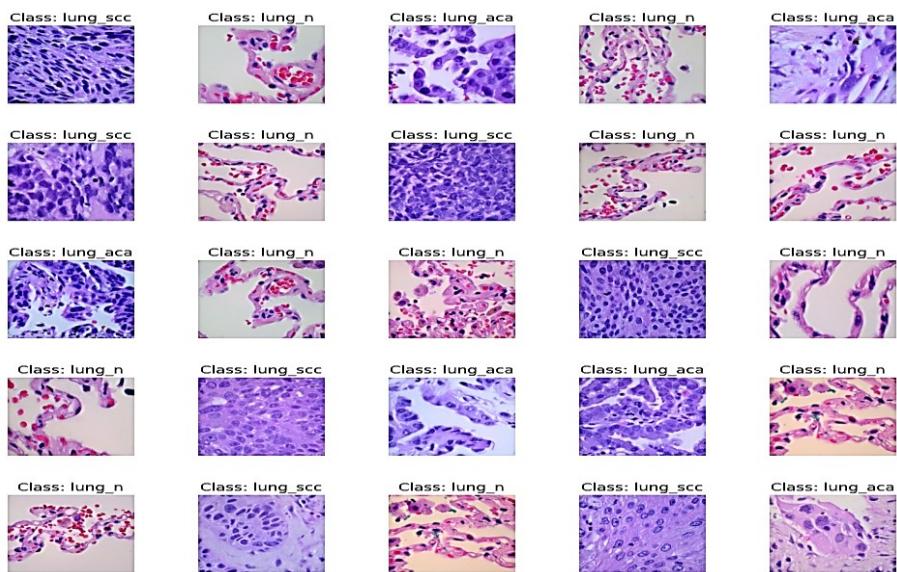
• شرح الكود

يوضح الكود المقدم كيفية تقسيم مجموعة البيانات إلى مجموعات التدريب والتحقق والاختبار باستخدام PyTorch. يستخدم تحويلات الصورة، مثل تغيير الحجم، والتحويل إلى موترات، والتطبيع. يتم تحميل مجموعة البيانات باستخدام "ImageFolder" من torchvision ، ويتم الحصول على أسماء الفئات. ثم يتم تقسيم مجموعة البيانات بشكل عشوائي إلى النسب المحددة. يحفظ الكود صور مجموعة بيانات الاختبار في مجلدات فئة منفصلة. أخيراً، تتم طباعة أحجام كل قسم. هذا الكود مفيد لتنظيممجموعات البيانات وإعدادها لمهام التعلم العميق خاصةً لتصنيف الصور.

• عرض بعض العينات قبل التدريب

```
# Create a subplot to display the images
import matplotlib.pyplot as plt
fig, axs = plt.subplots(5, 5, figsize=(10, 10))
fig.tight_layout()
# Iterate over the selected images and display them
for i, (image_path, class_label) in enumerate(selected_images):
    image = Image.open(image_path)
    # Display the image in the subplot
    ax = axs[i // 5][i % 5]
    ax.axis('off')
    ax.imshow(image)
    ax.set_title(f"Class: {class_label}")
# Show the plot
plt.show()
```

• نتيجة التنفيذ



(2) رقم

• تحميل بيانات التدريب وبيانات التحقق.

```
# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
```

• شرح الكود.

في هذه الخطوة، نحدد حجم الدفعة لمحمل البيانات. يحدد حجم الدفعة عدد العينات في كل دفعه صغيرة أثناء التدريب والتقدير.

هنا، نستخدم فئة `torch.utils.data.DataLoader` لإنشاء أداة تحميل بيانات لمجموعة بيانات التدريب. "train_dataset" هي مجموعة البيانات التي نريد تحميلها، وتحدد المعلمة "batch_size" عدد العينات في كل دفعه صغيرة. تقوم الوسيطة `shuffle = True` بتبديل ترتيب العينات في كل فتره لإدخال العشوائية وتحسين التدريب.

وبالمثل، في هذه الخطوة، نقوم بإنشاء أداة تحميل بيانات لمجموعة بيانات التحقق باستخدام فئة "torch.utils.data.DataLoader". "val_dataset" هي مجموعة البيانات التي نريد تحميلها ، وتحدد المعلمة "batch_size" عدد العينات في كل دفعه صغيرة. تضمن الوسيطة `shuffle = False` بقاء ترتيب العينات دون تغيير أثناء التحقق من الصحة.

تسمح لنا برامج تحميل البيانات بتكرار البيانات بشكل ملائم على دفعات صغيرة أثناء التدريب والتقدير. أنها توفر ميزات مثل خلط البيانات، والتجميع، وتحميل البيانات الموازي للتدريب الفعال لنماذج التعلم العميق.

الفصل التاسع : سرطان الرئة باستخدام نموذج ResNet-50

هذا كل شيء! لديك الآن برمج تحميل بيانات لمجموعات بيانات التدريب والتحقق من الصحة، والتي يمكن استخدامها لتكرار البيانات في مجموعة صغيرة أثناء التدريب والتقييم.

• استدعاء دالة الفقدان والمحسن.

```
# Define the loss function and optimizer
```

```
criterion = nn.CrossEntropyLoss
```

```
optimizer = optim.Adam(model.parameters, lr=0.001)
```

• شرح الكود.

تقيس وظيفة الخسارة أداء النموذج، ويقوم المحسن بتحديث معلمات النموذج. وظيفة الخسارة المستخدمة هي الانتروبيا المتقطعة، ومناسبة لمهام التصنيف. المحسن المستخدم هو Adam ، الذي يضبط معدل التعلم لكل معلمة على حدة. تم ضبط معدل التعلم على 0.001

• تدريب النموذج.

```
# Training loop
```

```
num_epochs = 25
```

```
train_losses = []
```

```
val_losses = []
```

```
val_accuracies = []
```

```
for epoch in range(num_epochs):
```

```
    model.train # Set the model to training mode
```

```
    running_loss = 0.0
```

```
    for images, labels in train_loader:
```

```
        images = images.to(device)
```

```
        labels = labels.to(device)
```

```
        optimizer.zero_grad
```

```
        outputs = model(images)
```

```
        loss = criterion(outputs, labels)
```

```
        loss.backward
```

```
        optimizer.step
```

```
        running_loss += loss.item * images.size(0)
```

```
    epoch_loss = running_loss / len(train_dataset)
```

```
    train_losses.append(epoch_loss)
```

```
    model.eval # Set the model to evaluation mode
```

```
    correct_predictions = 0
```

```
    total_predictions = 0
```

```
    with torch.no_grad:
```

```
        for images, labels in val_loader:
```

```
            images = images.to(device)
```

```
            labels = labels.to(device)
```

```
            outputs = model(images)
```

```
            _, predicted = torch.max(outputs.data, 1)
```

```
            total_predictions += labels.size(0)
```

```
            correct_predictions += (predicted == labels).sum.item
```

```

val_accuracy = correct_predictions / total_predictions
val_accuracies.append(val_accuracy)
val_loss = criterion(outputs, labels)
val_losses.append(val_loss.item)

print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation
Accuracy: {val_accuracy:.4f}")

```

• شرح الكود.

يوضح مقتطف الشفرة هذا حلقة تدريب نموذج التعلم العميق. فيما يلي ملخص لما يفعله كل جزء:

- 1. تهيئة حلقة التدريب :** يقوم الكود بتهيئة القوام الفارغة لتخزين خسائر التدريب ("خسائر التدريب") ، و خسائر التحقق ("val_losses") ، و دقة التحقق من الصحة ("val_accuracies").
- 2. Epoch Loop :** يتكرر الكود عبر عدد محدد من الحقب باستخدام وظيفة **.range (num_epochs)**.

3. مرحلة التدريب : داخل كل فترة ، يتم تعين النموذج على وضع التدريب باستخدام **model.train** . بعد ذلك ، لكل دفعه صغيرة من بيانات التدريب ، يقوم الكود بتنفيذ الخطوات التالية:

- ينقل صور الإدخال والسميات إلى **device** (وحدة المعالجة المركزية أو وحدة معالجة الرسومات) باستخدام ". إلى (device)".
- يمسح التدرجات من التكرار السابق باستخدام **optimizer.zero_grad** .

- يقوم بإعادة نشر الصور من خلال النموذج للحصول على المخرجات المتوقعة باستخدام النموذج (الصور) - يحسب الخسارة بين المخرجات المتوقعة وعلامات الحقيقة الأساسية باستخدام دالة الخسارة المحددة ("المعيار").

- للخلف نشر التدرجات اللونية من خلال النموذج باستخدام **loss.backward** " ويحدث اوزان النموذج باستخدام **optimizer.step** ".

- تراكم خسارة التشغيل بضرب قيمة الخسارة في حجم الدفعه (0) ((images.size (0)).

4. حساب مقاييس الحقيقة: بعد كل فترة ، يحسب الكود وتخزن متوسط خسارة التدريب عن طريق قسمة خسارة التشغيل على العدد الإجمالي للعينات في مجموعة بيانات التدريب ((train_dataset).len).

5. مرحلة التتحقق: يتم ضبط النموذج على وضع التقييم باستخدام **model.eval** . بعد ذلك ، لكل دفعه صغيرة من بيانات التتحقق من الصحة ، يقوم الكود بتنفيذ الخطوات التالية:

- ينقل صور الإدخال والسميات إلى **device**.
- يقوم بإعادة نشر الصور من خلال النموذج للحصول على المخرجات المتوقعة.
- يحسب فقدان التحقق بين المخرجات المتوقعة والسميات باستخدام دالة الخسارة المحددة.
- حساب عدد التنبؤات الصحيحة والعدد الإجمالي للتنبؤات.
- يحدث دقة التتحقق من خلال قسمة عدد التنبؤات الصحيحة على العدد الإجمالي للتنبؤات.

6. Metric Storage : ثُلُج الكود خسارة التتحقق المحسوبة (**val_loss.item**) ودقة التتحقق من صحة القوام الخاصة بها ("val_losses" و "val_accuracies").

7. التسجيل: في نهاية كل فترة ، يطبع الكود رقم الفترة الحالية وخسارة التدريب ودقة التتحقق. تقوم حلقة التدريب هذه بتدريب النموذج على عدد محدد من الحقب، وتحديث معلمات النموذج باستخدام **backpropagation** وتحسين الخسارة باستخدام المحسن المختار. كما أنه يقيم أداء النموذج في مجموعة التتحقق ويتبع مقاييس التدريب والتحقق من الصحة.

الفصل التاسع : سلطان الرنة باستخدام نموذج ResNet-50
 يرجى ملاحظة أن الكود يفترض أن متغير "device" قد تم تعريفه مسبقاً وأن النموذج والمعيار والمحسن وجميع البيانات قد تم إعدادها بشكل صحيح.
 • خزن النموذج النهائي.

```
# Save the trained model
torch.save(model.state_dict, "model.pth")
```

• النتائج مع مصفوفة الارباع.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
# Initialize variables for storing the predictions and true labels
all_predictions = []
all_labels = []
# Set the model to evaluation mode
model.eval()
# Disable gradient calculation for efficiency
with torch.no_grad:
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        # Append the predicted labels and true labels to the respective lists
        all_predictions.extend(predicted.tolist())
        all_labels.extend(labels.tolist())
# Calculate the confusion matrix
cm = confusion_matrix(all_labels, all_predictions)
# Plot the confusion matrix
classes = ['lung_aca', 'lung_n', 'lung_scc'] # Replace with the actual class labels
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
fmt = 'd'
thresh = cm.max / 2.
for i, j in np.ndindex(cm.shape):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
```

```

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout
plt.show

# Calculate precision, recall, F1-score, and support for each class
class_report = classification_report(all_labels, all_predictions, target_names=classes)
print("Classification Report:")
print(class_report)

```

٠ شرح الكود.

يحسب مقتطف الكود هذا ويمثل مصفوفة الارتباط ويقدم تقرير تصنیف لأداء النموذج في مجموعة التحقق من الصحة. فيما يلي ملخص لما يفعله كل جزء :

١. **التهيئة**: يستورد الكود المكتوبات الضرورية، بما في ذلك "numpy" و "sklearn.metrics.classification_report" و "matplotlib.pyplot" و "confusion_matrix".

٢. **تهيئة المتغيرات** : يقوم الكود بتهيئة القوائم الفارغة ("جميع التوقعات" و "جميع التصنیفات") لتخزين التسمیات المتوقعة والتسمیات الحقيقة، على التوالي.

٣. **وضع التقييم وتعطیل حساب التدرج** : يتم تعیین النموذج على وضع التقييم باستخدام `model.eval()` ، ويتم تعطیل حساب التدرج لتحسين الكفاءة باستخدام مدير السياق `torch.no_grad_`.

٤. **التنبؤات والعلامات الحقيقة** : لكل دفعۃ صغيرة من بيانات التحقق من الصحة، تنفذ الشفرة الخطوات التالية:

- ينقل صور الإدخال والتسمیات إلى `device` (وحدة المعالجة المركزية أو وحدة معالجة الرسومات).

- يقوم بإعادة نشر الصور من خلال النموذج للحصول على المخرجات المتوقعة.

- يستخرج الملصقات المتباعدة من خلال إيجاد فهرس قيمة الخرج القصوى باستخدام `torch.max`.

- إلحاد التسمیات المتوقعة والتسمیات الحقيقة بالقوائم المعنية ("جميع التوقعات" و "جميع التصنیفات").

٥. **مصفوفة الارتباط**: يحسب الكود مصفوفة الارتباط باستخدام `sklearn.metrics.confusion_matrix` من خلال توفير التسمیات الحقيقة (`all_labels`) والتسمیات المتوقعة (`all_predictions`) كمدخلات.

٦. **رسم مصفوفة الارتباط**: تم رسم مصفوفة الارتباط باستخدام `matplotlib.pyplot.imshow` . تم تعیین مخطط الألوان على "plt.cm.Blues" ، ويتم إضافة العناوين وشريط الألوان وتسمیات التجزئة للتوضیح.

٧. **تقریر التصنیف**: يحسب الكود الدقة والاسترجاع ودرجة F1 والدعم لكل فئة باستخدام `sklearn.metrics.classification_report` ("all_labels") والتسمیات المتوقعة ("all_predictions") مع تسمیات الفئات المستهدفة ("الفئات").

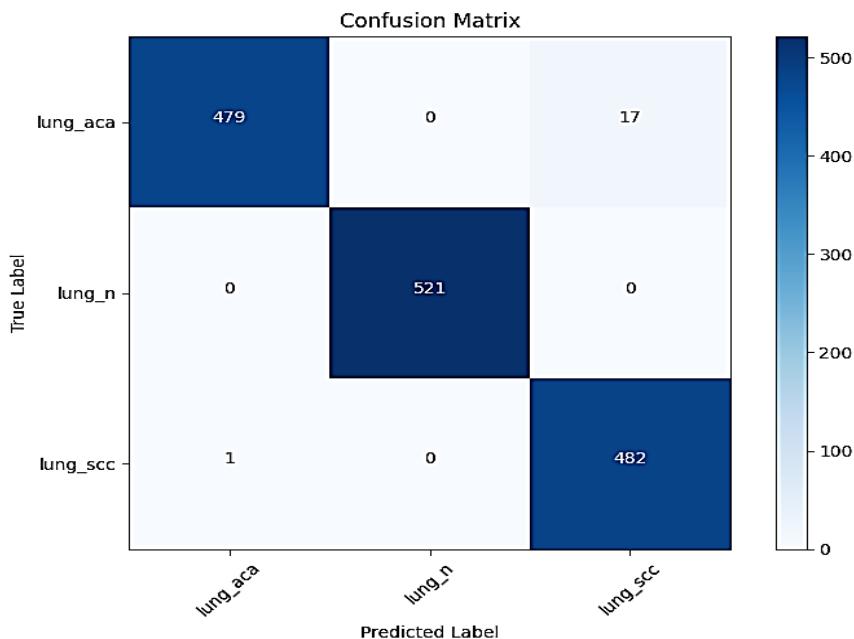
٨. **طباعة تقریر التصنیف**: يتم طباعة تقریر التصنیف باستخدام `print` .

توفر مصفوفة الارتباط تمثیلاً مرنیاً لأداء النموذج من خلال إظهار عدد الإيجابيات الحقيقة والإيجابيات الخاطئة والسلبيات الحقيقة والسلبيات الكاذبة لكل فئة. يوفر تقریر التصنیف مقاييس تفصیلية مثل الدقة والتذكر ودرجة F1 والدعم لكل فئة، مما يعطي نظرة ثاقبة على أداء النموذج في الفئات المختلفة.

يرجى ملاحظة أنك بحاجة إلى استبدال متغير "الفئات" بتسمیات الفئات الفعلية المستخدمة في مجموعة البيانات الخاصة بك.

الفصل التاسع : سرطان الرئة باستخدام نموذج ResNet-50

يمكن أن يساعدك هذا الكود في تقييم أداء النموذج الخاص بك وتحليله في مجموعة التحقق من الصحة، مما يوفر معلومات قيمة لمزيد من التحسينات أو اتخاذ القرار.



(3) رقم

الاستدلال النهائي للنموذج على بيانات الاختبار.

```
# Load the saved model for inference
#loaded_model = models.mobilenet_v2(pretrained=False)
loaded_model=models.resnet50(pretrained=False)
loaded_model.classifier[1] = nn.Linear(loaded_model.classifier[1].in_features,
num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval
# Load your test dataset
test_dataset = datasets.ImageFolder("/content/dataset/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1, shuffle=True)
# Select 5 images for testing
test_images = []
test_labels = []
with torch.no_grad:
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
```

```

_, predicted = torch.max(outputs.data, 1)
for i in range(images.size(0)):
    image = images[i].permute(1, 2, 0).cpu.numpy()
    label = predicted[i].item()
    class_name = test_dataset.classes[label]
    test_images.append(image)
    test_labels.append(class_name)
    if len(test_images) >= 5:
        break
    if len(test_images) >= 5:
        break
# Display the selected test images
for i in range(len(test_images)):
    image = test_images[i]
    label = test_labels[i]
    plt.imshow(image)
    plt.title(f"Predicted Class: {label}")
    plt.axis("off")
    plt.show

```

• شرح الكود.

يقوم مقتطف الشفرة هذا بتحميل نموذج محفوظ للاستدلال ويستخدمه للتنبؤ بفئات صور الاختبار. فيما يلي ملخص لما يفعله كل جزء:

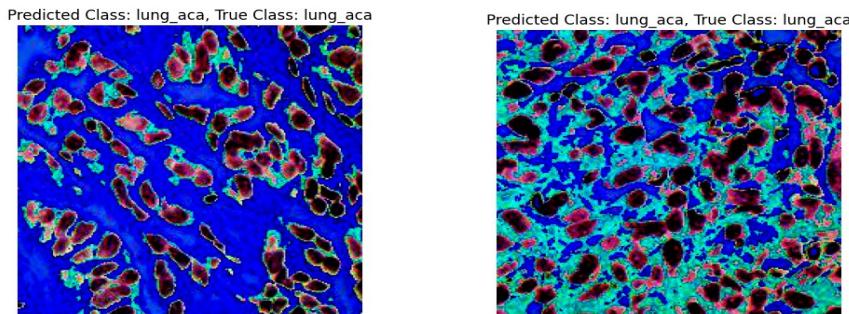
1. **تحميل النموذج**: يقوم الكود بتحميل نموذج ResNet-50 مدرب مسبقاً بدون الطبقة العلوية المتصلة بالكامل. ثم يعدل الطبقة الأخيرة لمطابقة عدد الفئات في مجموعة البيانات. يتم تحميل دكت حالة النموذج من ملف محفوظ (model.pth)، ويتم نقل النموذج إلى المحدد (device) وحدة المعالجة المركزية أو وحدة معالجة الرسومات. أخيراً، تم ضبط النموذج على وضع التقييم (model.eval()).

2. **تحميل مجموعة بيانات الاختبار**: يتم تحميل مجموعة بيانات الاختبار باستخدام "مجموعات البيانات. مجلد الصور" ، الذي يفترض أن الصور منتظمة في مجلدات منفصلة لكل فئة. يتم تطبيق التحويل المحدد ("التحويل") على الصور. يتم إنشاء أداة تحميل بيانات لمجموعة بيانات الاختبار ، باستخدام حجم دفعه من 1 وخلط البيانات.

3. **الاستدلال على صور الاختبار** : تقوم الشفرة بالاستدلال على مجموعة بيانات الاختبار عن طريق التكرار فوق محمل الاختبار. لكل صورة يتم تنفيذ الخطوات التالية:

- يتم نقل الصورة إلى device (الصور = الصور.إلى(device)) .
- يتم تنفيذ الانتشار الأمامي على النموذج المحمل للحصول على المخرجات المتوقعة (المخرجات = نموذج_محمل (صور)).
- يتم تحديد الفئة المتوقعة من خلال إيجاد مؤشر أقصى قيمة خرج (_ ، متوقعة = torch.max (outputs.data) ، (1)).
- يتم إلهاق الصورة وتسمية الفئة المتوقعة باسم الفئة المقابل بقائمة "test_images" و "test_labels".

4. عرض صور الاختبار : يتكرر الكود فوق صور الاختبار المحددة ويعرضها واحدة تلو الأخرى. يتم عرض الصورة وتسمية الفئة المتوقعة باستخدام plt.title و plt.imshow ، على التوالي. تم إيقاف تشغيل المحاور ("plt.axis (" off ")") ، ويتم عرض الصورة باستخدام plt.show .
يسمح لك هذا الكود بتحميل نموذج محفوظ، وإجراء الاستدلال على صور الاختبار، وتصور النتائج. يمكن أن يكون مفيداً للتحقق من أداء النموذج على البيانات غير المرئية واكتساب رؤى حول قراراته التنبؤية



صورة رقم (4)

خدمة تطبيق الويب.

يمكن إنشاء خدمة ويب للكشف عن سرطان الرئة باستخدام حزمة Gradio للاستدلال على النموذج النهائي. تتيح خدمة الويب هذه للمستخدمين تحميل صور الاشعة او الصور المقطعة لرئة المريض ، والتي تتم معالجتها بعد ذلك بواسطة النموذج المدرب للت辨ؤ باحتمالية الإصابة بسرطان الرئة . يتم عرض النتائج على المستخدم وتزويده بمعلومات قيمة عن وجود أو عدم وجود سرطان الرئة .

• الكود البرمجي.

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.models as models
import gradio as gr
from PIL import Image
# Load the saved model for inference
loaded_model = models.resnet50(pretrained=False)
num_classes = 3 # Replace with the actual number of classes in your dataset
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
loaded_model = loaded_model.to(device)
loaded_model.eval()
# Define the transformation applied to each image
transform = transforms.Compose([
    ...]
```

```

transforms.Resize(256),
transforms.CenterCrop(224),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])))

# Define a function for performing inference on the input image
def classify_image(input_image):
    img = Image.fromarray(input_image.astype('uint8'), 'RGB')
    img_tensor = transform(img).unsqueeze(0).to(device)
    outputs = loaded_model(img_tensor)
    _, predicted = torch.max(outputs.data, 1)
    label = predicted.item()
    class_names = ['lung_n', 'lung_aca', 'lung_sec']

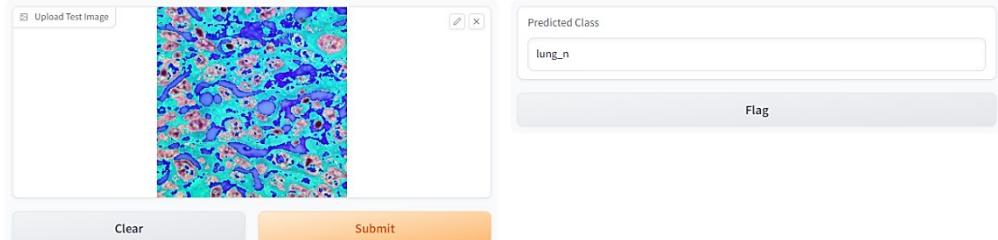
    class_name = class_names[label]
    return class_name

# Create a Gradio interface
image_input = gr.inputs.Image(type="numpy", label="Upload Test Image")
label_output = gr.outputs.Textbox(label="Predicted Class")
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output)
iface.launch(share=True)

```

ملاحظة:-

يرجى التأكد من مسارات موقع النموذج ان كنت تعمل في الحاسبة الشخصية. وتنفيذ الامر في سطر الاوامر بعد خزنء باي اسم (python lung_app.py).... ليظهر لك رابط لعنوان محلي انسخ الرابط في اي متصفح لنرى النتائج كما في الصورة.



صورة رقم (5)

رمز الاستجابة السريع



الملخص

في الختام، نستطيع ان نلخص ما تم شرحه في الكود البرمجي لتصنيف سرطان الرئة باستخدام بنية الشبكة المصصبة PyTorch ResNet-50. بال نقاط التالية:-

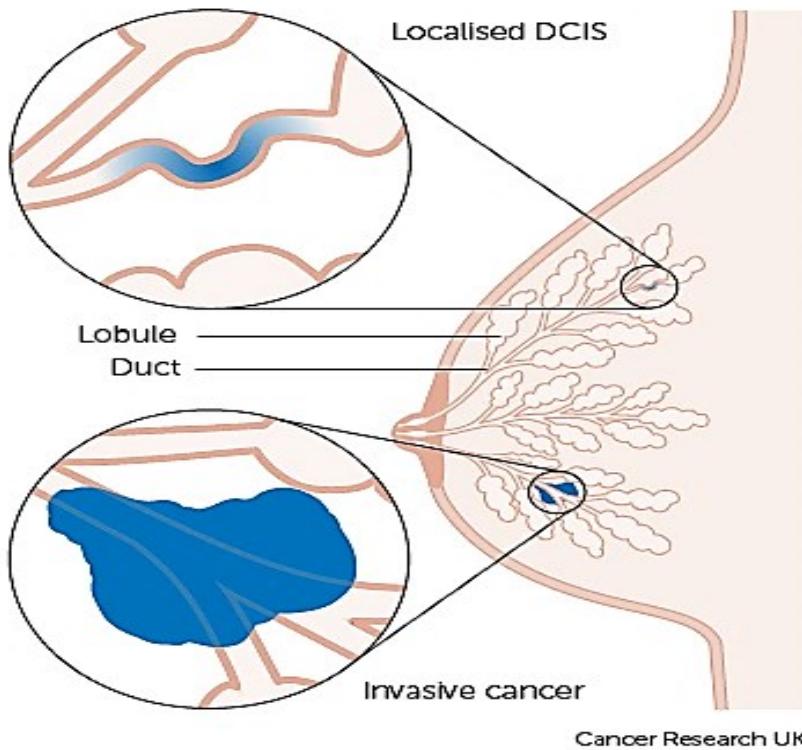
1. **هندسة النموذج:** يستخدم نموذج ResNet-50 كعمود أساسى لتصنيف سرطان الرئة. تم تعديل آخر طبقة متصلة بالكامل لتناسب مع عدد الفئات فى مجموعة البيانات.
2. **إعداد البيانات:** يعالج الكود إعداد مجموعة البيانات من خلال تطبيق التحولات المناسبة مثل تغيير الحجم والقص المركزي وتحويل الصور الى موترات وتطبيعها. يتم تقسيم مجموعة البيانات إلى مجموعات التدريب والتحقق والاختبار.
3. **التدريب:** يتم تدريب النموذج باستخدام حلقة تدريب تتكرر على مجموعة بيانات التدريب. وظيفة الخسارة المستخدمة هي فقدان الانتروبيا، ويتم استخدام محسن آدم لتحديث المعلمات.
4. **التقييم:** يتم تقييم أداء النموذج بناءً على مجموعة التحقق من خلال حساب الخسارة والدقة. يتم تسجيل خسائر التدريب والتحقق من الصحة لكل حقبة.
5. **مصفوفة الارباك وتقرير التصنيف:** يتم إنشاء مصفوفة الارباك وتقرير التصنيف لتقييم أداء النموذج في مجموعة التحقق من الصحة. توفر مصفوفة الارباك رؤى حول التسميات المتوقعة والحقيقة للنموذج، بينما يعرض تقرير التصنيف مقاييس مثل الدقة والاستدعاء ودرجة F1 والدعم لكل فئة.
6. **الاستدلال:** يتم تحميل النموذج المحفوظ للاستدلال، ويتم تمرير صور الاختبار عبر النموذج للتنبؤ بفئة سرطان الرئة. يتم عرض بعض صور الاختبار المحددة وفاتها المتوقعة.

بشكل عام، يوفر هذا الكود خط أنابيب شامل لتصنيف سرطان الرئة باستخدام نموذج ResNet-50 في PyTorch. ويعطي معالجة البيانات، وتدريب النموذج، والتقييم، والاستدلال، مما يسمح بالتنبؤات الدقيقة والتحليل لفئات سرطان الرئة.

الفصل العاشر
تصنيف سرطان الأقنية الغازية
Invasive Ductal Carcinoma Classification

- المقدمة.
- ما هو نقل التعلم؟.
- . ResNet-18 model
- البدء في تدريب النموذج.
- تطبيق ويب باستخدام مكتبة Gradio
- ال코드 البرمجي النهائي.
- الملخص.

ما هو سرطان الأقنية الغازية؟ سرطان الأقنية هو نوع شائع من سرطان الثدي يبدأ في الخلايا التي تبطن قنوات الحليب ، والتي تحمل حليب الثدي إلى الحلمة. هناك نوعان: سرطان الأقنية الغازية (IDC) و سرطان الأقنية الموضعي (DCIS) ، ويسمى أيضًا سرطان داخل القناة.



رقم (1)

+ ما هو نقل التعلم؟

ماذا يقصد بالتعلم عن طريق النقل ؟ هي طريقة تعلم الآلة حيث يتم إعادة استخدام نموذج تم تطويره مسبقاً لمهمة ما كنقطة بداية لنموذج في مهمة ثانية. إنه نهج وتقنيّة شائعة في التعلم العميق حيث يتم استخدام النماذج المدربة مسبقاً كنقطة انطلاق لمهام في مجال تطبيقات رؤية الكمبيوتر وفي مهام وتطبيقات معالجة اللغة الطبيعية نظراً لموارد الحوسبة والوقت الهائلة المطلوبة لتطوير نماذج الشبكة العصبية بشأن هذه المشكلات ومن القفزات الهائلة في المهارات التي يقدمونها بشأن المشكلات ذات الصلة.

+ نموذج .ResNet-18 model

بعد استخدام نموذج ResNet-18 لتصنيف سرطان الأقنية الغازية (IDC) طريقة واعدة في تحليل الصور الطبية. ResNet-18 عبارة عن بنية شبكة عصبية تلافيفية (CNN) معروفة بفعاليتها في مختلف مهام التعرف على الصور، بما في ذلك تحليل الصور الطبية.

إليك كيفية استخدام ResNet-18 لتصنیف IDC:

1. **إعداد مجموعة البيانات:** احصل على مجموعة بيانات من الصور الطبية التي تحتوي على عينات من حالات IDC والحالات غير IDC. تأكد من تسمية مجموعة البيانات بدقة.
2. **المعالجة المسبقة:** معالجة الصور مسبقاً للتأكد من أنها ذات حجم وتنسيق موحدين. تتضمن خطوات المعالجة المسبقة الشائعة تغيير الحجم والتقطيع والزيادة لتعزيز قدرة تعليم النموذج.
3. **بنية النموذج:** قم بتنفيذ بنية ResNet-18 باستخدام إطار عمل التعلم العميق مثل TensorFlow أو PyTorch. يتكون ResNet-18 من عدة طبقات تلفيفية مع اتصالات متبقية، مما يجعلها أعمق مع تخفيض مشكلة التدرج المتلاشي.
4. **نقل التعلم:** نظراً لأن مجموعات بيانات الصور الطبية غالباً ما تكون صغيرة، فإن الضبط الدقيق لنموذج ResNet-18 الذي تم تدريبيه مسبقاً يعد أسلوبًا شائعًا. يتضمن نقل التعلم استخدام نموذج تم تدريبيه مسبقاً (يتم تدريبيه عادةً على مجموعة بيانات كبيرة مثل ImageNet) وضبط معلماته في مجموعة بيانات IDC الخاصة بك. تساعد هذه العملية النموذج على تعلم الميزات ذات الصلة من صور IDC مع الاستفادة من المعرفة المكتسبة من النموذج المُدرب مسبقاً.
5. **التدريب:** قم بتقسيم مجموعة البيانات الخاصة بك إلى مجموعات تدريب وتحقق واختبار. قم بتدريب نموذج ResNet-18 على مجموعة التدريب، باستخدام تقنيات مثل نزول التدرج العشوائي (SGD) أو تحسين Adam. راقب أداء النموذج في مجموعة التحقق من الصحة لمنع التجهيز الزائد.
6. **التقييم:** قم بتقييم النموذج المُدرب في مجموعة الاختبار لتقييم مقاييس أدائه مثل الدقة والإحكام والاستدعاء ودرجة F1. بالإضافة إلى ذلك، قم بتحليل مصفوفة الارباك الخاصة بالنموذج لفهم أداء التصنيف الخاص به بالتفصيل.
7. **الضبط الدقيق والتحسين:** قم بتجربة المعلمات الفائقة، مثل معدل التعلم، وحجم الدفع، وإعدادات المحسن، لتحسين أداء النموذج بشكل أكبر. يمكن أن يؤدي ضبط بنية النموذج أو دمج تقنيات المجموعة أيضاً إلى تعزيز دقة التصنيف.
8. **النشر:** بمجرد رضاك عن أداء النموذج، قم بنشره في بيئة سريرية لتصنیف IDC. التأكد من أن عملية النشر تتوافق مع اللوائح والمعايير ذات الصلة بالبرامج الطبية.
9. **التحسين المستمر:** قم بتحديث النموذج وتحسينه باستمرار باستخدام بيانات ورؤى جديدة لتحسين أدائه والتكييف مع تحديات تصنیف IDC المتغيرة.

تذكر أنه عند التعامل مع البيانات الطبية، من الضروري ضمان خصوصية المريض والامتثال للإرشادات واللوائح الأخلاقية المتعلقة باستخدام البيانات ونشر النماذج. بالإضافة إلى ذلك، فإن إشراك الخبراء الطبيين في عملية تطوير النموذج يمكن أن يوفر رؤى قيمة وتحقق من الصحة.

البدء في تدريب النموذج.

• مجموعة البيانات.

بعد حصولك على مفتاح التخويل من منصة كاكل حان الان تنزيل مجموعة البيانات الصورية لسرطان الأقنية في الثدي.

Colab's file access feature
from google.colab import files

```
#retrieve uploaded file
uploaded = files.upload()
#print results
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))
# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d falahgatea/breast-ductalcarcinomadb
```

فك الضغط عن مجموعة البيانات وها نوين

```
!unzip /content/breast-ductalcarcinomadb.zip
!rm -r /content/breast-ductalcarcinomadb.zip
```

معرفة حجم البيانات.

```
#calculat number images
import os
def count_files(folder_path):
    file_count = 0
    for dirpath, dirnames, filenames in os.walk(folder_path):
        file_count += len(filenames)
    return file_count
# Specify the path to the folder you want to count the files in
folder_path = '/content/breast-ductalcarcinomadb'
total_files = count_files(folder_path)
print(f"Total number of files: {total_files}")
```

• استيراد المكتبات .

```
#importing libraries
from fastai import *
from fastai.vision import *
from fastai.metrics import error_rate
import os
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
```

```
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
import matplotlib.pyplot as plt
```

- **تطبيق النموذج على بيانات سرطان اقنية الثدي.**

```
# Load pre-trained ResNet-18 model
model = models.resnet18(pretrained=True)
# Modify the last layer to match the number of classes in your dataset
num_classes = 2 # Replace with the actual number of classes in your dataset
model.fc = nn.Linear(model.fc.in_features, num_classes)
# Set the device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
```

- **تنظيم الصور.**

يقوم الكود التالي بتنظيم الصور من مجموعة البيانات في مجلدين، "class0" و "class1" استناداً إلى تسميات الفئة الخاصة بها "0" و "1". يقوم بالمسح من خلال دليل يحتوي على مجلدات فرعية تسمى "0" و "1"، وينسخ ملفات الصور (.png أو .jpg) من هذه المجلدات الفرعية إلى أدلة الإخراج المعنية.

```
import os
import shutil

# Define your dataset root directory
dataset_root = '/content/IDC_regular_ps50_idx5'
# Create output directories for class 0 and class 1 images
class0_images_dir = 'class0'
class1_images_dir = 'class1'
os.makedirs(class0_images_dir, exist_ok=True)
os.makedirs(class1_images_dir, exist_ok=True)
# Define the class subfolder names
class0_subfolder = '0'
class1_subfolder = '1'
# Iterate through subfolders in the dataset root
```

الفصل العاشر: تصنیف سرطان الأقنية الغازية

for root, dirs, files in os.walk(dataset_root):

 for dir in dirs:

 if dir == class0_subfolder:

 for file in os.listdir(os.path.join(root, dir)):

 if file.endswith('.jpg') or file.endswith('.png'):

 source_path = os.path.join(root, dir, file)

 destination_path = os.path.join(class0_images_dir, file)

 shutil.copy(source_path, destination_path)

 elif dir == class1_subfolder:

 for file in os.listdir(os.path.join(root, dir)):

 if file.endswith('.jpg') or file.endswith('.png'):

 source_path = os.path.join(root, dir, file)

 destination_path = os.path.join(class1_images_dir, file)

 shutil.copy(source_path, destination_path)

print("Images are organized into class0 and class1 folders.")

• المعالجة المسابقة قبل التدريب .

يقوم جزء الكود هذا بالمهام التالية:

1. **تحويل البيانات**: يحدد سلسلة من التحويلات التي سيتم تطبيقها على الصور، بما في ذلك تغيير الحجم والتحويل إلى موترات والتقطيع.
2. **تحميل مجموعة البيانات**: تحميل مجموعة بيانات الصورة من المسار المحدد وتطبيق التحويلات المحددة.
3. **تقسيم مجموعة البيانات**: يقسم مجموعة البيانات إلى مجموعات تدريب وتحقق واختبار بناءً على النسب المنوية المحددة.
4. **حفظ مجموعة بيانات الاختبار**: يحفظ الصور من مجموعة بيانات الاختبار في المجلدات الخاصة بها داخل دليل "الاختبار".
5. **الإخراج**: يطبع أسماء الفصول وأحجام مجموعات التدريب والتحقق من الصحة والاختبار. ويستخدم مكتبة **torchvision** للتعامل مع مجموعة البيانات والتحولات ومكتبة **PIL** لحفظ الصور.

```
import torchvision.datasets as datasets
from torchvision.transforms import transforms
import os
from PIL import Image

# Define the transformations for the dataset
transform = transforms.Compose([
    transforms.Resize((224, 224)), # Resize the images to a specific size
    transforms.ToTensor(),       # Convert images to tensors
```

الفصل العاشر: تصنیف سرطان الاقنية الغازية

```
Invasive Ductal Carcinoma Classification  
transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize the images  
]  
  
# Load the original dataset  
dataset_path = "/content/dataset"  
train_dataset = datasets.ImageFolder(dataset_path, transform=transform)  
  
# Get the class names  
class_names = train_dataset.classes  
print("Class Names:", class_names)  
  
# Define the percentage of data for training, validation, and test sets  
train_split = 0.8 # 80% for training  
val_split = 0.1 # 10% for validation  
test_split = 0.1 # 10% for testing  
  
# Calculate the sizes of each split  
dataset_size = len(train_dataset)  
train_size = int(train_split * dataset_size)  
val_size = int(val_split * dataset_size)  
test_size = dataset_size - train_size - val_size  
  
# Split the dataset  
train_dataset, val_dataset, test_dataset = torch.utils.data.random_split(train_dataset,  
[train_size, val_size, test_size])  
  
# Save the test dataset to the "test" folder  
test_folder = "/content/dataset/test"  
os.makedirs(test_folder, exist_ok=True) # Create the folder if it doesn't exist  
for i, (image, label) in enumerate(test_dataset):  
    class_name = class_names[label]  
  
    class_folder = os.path.join(test_folder, class_name)  
    os.makedirs(class_folder, exist_ok=True) # Create class folder if it doesn't exist  
  
    filename = f"image_{i}.png" # Generate a unique filename with the .tif extension  
    image = transforms.ToPILImage()(image) # Convert tensor to PIL Image  
    image.save(os.path.join(class_folder, filename))  
  
# Print the sizes of each split  
print(f"Train set size: {len(train_dataset)}")  
print(f"Validation set size: {len(val_dataset)}")  
print(f"Test set size: {len(test_dataset)}")
```

بعد التنفيذ ظهر لنا النتائج كالتالي :-

Class Names: ['class0', 'class1']

Train set size: 222019

Validation set size: 27752

Test set size: 27753

• الكود البرمجي التالي:-

يحدد جزء الكود هذا أدوات تحميل البيانات لمجموعات بيانات التدريب والتحقق من الصحة. يتم استخدام أدوات تحميل البيانات لتجمیع البيانات وخلطها أثناء التدريب والتحقق من الصحة.

- **حجم الدفعه:** يحدد عدد العينات في كل دفعه.
- **Train Loader:** `DataLoader` لمجموعة بيانات التدريب مع تمكين التبديل العشوائي `(shuffle=True)`.
- **Validation Loader:** `DataLoader` لمجموعة بيانات التحقق من الصحة مع تعطيل التبديل `(shuffle=False)`.

يمكن استخدام أدوات تحميل البيانات هذه في حلقات التدريب للتكرار على دفعات من البيانات للتدريب التمودجي والتحقق من صحته.

```
# Define the data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
```

• الكود البرمجي التالي:-

يحدد جزء الكود هذا وظيفة الخسارة والمحسن لتدريب نموذج الشبكة العصبية.

- **وظيفة الخسارة:** يتم استخدام `nn.CrossEntropyLoss()` ، وهي مناسبة لمهام التصنيف ذات الفئات المتعددة.

- **المُحسن:** يُستخدم `optim.Adam` كخوارزمية التحسين. يقوم بتحديث معلمات النموذج بناءً على التدرجات المحسوبة أثناء الانتشار العكسي. تم ضبط معدل التعلم (`lr`) على `0.001`.

تعتبر هذه المكونات ضرورية لتدريب نموذج باستخدام التعلم الخاضع للإشراف، حيث يكون الهدف هو تقليل وظيفة الخسارة عن طريق ضبط معلمات النموذج باستخدام المُحسن.

```
# Define the loss function and optimizer
```

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

• الكود البرمجي التالي:-

يحدد جزء الكود هذا حلقة تدريب لتدريب نموذج شبكة عصبية لعدد محدد من الحقبات (num_epochs). وفيما يلي تفصيل لما يفعله:

- **وضع التدريب:** اضبط النموذج على وضع التدريب باستخدام().

- **مرحلة التدريب:**

- التكرار عبر دفعات من البيانات من "أداة تحميل التدريب".

- نقل البيانات إلى الجهاز المحدد (على سبيل المثال، وحدة المعالجة المركزية أو وحدة معالجة الرسومات).

- صفر التدرجات باستخدام().

- التمرير الأمامي: حساب تنبؤات النموذج باستخدام().

- حساب الخسارة باستخدام المعيار المحدد().

- التمرير للخلف: حساب التدرجات باستخدام().

- تحديث معلمات النموذج باستخدام().

- تجميع خسارة التشغيل للحقبة.

- **خسارة الحقبة:** احسب متوسط خسارة التدريب الحقبة وألحقه بـ "خسائر_التدريب".

- **وضع التقييم:** اضبط النموذج على وضع التقييم باستخدام().

- **مرحلة التحقق:**

- التكرار عبر دفعات من البيانات من val_loader.

- نقل البيانات إلى الجهاز المحدد.

- التمريرة الأمامية: حساب تنبؤات النموذج.

- حساب عدد التنبؤات الصحيحة وإجمالي التوقعات لحساب الدقة.

- حساب خسارة التحقق من الصحة.

- احسب دقة التتحقق وألحقها بـ val_accuracy.

- إلّا حاًل خسارة التتحقق بـ val_losses.

- اطبع رقم الحقبة وخسارة التدريب ودقة التتحقق لكل حقبة.

تقوم حلقة التدريب هذه بتدريب النموذج، وتقييمه على مجموعة التحقق من الصحة بعد كل فترة، وطباعة تقدم التدريب. يتم جمع فقدان التدريب ودقة التتحقق من الصحة لمزيد من التحليل أو التصور.

```
# Training loop
```

```
num_epochs = 10
```

```
train_losses = []
```

```

val_losses = []
val_accuracies = []
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)
    epoch_loss = running_loss / len(train_dataset)
    train_losses.append(epoch_loss)
    model.eval() # Set the model to evaluation mode
    correct_predictions = 0
    total_predictions = 0

```

```

with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total_predictions += labels.size(0)
        correct_predictions += (predicted == labels).sum().item()
    val_accuracy = correct_predictions / total_predictions
    val_accuracies.append(val_accuracy)
    val_loss = criterion(outputs, labels)
    val_losses.append(val_loss.item())
print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {epoch_loss:.4f}, Validation
Accuracy: {val_accuracy:.4f}")

```

• الكود البرمجي التالي :-

يحفظ جزء الكود هذا قاموس حالة النموذج المدرب في ملف يسمى "model.pth" باستخدام وظيفة PyTorch الخاصة بـ `torch.save()`. يحتوي قاموس الحالة على الاوزان التي تم تعلمها للنموذج، والتي يمكن تحميلها لاحقاً لإعادة إنشاء النموذج أو مواصلة التدريب.

```
# Save the trained model
```

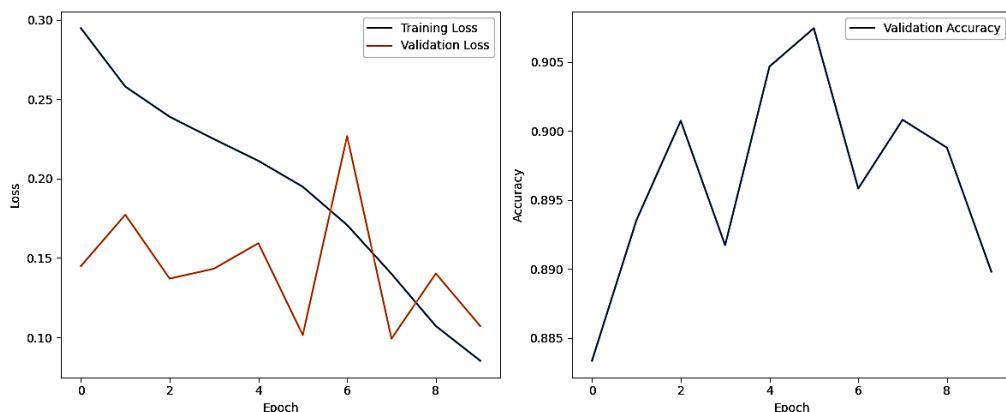
```
torch.save(model.state_dict(), "model.pth")
```

• الكود البرمجي التالي :-

يستخدم كود Python هذا `Matplotlib` لإنشاء شكل ذو مخططين فرعيين. ترسم الفرعية الأولى فقدان التدريب والتحقق من الصحة على مدى الحقبات، بينما الرسم الفرعية الثاني يمثل دقة التحقق على مدى الحقبات (epochs).

```
# Plotting the results
```

```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.tight_layout()
plt.show()
```



صورة رقم (2)

• الكود البرمجي التالي.

يسرد هذا الكود البرمجي أسماء الفئات الموجودة في دليل مجموعة البيانات (content/BreastCancerDB/) وبخزنها في المتغير class_names. يتكرر خلال كل دليل في دليل مجموعة البيانات ويلحق أسماء الدليل (أسماء الفئات) بالقائمة "أسماء_الفئات".

```
#the classes name list
dataset_path="/content/BreastCancerDB"
class_names = os.listdir(dataset_path)
for class_name in class_names:
    class_path = os.path.join(dataset_path, class_name)
    files = os.listdir(class_path)
print(class_names)
```

• الكود البرمجي التالي.

يحسب هذا الكود مصفوفة الارباك لنموذج التصنیف باستخدام مکتبة scikit-Learn

وهنا شرح موجز:-

1. قم باستيراد الدالة confusion_matrix من الوحدة النمطية sklearn.metrics.
 2. قم بتهيئة القوائم الفارغة all_labels و all_predictions لتخزين التسمیات المتوقعة والتسمیات الحقيقة، على التوالي.
 3. اضبط نموذج PyTorch على وضع التقيیم باستخدام "().model.eval" .
 4. قم بتعطیل حساب التدرج لتحقيق الكفاءة باستخدام ".0.torch.no_grad" .
 5. قم بالتکرار من خلال أداة تحمیل بيانات التحقق (val_loader) للحصول على مجموعات من الصور والتسمیات الحقيقة المقابلة لها.
 6. انقل الصور والتسمیات إلى المعالج المناسب (على سبيل المثال، GPU).
 7. قم بتمریر الصور عبر النموذج للحصول على المخرجات.
 8. قم بحساب التسمیات المتوقعة عن طريق أخذ القيمة الفصوی على طول البعد الثاني لموتور الإخراج.
 9. قم بتوسيع قائمتي "جميع_التنبؤات" و "جميع_التسمیات" بالتسمیات المتوقعة والصیحة، على التوالي.
 10. بعد تکرار جميع الدفعات، قم بحساب مصفوفة الارباك باستخدام .confusion_matrix(all_labels, all_predictions)
 11. اطبع مصفوفة الارباك.
- يساعد هذا الكود البرمجي في تقيیم أداء نموذج التصنیف من خلال تحلیل توزیع التسمیات المتوقعة والحقيقة عبر فئات مختلفة.

```
#confusion matrix
from sklearn.metrics import confusion_matrix
# Initialize variables for storing the predictions and true labels
all_predictions = []
all_labels = []
# Set the model to evaluation mode
```

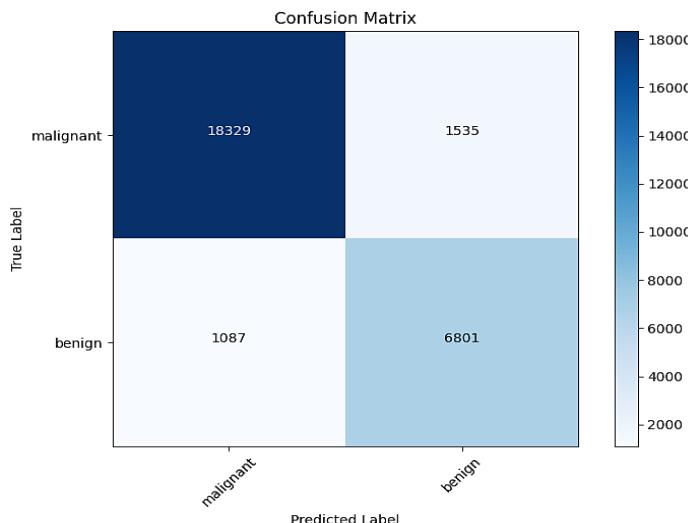
```

model.eval()
# Disable gradient calculation for efficiency
with torch.no_grad():
    for images, labels in val_loader:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        # Append the predicted labels and true labels to the respective lists
        all_predictions.extend(predicted.tolist())
        all_labels.extend(labels.tolist())
# Calculate the confusion matrix
cm = confusion_matrix(all_labels, all_predictions)
print("Confusion Matrix:")
print(cm)

```

ملاحظة:-

بامكانك اظهار مصفوفة الارباق وكما مذكورة في الكود البرمجي الرئيسي لعدم ذكره هنا بسبب تكراره سابقا وتنفيذ لترى النتائج كالاتي:-



صورة رقم (3)

• الكود البرمجي التالي :-

يوضح الكود هذا تحميل نموذج ResNet18 المدرب مسبقاً للاستدلال كعمل تطبيق وتشغيله وتجربته في الحاسبة الشخصية واستخدامه للتنبؤ بفنات بعض الصور من مجموعة بيانات اختبارية. وفيما يلي ملخص موجز:

1. **تحميل النموذج المحفوظ:** يقوم الكود بتحميل نموذج ResNet50 الذي تم تدريبيه مسبقاً بدون الطبقة العليا (الطبقة المتصلة بالكامل) باستخدام pretrained=False. ثم تقوم باستبدال models.resnet50(pretrained=False). ثم تقوم باستبدال الطبقة الأخيرة المتصلة بالكامل بطبقة جديدة مناسبة لعدد الفنات في مجموعة البيانات. يتم تحميل قاموس حالة النموذج من ملف يسمى "model.pth".
2. **إعداد مجموعة بيانات الاختبار والمحمل:** يقوم بإعداد مجموعة بيانات الاختبار وأداة تحميل البيانات باستخدام datasets.ImageFolder لتحميل الصور من الدليل "/content/dataset/test".
3. **الاستدلال على صور الاختبار:** يتكرر الكود من خلال مجموعة بيانات الاختبار لإجراء الاستدلال. ولكن صورة، تمررها عبر النموذج المحمل للحصول على التنبؤات. ثم يقوم باستخراج تسميات الفناء المتوقعة، إلى جانب الصور المقابلة، حتى يجمع 5 صور مع تسمياتها المتوقعة.
4. **عرض صور الاختبار بالتسميات المتوقعة:** أخيراً، يعرض الكود صور الاختبار المحددة واحدة تلو الأخرى مع تسميات الفناء المتوقعة باستخدام Matplotlib.

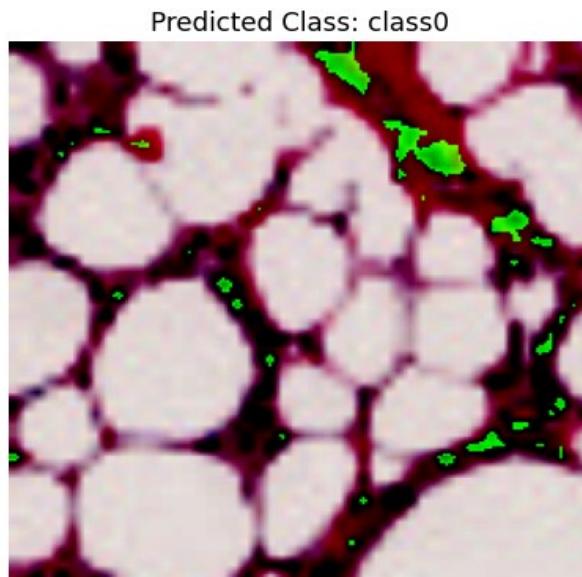
بعد مقطع التعليمات البرمجية هذا مفيداً لاختبار النموذج المدرب على البيانات غير المرئية وفحص أدائه بصرياً.

```
## Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("/content/model.pth"))
loaded_model = loaded_model.to(device)
loaded_model.eval()

# Load your test dataset
test_dataset = datasets.ImageFolder("/content/dataset/test", transform=transform)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1, shuffle=True)

# Select 5 images for testing
test_images = []
test_labels = []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        outputs = loaded_model(images)
        _, predicted = torch.max(outputs.data, 1)
        for i in range(images.size(0)):
```

```
image = images[i].permute(1, 2, 0).cpu().numpy()
label = predicted[i].item()
class_name = test_dataset.classes[label]
test_images.append(image)
test_labels.append(class_name)
if len(test_images) >= 5:
    break
if len(test_images) >= 5:
    break
# Display the selected test images
for i in range(len(test_images)):
    image = test_images[i]
    label = test_labels[i]
    plt.imshow(image)
    plt.title(f"Predicted Class: {label}")
    plt.axis("off")
    plt.show()
```



صورة رقم (4)
حيث تظهر الصورة باحتواها على سرطان الاقنية

+تطبيق ويب باستخدام مكتبة Gradio

ما هو Gradio؟ هي مكتبة Python تتيح لك إنشاء واجهات مستخدم (UIs) لنماذج التعلم الآلي بسرعة. فهو يبسط عملية إنشاء واجهات ويب تفاعلية لنماذجك، مما يتيح سهولة النشر والمشاركة مع الآخرين. يدعم Gradio مجموعة متنوعة من أنواع الإدخال مثل الصور والنصوص والصوت، مما يجعله متعدد الاستخدامات لأنواع مختلفة من النماذج.

باستخدام Gradio، يمكنك إنشاء واجهات حيث يمكن للمستخدمين إدخال البيانات ورؤيتها تنبؤات النموذج أو مخرجاته على الفور. فهو يوفر واجهة برمجة تطبيقات بسيطة لتحديد المدخلات والمخرجات والنماذج الأساسية، مما يزيل تعقيدات تطوير الويب.

• فيما يلي نظرة عامة مختصرة على ميزات Gradio الرئيسية:

1. سهولة التكامل: يتكامل Gradio بسلاسة مع إطار عمل التعلم الآلي الشائعة مثل TensorFlow وPyTorch وscikit-learn، مما يسمح لك بتنقل نماذجك بأقل قدر من التغييرات في التعليمات البرمجية.

2. مجموعة واسعة من أنواع الإدخال: يدعم Gradio أنواع الإدخال المختلفة مثل النصوص والصور والصوت والفيديو، مما يجعله مناسباً لمجموعة واسعة من مهام التعلم الآلي.

3. التفاعل في الوقت الفعلي: يمكن للمستخدمين التفاعل مع النموذج في الوقت الفعلي من خلال واجهة الويب، مما يجعله مثالياً للعروض التوضيحية والاختبار والأغراض التعليمية.

4. التخصيص: يقدم Gradio خيارات تخصيص لتصميم واجهة المستخدم، وتحديد واجهات الإدخال/الإخراج، وتكون التخطيط ليناسب تفضيلاتك.

5. النشر: بمجرد إنشاء واجهة المستخدم الخاصة بك باستخدام Gradio، يمكنك نشرها بسهولة لمشاركتها مع الآخرين أو دمجها في تطبيقات الويب الموجودة.

بشكل عام، Gradio هي أداة قوية لإنشاء نماذج أولية ومشاركة نماذج التعلم الآلي بسرعة من خلال واجهة بديهية وتفاعلية، دون الحاجة إلى معرفة واسعة بتطوير الويب.

• التنصيب.

عند نافذة سطر الأوامر نكتب الامر التالي:-

ملاحظة :-

إذا كان عملك في الحاسبة الشخصية يستخدم الأمر بدون وضع علامة التعجب أما إذا كان عملك على الخادم السحابي كوكل كولاب نستخدم الأمر التالي :

```
!pip install gradio
```

• الكود البرمجي التالي :-

يقوم هذا الكود بإنشاء واجهة ويب باستخدام Gradio لتصنيف الصور باستخدام نموذج ResNet18 المدرب مسبقاً في PyTorch. والذي تم حفظه أما في الخادم السحابي او في الحاسبة الشخصية.

1. استيراد المكتبات :

- يتم استيراد `torchvision.models` و `torchvision.transforms` و `torch.nn` و `torch` و `PyTorch` لوظائف التعلم العميق.
- يتم استيراد `gradio` لإنشاء واجهة الويب.
- يتم استيراد "الصورة" من `PIL` (مكتبة تصوير Python) لمعالجة الصور.

2. نموذج التحميل :

- يتم تحميل نموذج `ResNet18` الذي تم تدريبه مسبقاً وتكونيه للاستدلال.
- يتم استبدال الطبقة النهاية المتصلة بالكامل من النموذج بطبقة خطية جديدة ذات فترين للمخرجات.
- يتم تحميل أوزان النموذج من ملف `(model.pth)` وتعيينها إلى وحدة المعالجة المركزية.

3. تحويل الصور:

- يتم تحديد سلسلة من التحويلات باستخدام `torchvision.transforms.Compose` للمعالجة المسبقة للصورة المدخلة قبل تمريرها إلى النموذج.
- تتضمن هذه التحويلات تغيير الحجم، والاقتصاص المركزي، والتحويل إلى موتر، وتطبيع الصورة.

4. وظيفة الاستدلال:

- تم تعريف وظيفة `classify_image` لإجراء الاستدلال على الصورة المدخلة.
- يتم تحويل صورة الإدخال، الممثلة كمصفوفة `NumPy`، إلى صورة `PIL`.
- يتم تحويل الصورة باستخدام التحويلات المحددة وتمريرها عبر النموذج المحمول.
- يتم الحصول على تسمية الفن الموقعة وتعيينها باسم فن باستخدام "`فن اسم الفن`" المتوفرة.

5. واجهة المستخدم :User Interface

- يتم إنشاء مكون إدخال `image_input` لتحميل صور الاختبار.
- يتم إنشاء مكون الإخراج `label_output` لعرض تسمية الفن الموقعة.
- يتم إنشاء واجهة `iface` باستخدام وظيفة الاستدلال ومكونات الإدخال والإخراج والعنوان.
- يتم تشغيل الواجهة باستخدام `iface.launch()`، مما يسمح للمستخدمين بتحميل الصور ورؤية تسمية الفن الموقعة.

6. معلمات التشغيل:

- يمكّن وظيفة المشاركة، مما يسمح لك بمشاركة الواجهة مع الآخرين عبر عنوان `.URL`.
- يمكّن وضع التصحيح، الذي يوفر رسائل خطأ مفصلة في حالة حدوث أية مشكلات أثناء تنفيذ الواجهة.

ينشئ هذا الكود البرمجي واجهة ويب سهلة الاستخدام حيث يمكن للمستخدمين تحميل الصور والحصول على تنبؤات في الوقت الفعلي من نموذج `ResNet18` المدرب مسبقاً. إنه تطبيق عمل لمهام تصنیف الصور مثل تصنیف سرطان الأقنية الغازية. ويفضل العمل به في الحاسبة الشخصية بعد حفظ وخزن النموذج المدرب المسبق والتأكد من مسارات الحفظ. وتسمية الملف بـ `(app.py)` وعند سطر الأوامر كتابة الامر التالي:

(python app.py)

```

import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.models as models
import gradio as gr
from PIL import Image

# Load the saved model for inference
loaded_model = models.resnet18(pretrained=False)
num_classes = 2 # Assuming 2 classes based on class_names provided
loaded_model.fc = nn.Linear(loaded_model.fc.in_features, num_classes)
loaded_model.load_state_dict(torch.load("model.pth",
map_location=torch.device('cpu')))
loaded_model = loaded_model.eval()

# Define the transformation applied to each image
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) ])

# Define function for performing inference on the input image
def classify_image(img):
    img = Image.fromarray(img) # Convert NumPy array to PIL Image

    img_tensor = transform(img).unsqueeze(0)

    outputs = loaded_model(img_tensor)
    _, predicted = torch.max(outputs.data, 1)
    label = predicted.item()
    class_names = ['class0', 'class1']
    class_name = class_names[label]
    return class_name

# Create a Gradio interface
image_input = gr.Image(label="Upload Test Image")
label_output = gr.Textbox()
iface = gr.Interface(fn=classify_image, inputs=image_input, outputs=label_output,
title=" Invasive Ductal Carcinoma Classification ")

```

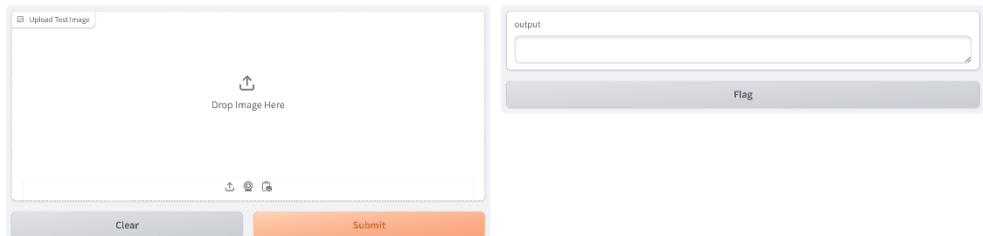
الفصل العاشر: تصنیف سرطان الاقنية الغازية

```
iface.launch(share=True,debug = True)
```

ملاحظة :-

عند تنفيذ الكود البرمجي اعلاه سوف يظهر رابط عنوان الويب لسيرفر معين عند كوكل كولاب او رابط عنوان محلي اذا كنت تعمل في الحاسبة الشخصية انسخ الرابط في اي متصفح لظهور لك الواجهة كما في الصورة أدناه قم برفع صورة لمرض سرطان الاقنية غير مدربة مسبقاً لكي ترى النتيجة المتوقعة النهائية ...

Invasive Ductal Carcinoma Classification



صورة رقم (5)

او كود واجهة المستخدم

الكود النهائي.



الملخص

في الختام، قدم فصل كتابنا هذا حلًا شاملًا لتصنيف سرطان الأقنية الغازية (IDC) باستخدام تقنيات التعلم العميق ونقل التعلم ، وتحديداً باستخدام نموذج **ResNet-18** الذي تم تدريسيه على صور تصوير الثدي. خلال هذا الفصل، قمنا بتغطية جوانب مختلفة من الحل، بما في ذلك التدريب النموذجي والتقييم والنشر عبر واجهة ويب سهلة الاستخدام.

أولاً، بحثنا في أهمية تصنیف IDC في تشخيص سرطان الثدي، مع تسلیط الضوء على الحاجة إلى طرق دقيقة وفعالة لمساعدة المتخصصين في الرعاية الصحية في هذه المهمة الحاسمة.

بعد ذلك، استكشفنا عملية التدريب النموذجية، حيث استخدمنا نقل التعلم باستخدام بنية **ResNet-18**، التي تم تدريبيها مسبقاً على ImageNet، للتكيف مع مهمة تصنیف IDC المحددة لدينا. من خلال ضبط النموذج على مجموعة بيانات مناسبة، كنا نهدف إلى التقاط ميزات ذات معنى من صور تصوير الثدي وتحسين أداء النموذج.

قمنا بتقييم النموذج المُدرِّب باستخدام مقاييس مثل الدقة ودرجات التحقق المتبادل، مما يوفر نظرة ثاقبة حول أدائه وقدرته على التعميم. تعمل هذه المقاييس كمؤشرات لموثوقية النموذج وفعاليته في عمل تنبؤات دقيقة.

وأخيراً، قمنا بنشر النموذج المُدرِّب باستخدام **Gradio**، وهي وحدة واجهة ويب سهلة الاستخدام للمستخدم ، مما يتيح تصنیف IDC في الوقت الفعلي للمهنيين الطبيين والباحثين. تتيح الواجهة للمستخدمين تحميل الصور وتلقي توقعات فورية، مما يسهل اتخاذ القرار السريع ويعزز كفاءة تشخيص سرطان الثدي.

بشكل عام، يقدم فصل كتابنا حلًا عمليًا ومؤثراً لتصنيف IDC، حيث يجمع بين قوة التعلم العميق وإمكانية الوصول إلى واجهات الويب. ونحن نعتقد أن نهجنا يحمل إمكانات كبيرة في تحسين تشخيص وعلاج سرطان الثدي، والمساهمة في نهاية المطاف في نتائج أفضل للمرضى وتعزيز البحث الطبي في هذا المجال الحيوي.

الفصل الحادي عشر الكشف عن سرطان العظام بتقنية Yolov8 Bone Tumor Detection By Using YOLV8

- المقدمة.
- ما هو نموذج Yolov8 ؟
- تطبيق على كشف سرطان العظام.
- ال코드 البرمجي للتدريب والتحقق.
- الاستدلال على صور الاختبار.
- تطبيق ويب باستخدام مكتبة Gradio.
- التطبيقات والافكار المستقبلية.
- ال코드 النهائي.
- الملخص.

المقدمة :-

ما هو مرض سرطان العظام ؟ في عالم الصحة والطب، يقف سرطان العظام كخصم هائل، حيث تعود جذوره إلى الهيكل العظمي للجسم البشري. تتميز هذه الحالة بالنمو غير الطبيعي للخلايا داخل الأنسجة العظمية، وتمثل تحدياً معقداً للأطباء والباحثين على حد سواء. من تأثيره المدمر المحتمل على سلامة الهيكل العظمي إلى قدرته على انتشار ورم خبيث، يتطلب سرطان العظام فحصاً يقظاً وأساليب مبتكرة للكشف والعلاج.

ضمن هذا المشهد، يكتسب البحث عن الكشف المبكر أهمية قصوى. إن تحديد سرطان العظام في الوقت المناسب لا يسهل التدخل الفوري فحسب، بل يحمل أيضاً القدرة على تحسين نتائج المرضى بشكل كبير. لقد كانت طرق التشخيص التقليدية، مثل الأشعة السينية، والأشعة المقطعة، والخزارات، بمثابة حلفاء أقوباء لفترة طويلة في هذا المعنى. ومع ذلك، في عصر التقدم التكنولوجي السريع، تظهر أدوات جديدة لتعزيز الأسلحة التشخيصية لدينا. ومن بين هذه الابتكارات، يمثل تكامل نماذج الذكاء الاصطناعي، وخاصة نموذج الكشف (YOLOv)، قفزة تحويلية إلى الأمام. يستخدم نموذج YOLOV قوة خوارزميات التعلم العميق لتحليل بيانات التصوير الطبي بسرعة ودقة غير مسبوقة. من خلال التحديد السريع للميزات الشاذة التي تشير إلى سرطان العظام في عمليات المسح الإشعاعي، تعمل أنظمة الذكاء الاصطناعي هذه على تمكين مقدمي الرعاية الصحية من خلال قدرات تشخيصية محسنة.

بينما نبدأ في استكشاف سرطان العظام باستخدام نماذج YOLOV، فإننا لا ننبعق في تعقيدات تقنية الذكاء الاصطناعي فحسب، بل أيضاً في الآثار العميقة المترتبة على رعاية المرضى. ومن خلال التعاون التأزرري بين المهنيين الطبيين ومطورى الذكاء الاصطناعى، نسعى إلى صياغة حدود جديدة في مكافحة سرطان العظام، إذاناً ببدء عصر الاكتشاف المبكر وتحسين التشخيص.



صورة رقم (1) قبل وبعد عملية الكشف

ما هو نموذج YOLOv8 ؟

هو أحدث نموذج في سلسلة خوارزمية - YOLO العائلة الأكثر شهرة للكشف المبكر عن الأشياء وتصنيفها في مجال رؤية الكمبيوتر (CV). مع الإصدار الأحدث، يستمر إرث YOLO من خلال توفير أحد النتائج لتحليلات الصور أو الفيديو، مع إطار عمل سهل التنفيذ. سنناقش في هذه المقالة: تطور خوارزميات YOLO التحسينات والتحسينات في تفاصيل تنفيذ YOLOv8 ونصائح التطبيقات.

ما هو YOLO ؟ أنت تنظر مرة واحدة فقط (YOLO) هي خوارزمية للكشف عن الأشياء تم تقديمها في عام 2015 في ورقة بحثية قام بها جوزيف ريدمون، وسانتوش ديفالا، وروس جيرشيك، وعلى فرهادي. كانت بنية YOLO بمثابة ثورة كبيرة في مجال الكشف عن الأشياء في الوقت الفعلى، متجاوزة سابقتها - الشبكة العصبية التلaffيفية القائمة على المنطقه YOLO (R-CNN). عبارة عن خوارزمية أحادية اللقطة تقوم بتصنیف کائن بشکل مباشر في مسار واحد من خلال وجود شبكة عصبية واحدة فقط تتبعاً بالمربيعات المحيطة واحتمالات الفئه باستخدام صوره كامله كمدخل. نموذج YOLO العائلي يتتطور باستمرار. أصدرت العديد من فرق البحث منذ ذلك الحين إصدارات مختلفة من YOLO ، وكان YOLOv8 هو الإصدار الأحدث (حالياً الإصدار 9).

• Ultralytics YOLOv8 +

أحدث إصدار من YOLO تم إصداره في يناير 2023. يتميز بدقة أعلى وسرعة أكبر. على سبيل المثال، حصل YOLOv8 (متوسط) على نتیجة mAP 50.2 عند 1.83 ملي ثانية على مجموعة بيانات COCO وأيضاً بجزمة YOLO v8 A100 TensorRT Python والتنفيذ المستند إلى CLI، مما يجعله سهل الاستخدام والتطوير. دعونا نلقي نظرة فاحصة على ما يمكن أن يفعله YOLOv8 ونستكشف بعضًا من تطوراته المهمة. النموذج المدرب مسبقًا YOLO v8 قادر على اكتشاف الكائنات في صورة أو فيديو مباشر مهام YOLOv8 يأتي YOLOv8 في خمسة أشكال مختلفة بناءً على عدد المعلمات - ناتو (n)، صغير (صغير)، متوسط (m)، كبير (l)، وكبير جدًا (x). يمكنك استخدام جميع المتغيرات للتصنیف واكتشاف الكائنات والتجزئة. تصنیف الصور يتضمن التصنیف التصنیف صورة بأكملها دون تحديد موضع الكائن الموجود داخل الصورة. يمكنك تنفيذ التصنیف باستخدام YOLOv8 عن طريق إضافة اللاحقة -cls إلى إصدار YOLOv8. على سبيل المثال، يمكنك استخدام yolov8n-cls.pt للتصنیف إذا كنت ترغب في استخدام إصدار الناتو.

• النماذج المتوفّرة في YOLOv8

توجد خمسة نماذج في كل فئة من نماذج YOLOv8 للكشف والتجزئة والتصنیف. YOLOv8 Nano هو الأسرع والأصغر، في حين أن (YOLOv8x) هو الأكثر دقة ولكنه الأبطأ فيما بينها.

YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
---------	---------	---------	---------	---------

صورة رقم(2)

حيث يقوم النموذج بكشف او اكتشاف الكائنات (الفئات) بتحديد موضع کائن داخل الصورة عن طريق رسم مربعات محیطة. ليس عليك إضافة أي لاحقة لاستخدام YOLOv8 للكشف. يتطلب التنفيذ فقط تحديد النموذج على أنه yolov8n.pt لاكتشاف الكائنات باستخدام متغير نوع النموذج المذكورة سابقاً. لكننا في هذا الفصل سوف لانتعق في البنية لهذا النموذج لانه خارج موضوع وسياق الكتاب وموضوع جداً مطول حول البنية الهيكلية بامكانك الاطلاع على الهيكلية في الورقة البحثية او البحث عنها في محرك البحث كوكل.

<https://arxiv.org/abs/2305.09972>

ستتعلم في هذا الفصل كيفية إنشاء مشروع لاكتشاف موقع مرض سرطان العظام في الركبة على مجموعة بيانات مخصصة، باستخدام أحدث تقنيات وخوارزمية (YOLOv8) الذي طورته (Ultralytics). سنستخدم تقنيات نقل التعلم لتدريب نموذجنا الخاص وتقديره وأدائها واستخدامه للاستدلال. طبعاً هذا التطبيق والمشروع العملي موجه للأشخاص الذين لديهم خلفية نظرية لخوارزميات اكتشاف الأشياء، والذين يسعون للحصول على ارشادات عملية للتنفيذ. يتم توفير محرر Jupiter سهل الاستخدام مع الرمز الكامل أدناه لراحتك. أو بإمكانك استخدام الخادم السحابي كوكل كولاب السريع والم مجاني وتم الحديث والكتابة عنه سابقاً في الفصول السابقة.

• تطبيق على كشف سرطان العظام.

• مجموعة البيانات.

بعد حصولنا وخذن مجموعة البيانات (راجع موضوع تنزيل مجموعة البيانات) أو ما تسمى Dataset سوف نبدأ بالخطوات البرمجية التالية:-

```
#yolov8 install pip
```

```
!pip install ultralytics
```

حيث يقوم هذا الكود بتنصيب مكتبة (Yolov8) لجميع النماذج المذكورة وهي كالتالي:-

- **Yolov8n**
- **Yolov8s**
- **Yolov8m**
- **Yolov8x**
- **Yolov8xl**

لكننا سوف نتعقب فقط في نموذج الكشف الصغير جداً (yolov8n.pt) وحاول بتجربة نماذج الكشف الأخرى لنرى الفرق والدقة والسرعة وحجم النموذج ومقاييس الدقة الأخرى.

• الكود البرمجي التالي:-

```
# upload Bone Tumor Dataset
from google.colab import files
# Prompt the user to upload a file
uploaded = files.upload()
# Loop through the uploaded files
for filename in uploaded.keys():
    print(f'Uploaded file: {filename}')
```

حيث يقوم الكود برفع البيانات المضغوطة والذي تم انزالها وخذنها في الحاسبة الشخصية والتي تحتوي على ثلاثة مجلدات وهي بيانات التدريب والتحقق والاختبار مع ملف التهيئة والذي يسمى بملفات (yaml) وهي جداً

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8
مهمة والتي تحدد مسارات المجلدات وعدد الفئات التي سوف يتم الكشف عنها واسماء الفئات كملخص لها ضمن اعدادات تدريب النموذج المخصص لنا في هذه الحالة لدينا فئة واحدة هي كشف السرطان بامكانك تحرير الملف وتعديل مسارات مجلدات الصور.

• الكود البرمجي التالي.

```
!unzip /content/bone_tumor.zip -d dataset  
!rm -r /content/bone_tumor.zip
```

وهو واضح فتح الضغط وخرن جميع المجلدات في مجلد الداتاسيت ومن ثم مسح المجلد المضغوط

• الكود البرمجي التالي.

حيث يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد وهو جدا مهم لمعرفة حجم مجموعة البيانات تحت التدريب.

```
import os  
  
def count_files_with_extension(folder_path, extension):  
    file_count = 0  
    for root, dirs, files in os.walk(folder_path):  
        for file in files:  
            _, file_extension = os.path.splitext(file)  
            if file_extension.lower() == extension:  
                file_count += 1  
  
    return file_count  
  
def main():  
    folders = ['train', 'valid', 'test']  
    image_extension = '.jpg'  
    for folder in folders:  
        folder_path = os.path.join(folder, 'images')  
        file_count = count_files_with_extension(folder_path, image_extension)  
        print(f"Folder: {folder}, Image Files: {file_count}")  
  
if __name__ == "__main__":  
    main()
```

بعد التنفيذ.

```
Folder: train, Image Files: 1880  
Folder: valid, Image Files: 89  
Folder: test, Image Files: 11
```

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8

• شرح الكود البرمجي بالتفصيل.

استيراد الوحدات المطلوبة:

تحديد الدالة `:count_files_with_extension`

تأخذ هذه الوظيفة وسيطتين `Folder_path` (مسار المجلد للبحث عن الملفات) والامتداد (امتداد الملف المراد حسابه). ويقوم بتهيئة عدد الملفات المتغير لتخزين عدد الملفات ذات الامتداد المحدد. ويستخدم الدالة `(os.walk()` للتنقل عبر كافة الأدلة والأدلة الفرعية الموجودة ضمن المجلد `_path` المحدد.

بالنسبة لكل ملف تم مواجهته، يقوم باستخراج امتداد الملف باستخدام `(os.path.splitext()` ومقارنته بالامتداد المقدم. إذا كانت متطابقة، فإنه يزيد `file_count`.

وأخيراً، تقوم بإرجاع العدد الإجمالي للملفات ذات الامتداد المحدد الموجود في المجلد.

تحديد الوظيفة الرئيسية:

تقوم هذه الوظيفة بتنسيق عملية العد لكل مجلد من المجلدات الرئيسية الثلاثة: "التدريب" و"التحقق" و"الاختبار".

يقوم بتهيئة قائمة المجلدات التي تحتوي على أسماء المجلدات الرئيسية.

بالنسبة لكل اسم مجلد في المجلدات، يقوم بإنشاء المسار إلى المجلد الفرعى "الصور" باستخدام `(os.path.join()`.

ثم يقوم بعد ذلك باستدعاء وظيفة `count_files_with_extension` بمسار المجلد الذي تم إنشاؤه وامتداد الصورة المحدد `('jpg')`.

وأخيراً، يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد.

يوفر هذا الكود البرمجي طريقة معيارية وقابلة للتطوير لحساب الملفات ذات الامتداد المحدد داخل مجلدات متعددة، مما يسهل المهام مثل المعالجة المسبقة للبيانات ولمشاريع التعلم الآلي والتعلم العميق ومعرفة عدد الصورة في كل مجلد مهم في التدريب .

• الكود البرمجي:.

```
#Read Dataset path for training  
!cat /content/dataset/data.yaml
```

• شرح الكود.

حالياً أنك تستخدم بينة Jupyter Notebook أو Google Colab للعمل مع مجموعة البيانات الخاصة بك.

يُستخدم الأمر `!cat` عادةً` في أنظمة التشغيل المشابهة لـ Unix وفي دفاتر ملاحظات Jupyter/Colab لعرض محتويات الملف.

دعنا نحلل ما يفطه هذا الأمر:

- يشير `!`` إلى أنه يجب تنفيذ الأمر التالي في غلاف النظام، وليس في Python .
- `cat`` هو أمر يستخدم لتسليسل وعرض محتويات الملفات.

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8
لذلك، عند تشغيل `cat /content/dataset/data.yaml`، فإنه يقرأ ويعرض محتويات الملف الموجود في `content/dataset/data.yaml`.

يحتوي الملف "data.yaml" على معلومات التكوين أو البيانات الوصفية حول مجموعة البيانات الخاصة في مشروعنا مثل المسارات إلى الصور، والتسميات المقابلة لها، وأي خطوات معالجة مسبقة، وما إلى ذلك. لظهور لنا نتيجة التنفيذ وهي فئة واحدة اسمها السرطان. وكما قلت سابقاً جداً مهم للحفاظ على المسارات وهي واضحة من ملف التكوين. (data.yaml).

```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 1
names: ['cancer']
```

• الكود البرمجي التالي:-

```
#import yolov8 detection modules lib.
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

• شرح الكود البرمجي.

سوف نقوم باستيراد وحدات الكشف عن YOLOv8 إلى جانب المكتبات الأخرى شائعة الاستخدام لرؤية الكمبيوتر وتحليل البيانات. فيما يلي تفاصيل كل بيان استيراد:

1. من **Ultralytics import YOLO**: يؤدي هذا إلى استيراد نموذج الكشف عن الكائنات YOLO من مكتبة Ultralytics. Ultralytics هي مكتبة توفر تطبيقات لنموذج التعلم العميق المختلفة لمهام رؤية الكمبيوتر، بما في ذلك اكتشاف الكائنات.
2. **import cv2**: يؤدي هذا إلى استيراد مكتبة OpenCV، وهي مكتبة رؤية كمبيوتر شائعة تستخدم لمهام معالجة الصور المختلفة، مثل قراءة الصور ومعالجتها، واكتشاف الكائنات، وتحليل الفيديو.
3. **import matplotlib.pyplot as plt**: يؤدي هذا إلى استيراد وحدة pyplot من مكتبة Matplotlib، وهي مكتبة تخطيط لإنشاء تصورات ثابتة ومتراكمة وتفاعلية في Python. يتم استخدامه بشكل شائع لتصور البيانات والصور.
4. **استيراد الباندا pd**: يؤدي هذا إلى استيراد مكتبة Pandas، وهي مكتبة قوية لمعالجة البيانات وتحليلها في Python. فهو يوفر هيكل البيانات ووظائفها للعمل مع البيانات المنظمة، مثل إطارات البيانات، التي تشبه الجداول في قاعدة البيانات أو جدول البيانات.

5. **import numpy as np**: يؤدي هذا إلى استيراد مكتبة NumPy، وهي حزمة أساسية للحوسبة الرقمية في Python. وهو يوفر الدعم للمصفوفات والمصفوفات الكبيرة وممتدة الأبعاد، إلى جانب مجموعة من الوظائف الرياضية للعمل على هذه المصفوفات بكفاءة.

من خلال هذه المكتبات والوحدات يمكنك الوصول إلى مجموعة واسعة من الأدوات والوظائف لتنفيذ اكتشاف الكائنات YOLOv8 ومعالجة الصور ومقاطع الفيديو وتصور النتائج وتحليل البيانات.

نموذج التدريب.

المستخدم هنا وكما قلت سابقا (yolov8n.pt) بامكانك تغيير نوع النموذج لترى الفرق. لكن اغلب المشاريع والتطبيقات الصغيرة يفضل استخدام نوع الناتو.

```
#Using any model yolov8 ()  
model = YOLO("yolov8n.pt") #you can using another models from yolov8
```

• شرح الكود.

تقوم بتهيئة نموذج اكتشاف الكائنات YOLOv8 باستخدام مكتبة Ultralytics . دعونا نحل الكود:

- YOLO("yolov8n.pt"): يقوم هذا السطر بإنشاء مثيل لنموذج YOLOv8. تحدد الوسيطة "yolov8n.pt" المسار إلى ملف الأوزان المدربة مسبقاً لنموذج YOLOv8. في هذه الحالة، يشير "yolov8n.pt" إلى متغير محدد من نموذج YOLOv8. يمكنك استبدال "yolov8n.pt" بالمسار إلى ملف الأوزان المدربة مسبقاً لأي متغير YOLOv8 آخر متوفّر في مكتبة Ultralytics . وهي كما تم ذكرها سابقاً.

من خلال تهيئة نموذج YOLOv8، يمكنك الآن استخدامه لتنفيذ مهام الكشف عن الكائنات على الصور أو مقاطع الفيديو. يتضمن ذلك اكتشاف الكائنات (فنا وعنصر السرطان هنا) وتحديد موقعها داخل الصور، بالإضافة إلى توفير معلومات حول تسميات الفناء ودرجات الثقة المرتبطة بكل فئة تم اكتشافه.

• الكود البرمجي التالي:

```
model.train(data='/content/dataset/data.yaml', epochs=100)
```

• شرح الكود البرمجي.

سوف نحاول تدريب نموذج YOLOv8 باستخدام مجموعة البيانات الخاصة بنا . حيث تقوم وظيفة `model.train` عادةً بتدريب نموذج YOLOv8 على مجموعة البيانات المحددة. دعونا نحل الكود :

- `data='/content/dataset/data.yaml': يحدد هذا المسار إلى ملف تكوين البيانات (data.yaml) الذي يحتوي على معلومات حول مجموعة البيانات الخاصة بنا مثل المسارات إلى صورة الملفات، والتعليقات

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8

التوصيحة المقابلة لها (إن أمكن)، وأسماء الفئات، وما إلى ذلك. سيستخدم النموذج هذا التكوين للتدريب على مجموعة البيانات الخاصة بنا.

- epochs=100: يحدد هذا الباراميتر عدد الحقبات (تكرارات التدريب) التي سيتم تدريب النموذج عليها. يتم تعريف الحقبة الواحدة على أنها تمريرة كاملة عبر مجموعة البيانات بأكملها أثناء التدريب. من خلال ضبط الحقبات = 100 ، فإننا تقوم بتوجيه النموذج للتدريب لمدة 100 حقبة.

بعد تشغيل هذا الكود، سيدأ نموذج YOLOv8 في التدريب على مجموعة البيانات الخاصة بنا باستخدام التكوين المحدد وسيتكرر على مجموعة البيانات لعدد محدد من الحقبات ، ويضبط أوزانها لتقليل الخسارة وتحسين أدائها في اكتشاف الكائنات (فئة السرطان).

يتطلب تدريب نموذج YOLOv8 عادةً موارد حاسوبية كبيرة وقد يستغرق وقتاً طويلاً، اعتماداً على حجم وتعقيد مجموعة البيانات الخاصة بنا بالإضافة إلى الأجهزة التي تستخدمها للتدريب.

تأكد من تنسيق مجموعة البيانات الخاصة بك بشكل صحيح وأن المسارات المحددة في ملف "data.yaml" صحيحة قبل بدء التدريب. بالإضافة إلى ذلك، تأكد من أن لديك موارد حاسوبية كافية ودعم GPU في حالة التدريب على مجموعة بيانات كبيرة.

• سوف نرى من عملية التدريب الخطوات التالية:-

يبعد أن عملية التدريب على نموذج YOLOv8 قد بدأت، ويُخضع النموذج حالياً للتدريب لمدة 100 حقبة. فيما يلي تفاصيل التقدم في التدريب والمعلومات الأساسية المقدمة:

1. **الأوزان المدرية مسبقاً المنقوله**: قام النموذج بنقل الأوزان من النماذج المدرية مسبقاً، وهي ممارسة شائعة لتهيئة النموذج بميزات تم التعرف عليها مسبقاً قبل الضبط الدقيق لمجموعة بيانات محددة.

2. **TensorBoard**: TensorBoard هي أداة تصور تستخدم لرصد وتحليل عملية التدريب. توفر الرسالة أمرًا لبدء TensorBoard وعرض تقدم التدريب في متصفح الويب.

3. **طبقة التجميد**: تم تجديد طبقة واحدة من النموذج، ومن المحتمل أن يمنع ذلك من التحديث أثناء التدريب. غالباً ما يتم إجراء طبقات التجميد للحفاظ على الميزات التي تم تعلمها مسبقاً في سيناريوهات تعلم النقل.

4. **الدقة المختلطة التلقائية (AMP)**: الدقة المختلطة التلقائية هي تقنية تستخدم لتسريع التدريب وتقليل استخدام الذاكرة عن طريق إجراء عمليات حسابية بدقة أقل (على سبيل المثال، نصف الدقة) حيثما أمكن ذلك.

5. **إعداد البيانات**: تم فحصمجموعات بيانات التدريب والتحقق من الصحة لجمع معلومات حول عدد الصور والتسميات المتاحة للتدريب والتحقق من الصحة. ربما تم تطبيق تقنيات زيادة البيانات مثل التمويه، والتعتيم المتوسط، وتحويل التدرج الرمادي، و CLAHE (مادلة الرسم البياني التكيفي المحدود للتبابن) لزيادة بيانات التدريب.

6. **اختيار المحسن**: تم تحديد المحسن (AdamW) ومعلماته تلقائياً بناءً على مجموعة البيانات وتكوين التدريب.

7. **تقديم التدريب**: يتم عرض تقدم التدريب كل فترة على حدة، مع عرض مقاييس مثل استخدام ذاكرة وحدة معالجة الرسومات، وفقدان الصندوق، وفقدان الفصل، وفقدان DFL (تعلم التصفية الديناميكي). بالإضافة إلى ذلك، يتم توفير عدد المثلثات (الكائنات) التي تم اكتشافها وحجم الصورة المستخدمة للتدريب.

8. **مقاييس التحقق**: بعد كل فترة، يتم حساب مقاييس التحقق، بما في ذلك الدقة (P)، والاستدعاء (R)، ومتوسط الدقة (mAP) عند عينات مختلفة (على سبيل المثال، IoU 0.5) لكل فئة وبشكل عام .

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8 بشكل عام، تسير عملية التدريب كما هو متوقع، حيث يتم تدريب النموذج على مجموعة البيانات المحددة وتقييمه بشكل دوري لتقدير أدائه. يمكن أن تساعد المعلومات المقدمة في مراقبة التقدم المحرز في عملية التدريب وفعاليتها.

• نتائج التدريب:-

يحتوي قاموس النتائج المقدم على مقاييس تقييم مختلفة لنموذج الكشف عن سرطان العظام. دعونا نفسر كل مقاييس:

:Precision .1

- تمثل الدقة نسبة المناطق السرطانية التي تم التنبؤ بها بشكل صحيح من بين جميع المناطق التي تم التنبؤ بأنها سرطانية. تشير درجة الدقة البالغة 0.898 تقريرًا إلى أن حوالي 89.8% من المناطق المصنفة على أنها سرطانية هي بالفعل سرطانية.

:Recall .2

- الاسترجاع، المعروف أيضًا باسم الحساسية، يقيس نسبة المناطق السرطانية الفعلية التي تم تحديدها بشكل صحيح بواسطة النموذج. تشير درجة الاستدعاء البالغة 0.735 تقريرًا إلى أن النموذج اكتشف حوالي 73.5% من المناطق السرطانية الفعلية.

:mAP50: .3

- يمثل mAP50 متوسط الدقة عند عتبة الثقة البالغة 50%. فهو يجمع بين الدقة والتركيز عبر عتبات الثقة المختلفة لتقدير أداء النموذج. تشير النتيجة التي تبلغ حوالي 0.823 إلى دقة إجمالية عالية في اكتشاف المناطق السرطانية عبر مستويات ثقة مختلفة.

:mAP50-95 .4

- يمثل mAP50-95 متوسط الدقة عبر عتبات الثقة المختلفة التي تتراوح من 50% إلى 95%. يوفر هذا المقياس رؤى حول دقة النموذج عبر نطاق أوسع من مستويات الثقة. تشير النتيجة التي تبلغ حوالي 0.413 إلى أن النموذج يحافظ على دقة عالية نسبيًا حتى عند عتبات الثقة الأعلى.

بشكل عام، تشير مقاييس التقييم هذه إلى أن نموذج الكشف عن سرطان العظام يحقق دقة عالية واسترجاعًا، مع أداء قوي عبر عتبات الثقة المختلفة. ومع ذلك، قد تكون هناك حاجة إلى مزيد من التحقق من الصحة والاختبار لتقدير مدى تعليم النموذج ومدى ملاءمته لتطبيقات العالم الحقيقي.

♣ الاستدلال على صور الاختبار.

• الكود البرمجي التالي:-

#Predict the model

```
results = model.predict(source='/content/dataset/test/images', save = True)
```

سوف نستخدم النموذج النهائي المُدرَّب والذي تم حذنه في مجلد الأوزان التابع للمجلد الرئيسي (runs) لإجراء تنبؤات واستدلال على مجموعة بيانات اختبارية. دعنا نحلل الكود :

الفصل الحادي عشر : الكشف عن سرطان العظام بتقنية Yolov8

- 'source='/content/dataset/test/images': يحدد هذا المسار إلى الدليل الذي يحتوي على صور الاختبار التي تريد إجراء التنبؤات عليها. سيقوم النموذج بتحليل الصور الموجودة في هذا الدليل وإنشاء تنبؤات للكائنات الموجودة في الصور.

- `save=True` : يشير هذا الباراميتر إلى ما إذا كان سيتم حفظ النتائج المتوقعة أم لا. من خلال تعين `save=True`، فإنك تقوم بتوجيهه النموذج لحفظ المربعات المحيطة المتوقعة وتصنيفات الصنف ودرجات الثقة لكل فئة وصنف تم اكتشافه.

بعد تشغيل هذا الكود، سيقوم النموذج الخاص بتحليل الصور الموجودة في الدليل المحدد (`/content/dataset/test/images`) وإنشاء تنبؤات للفئات الموجودة في تلك الصور. ستتضمن النتائج المتوقعة المربعات المحيطة (إحداثيات الفئة المكتشفة)، وتصنيفات الفئة (على سبيل المثال، "سرطان")، ودرجات الثقة (تشير إلى ثقة النموذج في تنبؤاته).



صورة (3) تظهر موقع السرطان من عظم الركبة

بامكانك ضغط جميع المجلدات ومجلد المقاييس والأوزان وخزنها في الحاسبة من خلال تنفيذ كود موجود ضمن الخادم السحابي كولاب لغرض تجربته وعمل الاستدلال لاي صورة اختبارية. وخاصة النموذج النهائي باسم `(best.pt)`.

• الكود البرمجي التالي:-

```
#PREDCTION: using Custom Trained best.pt in local
from ultralytics import YOLO
model=YOLO("best.pt")
#download from runs/detect/train/weights/best.pt supoose trained in GColab
results=model(source="Video.mp4",save=True,conf=0.4)
```

• شرح الكود:-

يستخدم الكود البرمجي هذا نموذج YOLOv8 الذي تم تحميله من "best.pt" لاكتشاف الكائنات الموجودة في ملف الفيديو "Video.mp4". يحدد الباراميتر conf حد الثقة لاكتشاف الكائنات، مما يضمن أن الاكتشافات ذات درجات الثقة الأعلى من 0.4 فقط هي التي تعتبر صالحة.

سيحتوي متغير النتائج على نتائج الاكتشاف، والتي قد تتضمن الفيديو المشروع مع المربعات المحيطة حول الفئة أو الصنف المكتشفة في الفيديو وهي فئة السرطان ، بالإضافة إلى المعلومات الأخرى ذات الصلة مثل اكتشافات حدود الصندوق المحاط بالفئة ودرجة الثقة.

تأكد من ضبط المسارات وأسماء الملفات وفقاً لبنية الدليل وأسماء الملفات المحددة. بالإضافة إلى ذلك، تأكد من أن ملف نموذج التحقق "best.pt" مدرب بشكل صحيح ومتاح للاستدلال.

• تطبيق ويب باستخدام مكتبة Gradio

بعد حصولنا على النموذج النهائي وخزنه في الحاسبة حان الوقت لتجربته في واجهة المستخدم كعمل تطبيق ويب لغرض التفاعل بين المستخدم والتطبيق. فيما يلي كود برمجي يحتوي على تفاصيل إضافية حول واجهة المستخدم باستخدام وحدة Gradio الخاصة بكود الكشف عن سرطان العظام:

• ملاحظة:- تم شرح كيفية تشغيل تطبيق الويب سابقاً وكيفية تنصيب وحدة Gradio

```
import gradio as gr
from ultralytics import YOLO
from PIL import Image
import os
import uuid

model = YOLO("best.pt")
def detect_objects(input_image_pil):
    # Ensure the input image is in RGB format
```

```
if input_image_pil.mode != 'RGB':
    input_image_pil = input_image_pil.convert('RGB')
# Generate a unique filename for the uploaded image
unique_filename = str(uuid.uuid4())
input_file_path = f"{unique_filename}.jpg"
input_image_pil.save(input_file_path)

# Process the image with the YOLO model
result = model(source=input_file_path, save=True, project="detect",
name="inference", exist_ok=True, conf=0.4)

# Display the result
processed_filename = f"{os.path.splitext(os.path.basename(input_file_path))[0]}.jpg"
```

```
# Load the processed image as a PIL Image
processed_image = Image.open(f"detect/inference/{processed_filename}")
return input_image_pil, processed_image
iface = gr.Interface(fn=detect_objects, inputs=gr.Image(type="pil"), outputs=["image",
"image"])
iface.launch(share=True)
```

• شرح واجهة المستخدم مع وحدة Gradio .:

يستخدم الكود المقدم وحدة Gradio لإنشاء واجهة سهلة الاستخدام للكشف عن سرطان العظام باستخدام نموذج YOLOv8. يعمل Gradio على تبسيط عملية إنشاء واجهات ويب تفاعلية لنماذج الذكاء الاصطناعي وكما تم شرحه في الفصول السابقة ، مما يجعلها في متناول كل من المطورين والمستخدمين النهائيين.

• تفاصيل الكود :

1. دمج نموذج YOLOv8 :

- تم دمج نموذجنا الذي تم تدريبه سابقا وخزنه في الحاسبة وهنا (best.pt) للكشف عن سرطان العظام في واجهة Gradio .
- عند تقييم صورة مدخلة، يقوم النموذج بمعالجتها لتحديد المناطق السرطانية المحتملة.

2. معالجة الصور وعرضها :

- يتم تحويل الصور المدخلة إلى تنسيق PIL وحفظها محلياً.
- يقوم النموذج بمعالجة الصورة ويولد نتيجة توضح الكائنات المكتشفة، بما في ذلك السرطانية المحتملة.
- يتم عرض كل من صورة الإدخال الأصلية والصورة المعالجة مع الكائنات المكتشفة في الواجهة.

3. أسماء ملفات الصور الفريدة :

- لمنع تعارض أسماء الملفات، يتم تعريف اسم فريد لكل صورة تم تحميلها باستخدام الوحدة النمطية `uuid`.

4. عتبة الثقة :

- تم تعريف عتبة الثقة البالغة 0.4 للكشف عن الكائنات. تعتبر الكائنات ذات درجة الثقة أعلى من هذا الحد اكتشافات صالحة.

5. إطلاق الواجهة :

- يتم تشغيل واجهة Gradio باستخدام وظيفة `gr.Interface` ، مع تحديد وظيفة الكشف (`detect_objects`) كتنسيق الإدخال والإخراج للصور.

اخزن الكود تحت ملف (app.py) وعند نافذة الأوامرنفذ الامر التالي (python app.py) سوف يتم عرض العنوان المحلي لشبكتك الخاصة انسخ رابط العنوان على اي متصفح . لظهور لك واجهة المستخدم كتطبيق ويب.



صورة(4) تظهر واجهة اليسار رفع الصورة وعلى اليمين تظهر صورتين الصورة الاعلى الرئيسية قبل المعالجة والصورة السفلی بعد المعالجة . تم قطع الصورة بسبب كبر الواجهة للتباين للقاري الكريم.

التطبيقات والأفكار المستقبلية

فيما يلي بعض التطبيقات والأفكار المستقبلية لتطبيق الكشف عن سرطان العظام باستخدام نموذج YOLOv8:

1. **برامج الفحص الآلي**: يمكن دمج التطبيق في برامج الفحص الآلي، مما يتيح إجراء فحص واسع النطاق لصور الأشعة السينية والرنين المغناطيسي للكشف المبكر عن سرطان العظام لدى السكان المعرضين للخطر.
2. **التطبيق عن بعد والاستشارات عن بعد**: تسهيل الاستشارات الطبية عن بعد من خلال السماح لمقدمي الرعاية الصحية بتحميل صور المرضى للتفسير والتشخيص عن بعد، خاصة في المناطق ذات الوصول المحدود إلى المرافق الطبية المتخصصة.
3. **التكامل مع الواقع المعزز (AR)**: استكشف تكامل تقنية الواقع المعزز لتركيب المناطق السرطانية المكتشفة على صور الأشعة السينية أو التصوير بالرنين المغناطيسي في الوقت الفعلي أثناء العمليات الجراحية، مما يساعد الجراحين في تحديد مكان الورم واستئصاله بدقة.
4. **تخطيط العلاج المخصص**: قم بتحسين التطبيق لت تقديم توصيات علاجية مخصصة بناءً على حجم الأورام المكتشفة وموقعها وشدةها، مما يؤدي إلى تحسين نتائج العلاج ومسارات رعاية المرضى.

5. **المراقبة الطولية:** تنفيذ ميزات المراقبة الطولية لتطور الورم والاستجابة للعلاج بمرور الوقت، مما يسمح للأطباء بتتبع التغيرات في حجم الورم وشكله لإجراء تعديلات علاجية مخصصة.
6. **منصات التشخيص التعاونية:** قم بتطوير منصات تشخيصية تعاونية حيث يمكن للعديد من المتخصصين في الرعاية الصحية الوصول بشكل آمن إلى صور المرضى وتحليلها، وتعزيز التعاون متعدد التخصصات واتخاذ القرارات على أساس الإجماع.
7. **التكامل مع السجلات الصحية الإلكترونية (EHR):** يمكنك دمج التطبيق مع أنظمة السجلات الصحية الإلكترونية لتبسيط عمليات سير العمل، وضمان الوصول السلس إلى بيانات تصوير المريض والتوثيق الآلي لنتائج التشخيص ضمن السجل الطبي للمريض.
8. **الأبحاث السريرية وتحليلات البيانات:** استفد من إمكانات تحليل بيانات التطبيق لإخفاء هوية بيانات المرضى وتجميعها لأغراض البحث السريري، مما يسهم في تطوير استراتيجيات علاجية جديدة ونماذج تنبؤية لإدارة سرطان العظام.
9. **أدوات التعليم والتدريب:** أنشئ موارد تعليمية ووحدات تدريبية باستخدام التطبيق لتنمية المتخصصين في الرعاية الصحية حول تفسير نتائج تصوير سرطان العظام واستخدام أدوات التشخيص المعتمدة على الذكاء الاصطناعي في الممارسة السريرية.
10. **التحسين المستمر للنموذج:** التحديث المستمر وتحسين نموذج YOLOv8 باستخدام تعليقات المستخدمين والتقدم البحثي المستمر، مما يضمن فعاليته ودقته في اكتشاف سرطان العظام والتكيف مع تحديات التشخيص المتطرفة.

ومن خلال استكشاف هذه التطبيقات والأفكار المستقبلية، يمكن أن يتطور تطبيق الكشف عن سرطان العظام إلى أداة متعددة الاستخدامات لا تساعد فقط في التشخيص المبكر وتحفيظ العلاج ولكنها تساهم أيضاً في التقدم في رعاية مرضى السرطان وأبحاثهم.

• الكود البرمجي النهائي.



الملخص

في الختام، يمثل تطبيق **YOLOv8** للكشف عن سرطان العظام تقدماً كبيراً في مجال التصوير الطبي. من خلال استخدام تقنيات التعلم العميق، المصممة خصيصاً لتعقيدات تحديد سرطان العظام، أظهر هذا الكتاب إمكانية التشخيص المبكر والدقيق. ومن خلال تسخير قوة **YOLOv8**، يستطيع متخصصو الرعاية الصحية الآن تحليل صور الأشعة السينية والرنين المغناطيسي بكفاءة، وتحديد المناطق المشبوهة التي تشير إلى الإصابة بسرطان العظام بسرعة.

تؤكد النتائج المقدمة في هذا الفصل من الكتاب على فعالية **YOLOv8** في اكتشاف سرطان العظام بدقة عالية ومعدلات تذكر. ومن خلال توفير رؤى وتقديرات احتمالية في الوقت الفعلي، تعمل هذه التقنية على تمكين الأطباء من اتخاذ قرارات مستنيرة على الفور، مما يؤدي إلى التدخلات في الوقت المناسب وتحسين نتائج المرضى.

وبالنظر إلى المستقبل، فإن دمج **YOLOv8** في الممارسة السريرية يبشر بإحداث ثورة في تشخيص سرطان العظام وتخطيط العلاج. يجب أن تركز المساعي البحثية المستقبلية على معالجة التحديات مثل ندرةمجموعات البيانات، وقابلية تفسير النماذج، والاعتبارات الأخلاقية لزيادة تعزيز فعالية هذا النهج المبكر وإمكانية الوصول إليه.

في جوهره، فإن اعتماد **YOLOv8** للكشف عن سرطان العظام يدل على تحول نموذجي نحو الرعاية الصحية الشخصية والاستباقية. ومن خلال الاستفادة من قدرات التعلم العميق، نحن على استعداد لقطع خطوات كبيرة في مكافحة سرطان العظام وتحسين نوعية الحياة في نهاية المطاف للأفراد المصابين.

الفصل الثاني عشر

تقنية تجزئة مرض سرطان الدماغ باستخدام نموذج Yolov8-Seg.

Brain Tumor Segmentation with Yolov8-Seg.

المقدمة.

ما هو نموذج تجزئة Yolov8-Seg.؟

تطبيق على كشف سرطان الدماغ.

نموذج التدريب.

الاستدلال على صور الاختبار.

تطبيق ويب باستخدام مكتبة Gradio.

الكود النهائي.

التطبيقات والافكار المستقبلية.

الملخص.

المقدمة:-

ثورة في التصوير العصبي للتشخيص والعلاج الدقيق في مجال التصوير الطبي، يعمل اندماج الذكاء الاصطناعي (AI) ورؤيه الكمبيوتر على تحفيز التغيرات التحويلية. ومن الأمثلة البارزة على هذا التأثير التكنولوجي اكتشاف أورام الدماغ وتقسيمها، والتي لها آثار بعيدة المدى على تشخيص وعلاج الاضطرابات العصبية. في هذا الفصل من الكتاب ، سوف نستكشف أهمية هذا النهج المبتكر، وتطبيقاته المتنوعة في مجال الرعاية الصحية، والمستقبل الواعد الذي يحمله في اكتشاف أورام الدماغ وتقسيمها.

ما هو سرطان ورم الدماغ؟

أورام الدماغ هي نمو غير طبيعي للخلايا داخل الدماغ. يمكن أن تكون حميدة (غير سرطانية) أو خبيثة (سرطانية).

و فيما يلي نظرة عامة مفصلة:

أنواع أورام الدماغ:

1. **أورام الدماغ الأولية:** تنشأ هذه الأورام في الدماغ ويمكن أن تكون حميدة أو خبيثة.
 - الأورام الدبقية: تنشأ من الخلايا الدبقية (الخلايا الداعمة للدماغ).
 - الأورام السحايا: تنشأ من السحايا، وهي الأغشية الواقية التي تغطي الدماغ والحبال الشوكي.
 - أورام الغدة النخامية: تتطور في الغدة النخامية في قاعدة الدماغ.
 - الأورام الشفائية: تنشأ من خلايا شوان، التي تنتج الغلاف المايليني المحيط بالأعصاب.
 - الأورام الأررمومية النخامية: تحدث عادةً عند الأطفال وتتشكل في الجزء السفلي الخلفي من الدماغ (الدماغين).
2. **أورام الدماغ الثانوية (أورام الدماغ النقiliّة):** تبدأ هذه الأورام في مكان آخر من الجسم وتنتشر (تنقل) إلى الدماغ. تشمل المواقع الأولى الشائعة سرطان الرئة والثدي والجلد (الورم الميلاني) والقولون والكلى.

الاعراض:-

- الصداع: غالباً ما يكون شديداً ويتافق مع النشاط أو في الصباح.
- النوبات: خاصة إذا حدثت لأول مرة في مرحلة البلوغ.
- الغثيان أو القيء: غير مرتبط بأمراض أخرى.
- تغيرات في الكلام أو الرؤية أو السمع: بما في ذلك صعوبة التحدث أو عدم وضوح الرؤية أو ازدواجها أو فقدان السمع.
- الضعف أو التنميل: عادةً في جانب واحد من الجسم.
- تغيرات في المزاج أو الشخصية: مثل الاكتئاب أو التهيج أو التغيرات المفاجئة في الشخصية.
- مشاكل في الذاكرة أو الإدراك: صعوبة في التركيز، أو فقدان الذاكرة، أو الارتباك.

التشخيص:-

1. اختبارات التصوير: التصوير بالرنين المغناطيسي أو التصوير المقطعي المحوسب لتصور موقع الورم وحجمه وخصائصه.
2. الخرزعة: إزالة عينة صغيرة من نسيج الورم لفحصها تحت المجهر لتحديد نوعه ودرجته.
3. الفحص العصبي: تقييم ردود الفعل وقوه العضلات والتنسيق والإحساس.

العلاج:-

1. العلاج الجراحي: تهدف إلى إزالة أكبر قدر ممكن من الورم دون الإضرار بأنسجة الدماغ المحيطة.
2. العلاج الإشعاعي: استخدام أشعة عالية الطاقة لقتل الخلايا السرطانية أو تقليل الأورام.
3. العلاج الكيميائي: أدوية لتدمير الخلايا السرطانية أو وقف نموها.
4. العلاج الموجي: الأدوية التي تستهدف على وجه التحديد جزيئات معينة تشارك في نمو السرطان.
5. العلاج المناعي: تعزيز جهاز المناعة في الجسم لمحاربة الخلايا السرطانية.
6. الرعاية الداعمة: إدارة الأعراض والآثار الجانبية لتحسين نوعية الحياة.

التكهن:-

- يختلف التخديص بشكل كبير اعتماداً على عوامل مثل نوع الورم وموقعه وحجمه والصحة العامة للفرد.
- قد يتم علاج بعض الأورام الحميدة بنجاح بالجراحة وحدها ويكون لها تشخيص جيد.
- الأورام الخبيثة بشكل عام أكثر عدوانية وقد تتطلب مجموعة من العلاجات. يمكن أن يكون التشخيص أقل ملاءمة، خاصة بالنسبة للأورام في مرحلة متقدمة أو تلك التي انتشرت.

الوقاية:-

- لا توجد طرق معروفة للوقاية من أورام الدماغ. ومع ذلك، فإن تجنب التعرض للإشعاع وبعض المواد الكيميائية قد يقلل من المماضط.

فجر الطب الدقيق: اكتشاف ورم الدماغ وتقسيمه.

تمثل أورام الدماغ، سواء كانت حميدة أو خبيثة، تحديات تشخيصية معقدة. يعد الاكتشاف في الوقت المناسب والتجزئة الدقيقة أمراً بالغ الأهمية لاستراتيجيات العلاج الدماغية وتحسين نتائج المرضى. تقليدياً، كانت عملية تحديد أورام الدماغ وتحديدها في الصور الطبية مهمة كثيفة العمالة وذاتية، وتعتمد غالباً على خبرة أخصائي الأشعة. أدخل الكمبيوتر، مسلحاً بخوارزميات الذكاء الاصطناعي، لإحداث ثورة في التصوير العصبي. ومن خلال تسيير قوة الرؤية الحاسوبية، يمكننا أتمتها وتعزيز دقة اكتشاف ورم الدماغ وتقسيمه. تمكن هذه التقنية متخصصي الرعاية الصحية من التعرف بسرعة على وجود الورم، وتحديد حدود الورم بدقة، والمساعدة في تحديد العلاج.

الدور الحيوي للكشف عن ورم الدماغ وتقسيمه.

لا يمكن المبالغة في أهمية الكشف الدقيق عن ورم الدماغ وتقسيمه:
تحسين نتائج المرضى: غالباً ما يؤدي الاكتشاف المبكر والتقسيم الدقيق إلى نتائج علاج أفضل، وتحسين نوعية الحياة، وتحسين معدلات البقاء على قيد الحياة للمرضى الذين يعانون من أورام الدماغ.
العلاج الدماغي: خطط علاجية مخصصة، مستنيرة ببيانات تجزئة دقيقة، تعمل على تحسين النتائج العلاجية وتقليل الأضرار الجانبية التي تتحقق بأنسجة الدماغ السليمة.
تعزيز إدارة الموارد: تعمل الآلية في التصوير العصبي على تقليل العبء الواقع على موارد الرعاية الصحية، مما يؤدي إلى تشخيصات أسرع وأكثر فعالية من حيث التكلفة.
الأبحاث العصبية: لا غنى عن بيانات التجزئة عالية الجودة في تعزيز فهمنا لأورام الدماغ، مما يؤدي إلى اختراقات محتملة في خيارات العلاج.

ما هو نموذج تجزئة Yolov8-Seg؟

لتجزئة الكائنات: نظرة عامة شاملة YOLOv8

YOLOv8، أحدث تكرار لعائلة YOLO (أنت تنظر مرة واحدة فقط) الشهيرة، يتميز الآن بقدرات تجزئة قوية إلى جانب براعته التقليدية في اكتشاف الأشياء. وهذا يعني أنه لا يمكنك فقط تحديد الكائنات داخل الصورة، بل يمكنك أيضًا تحديد حدودها بدقة، مما يؤدي إلى إنشاء أقعة مثالية للب溪سل.

فيما يلي تفاصيل لما يقدمه YOLOv8 إلى الجدول لتجزئة الكائنات:

• الميزات والمزايا الرئيسية:

البنية الموحدة: يستفيد YOLOv8 من بنية واحدة موحدة لكل من الاكتشاف والتجزئة، مما يبسط التدريب والنشر.
دقة وسرعة عالية: يوفر YOLOv8 دقة مذهلة على قدم المساواة مع نماذج التجزئة الحديثة، مع الحفاظ على سرعة توقيعه للتطبيقات في الوقت الفعلي.

النشر المرن: يمكن نشر النماذج بسهولة على منصات مختلفة، بما في ذلك وحدات المعالجة المركزية (CPU) ووحدات معالجة الرسومات (GPU) وحتى الأجهزة المحمولة، مما يجعلها قابلة للتكييف مع حالات الاستخدام المتعددة.

• الخيارات المتقدمة:

التجزئة الباتوبوكسية: يقوم YOLOv8 بالكشف عن الكائنات والتجزئة الدلالية في وقت واحد، مما يوفر فهماً شاملًا للمشهد.

العمود الفقري الفعال: يستخدم شبكة أساسية أكثر كفاءة، مما يقلل من التكلفة الحسابية ويحسن سرعة الاستدلال.

وظائف الخسارة المتقدمة: تستخدم وظائف الخسارة المتطوره المحسنة للتجزئة، مما يعزز أداء النموذج.

أساليب التدريب المحسنة: تتميز بمتغيرات تدريب محسنة لتحقيق تقارب أسرع وتعزيز أفضل.

• كيف تعمل:

تبني قدرات التجزئة في YOLOv8 من مجموعة من التقنيات:

البنية القائمة على المحوّلات: يتضمن YOLOv8 بنية قائمة على المحوّلات، مما يسمح بفهم أفضل للسياق والتقاط التبعيات طويلة المدى، مما يؤدي إلى أقعة تجزئة أكثر دقة.

شبكة هرمية الميزات (FPN): تمكّن شبكة FPN النموذج من استخراج الميزات بشكل فعال من مقاييس مختلفة، مما يؤدي إلى تحسين قدرته على اكتشاف الكائنات ذات الأحجام المختلفة وتقطيعها.

شبكة وحدة فك التشفير: تأخذ شبكة وحدة فك التشفير المتخصصة خرائط الميزات من العمود الفقري FPN وتحولها إلى أقعة تجزئة على مستوى الب溪سل.

• التطبيقات:

تفحص إمكانيات التجزئة في YOLOv8 نطاقاً واسعاً من التطبيقات عبر مختلف الصناعات:

التصوير الطبي: تقسيم الأعضاء والأورام والهيكل الأخرى بدقة لتشخيص الأمراض وتحليل العلاج.

القيادة الذاتية: تقسيم حارات الطريق وإشارات المرور والمشاة من أجل التنقل الآمن.

الروبوتات: تقسيم الكائنات للإمساك بها ومعالجتها في التطبيقات الآلية.

الزراعة: تقسيم المحاصيل والأعشاب الضارة والآفات لإدارة المحاصيل بكفاءة.

البيع بالتجزئة: تقسيم المنتجات على الرفوف لإدارة المخزون وتحليل سلوك العملاء.

حيث توفر إمكانيات التجزئة في YOLOv8 حلاً قوياً ومتنوعاً لاستخدامات تطبيقات متعددة. إن دقتها العالية وسرعته وموارنه تجعله خياراً مقتصداً لمهام تجزئة الكائنات في الوقت الفعلي. إن البنية الموحدة والوثائق الشاملة تجعلها في متاح للمطورين والباحثين على حد سواء.

• تطبيق على كشف سرطان الدماغ .

• مجموعة البيانات.

بعد حصولنا وخزن مجموعة البيانات (راجع موضوع تنزيل مجموعة البيانات) أو ماتسمى Dataset سوف نبدأ بالخطوات البرمجية التالية:-

```
#yolov8 install pip
```

```
!pip install ultralytics
```

حيث يقوم هذا الكود بتنصيب مكتبة (Yolov8) لجميع النماذج المذكورة وطبعاً نماذج شاملة للكشف والتجزئة والتصنيف وهي كالتالي:-
(هنا فقط نماذج التجزئة).

- **Yolov8n-seg**
- **Yolov8s-seg**
- **Yolov8m-seg**
- **Yolov8x-seg**
- **Yolov8xl-seg**

لكننا سوف نتعقب فقط في نموذج التجزئة الكبير (yolov8x-seg.pt) وحاول تجربة نماذج التجزئة الأخرى لترى الفرق والدقة والسرعة وحجم النموذج ومقاييس الدقة الأخرى.

• الكود البرمجي التالي:-

```
# upload Brain Tumor Dataset
from google.colab import files
# Prompt the user to upload a file
uploaded = files.upload()
# Loop through the uploaded files
for filename in uploaded.keys():
    print(f'Uploaded file: {filename}')
```

حيث يقوم الكود برفع البيانات المضغوطة والذي تم إزالتها وخزنها في الحاسبة الشخصية والتي تحتوي على ثلاثة مجلدات وهي بيانات التدريب والتحقق والاختبار مع ملف التهيئة والذي يسمى بملفات (yaml) وهي جداً

الفصل الثاني عشر: تقيية تجزئة مرض سرطان الدماغ باستخدام نموذج Yolov8-Seg مهمه والتي تحدد مسارات المجلدات وعدد الفئات التي سوف يتم الكشف عنها واسماء الفئات كملخص لها ضمن اعدادات تدريب النموذج الدماغص لنافي هذه الحالة لدينا فئة واحدة هي كشف السرطان بامكانك تحرير الملف وتعديل مسارات مجلدات الصور.

- **ال kod البرمجي التالي.**

```
!unzip /content/dataset.zip
```

```
!rm -r /content/dataset.zip
```

وهو واضح فتح الضغط وخزن جميع المجلدات في مجلد الداتايت ومن ثم مسح المجلد المضغوط

• **ال kod البرمجي التالي.**

حيث يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد وهو جدا مهم لمعرفة حجم مجموعة البيانات تحت التدريب.

```
import os
def count_files_with_extension(folder_path, extension):
    file_count = 0
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            _, file_extension = os.path.splitext(file)
            if file_extension.lower() == extension:
                file_count += 1

    return file_count

def main():
    folders = ['train', 'valid', 'test']
    image_extension = '.jpg'
    for folder in folders:

        folder_path = os.path.join(folder, 'images')
        file_count = count_files_with_extension(folder_path, image_extension)
        print(f"Folder: {folder}, Image Files: {file_count}")

if __name__ == "__main__":
    main()
```

بعد التنفيذ.

```
Folder: train, Image Files: 400
Folder: val, Image Files: 50
Folder: test, Image Files: 50
```

• شرح الكود البرمجي بالتفصيل.

استيراد الوحدات المطلوبة.

تحديد الدالة `count_files_with_extension`

تأخذ هذه الوظيفة وسيطتين `Folder_path` (مسار المجلد للبحث عن الملفات) والامتداد (امتداد الملف المراد حسابه). ويقوم بتهيئة عدد الملفات المتغير لتخزين عدد الملفات ذات الامتداد المحدد.

ويستخدم الدالة `(os.walk)` للتنقل عبر كافة الأدلة والأدلة الفرعية الموجودة ضمن المجلد `path` المحدد. بالنسبة لكل ملف تتم مواجهته، يقوم باستخراج امتداد الملف باستخدام `(os.path.splitext()` ومقارنته بالامتداد المقدم. إذا كانت متطابقة، فإنه يزيد `.file_count`.

وأخيراً، تقوم بإرجاع العدد الإجمالي للملفات ذات الامتداد المحدد الموجود في المجلد.

تحديد الوظيفة الرئيسية:

تقوم هذه الوظيفة بتنسيق عملية العد لكل مجلد من المجلدات الرئيسية الثلاثة: "التدريب" و"التحقق" و"الاختبار".

يقوم بتهيئة قائمة المجلدات التي تحتوي على أسماء المجلدات الرئيسية. بالنسبة لكل اسم مجلد في المجلدات، يقوم بإنشاء المسار إلى المجلد الفرعى "الصور" باستخدام `(os.path.join()`.

ثم يقوم بعد ذلك باستدعاء وظيفة `count_files_with_extension` بمسار المجلد الذي تم إنشاؤه وامتداد الصورة المحدد ('.jpg').

وأخيراً، يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد.

يوفر هذا الكود البرمجي طريقة معيارية وقابلة للتطوير لحساب الملفات ذات الامتداد المحدد داخل مجلدات متعددة، مما يسهل المهام مثل المعالجة المسابقة للبيانات ومشاريع التعلم الآلي والتعلم العميق ومعرفة عدد الصورة في كل مجلد مهم في التدريب .

• الكود البرمجي:.

```
#Read Dataset path for training
```

```
!cat /content/dataset/dataset.yaml
```

• شرح الكود.

حالياً أنك تستخدم بينة `Jupyter Notebook` أو `Google Colab` للعمل مع مجموعة البيانات الخاصة بنا. يستخدم الأمر `cat!` عادةً في أنظمة التشغيل المشابهة لـ `Unix` وفي دفتر ملاحظات `Jupyter/Colab` لعرض محتويات الملف.

دعنا نحلل ما يفعله هذا الأمر:

- يشير ! إلى أنه يجب تنفيذ الأمر التالي في خلاف النظام، وليس في `Python`.

- `cat` هو أمر يستخدم لتسلسل وعرض محتويات الملفات.

لذلك، عند تشغيل `!cat /content/dataset/dataset.yaml`، فإنه يقرأ ويعرض محتويات الملف الموجود في `/content/dataset/dataset.yaml`.

يحتوي الملف "data.yaml" على معلومات التكوين أو البيانات الوصفية حول مجموعة البيانات الخاصة في مشروعنا مثل المسارات إلى الصور، والتسميات المقابلة لها، وأي خطوات معالجة مسبقة، وما إلى ذلك. لنظهر لنا نتيجة التنفيذ وهي فئة واحدة اسمها السرطان. وكما قلت سابقاً جداً مهم لحفظ على المسارات وهي واضحة من ملف التكوين. (data.yaml)

```
train: ../train/images
val: ../valid/images
test: ../test/images
nc: 1
names: ['tumor']
```

نحو التدريب.

- الكود البرمجي التالي:-

```
!yolo task=segment mode=train model=yolov8x-seg.pt
data=/content/dataset/dataset.yaml epochs=100 imgsz=640
```

- شرح الكود البرمجي.

سوف نبدأ بتدريب نموذج YOLOv8-X لمهام التجزئة باستخدام معلمات محددة. دعنا نحل الكود القصير او الامر التالي :

- **yolo!**: يشير هذا إلى أنك تستخدم واجهة سطر أوامر أو برنامج نصي يتعرف على الأوامر المسبوقة بـ ".yolo". يمكن أن يكون إطار عمل أو مجموعة أدوات للعمل مع نماذج YOLO. وهنا واجهة الخادم السحابي كوكل كولاب.
- **task=segment**: يشير إلى أن المهمة التي تقوم بها هي التجزئة، والتي تتضمن تصنيف كل بناء في الصورة إلى فئة معينة.
- **mode=train**: يحدد أنك في وضع التدريب، مما يعني أنك تدرب نموذجاً جديداً بدلاً من استخدام نموذج تم تدريبه مسبقاً للاستدلال.
- **model=yolov8x-seg.pt**: يحدد بنية النموذج الذي تستخدمه. في هذه الحالة، يكون X-YOLOv8.
- **data=/content/dataset/dataset.yaml**: يشير "pt" في اسم الملف إلى أنه نموذج PyTorch.
- يحتوي ملف YAML هذا على معلومات حول مجموعة البيانات الخاصة بنا، مثل المسارات إلى الصور بالإضافة إلى تسميات الفئة أي إلى مجلدات (images,labels).
- **الحقبات والتكرار=100**: يحدد عدد فترات التدريب. الحقبة الواحدة عبارة عن تمريرة كاملة عبر مجموعة بيانات التدريب بأكملها.
- **imgsz=640**: يحدد حجم الصورة المدخلة للتدريب. من المحتمل أن يتم تغيير حجم الصور أو اقتاصتها بهذا الحجم قبل إدخالها في النموذج أثناء التدريب.
-

بشكل عام، يقوم هذا الأمر بإعداد عملية التدريب لنموذج YOLOv8-Seg لمهام التجزئة، وتحديد بنية النموذج، ومجموعة البيانات، وبارامترات التدريب، والبارامترات الفائقة.

• نتائج التدريب.

قد أكملت عملية التحقق من صحة نموذج التجزئة YOLOv8 الخاص بنا. فيما يلي تفاصيل نتائج التحقق من الصحة:

• **ملخص النموذج:** يشير ملخص نموذج YOLOv8x-seg إلى أنه يتكون من 295 طبقة ياجمالي 71,721,619 معلمة. يعمل النموذج على CUDA باستخدام وحدة معالجة الرسومات Tesla T4 مع ذاكرة تبلغ 15102 ميجابايت. التعقيد الحسابي للنموذج هو 343.7 GFLOPs.

• **مقاييس الكشف:** تعرض نتائج التحقق مقاييس الأداء لكل من تنبؤات المربع المحيط (المربع) وقناع التجزئة. بالنسبة للفئة "الكل"، والتي من المحتمل أن تمثل المقاييس المجمعة عبر جميع الفئات، يحقق النموذج المقاييس التالية:

- دقة الصندوق (P): 91.5%

- استدعاء الصندوق (R): 92.0%

- صندوق mAP50: 96.5%

- صندوق mAP50-95: 80.8%

- دقة قناع التجزئة (P): 91.5%

- استدعاء قناع التجزئة (R): 92.0%

- قناع mAP50: 96.5%

- قناع mAP50-95: 79.4%

• **سرعة الاستدلال:** يشير توزيع سرعة الاستدلال لكل صورة إلى الوقت المستغرق لخطوات المعالجة الدماغية:

- المعالجة المسبقة: 0.2 ملي ثانية

- الاستدلال: 32.6 ملي ثانية

- حساب الخسارة: 0.0 ملي ثانية

- مرحلة ما بعد المعالجة: 1.6 ملي ثانية

وهذا يعطي نظرة ثاقبة لفاءة النموذج أثناء الاستدلال.

• **النتائج:** يتم حفظ نتائج التتحقق في الدليل ".runs/segment/train".

بشكل عام، تشير نتائج التتحقق من الصحة إلى أن نموذج YOLOv8x-seg يعمل بشكل جيد من حيث الدقة والتنكير لكل من تنبؤات المربع المحيط وقناع التجزئة. كما أن سرعة الاستدلال معقولة أيضاً، مما يتيح تطبيقات الوقت الفعلي أو شبه الوقت الفعلي. تتحقق هذه النتائج من فعالية النموذج في مهام تجزئة ورم الدماغ.

• الاستدلال على صور الاختبار.

• الكود البرمجي

```
#test final custom model best.pt with test images
!yolo task=segment mode=predict
model=/content/dataset/runs/segment/train3/weights/best.pt conf=0.25
source='/content/dataset/test/images/y101.jpg.rf.37f034.jpg' save=true
```

ملاحظة:-

(يجب التأكيد من مسار النموذج المدرب النهائي مع مسار اي صور اختبارية)

• نتائج الاستدلال والتنبأ.

```
from IPython.display import display, Image
Image(filename='/content/dataset/runs/segment/predict/y101.jpg.rf.293853f3849b109d0b
4dd0c3e937f034.jpg', width=600)
```

حيث قمنا بإجراء الاستدلال على صورة ما باستخدام نموذج التجزئة YOLOv8 المدرب.
و فيما يلي تفصيل لنتائج الاستدلال:

• **معلومات الصورة:** تم اجراء الاستدلال على الصورة "y101.jpg.rf.293853f3849b109d0b4dd0c3e937f034.jpg" من مجموعة بيانات الاختبار الخاصة بنا. دقة الصورة هي 640x640.

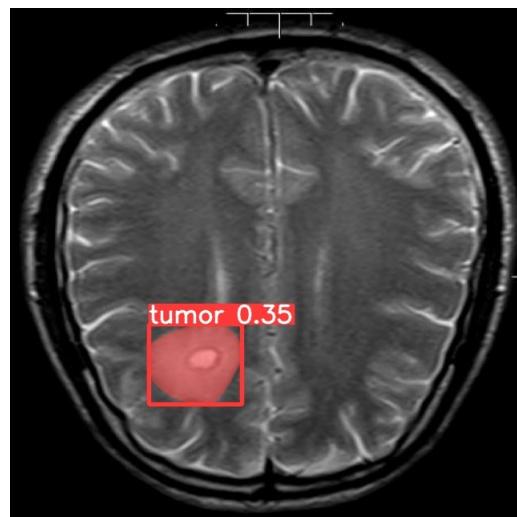
• **الاكتشاف:** اكتشف النموذج ورماً واحداً في الصورة مع وقت معالجة يبلغ 117.1 ملي ثانية. يشير هذا إلى أن النموذج قادر على تحديد الأورام بدقة داخل الصورة.

• **سرعة الاستدلال:** يوضح تفصيل سرعة الاستدلال لكل صورة الوقت المستغرق لخطوات المعالجة المختلفة:
- المعالجة المسبقة: 0.7 ملي ثانية
- الاستدلال: 117.1 ملي ثانية
- مرحلة ما بعد المعالجة: 680.3 ملي ثانية

يتوفر هذا نظرة ثافية للوقت الذي يستغرقه النموذج لمراحل المعالجة المختلفة.

• **النتائج:** يتم حفظ نتائج الاستدلال في الدليل "runs/segment/predict", والذي من المحتمل أن يحتوي على مخرجات التجزئة، بما في ذلك المربعات المحيطة أو الأقنعة، المتراكبة على صورة الإدخال.

بشكل عام، تشير نتائج الاستدلال إلى أن نموذج التجزئة **YOLOv8-Seg** نجح في التعرف على الورم داخل صورة الاختبار، مما يدل على فعاليته في مهام تجزئة ورم الدماغ.



صورة (1) حيث يظهر موقع الورم في الدماغ.

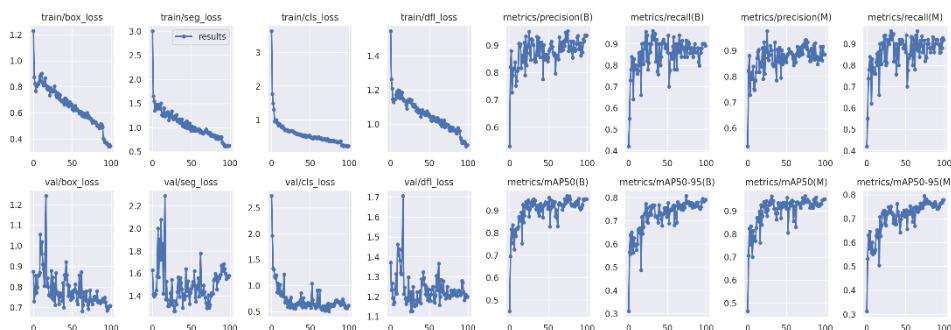
• الكود البرمجي

```
import os
import zipfile
from google.colab import files
zip_name="Brain-Tumor-Benchmark&weights"
# Directory path
dir_path = '/content/dataset/runs/segment/train/' # This will zip files from the current
directory in Colab. Adjust as needed.
# List of file extensions to zip
file_extensions = ['.pt']

# Create a new zip file
with zipfile.ZipFile(f'{zip_name}.zip', 'w') as zipf:
    # Walk the directory and filter files
    for foldername, subfolders, filenames in os.walk(dir_path):
        for filename in filenames:
            if any(filename.endswith(ext) for ext in file_extensions):
                zipf.write(os.path.join(foldername, filename), filename) # Add files to the zip
# Download the zip file
files.download(f'{zip_name}.zip')
```

• شرح الكود

خزن جميع النماذج المدربة والصور البيانية ونتائج التدريب وضغطها. يرجى التأكد من المسارات.



صورة (2) تظهر جميع النتائج والمقاييس

• التحقق من صحة نموذج تجزئة.

```
#validate custom model
!yolo task=segment mode=val
model=/content/dataset/runs/segment/train3/weights/best.pt
data=/content/dataset/dataset.yaml
```

• شرح الكود.

أجرينا التحقق من صحة نموذج تجزئة YOLOv8 الدماخصن الخاص بنا .
وفيمالي تفسير لنتائج التتحقق من الصحة :

• **ملخص النموذج:** النموذج المستخدم للتحقق من الصحة هو نموذج YOLOv8x-seg مخصص يحتوي على 295 طبقة وإجمالي 71,721,619 معلمة. يتم قياس التعقيد الحسابي عند 343.7 GFLOPs.

• **مجموعة البيانات:** تحتوي مجموعة بيانات التتحقق على 50 صورة، ولم يتم اكتشاف أي صور فاسدة. تمت عملية المسح لذاكرة التخزين المؤقت للتسليمات بنجاح، مما يشير إلى الاستعداد للتحقق من الصحة.

• **مقاييس الاكتشاف:** بالنسبة للفئة "الكل"، التي تمثل المقاييس المجمعة عبر جميع الفئات، حقق النموذج مقاييس الأداء التالية:

- دقة الصندوق (P): 93.9%
- استدعاء الصندوق (R): 91.6%
- صندوق mAP50: 96.5%
- صندوق mAP50-95: 80.9%
- دقة قناع التجزئة (P): 93.9%
- استدعاء قناع التجزئة (R): 91.6%
- قناع mAP50: 96.5%
- قناع mAP50-95: 79.6%

- سرعة الاستدلال:** يوضح تفصيل سرعة الاستدلال لكل صورة الوقت المستغرق لخطوات المعالجة المختلفة:
 - المعالجة المسبقة: 9.8 ملي ثانية
 - الاستدلال: 66.9 ملي ثانية
 - مرحلة ما بعد العملية: 19.0 ملي ثانية
- يوفّر هذا نظرة ثاقبة للوقت الذي يستغرقه النموذج لمراحل المعالجة الـ الدماغية المختلفة أثناء التحقق من الصحة.

بشكل عام، تشير نتائج التحقق من الصحة إلى أن نموذج التجزئة الدماغي الخاص بنا يعمل بشكل جيد، مع دقة عالية واستدعاء لكل من تنبؤات المربع المحيط بالورم وقناة التجزئة. بالإضافة إلى ذلك، فإن سرعة الاستدلال معقولة، مما يجعل النموذج مناسباً لتطبيقات العالم الحقيقي. تتحقق هذه النتائج من فعالية النموذج المخصص الخاص بنا لمهمات تجزئة أورام الدماغ.

• الكود البرمجي.

```
!yolo task=segment mode=predict
model=/content/dataset/runs/segment/train/weights/best.pt conf=0.25
source=/content/dataset/test/images save=true
```

• شرح الكود.

- يقوم الكود البرمجي بالاستدلال باستخدام نموذج التجزئة الدماغي YOLOv8 على مجموعة من صور الاختبار. فيما يلي تفاصيل أمر الاستدلال الذي قدمته:
- المهمة:** المهمة هي التجزئة، مما يشير إلى أنه تهدف إلى تقسيم الكائنات داخل الصور.
- الوضع:** تم ضبط الوضع للتنبؤ، مما يعني أنه تستخدم النموذج لإجراء تنبؤات بشأن البيانات غير المرئية.
- النموذج:** أنت تحدد المسار إلى نموذج تجزئة YOLOv8 المخصص، الموجود في ".content/dataset/runs/segment/train/weights/best.pt"
- ثقة الثقة :** تم تعين عتبة الثقة على 0.25، مما يعني أنه سيتم الاحتفاظ فقط بالتنبؤات التي لديها درجة ثقة أكبر من أو تساوي 0.25.
- المصدر:** تحدد الدليل الذي يحتوي على صور الاختبار، الموجود في ".content/dataset/test/images/"
- الحفظ :** ضبط الحفظ على الصحيح، مما يعني أنه سيتم حفظ نتائج الاستدلال.

سيقوم هذا الأمر بتشغيل الاستدلال على صور الاختبار باستخدام نموذج تجزئة YOLOv8 الدماغي الخاص بنا وسيتم حفظ النتائج لمزيد من التحليل أو التصور.

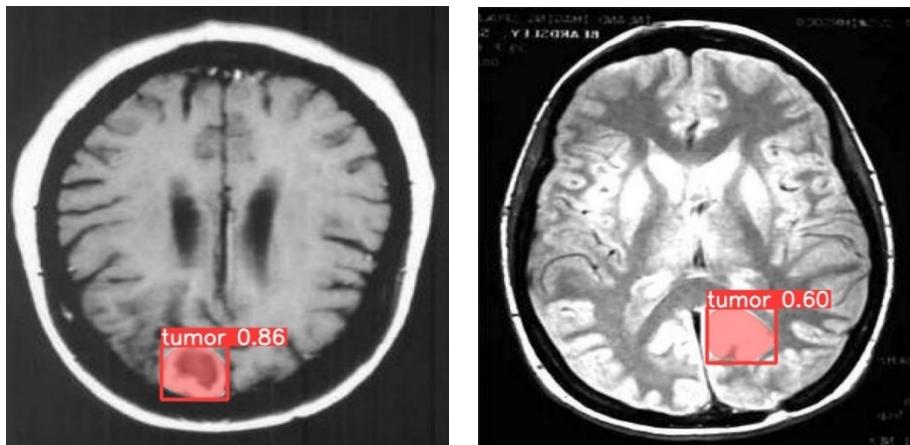
ملاحظة:- (يجب التأكد من المسارات للنموذج ومجلد الصور).

• الكود البرمجي.

```
import glob
from IPython.display import Image, display
for image_path in glob.glob('/content/dataset/runs/segment/predict/*.jpg')[3]:
    display(Image(filename=image_path, height=600))
    print("\n")
```

- **شرح الكود البرمجي.**

عرض الصور باستخدام سيرعرض الكود البرمجي الفصیر هذا الصور الثلاث الأولى الموجودة في الدليل المحدد.



صورة (3) تظهر موقع الورم في صور الاختبار لواجهة واستدلال المستخدم لصور اختبارية

- **تطبيق الويب للمستخدم.**

- **الكود البرمجي.**

```
import gradio as gr
from ultralytics import YOLO
from PIL import Image
import os
import uuid
model = YOLO("model/best.pt")
def seg_objects(input_image_pil):
    # Ensure the input image is in RGB format
    if input_image_pil.mode != 'RGB':
        input_image_pil = input_image_pil.convert('RGB')
    # Generate a unique filename for the uploaded image
    unique_filename = str(uuid.uuid4())
    input_file_path = f"{unique_filename}.jpg"
    input_image_pil.save(input_file_path)

    # Process the image with the YOLO model
    result = model(source=input_file_path, save=True, project="detect", name="inference",
exist_ok=True, conf=0.4)
    # Display the result
    processed_filename = f"{os.path.splitext(os.path.basename(input_file_path))[0]}.jpg"
```

```
# Load the processed image as a PIL Image
processed_image = Image.open(f"detect/inference/{processed_filename}")
# processed_image_small = processed_image.resize((int(processed_image.width / 4),
int(processed_image.height / 4)))

return input_image_pil, processed_image
iface = gr.Interface(fn=seg_objects, inputs=gr.Image(type="pil"), outputs=["image",
"image"])
iface.launch(share=True)
```

لإحتاج الى شرح الكود تم شرحه وتكراره في الفصول السابقة راجع مكتبة (Gradio)



الكود البرمجي النهائي.

رمز الاستجابة السريع

أخيرا.....

• الآفاق المستقبلية.

إن مستقبل الكشف عن أورام الدماغ وتقسيمها باستخدام الذكاء الاصطناعي يبشر بالخير على جبهات متعددة:

دقة محسنة: تعد التطورات المستمرة في الذكاء الاصطناعي بدقة أكبر في اكتشاف أورام الدماغ وتقسيمها، مما يقلل من النتائج الإيجابية والسلبية الكاذبة.

إمكانية الوصول على مستوى العالم: سوف يعمل التطبيب عن بعد، إلى جانب تشخيصات الذكاء الاصطناعي، على سد الفوارق في الرعاية الصحية من خلال جلب رعاية الخبراء إلى المناطق النائية أو التي تعاني من نقص الخدمات.

التدخل المبكر: ستتمكن النماذج التنبؤية المدعومة بالذكاء الاصطناعي من استراتيجيات التدخل المبكر لأورام الدماغ، مما قد يمنع الضرر العصبي.

الرعاية الصحية المتكاملة: سيستمر الذكاء الاصطناعي في التكامل بسلامة مع أنظمة الرعاية الصحية، مما يوفر دعماً شاملأً لمتخصصي الرعاية الصحية.

التحليلات التنبؤية: يقوم الذكاء الاصطناعي بتحليل مجموعات كبيرة من البيانات للتنبؤ بأنماط المرض، والمساعدة في تحديد الموارد ومنع تفشي المرض.

- **اكتشاف الأدوية:** يعمل التعلم الآلي على تسريع اكتشاف الأدوية من خلال نمذجة التفاعلات الجزيئية، مما يقلل الوقت والتكاليف بشكل كبير.
- **التطبيب عن بعد:** يعزز الذكاء الاصطناعي الرعاية الصحية عن بعد من خلال الاستشارات الافتراضية والتشخيص والمراقبة، خاصة في أوقات الأزمات.
- **الطب الشخصي:** يقوم الذكاء الاصطناعي بتصميم العلاجات بناءً على السمات الجينية الفردية والتاريخ الطبي، مما يؤدي إلى تحسين الفعالية وتقليل الآثار الجانبية.
- **الأهمية القصوى للكشف عن ورم الدماغ وتقسيمه.**

لا يمكن المبالغة في استخدام طرق الكشف عن أورام الدماغ وتقسيمها المدعومة بالذكاء الاصطناعي في التصوير الطبي. هذا النهج الثوري له آثار بعيدة المدى، ليس فقط في مجال علم الأعصاب ولكن عبر نطاق أوسع من الرعاية الصحية والبحوث الطبية. دعونا نتعمق أكثر في سبب أهمية هذا الابتكار:

- **التدخل المبكر ينفي الأرواح:** أحد الدوافع الرئيسية وراء أهمية اكتشاف أورام الدماغ بمساعدة الذكاء الاصطناعي هو إمكانية التدخل المبكر. تشتهر أورام الدماغ بقدرتها على البقاء بدون أعراض حتى تصل إلى مراحل متقدمة. ومن خلال تحديد الأورام في مراحلها الأولى، ي عمل الذكاء الاصطناعي على تمكين المتخصصين في الرعاية الصحية من بدء العلاج على الفور، مما يحسن بشكل كبير فرص تحقيق نتائج ناجحة.
- **إعادة تعريف الطب الدقيق:** الطب الدقيق هو مستقبل الرعاية الصحية، ويعتبر الكشف والتجزئة القائم على الذكاء الاصطناعي فعالين في جعله حقيقة واقعة. يتيح التحديد الدقيق لأورام الدماغ وتحديدها تفصيل العلاجات بدرجة غير مسبوقة. لم تعد العلاجات تدار على أساس تعليمات واسعة النطاق؛ يتم تخصيصها وفقًا للخصائص المحددة للورم الذي يعاني منه المريض.

- **تحسين جودة الحياة:** إلى جانب معدلات البقاء على قيد الحياة، يساهم اكتشاف أورام الدماغ وتقسيمها باستخدام الذكاء الاصطناعي في تحسين نوعية حياة المرضى. يمكن ضبط العلاجات بدقة لتقليل الضرر الذي يلحق بأنسجة الدماغ السليمة، مما يقلل من خطر الضعف الإدراكي، أو فقدان الحواس، أو العجز الحركي المرتبط غالباً بالعلاجات الغازية.
- **البحث وتطوير الأدوية:** تعد البيانات الناتجة عن تجزئة أورام الدماغ المعتمدة على الذكاء الاصطناعي بمثابة مصدر غني للبحث وتطوير الأدوية. تتيح هذه البيانات التي لا تقدر بثمن للعلماء استكشاف علاجات جديدة وتساهم في فهم أعمق لبيولوجيا أورام الدماغ، مما قد يؤدي إلى علاجات رائدة.
- **تحسين الموارد:** في عصر غالباً ما تكون فيه أنظمة الرعاية الصحية ممتدة إلى أقصى حدودها، فإن اعتماد التصوير العصبي من خلال الذكاء الاصطناعي توفر تحسيناً للموارد. يضمن التشخيص الأسرع والأكثر دقة الذي توفره تقنيات الذكاء الاصطناعي تخصيص موارد الرعاية الصحية بكفاءة.

• موضوعات أخرى ذات صلة:

في سياق الذكاء الاصطناعي في الرعاية الصحية وتصوير الأعصاب، تأتي العديد من المواضيع الأخرى ذات الصلة في المقدمة:

- **الاعتبارات الأخلاقية:** تشير الآثار الأخلاقية للذكاء الاصطناعي في مجال الرعاية الصحية، وخاصة في المجالات الحساسة مثل الكشف عن أورام الدماغ، تساؤلات حول خصوصية البيانات، والموافقة المستيرة، والتحيزات الخوارزمية.
- **دمج الذكاء الاصطناعي في الممارسة السريرية:** مناقشة التكامل السلس لتقنيات الذكاء الاصطناعي في سير العمل السريري ومعالجة التدريب والتعليم المطلوب لمختصي الرعاية الصحية للعمل جنباً إلى جنب مع أنظمة الذكاء الاصطناعي.
- **الأطر التنظيمية:** تعد الحاجة إلى إطار تنظيمي قوية لضمان سلامة وفعالية الأدوات الطبية المعتمدة على الذكاء الاصطناعي موضوعاً بالغ الأهمية. كيف يمكننا تحقيق التوازن بين الابتكار وسلامة المرضى؟
- **الرعاية التي تركز على المريض:** التأكيد على أهمية وضع المرضى في قلب الرعاية الصحية المدعومة بالذكاء الاصطناعي، وضمان أن تعمل التقنيات على تعزيز العلاقة بين الطبيب والمريض بدلاً من استبدالها.
- **التفاوتات العالمية في الرعاية الصحية:** قدرة الذكاء الاصطناعي على سد التفاوتات في الرعاية الصحية، وتوفير التشخيص والرعاية المتخصصة للمناطق والسكان المحرومين.
- **التعاون متعدد التخصصات:** أهمية التعاون متعدد التخصصات بين علماء الكمبيوتر والمهنيين الطبيين والباحثين لتطوير الذكاء الاصطناعي في مجال الرعاية الصحية والتصوير العصبي.

يستمر التأثر بين الرعاية الصحية والذكاء الاصطناعي في النمو، مدفوعاً بالتقدم في التعلم العميق وتحليل الصور وقدرات الأجهزة. ومع نضوج هذه التقنيات، يمكننا أن نتوقع المزيد من الابتكارات التي من شأنها تحسين رعاية المرضى، وتبسيط عمليات الرعاية الصحية، وجعل الرعاية الصحية في متناول نطاق أوسع من السكان.

• التحديات والاعتبارات الأخلاقية.

في حين أن YOLO تقدم وعداً هائلاً، إلا أنها تأتي مع تحديات، بما في ذلك خصوصية البيانات، والتحيز النمونجي، وال الحاجة إلى التحقق الصارم من الصحة. يجب أن تعالج الاعتبارات الأخلاقية القضايا المتعلقة بموافقة المريض والاستخدام المسؤول للذكاء الاصطناعي في الرعاية الصحية.

في الختام، يعد اكتشاف أورام العظام القائم على YOLO ابتكاراً رائداً في مجال التصوير الطبي، ويمثل جانباً واحداً فقط من ثورة الذكاء الاصطناعي في الرعاية الصحية. وبينما نواصل تسخير قدرات الذكاء الاصطناعي والرؤية الحاسوبية، يحمل المستقبل إمكانات هائلة لتحويل الرعاية الصحية على نطاق عالمي، وتحسين نتائج المرضى، وإعادة تشكيل الطريقة التي نتعامل بها مع التسخيص والعلاج الطبي.

الملخص

يقدم دمج نموذج YOLOv8 لتجزئة أورام الدماغ وسيلة واحدة في التصوير الطبي. وإليك خاتمة منظمة لهذا الفصل من الكتاب :

الاستنتاج:

الاستفادة من تجزئة YOLOv8 لتجزئة أورام الدماغ. يعد تجزئة أورام الدماغ مهمة حاسمة في التصوير الطبي، وهو ضروري للتشخيص وتنظيم العلاج ومراقبة المرض. يمثل استخدام YOLOv8 لهذا الغرض تقدماً كبيراً في هذا المجال. ومن خلال النقاط الرئيسية التالية، نستنتج مدى الفعالية والأفاق المستقبلية والاعتبارات المرتبطة بهذا النهج:

فعالية في تجزئة أورام الدماغ:

- ظهرت بنيّة YOLOv8، المصممة خصيصاً لمهمات التجزئة، دقة وكفاءة ملحوظة في اكتشاف أورام الدماغ وتقسيمها من الصور الطبية.
- من خلال تسخير قدرات التعلم العميق، يحقق YOLOv8 أداءً متطرّفاً من حيث دقة التجزئة وسرعتها، مما يتيح تحليل عمليات الفحص الطبي في الوقت الفعلي أو شبه الحقيقي.

الأفاق المستقبلية والتحسينات:

- يحمل البحث والتطوير المستمر في تجزئة أورام الدماغ المستندة إلى YOLOv8 إمكانات هائلة لمزيد من التحسين وتحسين دقة التجزئة.
- يمكن أن يؤدي دمج بيانات التصوير متعدد الوسائط، مثل التصوير بالرنين المغناطيسي والتصوير المقطعي المحوسب، إلى تعزيز قوة التنموذج وإمكانية تطبيقه عبر مجموعات متنوعة من المرضى وطرازات التصوير.
- استكشاف التقنيات المتقدمة مثل نقل التعلم وتكييف المجال قد يسهل تكيف النماذج المدربة مسبقاً مع إعدادات سريرية محددة، مما يقلل الحاجة إلى بيانات مصنفة واسعة النطاق.

اعتبارات التدريب والتحقق:

- تعد بروتوكولات التدريب والتحقق الصارمة ذات أهمية قصوى لضمان موثوقية وقابلية تعميم نماذج تجزئة أورام الدماغ المستندة إلى YOLOv8.
- تعد مجموعات البيانات الكبيرة والمشروحة ضرورية لتدريب النماذج القوية، مما يستلزم التعاون بين مؤسسات الرعاية الصحية والباحثين وعلماء البيانات لتنظيم مجموعات البيانات الشاملة مع الالتزام بلوائح خصوصية المريض.

- توفر تقنيات التحقق المتقطعة، إلى جانب المقاييس مثل معامل تشابه الثرد والتقاطع على الاتحاد، تقييمات كمية لأداء التنموذج وقدرات التعلم.

الواجهة والاستدلال للمستخدمين النهائيين:

- يعد تصميم واجهات سهلة الاستخدام للأطباء وأخصائيي الأشعة أمراً بالغ الأهمية للتكامل السلس لأدوات التجزئة المستندة إلى YOLOv8 في سير العمل السريري.
- خطوط أنابيب الاستدلال البسيطة، التي تتضمن تقنيات نشر النماذج الفعالة وتسريع الأجهزة الأمثل، تضمن الأداء في الوقت الفعلي أثناء تحليل الصور.
- التكامل مع منصات برامج التصوير الطبي الحالية، إلى جانب أدوات التصور البديهية، يسهل تفسير واستخدام نتائج التجزئة من قبل المتخصصين في الرعاية الصحية.

في الختام، فإن دمج تجزئة YOLOv8 في تجزئة أورام الدماغ يدل على نهج تحويلي نحو تعزيز دقة التشخيص ورعاية المرضى في علاج الأورام العصبية. إن التقدم المستمر في تطوير التماذج وتنظيممجموعات البيانات وتصميم الواجهة يعد بمستقبلٍ تصبح فيه أدوات التجزئة المعتمدة على الذكاء الاصطناعي أصولاً لا غنى عنها في الممارسة السريرية.

الفصل الثالث عشر

نموذج YOLOv8 للكشف مرض سرطان الجلد

Skin Cancer Detection.

المقدمة.

ما هو نموذج YOLOv8x؟

تطبيق على تصنیف مرض سرطان الجلد.

نموذج التدريب.

الاستدلال على صور الاختبار.

تطبيق ويب باستخدام مكتبة Gradio.

التطبيقات والافكار المستقبلية.

ال코드 النهائي.

الملخص.

المقدمة :-

يعد سرطان الجلد أحد أكثر أنواع السرطان شيوعاً على مستوى العالم، حيث يتم تشخيص ملايين الحالات الجديدة كل عام. يعد الاكتشاف المبكر والتشخيص الدقيق أمراً بالغ الأهمية للعلاج الفعال وتحسين نتائج المرضى. تقليدياً، يعتمد الكشف عن سرطان الجلد بشكل كبير على الفحوصات السريرية والغزارات، والتي يمكن أن تستغرق وقتاً طويلاً وغير دقيقة في بعض الأحيان. ومع ذلك، فإن التطورات في الذكاء الاصطناعي (AI) والتعلم الآلي (ML) تحدث ثورة في مجال طب الأمراض الجلدية، حيث تقدم أدوات وأساليب جديدة لتعزيز دقة التشخيص وكفاءته.

في هذا الفصل، ننتمق في الاستخدام المبكر لنموذج YOLOv8 (أنت تنظر مرة واحدة فقط، الإصدار 8) للكشف عن سرطان الجلد. YOLOv8 هي خوارزمية حديثة للكشف عن الأشياء تشتهر بسرعتها ودقتها، مما يجعلها مرشحاً واعداً لتحليل الصور الطبية. ومن خلال الاستفادة من YOLOv8، نهدف إلى تطوير نظام آلي قادر على تحديد أنواع مختلفة من سرطان الجلد من خلال صور تنظير الجلد بدقة عالية.

ما هو نموذج YOLOv8 ؟

هو أحدث نموذج في سلسلة خوارزمية - YOLO العائلة الأكثر شهرة لنماذج اكتشاف الكائنات وتصنيفها في مجال رؤية الكمبيوتر (CV). مع الإصدار الأحدث، يستمر إرث YOLO من خلال توفير أحدث النتائج لتحليلات الصور أو الفيديو، مع إطار عمل سهل التنفيذ. سنشانق في هذه المقالة: تطور خوارزميات YOLO التحسينات والتحسينات في تفاصيل تنفيذ YOLOv8 ونصائح التطبيقات.

ما هو YOLO ؟ أنت تنظر مرة واحدة فقط (YOLO) هي خوارزمية للكشف عن الأشياء تم تقديمها في عام 2015 في ورقة بحثية قام بها جوزيف ريدمون، وسانتوش ديفالا، وروس جيرشيك، وعلى فرهادي. كانت بنية YOLO بمثابة ثورة كبيرة في مجال الكشف عن الأشياء في الوقت الفعلي، متجاوزة سابقتها - الشبكة العصبية التلفيفية القائمة على المنطقة YOLO (R-CNN). عبارة عن خوارزمية أحادية اللقطة تقوم بتصنيف كائن يشكل مباشر في مسار واحد من خلال وجود شبكة عصبية واحدة فقط تتبع بالمربيعات المحاطة واحتمالات الفتنة باستخدام صورة كاملة كمدخل. نموذج YOLO العائلي يتطور باستمرار. أصدرت العديد من فرق البحث منذ ذلك الحين إصدارات مختلفة من YOLO ، وكان YOLOv8 هو الإصدار الأحدث (حالياً الإصدار 9).

Ultralytics YOLOv8 •

أحدث إصدار من YOLO تم إصداره في يناير 2023. يتميز بدقة أعلى وسرعة أكبر. على سبيل المثال، حصل YOLOv8 (متوسط) على نتيجة mAP 50.2 عند 1.83 ملي ثانية على مجموعة بيانات COCO وأ. A100 TensorRT v8. يتميز YOLOv8 أيضاً بحزمة Python والتنفيذ المستند إلى CLI، مما يجعله سهل الاستخدام والتطوير. دعونا نلقي نظرة فاحصة على ما يمكن أن يفعله YOLOv8 ونستكشف بعضاً من تطوراته المهمة. النموذج المُدرِّب مسبقاً YOLO v8 قادر على اكتشاف الكائنات في صورة أو فيديو مباشر مهم YOLOv8 يأتي في خمسة أشكال مختلفة بناءً على عدد المعلمات - نانو (n)، صغير (صغير)، متوسط (m)، كبير (l)، وكبير جداً (x). يمكنك استخدام جميع المتغيرات للتصنيف واكتشاف الكائنات والتجزئة. تصنيف الصور يتضمن التصنيف تصنيف صورة بأكملها دون تحديد موضع الكائن الموجود داخل الصورة.

يمكنك تنفيذ التصنيف باستخدام YOLOv8 عن طريق إضافة الملاحة -cls إلى إصدار YOLOv8 على سبيل المثال، يمكنك استخدام yolov8n-cls.pt للتصنيف إذا كنت ترغب في استخدام إصدار الناتو. توجد خمسة نماذج في كل فئة من نماذج YOLOv8 للكشف والتجزئة والتصنيف. YOLOv8 Nano هو الأسرع والأصغر، في حين أن YOLOv8 Extra Large هو الأكثر دقة ولكنه الأبطأ فيما بينها. (YOLOv8x)

YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
---------	---------	---------	---------	---------

صورة رقم(1)

حيث يقوم النموذج بكشف أو اكتشاف الكائنات (الفنان) بتحديد موضع كانن داخل الصورة عن طريق رسم مربعات محبيطة. ليس عليك إضافة أي لاحقة لاستخدام YOLOv8 للكشف. يتطلب التنفيذ فقط تحديد النموذج على أنه yolov8x.pt لاكتشاف الكائنات باستخدام متغير نوع النموذج المذكور سابقاً. لكننا في هذا الفصل سوف لنتعمق في البنية لهذا النموذج لأنّه خارج موضوع وساق الكتاب وموضوع جداً مطول حول البنية الهيكلية بامكانك الاطلاع على الهيكلية في الورقة البحثية. أو البحث عنها في محرك البحث كوكل.

<https://arxiv.org/abs/2305.09972>

ستتعلم في هذا الفصل كيفية إنشاء مشروع لاكتشاف موقع مرض سرطان الجلد على مجموعة بيانات مخصصة، باستخدام أحدث تقنيات وخوارزمية (YOLOv8) الذي طورته (Ultralytics). سنستخدم تقنيات نقل التعلم لتدريب نموذجنا الخاص وتقييم أدائه واستخدامه للاستدلال. طبعاً هذا التطبيق والمشروع العملي موجه للأشخاص الذين لديهم خلفية نظرية لخوارزميات اكتشاف الأشياء، والذين يسعون للحصول على إرشادات عملية للتنفيذ. يتم توفير محرر Jupiter سهل الاستخدام مع الرمز الكامل أدناه لراحتك. أو بإمكانك استخدام الخادم السحابي كوكل كولاب السريع والم مجاني وتم الحديث والكتابة عنه سابقاً في الفصول السابقة.

• تطبيق على كشف مرض سرطان الجلد.

• مجموعة البيانات.

بعد حصولنا وخذن مجموعة البيانات (راجع موضوع تنزيل مجموعة البيانات) أو ما تسمى Dataset سوف نبدأ بالخطوات البرمجية التالية:-

```
#yolov8 install pip
!pip install ultralytics
```

حيث يقوم هذا الكود بتنصيب مكتبة (Yolov8) لجميع النماذج المذكورة وهي كالتالي:-

- **Yolov8n**
- **Yolov8s**
- **Yolov8m**
- **Yolov8x**
- **Yolov8xl**

لكننا سوف نتعمق فقط في نموذج الكشف الكبير (yolov8x.pt) وحاول بتجربة نماذج الكشف الأخرى لترى الفرق والدقة والسرعة وحجم النموذج ومقاييس الدقة الأخرى.

• **ال코드 البرمجي التالي:-**

```
# upload Bone Tumor Dataset
from google.colab import files
# Prompt the user to upload a file
uploaded = files.upload()
# Loop through the uploaded files
for filename in uploaded.keys():
    print(f'Uploaded file: {filename}')
```

حيث يقوم الكود برفع الداتاسيت المضغوطة والذي تم انزلتها وخرزها في الحاسبة الشخصية والتي تحتوي على ثلاثة مجلدات وهي بيانات التدريب والتحقق والاختبار مع ملف التهيئة والذي يسمى بملفات (yaml) وهي جدا مهمة والتي تحدد مسارات المجلدات وعدد الفئات التي سوف يتم الكشف عنها واسماء الفئات كملخص لها ضمن اعدادات تدريب النموذج المخصص لنا في هذه الحالة لدينا فئة واحدة هي كشف السرطان بامكانك تحرير الملف وتعديل مسارات مجلدات الصور.

• **ال코드 البرمجي التالي.**

```
!unzip /content/dataset.zip -d dataset
!rm -r /content/dataset.zip
```

وهو واضح فتح الضغط وخرن جميع المجلدات في مجلد الداتاسيت ومن ثم مسح المجلد المضغوط

• **ال코드 البرمجي التالي.**

حيث يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد وهو جدا مهم لمعرفة حجم مجموعة البيانات تحت التدريب.

```
import os
def count_files_with_extension(folder_path, extension):
    file_count = 0
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            _, file_extension = os.path.splitext(file)
            if file_extension.lower() == extension:
                file_count += 1
    return file_count
```

```
def main():
    folders = ['train', 'valid', 'test']
```

```
image_extension = '.jpg'
for folder in folders:
    folder_path = os.path.join(folder, 'images')
    file_count = count_files_with_extension(folder_path, image_extension)
    print(f"Folder: {folder}, Image Files: {file_count}")
if __name__ == "__main__":
    main()
```

بعد التنفيذ.

```
Folder: train, Image Files: 1880
Folder: valid, Image Files: 89
Folder: test, Image Files: 11
```

• شرح الكود البرمجي بالتفصيل.

استيراد الوحدات المطلوبة:

تحديد الدالة `:count_files_with_extension`:

تأخذ هذه الوظيفة وسيطتين `Folder_path` (مسار المجلد للبحث عن الملفات) والامتداد (امتداد الملف المراد حسابه). ويقوم بتهيئة عدد الملفات المتغير لتخزين عدد الملفات ذات الامتداد المحدد.

ويستخدم الدالة `(os.walk()` للتنقل عبر كافة الأدلة والأدلة الفرعية الموجودة ضمن المجلد `path` المحدد. بالنسبة لكل ملف تتم مواجهته، يقوم باستخراج امتداد الملف باستخدام `(os.path.splitext()` ومقارنته بالامتداد المقدم. إذا كانت متطابقة، فإنه يزيد `file_count`.

وأخيراً، تقوم بإرجاع العدد الإجمالي لملفات ذات الامتداد المحدد الموجود في المجلد.

تحديد الوظيفة الرئيسية:

تقوم هذه الوظيفة بتنسيق عملية العد لكل مجلد من المجلدات الرئيسية الثلاثة: "التدريب" و"التحقق" و"الاختبار".

يقوم بتهيئة قائمة المجلدات التي تحتوي على أسماء المجلدات الرئيسية. بالنسبة لكل اسم مجلد في المجلدات، يقوم بإنشاء المسار إلى المجلد الفرعى "الصور" باستخدام `.(os.path.join`

ثم يقوم بعد ذلك باستدعاء وظيفة `count_files_with_extension` بمسار المجلد الذي تم إنشاؤه وامتداد الصورة المحدد `('jpg')`.

وأخيراً، يقوم بطباعة اسم المجلد والعدد المقابل لملفات الصور ذات الامتداد المحدد.

يوفر هذا الكود البرمجي طريقة معيارية وقابلة للتطوير لحساب الملفات ذات الامتداد المحدد داخل مجلدات متعددة، مما يسهل المهام مثل المعالجة المساعدة للبيانات ومشاريع التعلم الآلي والتعلم العميق ومعرفة عدد الصورة في كل مجلد مهم في التدريب.

• الكود البرمجي:.

```
#Read Dataset path for training
!cat /content/dataset/data.yaml
```

• شرح الكود.

- حالياً أنك تستخدم بيئة Jupyter Notebook أو Google Colab للعمل مع مجموعة البيانات الخاصة بك. يُستخدم الأمر `cat` عادةً في أنظمة التشغيل المشابهة لـ Unix وفي دفاتر ملاحظات Jupyter/Colab لعرض محتويات الملف.
- دعنا نحلل ما يفعله هذا الأمر:
- يشير `!` إلى أنه يجب تنفيذ الأمر التالي في غلاف النظام، وليس في Python.
 - `cat` هو أمر يستخدم لسلسلة عرض محتويات الملفات.

لذلك، عند تشغيل `!cat /content/dataset/data.yaml`، فإنه يقرأ ويعرض محتويات الملف الموجود في `.content/dataset/data.yaml/`.

يحتوي الملف "data.yaml" على معلومات التكوين أو البيانات الوصفية حول مجموعة البيانات الخاصة في مشروعنا مثل المسارات إلى الصور، والتسميات المقابلة لها، وأي خطوات معالجة مسبقة، وما إلى ذلك. لظهور لنا نتيجة التنفيذ وهي فئة واحدة اسمها السرطان. وكما قلت سابقاً جداً مهم لحفظ على المسارات وهي واضحة من ملف التكوين. (`data.yaml`)

```
train: ./train/images
val: ./valid/images
test: ./test/images
nc: 1
names: ['cancer']
```

• الكود البرمجي التالي:-

```
#import yolov8 detection modules lib.
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

شرح الكود البرمجي.

سوف نقوم باستيراد وحدات للكشف عن YOLOv8 إلى جانب المكتبات الأخرى شائعة الاستخدام لرؤية الكمبيوتر وتحليل البيانات. فيما يلي تفاصيل كل بيان استيراد:

1. من **YOLO** من **Ultralytics import YOLO**: يؤدي هذا إلى استيراد نموذج الكشف عن الكائنات **YOLO** من **Ultralytics**. Ultralytics هي مكتبة توفر تطبيقات لنماذج التعلم العميق المختلفة لمهام رؤية الكمبيوتر، بما في ذلك اكتشاف الكائنات.

2. يُؤدي هذا إلى استيراد مكتبة OpenCV، وهي مكتبة رؤية كمبيوتر شائعة تستخدم لمهام معالجة الصور المختلفة، مثل قراءة الصور ومعالجتها، واكتشاف الكائنات، وتحليل الفيديو.
3. يُؤدي هذا إلى استيراد وحدة `matplotlib.pyplot as plt` من مكتبة Matplotlib، وهي مكتبة تخطيط لإنشاء تصورات ثابتة ومتحركة وتفاعلية في Python. يتم استخدامه بشكل شائع لتصور البيانات والصور.
4. استيراد البيانات ك `pd`: يُؤدي هذا إلى استيراد مكتبة Pandas، وهي مكتبة قوية لمعالجة البيانات وتحليلها في Python. فهو يوفر هيكل البيانات ووظائفها للعمل مع البيانات المنظمة، مثل إطارات البيانات، التي تشبه الجداول في قاعدة البيانات أو جدول البيانات.
5. يُؤدي هذا إلى استيراد مكتبة NumPy، وهي حزمة أساسية للحوسبة الرقمية في Python. وهو يوفر الدعم للمصفوفات والمصفوفات الكبيرة ومتعددة الأبعاد، إلى جانب مجموعة من الوظائف الرياضية للعمل على هذه المصفوفات بكفاءة.

من خلال هذه المكتبات والوحدات يمكنك الوصول إلى مجموعة واسعة من الأدوات والوظائف لتنفيذ اكتشاف الكائنات YOLOv8 ومعالجة الصور ومقاطع الفيديو وتصور النتائج وتحليل البيانات.

نماذج التدريب.

المستخدم هنا وكما قلت سابقا (yolov8x.pt) بامكانك تغيير نوع النموذج لنرى الفرق. لكن اغلب المشاريع والتطبيقات الصغيرة يفضل استخدام نوع النانو.

```
#Using any model yolov8 ()  
model = YOLO("yolov8x.pt") #you can using another models from yolov8
```

شرح الكود.

نقوم بتهيئة نموذج اكتشاف الكائنات YOLOv8 باستخدام مكتبة Ultralytics.

دعونا نحل الكود:

- YOLO("yolov8x.pt"): يقوم هذا السطر بإنشاء مثيل لنموذج YOLOv8. تحدد الوسيطة "yolov8x.pt" المسار إلى ملف الأوزان المدربة مسبقاً لنموذج YOLOv8. في هذه الحالة، يشير "yolov8n.pt" إلى متغير محدد من نموذج YOLOv8. يمكنك استبدال "yolov8n.pt" بالمسار إلى ملف الأوزان المدربة مسبقاً لأي متغير YOLOv8 آخر متوفّر في مكتبة Ultralytics. وهي كما تم ذكرها سابقاً. من خلال تهيئة نموذج YOLOv8، يمكنك الآن استخدامه لتنفيذ مهام الكشف عن الكائنات على الصور أو مقاطع الفيديو. يتضمن ذلك اكتشاف الكائنات (فأة وعنصر السرطان هنا) وتحديد موقعها داخل الصور، بالإضافة إلى توفير معلومات حول تسميات الفأة ودرجات الثقة المرتبطة بكل فأة تم اكتشافه.

• الكود البرمجي التالي:-

```
model.train(data='/content/dataset/data.yaml', epochs=100)
```

• شرح الكود البرمجي.

سوف نحاول تدريب نموذج YOLOv8 باستخدام مجموعة البيانات الخاصة بنا . حيث تقوم وظيفة `model.train` عادةً بتدريب نموذج YOLOv8 على مجموعة البيانات المحددة. دعنا نحل الكود :

- (`data.yaml`): يحدد هذا المسار إلى ملف تكوين البيانات (`data.yaml`) الذي يحتوي على معلومات حول مجموعة البيانات الخاصة بنا مثل المسارات إلى صورة الملفات، والتعليقات التوضيحية المقابلة لها (ان أمكن)، وأسماء الفئات، وما إلى ذلك. سيستخدم النموذج هذا التكوين للتدريب على مجموعة البيانات الخاصة بنا.

- `epochs=100` : يحدد هذا الباراميتر عدد الحقبات (تكرارات التدريب) التي سيتم تدريب النموذج عليها. يتم تعريف الحقبة الواحدة على أنها تمريرة كاملة عبر مجموعة البيانات بأكملها أثناء التدريب. من خلال ضبط `الحقبات=100` ، فإننا نقوم بتوجيه النموذج للتدريب لمدة 100 حقبة.

بعد تشغيل هذا الكود، سيدأ نموذج YOLOv8 في التدريب على مجموعة البيانات الخاصة بنا باستخدام التكوين المحدد وسيتكرر على مجموعة البيانات لعدد محدد من الحقبات ، ويضبط أوزانها لتقليل الخسارة وتحسين أدائها في اكتشاف الكائنات (فلة السرطان).

يتطلب تدريب نموذج YOLOv8 عادةً موارد حاسوبية كبيرة وقد يستغرق وقتاً طويلاً، اعتماداً على حجم وتعقيد مجموعة البيانات الخاصة بنا بالإضافة إلى الأجهزة التي ستستخدمها للتدريب.

تأكد من تنسيق مجموعة البيانات الخاصة بك بشكل صحيح وأن المسارات المحددة في ملف "data.yaml" صحيحة قبل بدء التدريب. بالإضافة إلى ذلك، تأكد من أن لديك موارد حاسوبية كافية ودعم GPU في حالة التدريب على مجموعة بيانات كبيرة.

• سوف نرى من عملية التدريب الخطوات التالية:-

يبعد أن عملية التدريب على نموذج YOLOv8 قد بدأت، ويختبر النموذج حالياً للتدريب لمدة 100 حقبة. فيما يلي تفاصيل التقدم في التدريب والمعلومات الأساسية المقدمة:

1. الأوزان المدرية مسبقاً المنقولة: قام النموذج بنقل الأوزان من النماذج المدرية مسبقاً، وهي ممارسة شائعة لتهيئة النموذج بميزات تم التعرف عليها مسبقاً قبل الضبط الدقيق لمجموعة بيانات محددة.

2. TensorBoard: TensorBoard هي أداة تصور تستخدم لرصد وتحليل عملية التدريب. توفر الرسالة أمرًا لبدء TensorBoard وعرض تقدم التدريب في متصفح الويب.

3. طبقة التجميد: تم تجميد طبقة واحدة من النموذج، ومن المحتمل أن يمنع ذلك من التحديث أثناء التدريب. غالباً ما يتم إجراء طبقات التجميد للحفاظ على الميزات التي تم تعلمها مسبقاً في سيناريوات تعلم النقل.

4. الدقة المختلطة التقانية (AMP): الدقة المختلطة التقانية هي تقنية تستخدم لتسريع التدريب وتقليل استخدام الذاكرة عن طريق إجراء عمليات حسابية بدقة أقل (على سبيل المثال، نصف الدقة) حيثما أمكن ذلك.

5. إعداد البيانات: تم فحص مجموعات بيانات التدريب والتحقق من الصحة لجمع معلومات حول عدد الصور والensemبلات المتاحة للتدريب والتحقق من الصحة. ربما تم تطبيق تقنيات زيادة البيانات مثل التمويه، والتعميم المتوسط، وتحويل التدرج الرمادي، و CLAHE (معادلة الرسم البياني التكيفي المحدود للبيان) لزيادة بيانات التدريب.

6. اختيار المحسن: تم تحديد المحسن (AdamW) ومعلماته تلقائياً بناءً على مجموعة البيانات وتكوين التدريب.

7. تقدم التدريب: يتم عرض تقدم التدريب كل فترة على حدة، مع عرض مقاييس مثل استخدام ذاكرة وحدة معالجة الرسومات، وفقدان الصندوق، وفقدان الفصل، وفقدان DFL (تعلم التصفية الديناميكي). بالإضافة إلى ذلك، يتم توفير عدد المثلثات (الكائنات) التي تم اكتشافها وحجم الصورة المستخدمة للتدريب.

8. مقاييس التحقق: بعد كل فترة، يتم حساب مقاييس التحقق، بما في ذلك الدقة (P)، والاستدعاء (R)، ومتوسط الدقة (mAP) عند عتبات مختلفة (على سبيل المثال، IoU 0.5) لكل فئة وبشكل عام .

بشكل عام، تسير عملية التدريب كما هو متوقع، حيث يتم تدريب النموذج على مجموعة البيانات المحددة وتقييمه بشكل دوري لتقييم أدائه. يمكن أن تساعد المعلومات المقدمة في مراقبة التقدم المحرز في عملية التدريب وفعاليتها.

• نتائج التدريب:-

تحتوي قاموس النتائج المقدم على مقاييس تقييم مختلفة لنموذج الكشف عن سرطان الجلد. دعونا نفسر كل مقاييس:

:Precision .1

- تمثل الدقة نسبة المناطق السرطانية التي تم التنبؤ بها بشكل صحيح من بين جميع المناطق التي تم التنبؤ بأنها سرطانية. تشير درجة الدقة البالغة 0.898 تقريرًا إلى أن حوالي 89.8% من المناطق المصنفة على أنها سرطانية هي بالفعل سرطانية.

:Recall .2

- الاسترجاع، المعروف أيضًا باسم الحساسية، يقيس نسبة المناطق السرطانية الفعلية التي تم تحديدها بشكل صحيح بواسطة النموذج. تشير درجة الاستدعاء البالغة 0.735 تقريرًا إلى أن النموذج اكتشف حوالي 73.5% من المناطق السرطانية الفعلية.

mAP50 .3

- يمثل mAP50 متوسط الدقة عند عتبة الثقة البالغة 50%. فهو يجمع بين الدقة والتذكر عبر عتبات الثقة المختلفة لتقدير أداء النموذج. تشير النتيجة التي تبلغ حوالي 0.823 إلى دقة إجمالية عالية في اكتشاف المناطق السرطانية عبر مستويات ثقة مختلفة.

:mAP50-95 .4

- يمثل mAP50-95 متوسط الدقة عبر عتبات الثقة المختلفة التي تتراوح من 50% إلى 95%. يوفر هذا المقياس رؤى حول دقة النموذج عبر نطاق أوسع من مستويات الثقة. تشير النتيجة التي تبلغ حوالي 0.413 إلى أن النموذج يحافظ على دقة عالية نسبيًا حتى عند عتبات الثقة الأعلى.

بشكل عام، تشير مقاييس التقييم هذه إلى أن نموذج الكشف عن سرطان الجلد يحقق دقة عالية واسترجاعاً، مع أداء قوي عبر عينات الثقة المختلفة. ومع ذلك، قد تكون هناك حاجة إلى مزيد من التحقق من الصحة والاختبار لتقييم مدى تعليم النموذج ومدى ملاءمته لتطبيقات العالم الحقيقي.

الاستدلال على صور الاختبار.

• الكود البرمجي التالي::

#Predict the model

```
results = model.predict(source='/content/dataset/test/images', save = True)
```

سوف نستخدم النموذج النهائي المدرب والذي تم خزنه في مجلد الأوزان التابع للمجلد الرئيسي (runs) (إجراء تنبؤات واستدلال على مجموعة بيانات اختبارية. دعنا نحلل الكود :

- **'source=/content/dataset/test/images'**: يحدد هذا المسار إلى الدليل الذي يحتوي على صور الاختبار التي تريد إجراء التنبؤات عليها. سيقوم النموذج بتحليل الصور الموجودة في هذا الدليل وإنشاء تنبؤات للكائنات الموجودة في الصور.

- **'save=True'**: يشير هذا الباراميتر إلى ما إذا كان سيتم حفظ النتائج المتوقعة أم لا. من خلال تعين `save=True``، فإنك تقوم بتوجيهه النموذج لحفظ المربعات المحيطة المتوقعة واسميات الصنف، ودرجات الثقة لكل فئة وصنف تم اكتشافه.

بعد تشغيل هذا الكود، سيقوم النموذج الخاص بتحليل الصور الموجودة في الدليل المحدد (`"/content/dataset/test/images"`) وإنشاء تنبؤات للفئات الموجودة في تلك الصور. ستتضمن النتائج المتوقعة المربعات المحيطة (إحداثيات الفئة المكتشفة)، واسميات الفئة (على سبيل المثال، "سرطان")، ودرجات الثقة (تشير إلى ثقة النموذج في تنبؤاته).



صورة(2) تظهر موقع السرطان من عظم الركبة

بامكانك ضغط جميع المجلدات ومجلد المقاييس والأوزان وخرزها في الحاسبة من خلال تنفيذ كود موجود ضمن الخادم السحابي كولاب لغرض تجربته وعمل الاستدلال لاي صورة اختبارية. وخاصة النموذج النهائي باسم .(best.pt)

• الكود البرمجي التالي:-

```
#PREDCITION: using Custom Trained best.pt in local
from ultralytics import YOLO
model=YOLO("best.pt")
#download from runs/detect/train/weights/best.pt suppose trained in GColab
results=model(source="Video.mp4",save=True,conf=0.4)
```

• شرح الكود:-

يستخدم الكود البرمجي هذا نموذج YOLOv8 الذي تم تحميله من "best.pt" لاكتشاف الكائنات الموجودة في ملف الفيديو "Video.mp4". يحدد الباراميتر conf حد الثقة لاكتشاف الكائنات، مما يضمن أن الاكتشافات ذات درجات الثقة الأعلى من 0.4 فقط هي التي تعتبر صالحة.

سيحتوي متغير النتائج على نتائج الاكتشاف، والتي قد تتضمن الفيديو المشروع مع المربعات المحيطة حول الفئة أو الصنف المكتشفة في الفيديو وهي فئة السرطان ، بالإضافة إلى المعلومات الأخرى ذات الصلة مثل اكتشاف حدود الصندوق المحاط بالفئة ودرجة الثقة.

تأكد من ضبط المسارات وأسماء الملفات وفقاً لبنية الدليل وأسماء الملفات المحددة. بالإضافة إلى ذلك، تأكد من أن ملف نموذج التحقق "best.pt" مدرب بشكل صحيح ومتاح للاستخدام.

▪ تطبيق ويب باستخدام مكتبة Gradio .

بعد حصولنا على النموذج النهائي وخرزه في الحاسبة حان الوقت لتجربته في واجهة المستخدم كعمل تطبيق ويب لغرض التفاعل بين المستخدم والتطبيق. فيما يلي كود برمجي يحتوي على تفاصيل إضافية حول واجهة المستخدم باستخدام وحدة Gradio الخاصة بكود الكشف عن سرطان الجلد:

ملاحظة:- تم شرح كيفية تشغيل تطبيق الويب سابقاً وكيفية تنصيب وحدة Gradio .

```
import gradio as gr
from ultralytics import YOLO
from PIL import Image
import os
import uuid
model = YOLO("best.pt")
```

```

def detect_objects(input_image_pil):
    # Ensure the input image is in RGB format
    if input_image_pil.mode != 'RGB':
        input_image_pil = input_image_pil.convert('RGB')
    # Generate a unique filename for the uploaded image
    unique_filename = str(uuid.uuid4())
    input_file_path = f"{unique_filename}.jpg"
    input_image_pil.save(input_file_path)
    # Process the image with the YOLO model
    result = model(source=input_file_path, save=True, project="detect",
name="inference", exist_ok=True, conf=0.4)
    # Display the result
    processed_filename = f"{os.path.splitext(os.path.basename(input_file_path))[0]}.jpg"
    # Load the processed image as a PIL Image
    processed_image = Image.open(f"detect/inference/{processed_filename}")
    return input_image_pil, processed_image
iface = gr.Interface(fn=detect_objects, inputs=gr.Image(type="pil"), outputs=["image",
"image"])
iface.launch()

```

• شرح واجهة المستخدم مع وحدة Gradio .:

يستخدم الكود المقدم وحدة Gradio لإنشاء واجهة سهلة الاستخدام للكشف عن سرطان الجلد باستخدام نموذج YOLOv8. يعمل Gradio على تبسيط عملية إنشاء واجهات ويب تفاعلية لنماذج الذكاء الاصطناعي وكما تم شرحه في الفصول السابقة ، مما يجعلها في متناول كل من المطورين والمستخدمين النهائيين.

• تفاصيل الكود:

1. دمج نموذج YOLOv8 :

- تم دمج نموذجنا الذي تم تدريبه سابقا وخزنه في الحاسبة وهذا (best.pt) للكشف عن سرطان الجلد في واجهة Gradio .
- عند تلقي صورة مدخلة، يقوم النموذج بمعالجتها لتحديد المناطق السرطانية المحتملة.

2. معالجة الصور وعرضها:

- يتم تحويل الصور المدخلة إلى تنسيق PIL وحفظها محلياً.
- يقوم النموذج بمعالجة الصورة ويوفر نتيجة توضح الكائنات المكتشفة، بما في ذلك السرطانية المحتملة.
- يتم عرض كل من صورة الإدخال الأصلية والصورة المعالجة مع الكائنات المكتشفة في الواجهة.

3. أسماء ملفات الصور الفريدة:

- لمنع تعارض أسماء الملفات، يتم تعين اسم ملف فريد لكل صورة تم تحميلها باستخدام الوحدة النمطية `uuid` .

4. عتبة الثقة:

- تم تعين عتبة الثقة البالغة 0.4 للكشف عن الكائنات. تعتبر الكائنات ذات درجة الثقة أعلى من هذا الحد اكتشافات صالحة.

5. إطلاق الواجهة:

- يتم تشغيل واجهة Gradio باستخدام وظيفة `gr.Interface`، مع تحديد وظيفة الكشف `detect_objects` كتنسيق الإدخال والإخراج للصور.

آخر الكود تحت ملف (app.py) وعند نافذة الاوامر نفذ الامر التالي (python app.py) سوف يتم عرض العنوان المحلي لشبكتك الخاصة انسخ رابط العنوان على اي متصفح . لظهور لك واجهة المستخدم كتطبيق ويب.



صورة (3) تظهر واجهة اليسار رفع الصورة وعلى يسار الصورة تظهر صورتين الصورة الاعلى الرئيسية قبل المعالجة والصورة السفلى بعد المعالجة . تم قطع الصورة بسبب كبر الواجهة للتتبّع للقاري الكريم.

التطبيقات والأفكار المستقبلية.

فيما يلي بعض التطبيقات والأفكار المستقبلية لتطبيق الكشف عن سرطان الجلد باستخدام نموذج YOLOv8:

1. برامج الفحص الآلي: يمكنك دمج التطبيق في برامج الفحص الآلي، مما يتيح إجراء فحص واسع النطاق لصور الأشعة السينية والرنين المغناطيسي للكشف المبكر عن سرطان الجلد لدى السكان المعرضين للخطر.
2. التطبيب عن بعد والاستشارات عن بعد: تسهيل الاستشارات الطبية عن بعد من خلال السماح لمقدمي الرعاية الصحية بتحميل صور المرضى للتفسير والتشخيص عن بعد، خاصة في المناطق ذات الوصول المحدود إلى المرافق الطبية المتخصصة.

3. **التكامل مع الواقع المعزز (AR):** استكشف تكامل تقنية الواقع المعزز لتركيب المناطق السرطانية المكتشفة على صور الأشعة السينية أو التصوير بالرنين المغناطيسي في الوقت الفعلي أثناء العمليات الجراحية، مما يساعد الجراحين في تحديد مكان الورم واستصاله بدقة.
4. **تخطيط العلاج المخصص:** قم بتحسين التطبيق لت تقديم توصيات علاجية مخصصة بناءً على حجم الأورام المكتشفة وموقعها وشدة، مما يؤدي إلى تحسين نتائج العلاج ومسارات رعاية المرضى.
5. **المراقبة الطولية:** تنفيذ ميزات المراقبة الطولية لتطور الورم والاستجابة للعلاج بمرور الوقت، مما يسمح للأطباء تتبع التغيرات في حجم الورم وشكله لإجراء تعديلات علاجية مخصصة.
6. **منصات التشخيص التعاونية:** قم بتطوير منصات تشخيصية تعاونية حيث يمكن للعديد من المتخصصين في الرعاية الصحية الوصول بشكل آمن إلى صور المرضي وتحليلها، وتعزيز التعاون متعدد التخصصات واتخاذ القرارات على أساس الإجماع.
7. **التكامل مع السجلات الصحية الإلكترونية (EHR):** يمكنك دمج التطبيق مع أنظمة السجلات الصحية الإلكترونية لتبسيط عمليات سير العمل، وضمان الوصول السلس إلى بيانات تصوير المريض والتوثيق الآلي لنتائج التشخيص ضمن السجل الطبي للمريض.
8. **الأبحاث السريرية وتحليلات البيانات:** استفد من إمكانات تحليل بيانات التطبيق لإخفاء هوية بيانات المرضي وتجميعها لأغراض البحث السريري، مما يساهم في تطوير استراتيجيات علاجية جديدة ونمذجة تنبؤية لإدارة سرطان الجلد.
9. **أدوات التعليم والتدريب:** أنشئ مواد تعليمية ووحدات تدريبية باستخدام التطبيق لتنمية المتخصصين في الرعاية الصحية حول تفسير نتائج تصوير سرطان الجلد واستخدام أدوات التشخيص المعتمدة على الذكاء الاصطناعي في الممارسة السريرية.
10. **التحسين المستمر للنموذج:** التحديث المستمر وتحسين نموذج YOLOv8 باستخدام تعليقات المستخدمين والتقدم البحثي المستمر، مما يضمن فعاليته ودقته في اكتشاف سرطان الجلد والتكيف مع تحديات التشخيص المتطرفة.

ومن خلال استكشاف هذه التطبيقات والأفكار المستقبلية، يمكن أن يتطور تطبيق الكشف عن سرطان الجلد إلى أداة متعددة الاستخدامات لا تساعد فقط في التشخيص المبكر وتخطيط العلاج ولكنها تسهم أيضًا في التقدم في رعاية مرضي السرطان وأبحاثهم.

اللого البرمجي النهائي



الملخص

في هذا الفصل من الكتاب، استكشفنا الإمكانيات التحويلية لنموذج **YOLOv8x** في مجال الكشف عن سرطان الجلد، بدءاً من المراحل الأولية لجمع مجموعة البيانات وحتى نشر واجهة سهلة الاستخدام قادرة على تحليل صور الاختبار. بدأت رحلتنا بعملية دقيقة لجمع الصور بمناظر الجلد والتعليق عليها لإنشاء مجموعة بيانات شاملة تضمن مئنة ودقة نموذج **YOLOv8x**.

ركز جوهر الفصل على نموذج **YOLOv8x**، وهو تكرار متقدم لبنية **YOLO**، المعروفة بسرعتها ودققتها الاستثنائية في مهام اكتشاف الكائنات. لقد بحثنا في تعقيدات التدريب وضبط النموذج، مع التركيز على أهمية تحسين البارامترات واستخدام وظائف الخسارة المتموّلة ضمن النموذج الرئيسي لتعزيز أداء نموذجنا الخاص بنا. أظهرت عملية التقييم الصارمة التي قمنا بها، باستخدام مقاييس مثل الدقة والاستدعاء ودرجة F1 والتقطاع على الاتحاد (IoU)، قدرة النموذج على اكتشاف أنواع مختلفة من سرطان الجلد بدقة وموثوقية عالية.

وايضا ختمنا التطبيق بקוד برمجي قصير ومهم وهو تطوير واجهة مستخدم بديهية بستخدام مكتبة واجهات الويب وهي مكتبة (**Gradio**). تتيح هذه الواجهة للأطباء والمستخدمين تحميل صور الاختبار وتلقي تعليقات تشخيصية في الوقت الفعلي لأسم الفئة التي تم اكتشافها وهنا (cancer)، وبالتالي سد الفجوة بين تقنية الذكاء الاصطناعي المتقدمة والتطبيق السريري العملي. لقد سلطنا الضوء على أهمية مبادئ التصميم التي تركز على المستخدم في إنشاء واجهة ليست وظيفية فحسب، بل يمكن الوصول إليها بسهولة واستخدامها أيضاً.

وبالنظر إلى المستقبل، فإن مستقبل اكتشاف سرطان الجلد المعتمد على الذكاء الاصطناعي مشرق، ولكنه مليء بالتحديات والفرص. أحد الاتجاهات الواعدة هو دمج البيانات متعددة الوسائل، والجمع بين الصور التنبؤية الجلدية والمعلومات التشخيصية الأخرى مثل تاريخ المريض والبيانات الوراثية لتحسين دقة التشخيص. يعده تعزيز إمكانية تفسير نماذج الذكاء الاصطناعي مجالاً بالغ الأهمية، مما يضمن ثقة الأطباء في عملية صنع القرار في الذكاء الاصطناعي وفهمها. ستكون معالجة المخاوف الأخلاقية والخصوصية أمراً بالغ الأهمية ونحن نسعى جاهدين لتطبيق هذه التقنيات في البيانات السريرية في العالم الحقيقي.

علاوة على ذلك، فإن التقدم في التعلم الموحد يمكن أن يمكن من تطوير نماذج مدربة علىمجموعات بيانات متنوعة من مؤسسات متعددة دون المساس بخصوصية المريض. يمكن لهذا النهج أن يعزز بشكل كبير إمكانية تعليم نماذج الذكاء الاصطناعي، مما يجعلها أكثر قوة عبر مختلف المجموعات السكانية والديموغرافية والجغرافية.

وفي الختام، فإن تطبيق نموذج **YOLOv8x** في الكشف عن سرطان الجلد يمثل قفزة كبيرة إلى الأمام في تحليل الصور الطبية. بدءاً من الإعداد الدقيق لمجموعات البيانات وحتى نشر واجهة تشخيصية سهلة الاستخدام، قمنا بعرض سير العمل الشامل المطلوب لتسخير قوة الذكاء الاصطناعي في طب الأمراض الجلدية. ومع استمرارنا في تحسين هذه التقنيات ومواجهة التحديات المقبلة، أصبح إمكانية إحداث ثورة في تشخيص سرطان الجلد وتحسين نتائج المرضى ممكنة التحقيق بشكل متزايد. يعد هذا الكتاب بمثابة دليل وإلهام للباحثين والأطباء وممارسي الذكاء الاصطناعي الذين يكرسون جهودهم لتطوير مجال التشخيص الطبي من خلال حلول الذكاء الاصطناعي المبتكرة.

المقدمة.

نقل التعلم ونموذج ResNet50

تدريب النموذج.

خدمة تطبيق الويب.

الكود النهائي.

الملخص.

ما هو مرض هشاشة عظام الركبة؟ يتم تعريف هشاشة العظام في الركبة عن طريق تدهور الغضروف المفصلي للركبة، وهي مادة مرنّة وزلقة تحمي العظام عادة من احتكاك المفاصل والاصدمات. تتضمن الحالة أيضًا تغيرات في العظام الموجودة أسفل الغضروف ويمكن أن تؤثر على الأنسجة الرخوة القربيّة. يعد التهاب مفاصل الركبة هو النوع الأكثر شيوعاً من التهاب المفاصل الذي يسبب آلام الركبة وغالباً ما يشار إليه ببساطة باسم التهاب مفاصل الركبة. يمكن للعديد من أنواع التهاب المفاصل الأخرى الأقل شيوعاً أن تسبب آلاماً في الركبة، بما في ذلك التهاب المفاصل الروماتويدي والنقرس الكاذب والتّهاب المفاصل التفاعلي. من الضروري إجراء بعض الإجراءات الطبية للتعرف على هذا المرض، مثل الأشعة السينية أو التصوير بالرنين المغناطيسي، حيث يمكن تقييم الفقد في المباعدة بين المفاصل، وبالتالي الإشارة إلى مدى خطورة المرض.

تم تصنیف شدة هشاشة العظام إلى 5 مستويات على أساس درجة KL، من المستوى الصحي إلى المستوى الشديد، حيث كلما زادت درجة الشدة، قلت المسافة بين المفصل.

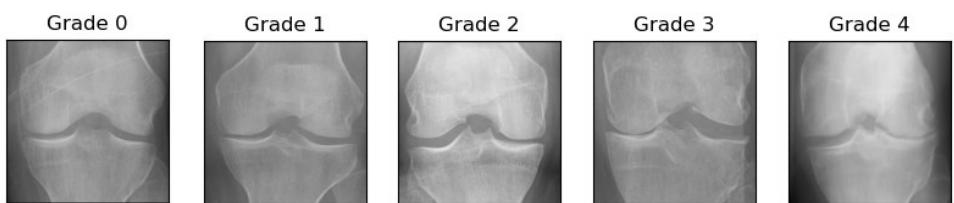
• مجموعة البيانات (Dataset).

تحتوي مجموعة البيانات هذه على بيانات الأشعة السينية للركبة للكشف عن مفصل الركبة وتصنیف KL للركبة. أوصاف الدرجة هي كما يلي:

• الدرجة 0: صورة الركبة الصحية.

- **الدرجة الأولى (مشكوك فيها):** تضيق مفصل مشكوك فيه مع احتمال حدوث شفة عظمية.
- **الدرجة الثانية (الحد الأدنى):** وجود مؤكد للنابتات العظمية واحتمال تضيق مساحة المفصل.
- **الدرجة 3 (مُعتَدِل):** نباتات عظمية متعددة، تضيق واضح في مساحة المفصل، مع تصلب خفيف.
- **الدرجة الرابعة (شديدة):** وجود نباتات عظمية كبيرة، وتضيق كبير في المفاصل، وتصلب حاد.

- **Grade 0: Healthy**
- **Grade 1: Doubtful**
- **Grade 2: Minimal**
- **Grade 3: Moderate**
- **Grade 4: Severe**



صورة رقم(1) توضح الصورة التالية المستويات المختلفة من مجموعة بيانات التهاب مفاصل الركبة مع تصنیف الخطورة.

نقل التعلم ونموذج ResNet50

نقل التعلم هو أسلوب في التعلم الآلي حيث يتم تطبيق المعرفة المكتسبة من تدريب نموذج على مهمة واحدة على مهمة مختلفة ولكن ذات صلة. يكون ذلك مفيداً بشكل خاص عندما تكون لديك بيانات محدودة لمهمتك المستهدفة، حيث يمكنك الاستفادة من المعرفة المستفادة من نموذج تم تدريسه مسبقاً على مجموعة بيانات كبيرة.

عبارة عن بنية شبكة عصبية تلaffيفية شائعة (CNN) معروفة بأدائها المذهل في مهام تصنيف الصور. ويستخدم "الكتل المتبقية" لتسهيل تدفق التدرجات وتحسين استقرار التدريب، مما يسمح بشبكات أعمق دون تدهور الأداء.

• الجمع بين نقل التعلم وResNet50

باستخدام ResNet50 كنموذج تم تدريسه مسبقاً لنقل التعلم، يمكنك الاستفادة من تمثيلات الميزات الغنية التي تم تعلمها من مجموعة بيانات صور واسعة مثل ImageNet. وإليك كيف يعمل:

1. **ResNet50 المُدرب مسبقاً:** لقد تعلم نموذج ResNet50 المُدرب مسبقاً كيفية تحديد الميزات واستخراجها من الصور.

2. **تجميد الطبقات السابقة:** لتكيف النموذج مع مهمتك المحددة، يمكنك بشكل عام تجميد الطبقات الأولية لـ ResNet50، المسئولة عن استخراج الميزات ذات المستوى المنخفض. يؤدي هذا إلى الحفاظ على المعرفة العامة للنموذج حول ميزات الصورة.

3. **استبدال/ضبط الطبقات النهائية:** يتم استبدال الطبقات النهائية لـ ResNet50، المسئولة عن التصنيف، أو ضبطها لتناسب مهمتك المحددة. قد يتضمن ذلك إضافة طبقات جديدة لفناتيك المحددة أو تعديل الطبقات الموجودة للتعرف على بياناتك الفريدة.

• المزايا:

تدريب أسرع: يمكنك تدريب النموذج الخاص بك بشكل أسرع حيث يتم تدريب الطبقات الأولية مسبقاً.

أداء أفضل: غالباً ما تؤدي الاستفادة من الميزات التي تم تدريسيها مسبقاً إلى أداء أفضل، خاصة مع بيانات التدريب المحدودة.

متطلبات بيانات منخفضة: يتطلب نقل التعلم بيانات أقل مقارنة بتدريب النموذج من البداية.

• التطبيقات:

يتم استخدام نقل التعلم باستخدام ResNet50 على نطاق واسع في العديد من التطبيقات المعتمدة على الصور، مثل:

تصنيف الصور: تحديد الكائنات والفنانات المختلفة في الصور.

اكتشاف الكائنات: تحديد موقع الكائنات وتصنيفها داخل الصور.

تقسيم الصورة: إنشاء أقنعة على مستوى البكسل لتقسيم مناطق مختلفة داخل الصورة.

يوفر الجمع بين تعلم النقل وResNet50 طريقة قوية وفعالة للمهام المتعلقة بالصور. فهو يسمح لك بالاستفادة من المعرفة المدربة مسبقاً لتكيف التماذج بسرعة مع تطبيقاتك المحددة، مما يتطلب بيانات ووقتاً أقل مع تحقيق نتائج مذهلة.

تدريب النموذج:-

• الكود البرمجي.

بعد حصولك على مفتاح التخويل من منصة كاكل حان الان تنزيل مجموعة البيانات الصورية لمفاصل الركبة

```
# Colab's file access feature
```

```
from google.colab import files
```

```
#retrieve uploaded file
```

```
uploaded = files.upload()
```

```
#print results
```

```
for fn in uploaded.keys():
```

```
    print('User uploaded file {name} with length {length} bytes'.format(  
        name=fn, length=len(uploaded[fn])))
```

```
# Then move kaggle.json into the folder where the API expects to find it.
```

```
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
```

• الكود البرمجي.

```
! kaggle datasets download -d falahgatea/knee-osteoarthritis-dataset-with-severity-grading
```

سحب مجموعة البيانات من منصة كاكل. بإمكانك استخدام طريقة اخرى بتنزيل المجموعة في الحاسبة الشخصية ورفعها الى الخادم السحابي كوكل كولاب وهذه افضل طريقة لحفظ على مجموعة البيانات لتطبيقات اخرى.

• الكود البرمجي.

```
!unzip /content/knee-osteoarthritis-dataset-with-severity-grading.zip
```

```
!rm -r /content/knee-osteoarthritis-dataset-with-severity-grading.zip
```

فتح الضغط عن مجموعة البيانات ووضعها في مجلد (dataset)

• الكود البرمجي.

```
import os
```

```
base_dir = '/content/dataset'
```

```
train_path = os.path.join(base_dir, 'train')
```

```
valid_path = os.path.join(base_dir, 'val')
```

```
test_path = os.path.join(base_dir, 'test')
```

تحديد مسارات مجلد التدريب والاختبار والدقة.

```
model_name = ResNet50
class_names = [ Healthy , Doubtful , Minimal , Moderate , Severe ]
target_size = (224, 224)
```

```
epochs = 50
batch_size = 256
img_shape = (224, 224, 3)
# Save model
save_model_ft = os.path.join( models , f model_{model_name}_ft.hdf5 )
```

• شرح الكود

فيما يلي تفاصيل كل جزء من الكود :

1. `model_name = ResNet50`

- يستخدم هذا المتغير لتحديد اسم النموذج الذي تم تدريبه مسبقاً والذي سيتم استخدامه وهو ResNet50 عبارة عن بنية شبكة عصبية تلافية (CNN) مستخدمة على نطاق واسع ومعروفة بطبقاتها الخمسين وإطار التعلم المتبقي. تحظى بشعبية كبيرة في مهام تصنیف الصور المختلفة نظرًا لتوازنها بين العمق والكفاءة الحسابية.

2. `= class_names` [صحي ، مشكوك فيه ، حد أدنى ، معتدل ، شديد]

- تحدد هذه القائمة الفئات أو الفئات المختلفة التي سيتنا به النموذج. في هذا السياق، يبدو أنه يتم استخدام النموذج في شكل ما من أشكال تصنیف الصور الطبية حيث يتم تصنیف الصور إلى خمس فئات مختلفة: صحية، مشكوك فيها، حد أدنى، معتدل، وشديد.

3. `حجم الهدف = (224, 224)`

- يحدد هذا الصنف الحجم المستهدف الذي سيتم تغيير حجم كافة الصور المدخلة إليه. بالنسبة لـ ResNet50، يكون حجم الإدخال القياسي 224×224 بكسل. وهذا يضمن أن جميع الصور التي يتم إدخالها في النموذج لها نفس الأبعاد، وهو أمر ضروري للتدريب والتقييم المتسقين للنموذج.

4. `الحقبات = 50`

- يحدد هذا العدد الصحيح عدد الحقبات التي سيتم تدريب النموذج عليها. الحقبة عبارة عن تمريرة كاملة عبر مجموعة بيانات التدريب بأكملها. تعین هذا على 50 يعني أن النموذج سوف يتكرر على مجموعة البيانات 50 مرة أثناء التدريب.

5. `حجم الدفع = 256`

- يحدد هذا العدد الصحيح عدد العينات التي سيتم نشرها عبر الشبكة مرة واحدة. حجم الدفع 256 يعني أن النموذج سيعالج 256 صورة في المرة الواحدة قبل تحديث الأوزان. يمكن أن تؤدي أحجام الدفعات الأكبر إلى تحسين كفاءة التدريب ولكنها تتطلب المزيد من الذاكرة.

6. `(3,224,224) = img_shape`

- يحدد هذا الصنف شكل الصور المدخلة. الشكل (224, 224, 3) يتوافق مع الصور التي يبلغ طولها وعرضها 224 بكسل مع 3 قنوات ألوان (أحمر، أخضر، أزرق). يعد هذا الشكل قياساً لنماذج مثل ResNet50 التي تتوقع صور RGB كمدخلات.

7. save_model_ft = os.path.join(models , f model_{model_name}_ft.hdf5) .

- ينشئ هذا السطر من التعليمات البرمجية مسار ملف لحفظ النموذج المدرب. تضمن وظيفة `os.path.join` إنشاء المسار بشكل صحيح لنظام التشغيل. يتم استخدام `f-string` (سلسلة منسقة) لإدراج متغير `model_name` (وهو `ResNet50` ديناميكياً في اسم الملف). مسار الملف الناتج هو `models/model_ResNet50_ft.hdf5`، حيث سيتم حفظ النموذج بتنسيق `HDF5`، وهو تنسيق شائع لحفظ نماذج `Keras`.

باختصار، تقوم هذه المتغيرات والإعدادات بتكوين نموذج الشبكة العصبية، وتحديد خصائص الصورة المدخلة، وتحديد معلمات التدريب، وإعداد المسار لحفظ النموذج المدرب.

• الكود البرمجي.

```
import os
import timeit
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from sklearn.metrics import (accuracy_score, balanced_accuracy_score,
                             classification_report, confusion_matrix)
from sklearn.utils.class_weight import compute_class_weight
```

• شرح الكود.

فيما يلي توضيحات مختصرة لكل من هذه الواردات:

1. استيراد نظام التشغيل

- يوفر طريقة للتفاعل مع نظام التشغيل (على سبيل المثال، مسارات الملفات).

2. وقت الاستيراد

- يسمح بتوقيت تنفيذ مقطفات التعليمات البرمجية الصغيرة لقياس الأداء.

3. استيراد plt كـ matplotlib.pyplot

- يستخدم لإنشاء تصورات ثابتة وتفاعلية ومحركة في بايثون.

4. استيراد np كـ numpy

- يدعم المصفوفات والمصفوفات الكبيرة ومتعددة الأبعاد، بالإضافة إلى مجموعة من الدوال الرياضية للعمل على هذه المصفوفات.

5. استيراد sns كـ seaborn

- مكتبة تصور البيانات تعتمد على `matplotlib`، وتتوفرواجهة عالية المستوى لرسم رسومات إحصائية جذابة.

6. استيراد tf كـ Tensorflow

- مكتبة مفتوحة المصدر للتعلم الآلي والتعلم العميق، تُستخدم عادةً لبناء الشبكات العصبية وتدربيها.

7. من استيراد `sklearn.metrics` (درجة_الدقة، ودرجة_دقة_التوازن، وتقرير_التصنیف، و_مصفوفة_الارتبك)

- `accuracy_score` : حساب دقة النموذج.

- `balanced_accuracy_score` : يحسب الدقة المتوازنة، مع الأخذ في الاعتبار عدم توازن الفئة.

- `classification_report` : يُنشئ تقريراً تفصيلياً عن الدقة والاستدعاء ودرجة F1 وما إلى ذلك.

- `confusion_matrix` : إنشاء مصفوفة لتقییم أداء نموذج التصنیف.

8. من `compute_class_weight` استيراد `sklearn.utils.class_weight`

- حساب أوزان الفئة لمعالجة عدم توازن الفئة في مجموعات البيانات، مما يجعل تدريب النموذج أكثر فعالية.

• الكود البرمجي.

```
aug_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    preprocessing_function=tf.keras.applications.resnet50.preprocess_input,  
    horizontal_flip=True,  
    brightness_range=[0.3, 0.8],  
    width_shift_range=[-50, 0, 50, 30, -30],  
    zoom_range=0.1,  
    fill_mode=nearest)  
noaug_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    preprocessing_function=tf.keras.applications.resnet50.preprocess_input,)
```

• شرح الكود.

شرحًا قصيراً لكل جزء من الكود:

1. (...) `aug_datagen = tf.keras.preprocessing.image.ImageDataGenerator` :

- `ImageDataGenerator` : يستخدم لإنشاء دفعات من بيانات الصور الموتّرة مع زيادة البيانات في الوقت الفعلي.

`preprocessing_function=tf.keras.applications.resnet50.preprocess_input` : إدخال العمليات المسبقة لـ ResNet50

- `horizontal_flip=True` : لقلب الصور أفقياً بشكل عشوائي.

- `brightness_range=[0.8,0.3]=brightness_range` : تغيير سطوع الصور بشكل عشوائي داخل النطاق.

- `width_shift_range=[30,-50,0,50,0,50-]=width_shift_range` : يقوم بازاحة الصور أفقياً بشكل عشوائي حسب قيم البكسل المحددة.

- `zoom_range=0.1` : تكبير الصور بشكل عشوائي بنسبة تصل إلى 10%.

- `fill_mode= nearest` : يملأ وحدات البكسل المفقودة بعد التحويلات باستخدام قيم أقرب وحدات البكسل.

2. (...) `noaug_datagen = tf.keras.preprocessing.image.ImageDataGenerator` :

- `ImageDataGenerator` : يُنشئ دفعات من بيانات الصورة الموتّرة دون زيادة البيانات.

`preprocessing_function=tf.keras.applications.resnet50.preprocess_input`
: إدخال العمليات المسبقة لـ ResNet50، مما يضمن الاتساق في كيفية معالجة الصور مسبقاً.

يقوم هذا الكود القصير بإنشاء مولدات لتحميل بيانات الصورة وزيادتها بشكل اختياري أثناء تدريب النموذج.

• **الكود البرمجي.**

```
train_generator = aug_datagen.flow_from_directory(  
    train_path, class_mode=categorical, target_size=target_size, shuffle=True )
```

```
valid_generator = noaug_datagen.flow_from_directory(  
    valid_path,  
    class_mode=categorical,  
    target_size=target_size,  
    shuffle=False,)
```

• **شرح الكود.**

شرحًا قصيراً لكل جزء من الكود:

`(...)train_generator = aug_datagen.flow_from_directory` .1

- `aug_datagen` : يستخدم منشئ زيادة البيانات المحدد مسبقاً.

- `flow_from_directory` : تحميل الصور من الدليل على دفعات.

- `train_path` : المسار إلى دليل بيانات التدريب.

- `class_mode= categorical` : يضبط الوضع لتصنیف الصور إلى فئات متعددة.

- `target_size=target_size` : تغيير حجم الصور إلى الحجم المستهدف المحدد (على سبيل المثال، 224 × 224 بكسل).

- `shuffle=True` : خلط البيانات بشكل عشوائي لضمان العشوائية أثناء التدريب.

`(...)valid_generator = noaug_datagen.flow_from_directory` .2

- `noaug_datagen` : يستخدم المولد غير المعزز المحدد مسبقاً.

- `flow_from_directory` : تحميل الصور من الدليل على دفعات.

- `valid_path` : المسار إلى دليل بيانات التحقق من الصحة.

- `class_mode= categorical` : يضبط الوضع لتصنیف الصور إلى فئات متعددة.

- `target_size=target_size` : تغيير حجم الصور إلى الحجم المستهدف المحدد (على سبيل المثال، 224 × 224 بكسل).

- `shuffle=False` : لا يقوم بخلط البيانات، مع الحفاظ على ترتيب التقييم.

• **الكود البرمجي.**

```
y_train = train_generator.labels  
y_val = valid_generator.labels
```

• شرح الكود.

شرحًا قصيراً للكود القصير:

`y_train = Train_generator.labels .1`

: يسترد تسميات الفصل لصور التدريب التي تم إنشاؤها بواسطة `train_generator`

`y_train : يخزن تسميات التدريب هذه في المتغير y_train -`

`y_val = valid_generator.labels .2`

: يسترد تسميات الفئة لصور التحقق من الصحة التي تم إنشاؤها بواسطة `valid_generator`

`y_val : يخزن تسميات التتحقق هذه في المتغير val -`

باختصار، تقوم سطور التعليمات البرمجية هذه باستخراج وتخزين تسميات الفئات لمجموعات بيانات التدريب والتحقق من الصحة من مولدات البيانات الخاصة بها.

• الكود البرمجي.

```
unique, counts = np.unique(y_train, return_counts=True)
print(Train: , dict(zip(unique, counts)))
```

```
class_weights = compute_class_weight(
    class_weight='balanced', classes=np.unique(y_train), y=y_train
)
train_class_weights = dict(enumerate(class_weights))
print(train_class_weights)
```

• شرح الكود.

شرح مختصر لكل جزء من الكود:

`np.unique(y_train, return_counts=True) .1`

- `np.unique(y_train, return_counts=True)` : يبحث عن تسميات الفئات الفريدة في `y_train` ويحسب تكراراتها.

- فريد : يخزن تسميات الفئة الفريدة.

- استرجاع العدد : تخزن أعداد كل تصنيف فئة.

`(dict(zip(unique,counts)),"التدريب: ",)` .2

- `dict(zip(unique, counts))` : إنشاء قاموس يحتوي على تسميات الفئات كمفاتيح وأعدادها كقيم.

- طباعة القاموس الذي يوضح توزيع تسميات الفصل في مجموعة التدريب.

`class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train) .3`

- `compute_class_weight` : يحسب الأوزان لكل فئة لمعالجة عدم توازن الفئة.

- يضمن حساب الأوزان لموازنة الفئات.

- `classes=np.unique(y_train)` : لتمرير تسميات الفصل الفريدة.

- `y=y_train` : اجتياز تسميات التدريب.

الفصل الرابع عشر: تصنیف مرض هشاشة مفاصل الرکبة و درجة الخطورة

4. **train_class_weights = dict(enumerate(class_weights))**

-)**dict(enumerate(class_weights))**: إنشاء قاموس تكون فيه فهارس الفئة مفاتيح والأوزان المقابلة لها هي قيم.

- **train_class_weights** : يخزن قاموس أوزان الفصل.

5. **(train_class_weights)** طباعة

- طباعة القاموس الذي يوضح أوزان الحصص المحسوبة لموازنة الحصص أثناء التدريب.

باختصار، تقوم هذه الأسطر من التعليمات البرمجية بحساب وطباعة توزيع تسميات الفصل في مجموعة التدريب وحساب أوزان الفصل لمعالجة أي خلل.

• الكود البرمجي.

```
classes = np.unique(y_train)
# Callbacks
early = tf.keras.callbacks.EarlyStopping(
    monitor= val_loss , min_delta=0.01, patience=8,
    restore_best_weights=True )
plateau = tf.keras.callbacks.ReduceLROnPlateau(
    monitor= loss , factor=0.1, min_delta=0.01,
    min_lr=1e-10, patience=4, mode= auto )
```

• شرح الكود.

تقوم هذه السطور بإعداد استخراج تصنیف الفصل وتحديد عمليات الاسترجاعات للتوقف المبكر وتقليل معدل التعلم لتحسين عملية التدريب واستقرارها.

شرح مختصر لكل جزء من الكود:

1. **np.unique(y_train)**

- يبحث عن تسميات الفئة او الصنف الفريدة ويخزنها في بيانات التدريب **y_train**

2. **EarlyStopping(...)**

- التوقف المبكر : يوقف التدريب مبكراً إذا لم يتحسن المقياس الذي يتم مراقبته (فقد التحقق من الصحة).

- **monitor= val_loss** : يراقب فقدان التحقق من الصحة.

- **min_delta=0.01** : الحد الأدنى للتغيير للتأهل لتحسين.

- **الصبر=8** : عدد الفترات التي لم يحدث فيها تحسن وبعدها توقف التدريب.

- **restore_best_weights=True** : استعادة أوزان النموذج من العصر مع أفضل فقدان للتحقق من الصحة.

3. **ReduceLROnPlateau = plateau**

- **ReduceLROnPlateau** : يقلل من معدل التعلم عندما يتوقف مقياس (خسارة) مراقب عن التحسن.

- **monitor= loss** : يراقب خسارة التدريب.

- **العامل=0.1** : العامل الذي سيتم من خلاله تخفيض معدل التعلم.

- **min_delta=0.01** : الحد الأدنى للتغيير للتأهل لتحسين.

- **min_lr=1e-10** : الحد الأدنى لمعدل التعلم.

- الفصل الرابع عشر: تصنيف مرض هشاشة مفاصل الركبة ودرجة الخطورة
- الفترة=4 : عدد الفترات التي لم يحدث فيها تحسن وبعدها انخفض معدل التعلم.
 - mode= auto : يقرر تلقائياً ما إذا كان سيتم البحث عن الحد الأقصى أو الحد الأدنى للمقياس.

• الكود البرمجي.

```
model = tf.keras.applications.ResNet50(
    input_shape=(img_shape),
    include_top=False,
    weights=imagenet,)
```

• شرح الكود.

يقوم هذا الكود بتهيئة نموذج ResNet50 بشكل إدخال محدد، بدون طبقة التصنيف العليا، وباستخدام أوزان مدربة مسبقاً من ImageNet. غالباً ما يستخدم هذا التكوين لنقل التعلم حيث يتم ضبط النموذج الأساسي بشكل دقيق لمهمة محددة.

بالتأكيد، إليك شرحاً قصيراً لكل جزء من الكود:

1. النموذج = (...)(tf.keras.applications.ResNet50(...)) : تحميل بنية نموذج ResNet50
2. input_shape=(img_shape) : يضبط شكل إدخال النموذج على (img_shape)، والذي يكون عادةً (224, 224, 3) للصور بحجم 224×224 مع 3 قنوات ألوان (RGB).
3. include_top=False : يستثني الطبقات العليا المتصلة بالكامل بالنماذج. يعد هذا مفيداً لاستخراج الميزات أو عند إضافة طبقات مخصصة لمهمة محددة.
4. weights=imagenet : تحميل الأوزان المدربة مسبقاً من مجموعة بيانات ImageNet، مما يسمح للنموذج بالاستفادة من الميزات المستفادة من مجموعة بيانات كبيرة.

• الكود البرمجي.

```
for layer in model.layers:
    layer.trainable = True

model_ft = tf.keras.models.Sequential([
    model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(5, activation=softmax),
])
model_ft.summary()
```

• شرح الكود.

يقوم هذا الكود بضبط نموذج ResNet50 بالكامل عن طريق جعل جميع طبقاته قابلة للتدريب، ثم يبني نموذجاً جديداً مع متوسط عالمي إضافي للتجميع والتسلسلي وطبقات الإخراج الكثيفة لمهمة تصنيف من 5 فئات.
شرح مختصر لكل جزء من الكود:

1. - جعل جميع الطبقات في النموذج الأساسي ResNet50 قابلة للتدريب، مما يسمح بتحديث أوزانها أثناء التدريب.

(...)model_ft = tf.keras.models.Sequential .2

- tf.keras.models.Sequential : إنشاء نموذج تسلسلي جديد عن طريق تكديس الطبقات.

3. النموذج

- يضيف النموذج الأساسي ResNet50 (مع تعين جميع الطبقات الآن على أنها قابلة للتدريب) طبقة أولى.

(tf.keras.layers.GlobalAveragePooling2D) .4

- يضيف طبقة تجميع عالمية متوسطة لتقليل الأبعاد المكانية لخريطة المعالم من النموذج الأساسي إلى قيمة واحدة لكل خريطة معلم.

(0.2)tf.keras.layers.Dropout .5

- إضافة طبقة تسرّب بمعدل تسرّب يبلغ 0.2 للمساعدة في منع التجهيز الزائد عن طريق تعين 20% من وحدات الإدخال بشكل عشوائي على 0 أثناء كل تحديث أثناء التدريب.

tf.keras.layers.Dense(5,activation='softmax') .6

- إضافة طبقة إخراج متصلة بالكامل (كثيفة) تحتوي على 5 وحدات (المقابلة للفئات الخمس) ووظيفة تنشيط softmax لتصنيف متعدد الفئات.

()model_ft.summary .7

- طباعة ملخص لبنية النموذج، بما في ذلك الطبقات وعدد الأوزان.

• الكود البرمجي.

```
model_ft.compile(  
    optimizer=adam, loss=categorical_crossentropy, metrics=[accuracy] )
```

```
start_ft = timeit.default_timer()  
history = model_ft.fit(  
    train_generator,  
    epochs=epochs,  
    batch_size=batch_size,  
    callbacks=[early, plateau],  
    validation_data=valid_generator,  
    class_weight=train_class_weights,  
    verbose=1, )  
stop_ft = timeit.default_timer()
```

• شرح الكود.

يقوم هذا الكود بتجمیع النموذج المضبوط بدقة (`model_ft`) وتدريبه وتقيیمه باستخدام المحسن المحدد ووظيفة الخسارة والمقایيس، إلى جانب عمليات الاسترجاعات الإضافیة وأوزان الفئات لتحسين أداء التدريب. يتم أيضاً تسجیل الوقت المستغرق للتدريب.

شرح مختصر لكل جزء من الكود:

1. `: (...)model_ft.compile`

- تجمیع " `model_ft` " مع المحسن المحدد ووظيفة الخسارة والمقایيس.
- `optimizer=adam` : يستخدم محسن Adam للنزول المتدرج.
- `loss=categorical_crossentropy` : يضبط دالة الخسارة على انتروپیا متقطعة فنوية، مناسبة للتصنیف متعدد الفئات.
- `metrics=[accuracy]` : تحد الدقة كمقیاس للتقيیم.

2. `: ()start_ft = timeit.default_timer`

- يسجل وقت البدء قبل تدريب النموذج المضبوط

3. `3. سجل التاريخ = (...)model_ft.fit`

- تدريب النموذج المضبوط (`model_ft`) على بيانات التدريب.
- `train_generator` : يوفر دفعات من بيانات التدريب.
- `epochs` : يحدد عدد فترات التدريب.
- `batch_size=batch_size` : يضبط حجم الدفعه للتدريب.
- `callbacks=[early, Plateau]` : يستخدم عمليات الاسترجاعات الخاصة بالتوقف المبكر وتقلیل معدل التعلم أثناء التدريب.
- `validation_data=valid_generator` : يستخدم بيانات التحقق لتقيیم النموذج.
- `class_weight=train_class_weights` : يطبق أوزان الفئه والصنف للتعامل مع عدم توازن الفئه أثناء التدريب.
- `verbose=1` : يعرض تحدیثات تقدم التدريب.

4. `: ()stop_ft = timeit.default_timer`

- يسجل وقت التوقف بعد تدريب النموذج المضبوط.

لتظهر لنا نتائج التنفيذ كالاتي...

Epoch 1/50

181/181 [=====] - 154s 630ms/step - loss:

1.5527 - accuracy: 0.3022 - val_loss: 11.7481 - val_accuracy: 0.2446 - lr: 0.0010

هذا الناتج هو من الحقبة الأولى لتدريب النموذج المضبوط. وهذا تفصیل:

- **الحقبة 50/1** : يشير إلى أنه العصر الأول من إجمالي 50 عصرًا.
- **181/181** [=====] : يمثل التقدم عبر الدفعات. في هذه الحاله، تمت معالجة 181 دفعه من إجمالي 181 دفعه.
- **630154 s ملي ثانية/خطوة** : يعرض الوقت المنقضي لكل خطوه (أو دفعه) أثناء التدريب. استغرق الأمر حوالي 154 ثانية و 630 ملي ثانية لمعالجة كل دفعه.

الفصل الرابع عشر: تصنیف مرض هشاشة مفاصل الرکبة ودرجة الخطورة

- **الخسارة:** 1.5527 : الخسارة التدريبية لهذه الحقبة هي 1.5527. هذه هي قيمة دالة الخسارة (الإنتروربيا الفنوية) المحسوبة على بيانات التدريب.
- **الدقة:** 0.3022 : دقة التدريب لهذا العصر هي 0.3022 مما يدل على أن ما يقارب 30.22 % من عينات التدريب تم تصنیفها بشكل صحيح.
- **خسارة التحقق:** val_loss: 11.7481 : خسارة التتحقق لهذه الحقبة هي 11.7481. هذه هي قيمة دالة الخسارة المحسوبة على بيانات التتحقق من الصحة.
- **دقة التتحقق:** val_accuracy: 0.2446 : دقة التتحقق لهذا الحقبة هي 0.2446، مما يشير إلى أن ما يقرب من 24.46 % من عينات التتحقق قد تم تصنیفها بشكل صحيح.
- **lr:** 0.0010 : يشير إلى معدل التعلم المستخدم في هذه الحقبة. في هذه الحالة، هو 0.0010.

بشكل عام، يبدو أن أداء النموذج ليس الأمثل في الفترة الأولى، مع خسائر عالية نسبياً ودقة منخفضة لكل من بيانات التدريب والتحقق من الصحة.

• الكود البرمجي.

```
execution_time_ft = (stop_ft - start_ft) / 60  
print(" fModel {model_name} fine tuning executed in {execution_time_ft:.2f} minutes")
```

• شرح الكود.

يقوم هذا الكود القصير بحساب وطباعة وقت تنفيذ عملية الضبط الدقيق بالدقائق. فيما يلي تفاصيل الكود:

- execution_time_ft = (stop_ft - start_ft) / 60 : يتم حساب إجمالي وقت التنفيذ لعملية الضبط الدقيق عن طريق طرح وقت البدء (start_ft) من وقت التوقف (stop_ft) ثم القسمة على 60 لتحويل الوقت إلى دقائق. يتم تخزين النتيجة في المتغير "Execution_time_ft".
- print(" fModel {model_name} fine tuning executed in {execution_time_ft:.2f} minutes") : طباعة سلسلة منسقة تشير إلى وقت تنفيذ عملية الضبط الدقيق.

• الكود البرمجي.

```
model_ft.save(save_model_ft)
```

• شرح الكود.

يحفظ سطر التعليمات البرمجية هذا النموذج المضبوط بدقة (model_ft) في مسار الملف المحدد (save_model_ft). وإليك ما يفعله:

- model_ft.save(save_model_ft) : يحفظ النموذج المضبوط بدقة (model_ft) في مسار الملف المحدد بواسطة save_model_ft . يتم حفظ النموذج بتنسيق تنسيق البيانات الهرمي (HDF5)، والذي يستخدم بشكل شائع لتخزين وتبادل كميات كبيرة من البيانات الرقمية.

يضمن هذا الكود القصير حفظ النموذج المذرب على وفي الحاسبة الشخصية للمستخدم مما يسمح بتحميله وإعادة استخدامه لاحقاً للاستدلال أو التدريب الإضافي دون الحاجة إلى إعادة التدريب من البداية وهو جداً مهم.

• الكود البرمجي.

```
def get_plot_loss_acc(model, model_name):  
    fig = plt.figure()
```

```
plt.subplot(2, 1, 1)
```

```
plt.plot(model.history.history[loss])
plt.plot(model.history.history[val_loss])
plt.title(f'{model_name} \n\n model loss')
plt.ylabel(loss)
plt.xlabel(epoch)
plt.legend([train, valid], loc=upper right)
```

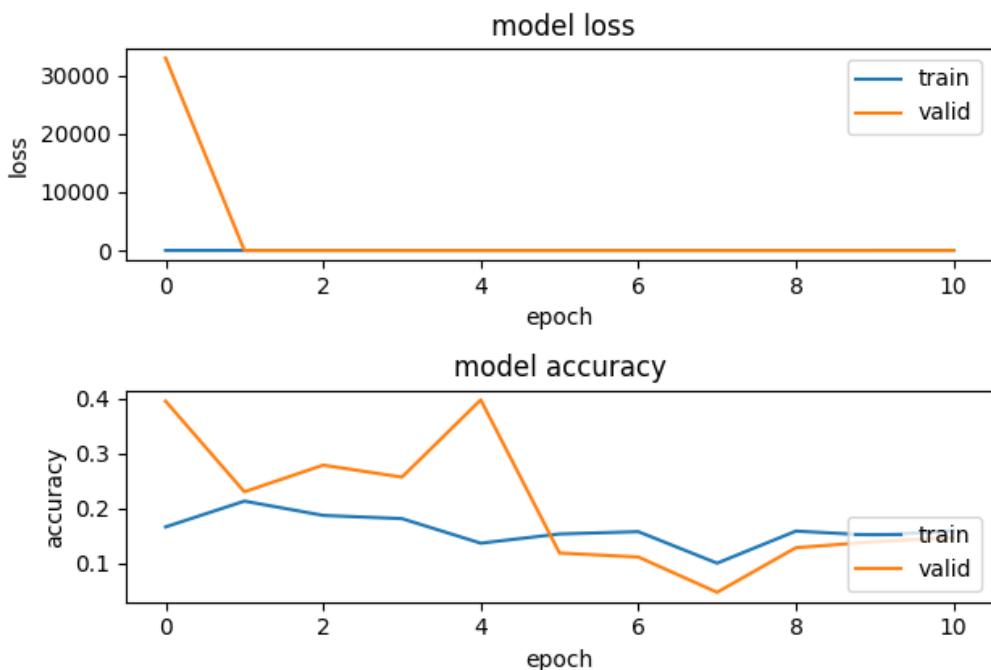
```
plt.subplot(2, 1, 2)
plt.plot(model.history.history[accuracy])
plt.plot(model.history.history[val_accuracy])
plt.title(model accuracy)
plt.ylabel(accuracy)
plt.xlabel(epoch)
```

```
plt.legend([train, valid], loc=lower right)
plt.tight_layout()
```

• شرح الكود البرمجي:-

ترسم هذه الدالة (الوظيفة) منحنیات فقدان التدريب والتحق والدقة لنموذج معین. يقوم بانشاء شکل بمخططین فرعيین أحدهما للخسارة والآخر للدقة، ويرسم المقایيس المقابلة على مدى الحقبات.تأخذ الدالة(الوظيفة) بارامیترات : النموذج المدرب (model) واسم النموذج (model_name).

ResNet50 Fine Tuning



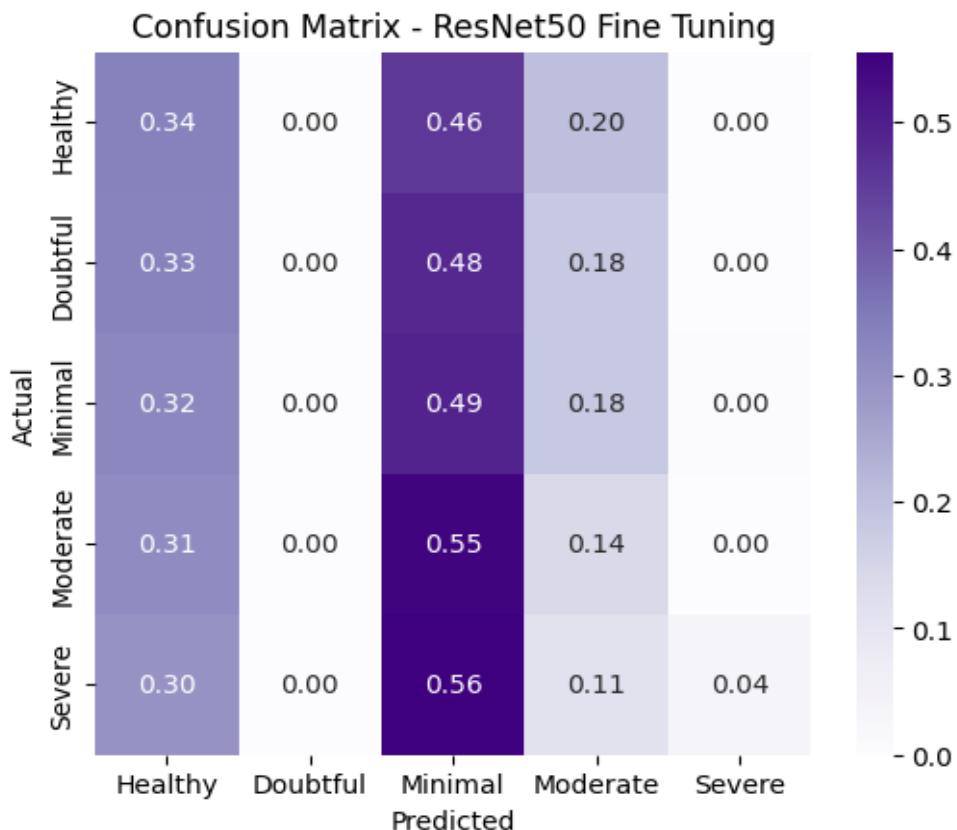
صورة رقم(2) للدقة والخسارة لنموذج التدريب

- الكود البرمجي.

```
def compute_confusion_matrix(
    ytrue, ypred, class_names, model_name):
    cm = confusion_matrix(
        y_true=ytrue.labels,
        y_pred=np.argmax(ypred, axis=1), )
```

```
cmn = cm.astype(float) / cm.sum(axis=1)[:, np.newaxis]
plt.subplots(figsize=(6, 5))
sns.heatmap(
    cmn,
    annot=True,
    fmt=.2f,
    cmap=Purples,
    xticklabels=class_names,
    yticklabels=class_names, )
plt.title(f'Confusion Matrix - {model_name}')
```

```
plt.ylabel(Actual)
plt.xlabel(Predicted)
plt.show(block=False)
```



صورة رقم(3) مصفوفة الارباك

• شرح الكود.

تقوم هذه الوظيفة بحساب وتصور مصفوفة الارباك لمجموعة معينة من التسميات الحقيقة والمتواعدة. يتم إدخال التسميات الحقيقة (`ytrue`)، والتسميات المتوقعة (`ypred`)، وأسماء الفئات، واسم النموذج (`model_name`). يتم حساب مصفوفة الارباك باستخدام الدالة `confusion_matrix` من `scikit-learn`. يتم بعد ذلك تطبيق المصفوفة ورسمها كخريطة حرارية باستخدام وظيفة "الخريطة الحرارية" الخاصة بـ `seaborn`. تعرض الوظيفة مصفوفة الارباك مع التعليقات التوضيحية وتسميات الفئات والعنوانين والتسميات المناسبة.

• الكود البرمجي.

```
def get_evaluate(data, name, model):
    score_model = model.evaluate(data, verbose=1)
```

```
print(f'{name} loss: {score_model[0]:.2f}')
print(f'{name} accuracy: {score_model[1]:.2f})
```

- شرح الكود.

تقوم هذه الوظيفة بتقييم أداء نموذج معين على البيانات المحددة. يقوم بحساب فقدان دقة النموذج على البيانات المقدمة باستخدام طريقة "التقييم" للنموذج. تقوم الوظيفة بطباعة درجات الخسارة والدقة مع الاسم المقابل لتحديد الهوية.

```
26/26 [=====] - 6s 218ms/step - loss: 1.6012 - accuracy:  
0.2785  
Valid loss: 1.60  
Valid accuracy: 0.28
```

- الكود البرمجي.

```
def get_predict(data, model):
    predict_model = model.predict(data)
    return predict_model
```

- شرح الكود.

تقوم هذه الوظيفة بعمل تنبؤات باستخدام نموذج معين على البيانات المقدمة. ويستخدم طريقة "التنبؤ" للنموذج لإنشاء تنبؤات لبيانات الإدخال. تقوم الدالة بإرجاع المخرجات المتوقعة التي تم إنشاؤها بواسطة النموذج.

- الكود البرمجي.

```
def get_metrics(y_test, y_pred, model_name):
    acc = accuracy_score(y_test, y_pred)
    bal_acc = balanced_accuracy_score(y_test, y_pred)
    print(f'Accuracy Score - {model_name}: {acc:.2f}')
    print(f'Balanced Accuracy Score - {model_name}: {bal_acc:.2f}')
    print()
    print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.41	0.34	0.37	328
1	0.00	0.00	0.00	153
2	0.26	0.49	0.34	212
3	0.10	0.14	0.12	106
4	0.33	0.04	0.07	27
accuracy			0.28	826
macro avg	0.22	0.20	0.18	826
weighted avg		0.25	0.28	0.25
				826

• شرح الكود.

تقوم هذه الوظيفة بحساب وطباعة مقاييس التقييم المختلفة لنموذج التصنیف. يتم إدخال التسمیات الحقیقیة (.(model_name)، والتسمیات المتوقعة (y_pred)، واسم النموذج (y_test

تحسب الوظيفة درجة الدقة (acc) ودرجة الدقة المتوازنة (bal_acc) باستخدام الدالین scikit-learn و accuracy_score من balanced_accuracy_score. ثم يقوم بطباعة هذه الدرجات مع اسم النموذج.

بالإضافة إلى ذلك، تقوم الوظيفة بطباعة تقریر تصنیف يتضمن الدقة والاستدعاة ودرجة F1 والدعم لكل فئة، مما یوفر ملخصاً تفصیلیاً لأداء النموذج.

• خدمة تطبيق الويب.

• الكود البرمجي.

```
import gradio as gr
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np
#Load your Keras model (replace with your actual model loading)
model = tf.keras.models.load_model("model/model.hdf5") # Load your model
IMG_WIDTH = 224 # Replace with the input size of your model
IMG_HEIGHT = 224 # Replace with the input size of your model
#Define class labels
class_labels} =
" :0 Grade 0: Healthy,"
" :1 Grade 1: Doubtful,"

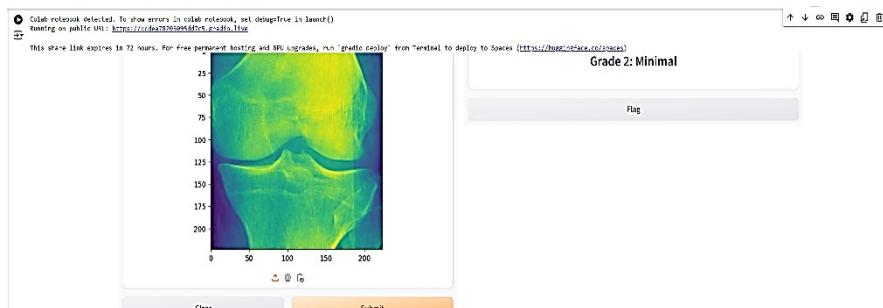
" :2 Grade 2: Minimal,"
" :3 Grade 3: Moderate,"
" :4 Grade 4: Severe{"
def predict_image(image_file):
    """ Predicts the grade from an uploaded image""".
    # Load and preprocess the image (Gradio provides the image data directly)
    img = image.img_to_array(image_file) # Directly use the image data from Gradio
    img = tf.image.resize(img, (IMG_WIDTH, IMG_HEIGHT))
    img = img / 255.0
    img = np.expand_dims(img, axis=0) # Add batch dimension
    # Make prediction using the model
    prediction = model.predict(img)[0]
    # Get the predicted class index
```

الفصل الرابع عشر: تصنیف مرض هشاشة مفاصل الرکبة و درجة الخطورة

```
predicted_class_index = np.argmax(prediction)
# Get the class label based on the index
predicted_class_label = class_labels[predicted_class_index]
return predicted_class_label
#Create the Gradio interface
iface = gr.Interface(
    fn=predict_image,
    inputs=gr.Image(label="Upload Image"),
    outputs=gr.Label(label="Predicted Grade"),
    title="Grade Classification Model",
    description="Upload an image to predict its grade".
#Launch the interface
iface.launch()
```

• شرح الكود.

تم شرح الاداء (**Gradio**) سابقا في الفصول السابقة. تم استخدام نموذج كيراس المدرب المسبق على نموذج . يرجى التأكد من مسار النموذج النهائي بعد التأكد من الدقة واعادة تدربيبة مرة اخرى . (**Resnet50**)



صورة رقم(4) واجهة المستخدم



الكود البرمجي النهائي.

الملخص

في الختام، فإن استخدام نموذج **ResNet50** يمثل وسيلة واحدة لتعزيز تشخيص وعلاج التهاب وهشاشة مفاصل الركبة (OA). ومن خلال تسخير قوة الذكاء الاصطناعي، يمكننا تعزيز دقة وكفاءة تحليل التصوير الطبي، لا سيما في تحديد العلامات الدقيقة للتهاب ومعرفة درجة هشاشة المفاصل العظمي في الركبة من الأشعة السينية أو فحوصات التصوير بالرنين المغناطيسي.

إن الطبيعة المعقّدة للتهاب وهشاشة المفاصل العظمي في الركبة، والتي تتميز بالتدور التدريجي للضرووف المفصلي والتغيرات في العظام والأنسجة الرخوة المحيطة، تؤكد على أهمية أدوات التشخيص الدقيقة. تظل الطرق التقليدية مثل الأشعة السينية والتصوير بالرنين المغناطيسي حيوية لتقدير تباعد المفاصل وتحديد مدى خطورة الحالة، كما يتضح من نظام تسجيل KL المعتمد به.

من خلال دمج تقنيات الذكاء الاصطناعي مثل **ResNet50**، يمكن لمختصي الرعاية الصحية زيادة قدراتهم التشخيصية، مما يتيح الكشف والتدخل المبكر للتهاب المفاصل العظمي في الركبة. ومن خلال التقسيم الطبقي الدقيق لشدة المرض، يمكن للأطباء تصميم خطط علاجية لكل مريض على حدة، وتحسين النتائج وتحسين نوعية الحياة.

علاوة على ذلك، فإن إمكانات الذكاء الاصطناعي تمتد إلى ما هو أبعد من التشخيص، حيث توفر طرقًا للندجنة التنبؤية والطب الشخصي في إدارة التهاب المفاصل في الركبة. ومن خلال الاستفادة منمجموعات البيانات الضخمة والخوارزميات المتقدمة، يحمل الذكاء الاصطناعي وعداً بإحداث ثورة في كيفية تعاملنا مع هذا الشكل السائد من التهاب المفاصل وعلاجه.

في الجوهر، يمثل تكامل نموذج **ResNet50** خطوة مهمة إلى الأمام في مكافحة التهاب مفاصل الركبة، وتمكن مقدمي الرعاية الصحية بأدوات مبتكرة لتعزيز دقة التشخيص، وتحسين رعاية المرضى، والتخفيف في نهاية المطاف من عبء هذه الحالة المنهكة.

باختصار، يبشر تكامل تقنيات الذكاء الاصطناعي بوعود كبيرة لإحداث ثورة في تشخيص وإدارة وعلاج التهاب وهشاشة مفاصل الركبة من خلال تمكين التصنيف الدقيق لدرجات الخطورة وتسهيل أساليب الرعاية الشخصية. ومن خلال الجهود التعاونية بين الأطباء والباحثين والمهندسين والممرضى، يمكننا تسخير الإمكانيات الكاملة للذكاء الاصطناعي لمواجهة التحديات المتغيرة التي يفرضها التهاب مفاصل الركبة وتحسين نتائج المرضى ونوعية حياتهم في نهاية المطاف.

يؤدي دمج الذكاء الاصطناعي (AI) والتعلم الآلي (ML) في التصوير الطبي إلى تحويل مشهد الرعاية الصحية، مما يوفر إمكانات غير مسبوقة في التشخيص وتحطيط العلاج ومراقبة المرضى. يقام هذا الكتاب استكشافاً متمعماً لكيفية تسخير Python، جنباً إلى جنب مع نماذج الذكاء الاصطناعي المتقدمة مثل ResNet وYOLOv8، لأداء المهام الحاسمة في التصوير الطبي، بما في ذلك التصنيف والتجزئة والكشف.

• المأخذ الرئيسية.

1. بايثون في التصوير الطبي:

- برزت لغة بايثون كلغة مفضلة في التصوير الطبي بسبب مكتباتها الواسعة وسهولة استخدامها ودعم المجتمع القوي لها. تعمل المكتبات مثل OpenCV وSciPy وNumPy والأطر المتخصصة مثل PyTorch وTensorFlow على تمكين التطوير الفعال ونشر نماذج الذكاء الاصطناعي.

2. نماذج ResNet:

- أحدثت ResNet (الشبكات المتبقية) ثورة في مهام تصنیف الصور بقدرتها على تدريب شبکات عمیقة جداً دون الوقوع في مشاکل اختفاء التدرجات. وهذا يجعلها مناسبة للغاية لتطبیقات التصوير الطبی حيث يعده التصنيف الدقيق للأمراض المعقدة أمراً بالغ الأهمیة.
- أظهر الكتاب استخدام ResNet في سیاقات التصوير الطبی المختلفة، وأظهر قوتها في مهام مثل تصنیف الأمراض من خلال الأشعة السینیة والتصوير بالرنین المغناطیسي.

3. YOLOv8 للكشف والتجزئة:

- أثبتت نماذج YOLO (أنت تنظر مرة واحدة فقط)، وخاصة YOLOv8، فعاليتها في اكتشاف الكائنات ومهام التجزئة في الوقت الفعلي. إن قدرتهم على التنبؤ بالمربعات المحيطة واحتمالات الفنة في وقت واحد تسمح بالتحليل الفعال للصور الطبية.
- إن التقدم الذي حققه YOLOv8 في السرعة والدقة يجعله مناسباً للكشف عن الحالات الشاذة في عمليات الفحص الطبی، وبالتالي تسهيل التشخيص والتدخل المبكر.

4. التصنيف والتجزئة والكشف:

- يساعد التصنيف في تشخيص الأمراض من خلال تصنیف الصور إلى فئات محددة مسبقاً.
- يؤدي التقسيم إلى تقسيم الصورة إلى أجزاء ذات معنى، مثل أنواع الأنسجة المختلفة أو المناطق المرضية، وهو أمر بالغ الأهمية للتحليل التفصيلي وتحطيط العلاج.
- يحدد الاكتشاف ويحدد المناطق ذات الاهتمام، مثل الأورام أو الآفات، مما يتبع أساليب العلاج المستهدفة.

• نواتج عملية.

- **دقة التشخيص المحسنة:** تعمل نماذج الذكاء الاصطناعي على تحسين دقة التشخيص واتساقه بشكل كبير، مما يقلل من احتمالية الخطأ البشري والتباطؤ في التفسير.
- **السرعة والكفاءة:** يعمل التحليل الآلي للصور على تسريع عملية التشخيص، مما يسمح باتخاذ القرار بشكل أسرع والتدخل في الوقت المناسب.
- **تحسين الموارد:** يمكن للأدوات المستندة إلى الذكاء الاصطناعي تخفيف عبء العمل عن أخصائيي الأشعة ومتخصصي الرعاية الصحية، وتمكينهم من التركيز على الحالات الأكثر تعقيداً ورعاية المرضى.
- **الطب المخصص:** من خلال تحليلمجموعات البيانات الكبيرة، يمكن لنموذج الذكاء الاصطناعي تحديد الأنماط التي تدعم خطط العلاج المخصصة والمصممة خصيصاً لملفات تعريف المرضى الفردية.

• الاتجاهات المستقبلية.

- **التحسين المستمر للنماذج:** يعد البحث والتطوير المستمر أمراً ضرورياً لتحسين نماذج الذكاء الاصطناعي، مما يضمن مواكبة تقييمات التصوير الطبي المتقدمة ومجموعات البيانات المتغيرة.
- **التكامل مع سير العمل السريري:** سيؤدي التكامل السلس للأدوات الذكاء الاصطناعي في سير العمل السريري إلى تعزيز فائدتها، مما يجعلها لا غنى عنها في الممارسة الطبية الروتينية.
- **الاعتبارات الأخلاقية والتنظيمية:** مع تزايد انتشار الذكاء الاصطناعي، ستكون معالجة المخاوف الأخلاقية والالتزام بالمعايير التنظيمية أمراً بالغ الأهمية لضمان سلامة المرضى وخصوصية البيانات.
- **التعاون متعدد التخصصات:** سيؤدي التعاون بين الباحثين في مجال الذكاء الاصطناعي ومتخصصي الرعاية الصحية وصانعي السياسات إلى دفع عجلة الابتكار وتسهيل اعتماد تقييمات الذكاء الاصطناعي في التصوير الطبي.

• أفكار اخيرة.

لقد زود هذا الكتاب القراء بالمعرفة والمهارات العملية لتطبيق نماذج الذكاء الاصطناعي المتقدمة في التصوير الطبي. يوفر الجمع بين المفاهيم النظرية وأمثلة الترميز العملية وتطبيقات العالم الحقيقي أساساً شاملأً للاستفادة من الذكاء الاصطناعي في الرعاية الصحية. ومع استمرار تقدم التكنولوجيا، فإن دور الذكاء الاصطناعي في التصوير الطبي سوف ينمو، مما يهد بمستقبل تكون فيه الرعاية الصحية أكثر دقة وكفاءة وتحصيناً.

السيرة الذاتية للمؤلف

Falah .G.Salih Resume

Education

- 1- B.Sc. In Physics' Science, University of Baghdad. (1986-1987)
- 2- Diploma in Ceramic Art in the Popular Arts Center/Baghdad (1995-1996).
- 3- Programmer from 1987 until now.

Computer Skills and programming languages:

- 1-Visual C++. And Visual Basic .Net
- 2- ASP Server Side Programming.
- 3-Java Script. for web pages.
- 4-Java for desktop.
- 5-MYSQL Server (Data Base systems). for IBM Co.
- 6- Developing Microsoft ASP.NET Web Application using Visual Studio.Net & ADO.NET Components for database systems.
- 7- Microsoft SQL Server (version 2000 & 2005) & Database Search Engines Systems.
- 8-PHP Server Side Programming (PHP Nuke and Forum for MS).
- 9- Static Pages Programming Languages (HTML & DHTML).
- 10- ASP.NET Server Side Programming with MS SQL Server.
- 11- Oracle SQL Database 10g
- 12- ArcView and Arc Map for GIS Application for spatial data analysis.
- 13-Microcontroller apps. (Arduino & Esp8266 MCU & Raspberry pi) 2014-
- 13- Android applications. (2011-)
- 14- Python for AI applications. (2017-)
- 15- Artificial intelligence in deep learning and computer vision applications.
- 16-flutter and dart for Android applications. (2020-)
- 17- AI Artificial Intelligence Model Developer (2018-) for Art Field.

Certificates: -

- 1- Microsoft Certified Professional (MCP). Mar 13, 2007.
- 2- Microsoft Certified Application Developer (MCAD). May 10 2007
<https://tinyurl.com/2x3hfkwp>
- 3- Microsoft Certified Solution Developer (MCSD). Aug. 12 2007.
- 4- Data Analysis with Python
- 5- Python 101 for Data Science
- 6- Deep Learning with TensorFlow

بإمكانك مشاهدة جميع الشهادات



مشاهدة جميع مشاريعي في المجالات التالية

- Education field
- Health field
- Army field
- Industrial field
- General app.
- Agricultural app.
- Artificial Intelligence app.



بإمكانك مشاهدة جميع مشاريعي

<https://tinyurl.com/2p8cejwa>

My website:

- Blog: <https://iraqprogrammer.wordpress.com>
- Email: falahgs07@gmail.com
- AI4Art Models: <https://huggingface.co/Falah>
- AI4Art Models: <https://civitai.com/user/falahgs/models>
- YouTube: <https://www.youtube.com/c/FalahgsGate>
- Amazon: <https://www.amazon.com/stores/author/B0BYHXL7R/>
- Github: <https://github.com/falahgs>
- PyPi: <https://pypi.org/user/falahgs/>
- Facebook: <https://www.facebook.com/falahgs4ai>
- Telegram: https://t.me/falahgs_dl_cv
- LinkedIn: <https://www.linkedin.com/in/falah-gatea-060a211a7/>
- Twitter: <https://twitter.com/FalahGatea>

- **NightCafe AI Art:** <https://creator.nightcafe.studio/u/FalahGS>
- **Artstation AI Art :** <https://www.artstation.com/falahgs>
- **Medium Posts:** <https://medium.com/@falahgs>
- **Instagram:** <https://www.instagram.com/falahgs4ai/>

1. DataCamp, First Chapter of [Biomedical Image Analysis in Python Course](#), First Chapter, Exploration.
2. C. Rossant, [IPython Interactive Computing and Visualization Cookbook](#), (2018), [Mastering widgets in the Jupyter Notebook](#).
3. ImageIO, GitHub, ImageIO-Plugin-DICOM, (2022), DICOM Attributes, [MINIDICT](#).
4. <https://www.dlogy.com/blog/transfer-learning-with-efficientnet/>
5. <https://arxiv.org/abs/1905.11946>
6. <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>
7. <https://www.worldometers.info/coronavirus/>
8. https://colab.research.google.com/drive/1ATgpjP6yJvHCY8mhzNhAsFAR_yrBQWuA?usp=sharing
9. <https://www.kaggle.com/paultimothymooney/breast-histopathology-images?>
10. <https://pubmed.ncbi.nlm.nih.gov/27563488/>
11. <https://spie.org/Publications/Proceedings/Paper/10.1117/12.2043872?SSO=1>
12. Data Exploration: <https://jovian.ml/karthicksothivelr/breast-cancer-classification-data-exploration>
13. Logistic Regression: <https://jovian.ml/karthicksothivelr/breast-cancer-classification-logistic-regression-no-augmentation>
14. Logistic Regression (With Augmentation): <https://jovian.ml/karthicksothivelr/breast-cancer-classification-logistic-regression>
15. DNN: <https://jovian.ml/karthicksothivelr/breast-cancer-classification-dnn-no-augmentation/v/1>
16. CNN: <https://jovian.ml/karthicksothivelr/breast-cancer-classification-cnn-no-augmentation/v/2>
17. ResNet34: <https://jovian.ml/karthicksothivelr/breast-cancer-classification-resnet34-no-augmentation/v/4>
18. ResNet34 (With Augmentation): <https://jovian.ml/karthicksothivelr/breast-cancer-classification-resnet34>
19. ResNet34 (WRS): <https://jovian.ml/karthicksothivelr/breast-cancer-classification-resampling-resnet34-no-augmentation>
20. ResNet34 (SMOTE): <https://jovian.ml/karthicksothivelr/breast-cancer-classification-smote-resampling-resnet34-no-augmentation/v/2>
21. <https://docs.ultralytics.com/models/yolov8/>
22. [https://www.researchgate.net/publication/348248500 ResNet 50](https://www.researchgate.net/publication/348248500)
23. <https://keras.io/api/applications/mobilenet/>
24. <https://www.gradio.app/docs>
25. <https://www.gradio.app/guides/theming-guide>