جامعة حلب
كلية الهندسة الكهربائية والإلكترونية
قسم هندسة الحواسيب

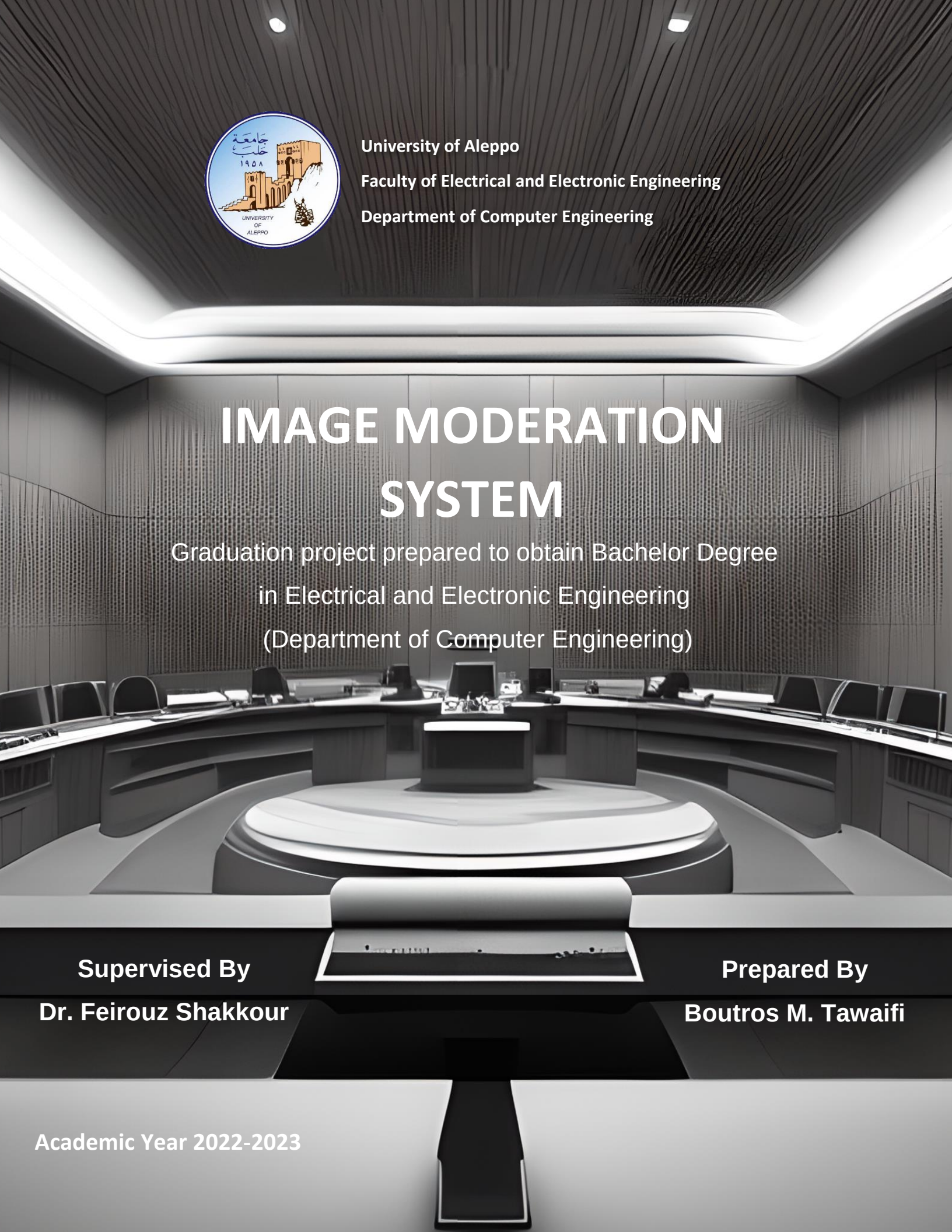
نظام مراقبة للصور

مشروع تخرج أعد لنيل الإجازة في الهندسة الكهربائية والالكترونية
(قسم هندسة الحواسيب)

إعداد الطالب
بطرس طوائفي

إشراف
د. فيروز شكور

العام الدراسي 2022-2023



University of Aleppo
Faculty of Electrical and Electronic Engineering
Department of Computer Engineering

IMAGE MODERATION SYSTEM

Graduation project prepared to obtain Bachelor Degree
in Electrical and Electronic Engineering
(Department of Computer Engineering)

Supervised By
Dr. Feirouz Shakkour

Prepared By
Boutros M. Tawaifi

Academic Year 2022-2023

إهداء

إلى والديَّ الذينَ غرسا فيَّ أهمية العمل الجاد والتفاني الدائم، هذا المشروع هو تكريمٌ لدعمكم ولتضحياتكم التي لا تعد ولا تحصى.

إلى أساتذتي، الذين كانت إرشاداتهم وتوجيهاتهم هي أساس تعلمي، حكمتكم هي منارة طريقي.

إلى أصدقائي، الذين وفروا الراحة في الأوقات الصعبة وشاركوني الضحك في اللحظات السعيدة، صداقتكم محفوظة.

وأخيراً، إلى كل من آمن بي، كان إيمانكم هو الريح تحت أجنحتي الذي دفعتني للأمام في الأوقات الصعبة.

هذا المشروع هو ثمرة العديد من ساعات الجهد، التي تم تغذيتها بالحب والتشجيع والإيمان من هؤلاء الأشخاص الاستثنائيين في حياتي.

إنه بكل الشكر والاحترام العميقين، أهدي هذا المشروع الجامعي لكل منكم.

بطرس طوائفي

Table of Content

الفصل الأول المقدمة

1.1 مقدمة

في زمننا الرقمي المعاصر، باتت الصور تحتل مكاناً مركزياً في المشهد البيئي للبيانات الذي نعيش ضمنه. الصور تُستخدم الآن كلغة مرئية للتعبيرعن أفكارنا ومشاعرنا وتجاربنا. وتتضاعف كميات الصور المنتجة والمستخدمة في عدة مجالات من الحياة مثل وسائل التواصل الاجتماعي، والترفيه، والرعاية الصحية وغيرها، مما يجعل تصنيف الصور بطريقة فعالة ودقيقة أمراً لا بد منه.

خلال السنوات القليلة الماضية، شهدنا تقدماً كبيراً في تقنيات الرؤية الحاسوبية، بفضل التطور في التعلم العميق واستخدام الشبكات العصبية التلافيفية. هذه التقنيات المتقدمة قد أسهمت في تحسين الدقة والقدرات في أنظمة التعرف على الصور وتصنيفها. وكنتيجة لهذه التطورات، أصبحت أنظمة التعرف على الصور قادرة على التفاعل مع المحتوى المرئي بشكل أكثر فاعلية، واستيعاب التفاصيل الدقيقة والأنماط المعقدة.

الآن، يعمل الباحثون والمطورون على دمج البيانات النصية والبصرية في نماذج متقدمة. يستهدف هذا الدمج تحقيق فهم أعمق وأكثر دقة للصور وتصنيفها بطريقة أكثر كفاءة. بفضل هذه النماذج المتقدمة، يمكننا بناء تطبيقات ذكية وسهلة الاستخدام تستفيد من قدرات التعرف على الصور وتصنيفها. تتمتع هذه النماذج بالقدرة على فهم المحتوى البصري والنصي، وتقديم رؤى شاملة ومعرفة دقيقة. تتيح هذه التقنيات فرص جديدة لتحسين التفاعل وتعزيز الفهم والاستفادة من الصور في عصرنا الرقمي.

1.2 نظرة عامة على المشروع

المشروع يسعى لابتكار نظام مراقبة للصور يتميز بالمرونة والاقتصادية، وهو موجه أساساً لمراقبة المحتوى الذي ينشئه المستخدمين مع التركيز بشكل خاص على المحتوى المسيء. يتميز النظام بالقدرة على تحديد وتصنيف محتوى الصور، يتحقق هذا عن طريق دمج نموذج CLIP من OpenAI مع واجهة برمجة التطبيقات RESTful غير المتزامنة، التي سيتم تصميمها بواسطة FastAPI، مما يسمح بالتفاعل مع النموذج عبر طلبات HTTP.

1.3 أهداف المشروع

الهدف الرئيسي للمشروع هو تقييم فعالية النموذج المتقدم CLIP المدرب مسبقاً في سياق مراقبة الصور وتصنيفها وتوضيح كيفية الاستفادة منه لبناء نظام رقابة للصور اقتصادي، ويتم ذلك عن طريق استخدام FastAPI لتطوير API غير متزامنة تقبل بيانات الصور على شكل ملفات يتم تحميلها أو URL ليتم تصنيفها من قبل النموذج. كما يسعى المشروع لتوضيح قدرة CLIP والنظام الذي يستخدمه على التكيف مع سيناريوهات تصنيف متعددة، وبالتالي توسيع الاستفادة من تقنيات تصنيف الصور في الاستخدامات العملية. يعتبر هذا النظام، بالنظر إلى اقتصاديته وقدرته على التعامل مع التصنيفات الجديدة بمرونة ودون الحاجة لتدريب مكثف، خطوة مهمة نحو خلق بيئة أكثر أماناً على الإنترنت.

1.4 الأدوات البرمجية

سيتم استخدام الأدوات البرمجية التالية:

- **Python:** تم اختيارها لبساطتها و قدرة استخدامها لمجموعة واسعة من المكتبات المتاحة للتعلم الآلي وتطوير الويب.
- **FastAPI:** إطار عمل ويب عالي الأداء لبناء واجهات برمجة التطبيقات مع Python، يستخدم لبناء واجهة برمجة التطبيقات RESTful.
- **PyTorch:** يوفر حسابات tensor مع مسارع قوي لوحدة معالجة الرسوميات والشبكات العصبية العميقة، ويعمل كأساس لـCLIP.
- **نموذج CLIP الخاص بـ OpenAI:** قادر على فهم كل من الصور والبيانات النصية، فهو مناسب للغاية للمهام التي تتطلب فهمًا مشتركًا لهذه الأنواع من البيانات.
- **Pydantic:** مكتبة للتحقق من صحة البيانات في Python، تستخدم للتحقق من صحة جسم طلب HTTP في تطبيق FastAPI.
- **Uvicorn:** خادم ASGI يعمل على تشغيل تطبيقات FastAPI ، ضروري لتقديم تطبيق FastAPI.

الفصل الثاني القسم النظري

2.1 التصنيف الصوري: التطور من الخوارزميات التقليدية إلى التعلم العميق

2.1.1 من الخوارزميات التقليدية إلى الشبكات العصبية التلافيفية (CNNs)

من الناحية التاريخية، اعتمد التصنيف الصوري على الخوارزميات التقليدية مثل (k-NN) و (SVM) وأشجار القرار، التي تتطلب هندسة الميزات يدوياً مثل الهيستوغرامات اللونية وصفوف البنية وكاشفات الحواف. ومع ذلك، واجهت هذه الخوارزميات صعوبات في التعامل مع تعقيد وتباين مجموعات البيانات الصورية في العالم الحقيقي. أدى ظهور التعلم العميق إلى ظهور الشبكات العصبية التلافيفية (CNNs)، مما أتاح الاستخراج التلقائي للتمثيلات الهرمية من قيم البيكسل الخام، وبالتالي تعزيز الدقة من خلال التعامل مع أنماط الصور المعقدة.

2.1.2 التطورات والقيود في هياكل الشبكات العصبية التلافيفية

دمجت نماذج CNNs بارزة مثل LeNet و AlexNet و VGGNet و GoogLeNet و ResNet استراتيجيات مثل:

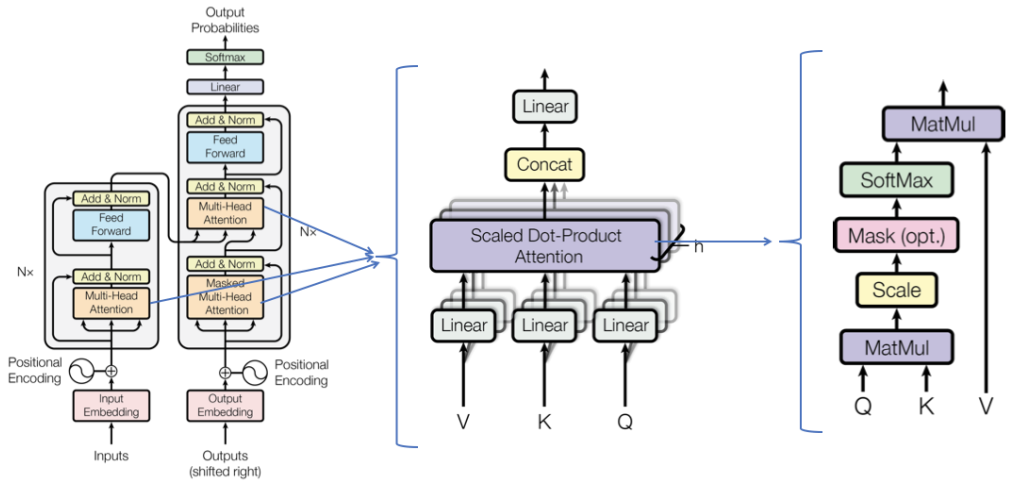
skip connections, residual learning, and network-in-network structure

لتحسين الأداء. ومع ذلك، تواجه CNNs مشاكل فيما يتعلق بتوافر البيانات الموسومة المحدودة والمهام التي تتطلب فهمًا مشتركًا للوضعيات الصورية والنصية.

2.2 نماذج Transformer في تطبيقات تصفية الصور: دراسة مقارنة

2.2.1 نظرة عامة وآلية الاهتمام الذاتي

وجدت نماذج (المحول) Transformer المصممة في البداية للمعالجة اللغوية الطبيعية (NLP) أهمية في مهام الرؤية. تسمح آلية الاهتمام الذاتي بهذه النماذج بتعلم التبعيات بعيدة المدى بين عناصر الإدخال، مما يجعلها مناسبة لإعطاء التوصيف للصور والإجابة على الأسئلة المرئية.



2.2.2 تطبيقات الرؤية والتحديات

النماذج المستندة إلىTransformers، مثل Vision Transformer (ViT)، قد أظهرت وعودًا كبيرة في مهام الرؤية المختلفة، بما في ذلك التصنيف الصور، الرد على الأسئلة البصرية، واكتشاف الأشياء. ومع ذلك، لا يزال هناك بعض التحديات التي يجب التغلب عليها، مثل الحاجة إلى كمية كبيرة من البيانات للتدريب. رغم هذه التحديات، فإن المحولات قد أظهرت مرونة عالية ودقة عالية في التصنيف.

2.2.3 نماذج Transformer في رقابة الصور: الإيجابيات والسلبيات

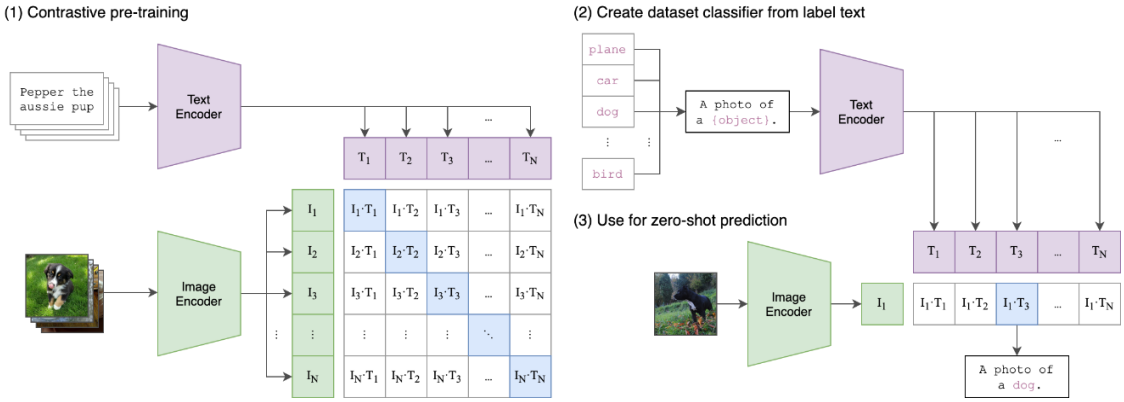
2.2.3.1 المزايا

1. **التبعية بعيدة المدى:** يمكن للمحولات التقاط التبعيات بعيدة المدى بين وحدات البكسل في الصورة. هذا أمر بالغ الأهمية لمهام الرقابة على الصور حيث يمكن أن ينتشر المحتوى الضار عبر وحدات بكسل متعددة. على سبيل المثال، يمكن استخدام محول لتحديد الصور التي تحتوي على محتوى اباحي، حتى لو لم يكن المحتوى مرئيًا بوضوح في أي بكسل واحد.
2. **تحمل الضوضاء:** تُظهر المحولات قوة أكبر لمقاومة للضوضاء مقارنة بنماذج تصنيف الصور التقليدية مثل شبكات CNN. هذا لأن المحولات لا تعتمد على الميزات المحلية لعمل تنبؤات. بدلاً من ذلك، تتعلم تمثيل الصورة بأكملها كمتجه واحد. هذا يجعلها أقل عرضة للضوضاء والتشوهات التي قد تكون موجودة في وحدات البكسل الفردية.
3. **التعميم على الصور الجديدة:** يمكن تدريب المحولات على مجموعات بيانات كبيرة من الصور. هذا يسهل التعميم على الصور الجديدة. على سبيل المثال، يمكن تدريب المحولات على مجموعة بيانات لصور القطط والكلاب. بمجرد تدريبه، يمكن استخدامه لتصنيف صور جديدة للقطط والكلاب، حتى لو لم تكن الصور موجودة في مجموعة بيانات التدريب.

2.2.3.2 العيوب

1. **التكلفة الحسابية:** أحد العوائق الرئيسية للمحولات هو تكلفتها الحسابية، أثناء التدريب والنشر. هذا لأن المحولات لديها عدد كبير من المعلمات (Parameters). على سبيل المثال، يحتوي نموذج BERT Base على 110 مليون معلمة.
2. **الدقة:** قد تكون المحولات أقل دقة من نماذج تصنيف الصور التقليدية، خاصة في المهام التي تتضمن اكتشاف الكائنات. وذلك لأن المحولات لا تقوم بشكل صريح بنمذجة العلاقات المكانية بين وحدات البكسل. على سبيل المثال، قد يواجه المحول صعوبة في تحديد موقع قطة في صورة ما، حتى لو كان قادرًا على تصنيف الصورة بشكل صحيح على أنها تحتوي على قطة.

2.3 نظرة شاملة على CLIP لشركة OpenAI



نموذج CLIP (Contrastive Language-Image Pretraining) لشركة OpenAI قد غير تصنيف الصور من خلال دمجه لتمثيلات الصور مع النص (Labels) من مجموعة بيانات ضخمة. هذا الفهم متعدد الوسائط يتيح لـ CLIP تصنيف الصور في فئات متنوعة بدون تدريب مسبق عليها جميعاً.

يعمل نموذج CLIP عن طريق تحويل الصورة والنص إلى متجهات باستخدام نموذج Vision Transformer ونموذج Language Transformer على التوالي. ثم يتم مقارنة هذه المتجهات، ويتم تنبؤ فئة الصورة استنادًا إلى الزوج الأكثر تشابهًا. تتأثر دقة هذا التنبؤ بشكل كبير باختيار التصنيفات النصية (Labels) المرادة، مما يؤكد أهمية اختيار تصنيفات ذات صلة بشكل كافي.

تم تدريب نماذج CLIP على مجموعات بيانات مختلفة مثل ImageNet وVisual Genome وCOCO وPlaces وFlickr30k وConceptual Captions. كل مجموعة بيانات لها نقاط قوة وضعف فريدة من نوعها، مما يؤكد على وجوب اختيار نموذج CLIP استنادًا إلى المهمة المطلوبة.

2.3.1 مزايا استخدام CLIP في رقابة الصور

1. **المرونة:** يتيح التصنيف بدون تدريب في CLIP تصنيف الصور بدون تدريب مسبق على الفئة، مما يجعله قابلاً للتكيف مع مهام متنوعة. عادةً ما يتم تدريب شبكات CNNs التقليدية على مهمات محددة للتصنيف الصوري.
2. **دقة محسنة:** يتيح التعلم المشترك لتمثيلات الصور والنص في CLIP فهم الصور وتصنيفها في سياق واصفاتها النصية، مما يحسن الدقة.
3. **التخصيص:** توافر نماذج CLIP المدربة على مجموعات بيانات مختلفة يسمح بالمرونة والتخصيص استنادًا إلى المهمة المحددة.

2.3.2 عيوب استخدام CLIP في رقابة الصور:

1. **اعتمادية التصنيف:** تعتمد دقة تنبؤات CLIP بشكل كبير على صلة وكمية التصنيفات المقدمة.
2. **انحيازيات المجموعة البيانات:** قد تحتوي بعض مجموعات البيانات المستخدمة لتدريب نماذج CLIP على انحيازيات ذات طابع خاص، مما يمكن أن يقيد قابلية تطبيق النموذج على سياقات متنوعة.
3. **مكلفة من حيث الموارد:** يتطلب تدريب نماذج CLIP موارد حسابية واسعة وكمية كبيرة من البيانات، مما قد يقيد تطبيقها. يعود ذلك إلى استخدام المحولات التي تستخدم الاهتمام الذاتي وهي عملية تتطلب الكثير من القدرة

الحسابية، بالمقارنة مع العمليات الاقتصادية أكثر Convolutions المستخدمة في شبكات CNNs. بالإضافة إلى ذلك، تتطلب المحولات مجموعة بيانات أكبر للتدريب مقارنة بشبكات CNNs، حيث يجب أن تتعلم المحولات التبعيات بعيدة المدى في الإدخال، في حين يمكن لشبكات CNNs التعامل مع التبعيات قصيرة المدى بكمية أقل من البيانات.

2.3.2 الجداول: مقارنة الأداء

جدول 1: مقارنة الدقة بين نموذج Vision Transformer (ViT-B16) و ResNet-50

Model	Top-1 Accuracy	Top-5 Accuracy
ViT-B16	82.5٪	96.0٪
ResNet-50	78.3٪	93.6٪

جدول 2: مقارنة الدقة بين نماذج CLIP على ImageNet

Model	Top-1 Accuracy	Top-5 Accuracy	Computational cost of training	Computational cost of inference	Number of parameters	Source	Testing Dataset	RAM required
CLIP (ImageNet)	84.2%	96.7%	2 weeks on 4 TPUv4 Pods	100 ms	340M	Radford et al. (2021)	ImageNet	340 MB
CLIP-ViT-B16	86.1%	97.4%	4 weeks on 4 TPUv4 Pods	150 ms	838M	Radford et al. (2021)	ImageNet	838 MB
CLIP-ViT-L16	87.2%	98.1%	8 weeks on 4 TPUv4 Pods	200 ms	3.4B	Radford et al. (2021)	ImageNet	3.4 GB
CLIP-ViT-H16	88.2%	98.6%	12 weeks on 4 TPUv4 Pods	250 ms	11B	Radford et al. (2021)	ImageNet	11 GB
openai/clip-vit-base-patch32	86.1%	97.4%	1 week on 4 TPUv4 Pods	100 ms	838M	Hugging Face	Conceptual Captions	838 MB
openai/clip-vit-large-patch14	88.8%	99.0%	2 weeks on 4 TPUv4 Pods	150 ms	3.4B	Hugging Face	Conceptual Captions	3.4 GB
openai/clip-vit-base-patch16	87.2%	98.1%	2 weeks on 4 TPUv4 Pods	200 ms	3.4B	Hugging Face	Conceptual Captions	3.4 GB

ملاحظة: يجب تفسير هذه النتائج بحذر بسبب الاختلافات في مجموعات البيانات ومقاييس التقييم. ومع ذلك، تشير إلى قدرة CLIP على التنافس في التصنيف الصوري.

يبرز نموذج openai/clip-vit-base-patch32 بتوازنه بين السرعة والدقة، حيث يحقق دقة الصنف الأول بنسبة 86.1٪ ودقة الصنف الخامس بنسبة 97.4٪، وتكلفة التنفيذ الحسابية تبلغ 100 ملي ثانية.

2.4 تقنية Python

2.4.1 تعريف Python

Python هي لغة برمجة مفتوحة المصدر تشتهر ببساطتها وسهولة قراءتها. تُستخدم على نطاق واسع في تطوير الويب والحوسبة العلمية وتحليل البيانات والذكاء الاصطناعي وغيرها، وتحتوي على مكتبة قياسية كبيرة توفر العديد من الوظائف والوحدات المُعدَّة مُسبقًا لمهام مختلفة.

2.4.2 لماذا Python

1. **سهولة التعلم والقراءة:** تتميز Python بتركيبتها البسيطة وهياكلها الكودية النظيفة والقابلة للقراءة، مما يجعلها سهلة للمبتدئين لفهم الكود وكتابته.
2. **مرونة وقوة:** يمكن استخدام Python لمجموعة واسعة من التطبيقات، بدءًا من تطوير الويب إلى تحليل البيانات وتعلم الآلة. إنها توفر مكتبات وإطارات عمل شاملة تبسط المهام المعقدة.
3. **مجتمع مطورين كبير:** يتمتع Python بمجتمع واسع ونشط من المطورين الذين يساهمون في نموه. يقدم هذا المجتمع الدعم والموارد والعديد من المكتبات وإطارات العمل من جهات خارجية.
4. **قابلية التوافق عبر المنصات:** يتوفر Python على أنظمة تشغيل رئيسية مثل Windows و macOS و Linux، مما يتيح للمطورين كتابة كود يعمل بسلاسة عبر منصات مختلفة.
5. **قدرات التكامل:** يمكن لـ Python التكامل بسهولة مع لغات وأطر عمل أخرى مثل C و C++ وJava ، مما يجعله مرناً لبناء تطبيقات تتطلب تقنيات متعددة.
6. **القابلية للتوسع:** يدعم Python البرمجة الكائنية والتنظيم القائم على الوحدات، مما يتيح للمطورين بناء تطبيقات قابلة للتوسع وسهلة الصيانة.
7. **التنوع:** يمكن استخدام Python لمجموعة واسعة من التطبيقات، بدءًا من تطوير الويب إلى الحوسبة العلمية والذكاء الاصطناعي .

2.4.3 تعريف المتغيرات في Python

يمكن تعريف المتغيرات في Python بالطرق التالية:

- **باستخدام اسم المتغير مباشرة:**

```
x = 5
```

- **باستخدام "=" :**

```
x = y = 10
```

- **باستخدام أنواع البيانات الصريحة:**

```
x: int = 5
```

- **تعدد تعريف المتغيرات:**

```
x, y, z = 1, 2, 3
```

2.4.4 أنواع البيانات في Python

يدعم Python العديد من أنواع البيانات، بما في ذلك:

- **السلاسل النصية:** تُستخدم لتمثيل البيانات النصية.

مثال:

```
name = "John Doe"
```

- **الأرقام:** تُستخدم لتمثيل البيانات الرقمية.

مثال:

```
age = 25
```

- **القوائم:** تُستخدم لتخزين عناصر متعددة في مجموعة مرتبة.

مثال:

```
fruits = ["تفاح", "موز", "برتقال"]
```

- **القواميس:** تُستخدم لتخزين أزواج المفاتيح والقيم.

مثال:

```
person = {"الاسم": "جون", "العمر": 30, "المدينة": "نيويورك"}
```

- **القيم المنطقية:** تُستخدم لتمثيل القيم المنطقية (صحيح أو خطأ).

مثال:

```
is_active = True
```

- **الأزواج:** تُستخدم لتخزين عناصر متعددة في مجموعة مرتبة غير قابلة للتعديل.

مثال:

```
coordinates = (10, 20)
```

هذه أمثلة فقط عن أنواع البيانات المتاحة في Python توفر اللغة مجموعة غنية من أنواع البيانات المدمجة وتسمح أيضًا للمطورين بإنشاء أنواع بيانات مخصصة باستخدام الفئات والكائنات.

2.5 استخدام FastAPI لتطوير واجهات برمجة التطبيقات

يعد FastAPI إطار عمل ويب حديث ذو فعالية عالية للغة Python، معروف بسرعته ومرونته وأدائه. يستغل FastAPI معيار ASGI (واجهة بوابة الخادم الغير متزامنة)، مما يتيح التعامل بسلاسة مع الطلبات الغير متزامنة. وعلاوة على ذلك، يتبع FastAPI نمط العمارة REST (نقل الحالة التمثيلية)، مما يضمن التوسعية والتوافقية وعدم الاحتفاظ بالحالة في تطوير واجهات برمجة التطبيقات. بالإضافة إلى REST، يمكن لـ FastAPI أن يمكّن المطورين من إنشاء واجهات برمجة تطبيقات GraphQL وWebSocket، مما يوسع مرونته وإمكانياته التطبيقية.

2.5.1 مزايا FastAPI

1. **السرعة والكفاءة:** FastAPI هو أحد أسرع إطارات الويب المتاحة. في دراسة من قبل مطوره اظهر أنه يمكنه التعامل مع ما يصل إلى 10,000 طلب متزامن في الثانية. يعود ذلك إلى دعمه لـ HTTP/2 ومكتبة asyncio في Python. تمكن هذه الميزات FastAPI من معالجة طلبات متعددة بشكل متزامن، مما يعزز الأداء.
2. **الأمان والمصادقة:** يتضمن FastAPI عددًا من ميزات الأمان المدمجة، مثل المصادقة باستخدام OAuth2 وCookie Sessions والحماية ضد التهديدات الشائعة مثل حقن SQL و (XSS). تساعد هذه الآليات في حماية بيانات المستخدم ومنع الوصول غير المصرح به إلى الموارد.
3. **سهولة الاستخدام:** تم تصميم FastAPI ليكون سهل الاستخدام. لديه بناء بسيط وبديهي يجعل من السهل كتابة واجهات برمجة التطبيقات باستخدامه. يتضمن FastAPI أيضًا مولد وثائق شامل يقوم بتوليد الوثائق تلقائيًا لواجهة برمجة التطبيقات الخاصة بك. يتيح ذلك للمطورين فهم كيفية استخدام واجهة برمجة التطبيقات الخاصة بك بسهولة.
4. **المرونة:** FastAPI هو إطار عمل مرن للغاية. يمكن استخدامه لإنشاء مجموعة واسعة من واجهات برمجة التطبيقات، بما في ذلك واجهات برمجة التطبيقات القائمة على REST وواجهات برمجة التطبيقات القائمة على GraphQL وواجهات برمجة التطبيقات القائمة على WebSocket. يدعم FastAPI أيضًا مجموعة واسعة من تنسيقات البيانات، بما في ذلك JSON وXML وYAML.

2.6 تقنية PyTorch

2.6.1 تعريف PyTorch

PyTorch هي مكتبة لتعلم الآلة مفتوحة المصدر مبنية على مكتبة Torch. توفر بنية تحتية مرنة وفعالة لبناء وتدريب نماذج التعلم العميق. تُستخدم PyTorch على نطاق واسع في مجالات مختلفة، بما في ذلك رؤية الحواسيب، معالجة اللغة الطبيعية وتعلم التعزيز. يوفر PyTorch رسومات حسابية ديناميكية وتفاضل تلقائي وتسريع الوحدة المركزية، مما يجعلها خيارًا شائعًا بين الباحثين والممارسين في مجال التعلم العميق.

2.6.2 مزايا PyTorch

هناك عدة أسباب تجعل PyTorch مفضلة لدى ممارسي التعلم الآلي:

1. **رسومات حسابية ديناميكية:** يستخدم PyTorch رسومات حسابية ديناميكية، مما يتيح بناء النماذج بطرق أكثر مرونة وسهولة مقارنة بالأطر الثابتة. يسهل هذا التصميم التصحيح والتجريب والتحكم الديناميكي وتحديد البنى المعقدة.
2. **بنية البرمجة البايثونية والبديهية:** تم تصميم PyTorch بناءً على البايثون، مما يوفر واجهة برمجة سهلة وبديهية. يسهل هذا التصميم التعلم والاستخدام، خاصة بالنسبة لمطوري البايثون.
3. **تسريع الوحدة المركزية بشكل فعال:** يستفيد PyTorch من قوة وحدات المعالجة المركزية لتسريع عمليات التعلم العميق. يوفر التكامل السلس مع CUDA، مما يتيح التدريب والتصنيف على الوحدات المركزية بأداء عالٍ.
4. **مجتمع قوي وبيئة متكاملة:** يتمتع PyTorch بمجتمع نشط من الباحثين والمطورين الذين يساهمون في تطويره. يوفر هذا المجتمع الدعم ويشارك المعرفة والموارد، ويوفر مجموعة واسعة من النماذج المدربة مسبقًا والمكتبات والأدوات.
5. **قابلية للتوسع والتخصيص:** يوفر PyTorch بنية تحتية مرنة تتيح للمستخدمين تخصيص وتوسيع المكتبة وفقًا لاحتياجاتهم الخاصة. يشمل ذلك بناء وحدات الشبكة العصبية المخصصة وتحديد توابع الفقدان الجديدة وتنفيذ خوارزميات التحسين المخصصة.
6. **التكامل مع المكتبات الشهيرة:** يتكامل PyTorch بسهولة مع المكتبات البايثونية الشهيرة للحوسبة العلمية ومعالجة البيانات مثل NumPy وPandas وscikit-learn. هذا يتيح المعالجة السلسة للبيانات والتحليل قبل إدخالها في نماذج PyTorch.
7. **قدرات نقل التعلم:** يدعم PyTorch نقل التعلم، مما يتيح للمطورين استخدام النماذج المدربة مسبقًا وتعديلها لمهام محددة. يوفر هذا الأمر توفير الوقت والموارد الحسابية في تدريب النماذج من البداية.

2.6.3 مقدمة حول المتغيرات في PyTorch

في PyTorch، المتغيرات هي الوحدات الأساسية لتخزين وتلاعب البيانات. يمكن تعريف المتغيرات وتلاعبها باستخدام الخطوات التالية:

- **إنشاء تنسور:** التنسور هو مصفوفة متعددة الأبعاد يمكن أن تمثل قيمًا مجردة أو متجهات أو مصفوفات أو مصفوفات عالية الأبعاد.

```
x = torch.tensor([3.0 ,2.0 ,1.0])
```

- **لف التنسور بالمتغير:** يتيح لف التنسور بالمتغير إجراء التفاضل التلقائي وحساب التدرجات أثناء الانتشار العكسي.

```
x_var = torch.autograd.Variable(x, requires_grad=True)
```

- **إجراء العمليات على المتغير:** تدعم المتغيرات العديد من العمليات الرياضية مثل الجمع والطرح والضرب والقسمة.

```
y_var = 2 * x_var + 1
```

- **الوصول إلى القيم والتدرجات:** تخزن المتغيرات كل من القيم والتدرجات للتنسورات. تمثل القيم البيانات الفعلية، بينما تمثل التدرجات مشتقة المتغير بالنسبة لمتغير هدف معين.

```
print(y_var.data) :الناتج #    tensor([.7 ,.5 ,.3])
print(y_var.grad) :الناتج #    None
```

- **إجراء الانتشار العكسي:** عملية الانتشار العكسي في PyTorch تستخدم لحساب التدرجات (المشتقات) لدالة معينة بالنسبة للمتغيرات التي يتم تطبيقها عليها هذه الدالة.

```
y_var.sum().backward()
```

- **الوصول إلى التدرجات بعد الانتشار العكسي:**

```
print(x_var.grad) :الناتج #    tensor([.2 ,.2 ,.2])
```

ملاحظة: تقدم PyTorch نموذجًا جديدًا يُطلق عليه اسم التناظر والرسومات الحسابية، وهي مختلفة عن المتغيرات التقليدية في لغة Python. التنسورات هي البنية الأساسية المستخدمة في PyTorch، بينما المتغيرات توفر وسيلة للتعامل مع التنسورات وتمكين التفاضل التلقائي.

2.7 تقنية Pydantic

2.7.1 تعريف Pydantic

Pydantic هي مكتبة Python للتحقق من البيانات وتحويلها وتوثيقها. توفر Pydantic واجهة برمجة التطبيقات (API) للتعامل مع بيانات الإدخال والمخرجات بشكل آمن وفعال. تعتمد Pydantic على التعريف الصريح للنماذج (Models) والتحقق التلقائي من صحة البيانات وتحويلها والتلاعب بها وفقًا لقواعد محددة.

2.7.2 مزايا Pydantic

توفر العديد من المزايا التي تجعلها خيارًا مفضلًا للتعامل مع بيانات الإدخال والمخرجات في تطبيقات Python:

1. **تحقق من البيانات:** يسمح Pydantic بتحديد نماذج البيانات (Models) والتحقق تلقائياً من صحة البيانات المرسلة أو المستلمة.
2. **تحويل البيانات:** يمكن لـ Pydantic تحويل البيانات المدخلة إلى كائنات Python وبالعكس.
3. **توثيق البيانات:** يوفر Pydantic وسيلة سهلة لتوثيق البيانات والتعامل معها، مما يسهل فهم واستخدام البيانات بشكل صحيح.
4. **تحقق الثقة:** باستخدام Pydantic، يمكن التأكد من أن البيانات المستخدمة في التطبيق تتوافق مع التعاريف المحددة للنماذج.
5. **سهولة الاستخدام:** توفر Pydantic واجهة سهلة الاستخدام لتحديد النماذج والتعامل مع البيانات، وتقدم أيضًا رسائل الخطأ التفصيلية في حالة عدم تطابق البيانات.
6. **تعزيز الإنتاجية:** يساعد Pydantic في تسريع عملية تطوير التطبيقات من خلال التحقق التلقائي من البيانات وتوفير أدوات قوية للتلاعب بها.
7. **التوافق مع البيانات:** يمكن استخدام Pydantic مع مصادر البيانات المختلفة والتكامل مع تقنيات أخرى في بيئة التطبيق.

2.7.3 تعريف النماذج في Pydantic

في Pydantic، يتم تعريف النماذج باستخدام الكائنات المشتقة من pydantic.BaseModel ويتم تحديد حقول النموذج وأنواع البيانات المتوقعة لكل حقل.

مثال:

```
from pydantic import BaseModel


class Person(BaseModel):
    name: str
    age: int
    email: str
```

في المثال أعلاه، تم تعريف نموذج Person باستخدام BaseModel من Pydantic يحتوي النموذج على ثلاثة حقول : name و age وemail، وتحدد أنواع البيانات المتوقعة لكل حقل.

تستخدم Pydantic التعاريف المحددة في النماذج للتحقق من صحة البيانات وتحويلها وتوثيقها بشكل آمن وفعال.

الفصل الثالث القسم العملي

3.1 دليل التثبيت

3.1.1 تثبيت Python

1. قم بتنزيل مثبت Python من الموقع الرسمي: https://www.python.org/downloads

2. قم بتشغيل ملف المثبت واتبع التعليمات. تأكد من تحديد المربع الذي يقول "إضافة Python إلى PATH" قبل النقر فوق تثبيت الآن.

3. بعد التثبيت، يمكنك التحقق من نسخة Python وpip عن طريق فتح موجه أوامر جديد وكتابة:

```
python --version pip --version
```

3.1.2 تثبيت الاعتماديات

لتثبيت الاعتماديات، ستحتاج إلى استخدام pip، وهو مدير حزم لـ Python والذي يأتي مع تثبيت Python. هنا شرح عن الاعتماديات والأوامر التي ستحتاج لتشغيلها في ال Terminal لتثبيت كل منها:

1. **FastAPI:** FastAPI هو إطار عمل حديث وسريع (عالي الأداء) لبناء واجهات برمجة التطبيقات مع Python 3.6+ بناءً على تلميحات النوع القياسية في Python.

```
pip install fastapi
```

2. **Pillow:** Pillow (PIL) هو نسخة من PIL (مكتبة الصور في Python). إنه يضيف بعض الميزات الودية للمستخدم مثل تحميل الصور مباشرة.

```
pip install pillow
```

3. **Pydantic:** Pydantic هو مكتبة تحقق من صحة البيانات. يستخدم لتأكيدات النوعية لـ Python للتحقق من أن هياكل البيانات (مثل JSON) تطابق تنسيق مرغوب.

```
pip install pydantic
```

4. **Transformers:** يوفر Transformers آلاف النماذج المدربة مسبقاً لإجراء المهام على النصوص مثل التصنيف، واستخراج المعلومات، والإجابة على الأسئلة، والتلخيص، والترجمة، وتوليد النصوص، وأكثر من ذلك في أكثر من 100 لغة.

```
pip install transformers
```

5. **Torch:** Torch هو إطار عمل للحوسبة العلمية يقدم دعمًا واسعًا لخوارزميات التعلم الآلي.

```
pip install torch
```

6. **Requests:** تسمح لك Requests بإرسال طلبات HTTP/1.1. معها، يمكنك إضافة محتوى مثل الرؤوس، وبيانات النموذج، والملفات المتعددة الأجزاء، والمعلمات عبر مكتبات Python البسيطة إلى طلبات HTTP.

```
pip install requests
```

3.2 اختيار نموذج CLIP المناسب

أفضل نموذج للسرعة العالية والدقة هو openai/clip-vit-base-patch32. لديه دقة top-1 تبلغ 86.1٪ ودقة top-5 تبلغ 97.4٪، ولديه تكلفة حوسبة للاستدلال تبلغ 100 مللي ثانية. هذا يجعله خيارًا جيدًا لكل من السرعة والدقة، ويمكننا التحقق من استخدام الذاكرة العشوائية له مع الكود التالي:

```
from transformers import CLIPProcessor, CLIPModel
import os
import psutil

start_memory = psutil.virtual_memory().used

model_name = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(model_name)
clip_model = CLIPModel.from_pretrained(model_name)
end_memory = psutil.virtual_memory().used

memory_used = end_memory - start_memory
```

يتضمن **الفصل 5** سكريبت كامل لاختبار أداء ودقة النموذج المختار.

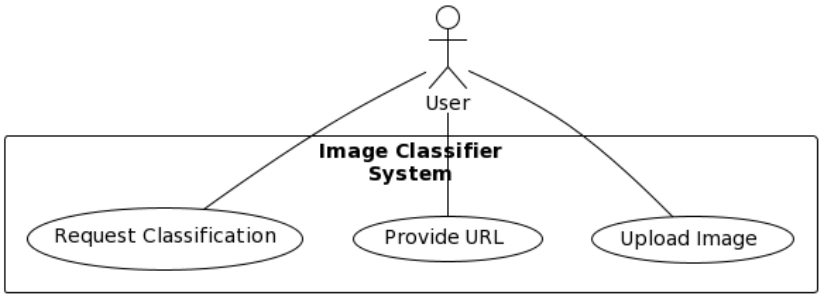
3.3 شرح التطبيق

1. يتم استيراد المكتبات اللازمة، بما في ذلك FastAPI لخادم الويب، وPIL لمعالجة الصور، وPydantic للتحقق من صحة البيانات، وtransformers لنموذج CLIP والمعالج، وغيرها من الأدوات المساعدة للوظائف المختلفة.
2. يتم تهيئة تطبيق FastAPI لمعالجة الطلبات الواردة.
3. يتم تعريف متغير `model_name`، محددًا النموذج CLIP المدرب مسبقًا للتحميل، ويتم تحميل نموذج CLIP والمعالج من مكتبة `transformers`.
4. يتم تعريف نموذج Pydantic باسم `PredictionResult` لتنظيم الرد من النقطة النهائية يتضمن التنبؤ، واحتماليات كل تصنيف، والحمولة الأصلية.
5. يتم تعريف نقطة النهاية `/classify/` باستخدام المُزين `app.post`. تقبل هذه النقطة النهائية طلبات HTTP POST.
6. داخل وظيفة `classify_image`، يتم تحليل معلمات الطلب لاستخراج موقع الصورة، والمصنف، والفئات للتنبؤ بها.

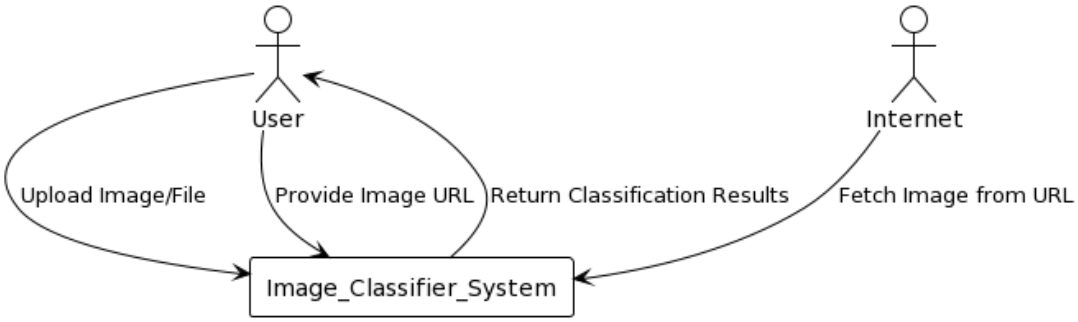
7. تتحقق الوظيفة إذا كان يتم توفير ملف صورة أو URL للموقع. إذا لم يكن الأمر كذلك، فإنها تثير استثناء HTTPException مع رسالة خطأ مناسبة.
8. يتم تحليل الفئات (التصنيفات) من الطلب وتخزينها كقائمة.
9. يتم معالجة الصورة بناءً على ما إذا كان يتم توفير ملف تم تحميله أو URL. إذا تم تحميل ملف صورة، يتم فتحه وتغيير حجمه باستخدام PIL إلى أبعاد 224 x 224 بكسل، وهو حجم الإدخال المتوقع للنموذج. إذا تم توفير URL، يتم إرسال طلب GET لاسترداد الصورة، ومن ثم يتم فتحها وتغيير حجمها.
10. يتم تمرير الصورة المعالجة والفئات إلى المعالج CLIP لإعدادها للنموذج. الوظيفة `processor` تأخذ النص (الفئات) وإدخالات الصورة، تعيد أجسام Tensor من PyTorch.
11. يتم تمرير الإدخالات المعالجة إلى نموذج CLIP، الذي يولد إخراجًا يحتوي على توقعات النموذج لكل صورة.
12. يتم تمرير إخراجات النموذج من خلال وظيفة softmax لتحويلها إلى احتمالات. تطبق الوظيفة `F.softmax` من PyTorch softmax على البعد الأخير من Tensor وتعيد قائمة من الاحتمالات.
13. يتم دمج الفئات والاحتمالات في قاموس يُسمى `label_probs` باستخدام وظائف `zip` و `dict`.
14. يتم تحديد الفئة المتوقعة (التصنيف) من خلال العثور على الفئة ذات الاحتمال الأعلى باستخدام الوظيفة `max` والمعامل `key` مضبوط على `label_probs.get`.
15. يتم إنشاء كائن `PredictionResult`، يحتوي على التنبؤ، واحتمالات التصنيف، والحمولة الأصلية (الموقع والمصنف).
16. يتم إرجاع كائن `PredictionResult` كرد من النقطة النهائية.

3.3.1 مخططات DFD, UML

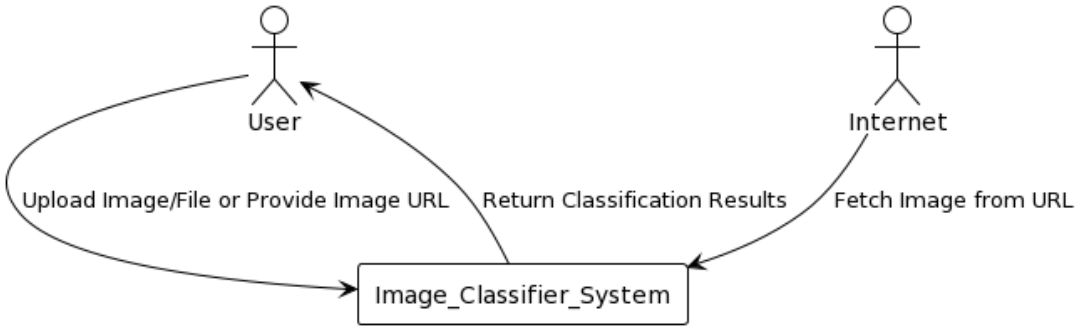
Use Case Diagram - Image Classifier System

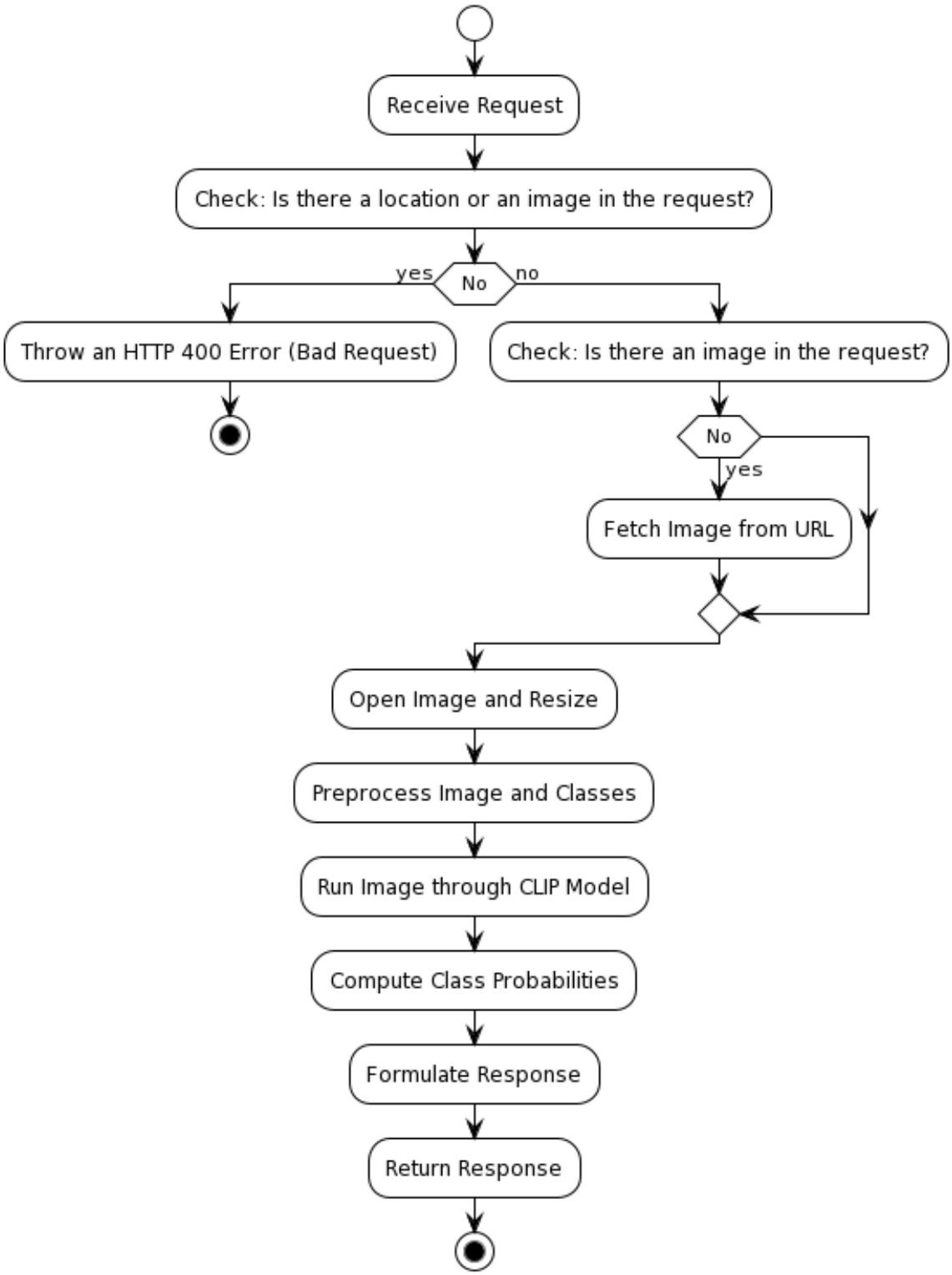


Context DFD - Image Classifier System



Level 0 DFD - Image Classifier System




UML Diagram - Image Classifier System
Receive Request
Check: Is there a location or an image in the request?
yes
No
no
Throw an HTTP 400 Error (Bad Request)
Check: Is there an image in the request?
No
yes
Fetch Image from URL
Open Image and Resize
Preprocess Image and Classes
Run Image through CLIP Model
Compute Class Probabilities
Formulate Response
Return Response

3.4 تجهيز التطبيق

نقوم في البداية بتهيئة تطبيق FastAPI واستيراد التبعيات اللازمة. تتضمن هذه التبعيات FastAPI نفسه، وأصناف الأدوات المساعدة للتعامل مع الطلبات وتحميل الملفات، وPydantic للتحقق من صحة البيانات، والمكتبات اللازمة لاستخدام نموذج CLIP.

```
from fastapi import FastAPI, Form, File, UploadFile, HTTPException
from PIL import Image
from pydantic import BaseModel
from transformers import CLIPProcessor, CLIPModel
import torch.nn.functional as F
import io
from urllib.parse import urlparse
import requests
```

3.4.1 توابع المساعدة

نحن نحدد أيضًا وظائف المساعدة get_image_from_upload و get_image_from_url للتعامل مع معالجة الصور. هاتان الوظيفتان تحاولان فتح وتغيير حجم الصورة وتثيران HTTPException في حالة وجود أي مشاكل.

```
async def get_image_from_upload(image: UploadFile):
    try:
        return Image.open(io.BytesIO(await image.read())).resize((224, 224))
    except IOError:
        raise HTTPException(status_code=400, detail="Invalid image file.")

async def get_image_from_url(url: str):
    if urlparse(url).scheme not in ['http', 'https']:
        raise HTTPException(status_code=400, detail="Invalid image URL.")
    try:
        response = requests.get(url)
        response.raise_for_status()
        return Image.open(io.BytesIO(response.content)).resize((224, 224))
    except (requests.RequestException, IOError):
        raise HTTPException(status_code=400, detail="Unable to download image
from the provided URL.")
```

3.5 تحميل نموذج CLIP والمعالج المقابل

يتم تحميل نموذج CLIP والمعالج المقابل من مكتبة transformers, model_name يحدد النموذج المدرب مسبقا للتحميل. هذه العملية تحدث مرة واحدة عند بدء تشغيل الخادم.

```
model_name = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(model_name)
clip_model = CLIPModel.from_pretrained(model_name)
```

3.6 تحديد نماذج البيانات

يتم استخدام نماذج Pydantic للتحقق التلقائي من صحة البيانات. هنا نحدد نموذج PredictionResult لهيكلة الاستجابة من نقطة النهاية الخاصة بنا.

```
class PredictionResult(BaseModel):
    prediction: str
    labelProbabilities: dict
    originalPayload: dict
```

3.7 تنفيذ نقطة النهاية لتصنيف الصور

لقد استخدمنا Form و File من FastAPI لتلقي location، classifier، classes، و image مباشرة كمعاملات في نقطة النهاية.

```
@app.post("/classify/")
async def classify_image(
    location: Optional[str] = Form(None),
    classifier: Optional[str] = Form("clip"),
    classes: Optional[str] = Form("nsfw, sfw"),
    image: UploadFile = File(None)
):
```

3.8 معالجة الإدخال

تم وضع معالجة الأخطاء لضمان توفير location أو image. يتم استخراج الفئات (التصنيفات) للتصنيف وتقسيمها إلى قائمة.

```
if not location and not image:
    raise HTTPException(status_code=400, detail="Please provide either 'locati
on' or 'image'.")

classes = classes.strip(' ').split(', ')
```

3.9 التعامل مع تحميل الصورة

إذا تم تحميل ملف صورة، يتم قراءته في الذاكرة، وتحويله إلى بايتات، ثم تحويله إلى كائن صورة PIL. إذا تم توفير URL، يتم إرسال طلب GET إلى URL، ويتم قراءة بيانات الصورة من الرد، ثم تحويلها إلى كائن صورة PIL.

```
if image:
    pil_image = await get_image_from_upload(image)
else:
    pil_image = await get_image_from_url(location)
```

3.10 معالجة المدخلات وإجراء التوقعات

يتم معالجة المدخلات باستخدام CLIPProcessor، ثم تمريرها إلى نموذج CLIP لإنتاج المخرجات

. ثم يتم تحويل المخرجات إلى احتمالات باستخدام وظيفة softmax.

```
inputs = processor(text=classes, images=pil_image, return_tensors="pt",
padding=True)
outputs = clip_model(**inputs)
probs = F.softmax(outputs.logits_per_image, dim=-1).tolist()[0]
```

3.11 بناء الرد

يتم بناء الرد باستخدام نموذج PredictionResult. يتم اختيار التصنيف ذو الاحتمال الأعلى بوصفه التوقع. يحتوي الرد على التوقع، والاحتمالات لكل تصنيف، والحمولة الأصلية.

```
label_probs = dict(zip(classes, probs))
prediction = max(label_probs, key=label_probs.get)
result = PredictionResult(
    prediction=prediction,
    label_probabilities=label_probs,
    original_payload={
        "location": location,
        "classifier": "clip"
    }
)

return {"result": result}
```

3.12 الرمز الكامل

```
from typing import Optional
from fastapi import FastAPI, Form, File, UploadFile, HTTPException
from PIL import Image
from pydantic import BaseModel
from transformers import CLIPProcessor, CLIPModel
import torch.nn.functional as F
import io
from urllib.parse import urlparse
import requests

app = FastAPI()

model_name = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(model_name)
clip_model = CLIPModel.from_pretrained(model_name)

class PredictionResult(BaseModel):
    prediction: str
    label_probabilities: dict
    original_payload: dict

async def get_image_from_upload(image: UploadFile):
    try:
        return Image.open(io.BytesIO(await image.read())).resize((224, 224))
    except IOError:
        raise HTTPException(status_code=400, detail="Invalid image file.")

async def get_image_from_url(url: str):
    if urlparse(url).scheme not in ["http", "https"]:
        raise HTTPException(status_code=400, detail="Invalid image URL.")
    try:
        response = requests.get(url)
        response.raise_for_status()
        return Image.open(io.BytesIO(response.content)).resize((224, 224))
    except (requests.RequestException, IOError):
        raise HTTPException(
            status_code=400, detail="Unable to download image from the provide
d URL."
        )

@app.post("/classify/")
async def classify_image(
    location: Optional[str] = Form(None),
    classifier: Optional[str] = Form("clip"),
    classes: Optional[str] = Form("nsfw, sfw"),
```

```
    image: UploadFile = File(None),
):
    # Check if either 'location' or 'image' is provided
    if not location and not image:
        raise HTTPException(
            status_code=400, detail="Please provide either 'location' or 'imag
e'."
        )
    # Parse classes string into a list
    classes = classes.strip(" ").split(", ")
    # Process the image based on whether it is uploaded or from a URL
    if image:
        pil_image = await get_image_from_upload(image)
    else:
        pil_image = await get_image_from_url(location)
    # Prepare inputs for the CLIP model
    inputs = processor(
        text=classes, images=pil_image, return_tensors="pt", padding=True
    )
    # Pass inputs to the CLIP model and generate outputs
    outputs = clip_model(**inputs)
    # Convert the outputs to probabilities using softmax
    probs = F.softmax(outputs.logits_per_image, dim=-1).tolist()[0]
    # Combine classes and probabilities into a dictionary
    label_probs = dict(zip(classes, probs))
    # Find the class with the highest probability as the prediction
    prediction = max(label_probs, key=label_probs.get)
    # Create a PredictionResult object to structure the response
    result = PredictionResult(
        prediction=prediction,
        label_probabilities=label_probs,
        original_payload={"location": location, "classifier": "clip"},
    )
    return {"result": result}
```

3.13 اختبار التطبيق باستخدام Postman

Postman هو أداة شهيرة لاختبار الواجهات البرمجية للتطبيقات (APIs). فيما يلي كيفية استخدامه لاختبار تطبيق FastAPI:

1. **بدء خادم FastAPI:** قم بتشغيل تطبيق FastAPI على جهاز الكمبيوتر الخاص بك. افتح Terminal، انتقل إلى الدليل الذي يحتوي على البرنامج النصي الخاص بك، وقم بتشغيل الأمر uvicorn main:app --port 8000 --reload يفترض أن البرنامج النصي الخاص بك يسمى main.py، وسوف يعمل الخادم على المنفذ 8000.

2. **تثبيت وفتح Postman:** إذا لم تقم بذلك بالفعل، قم بتنزيل وتثبيت Postman من موقعهم الرسمي https://www.postman.com/downloads بمجرد التثبيت، افتح Postman.

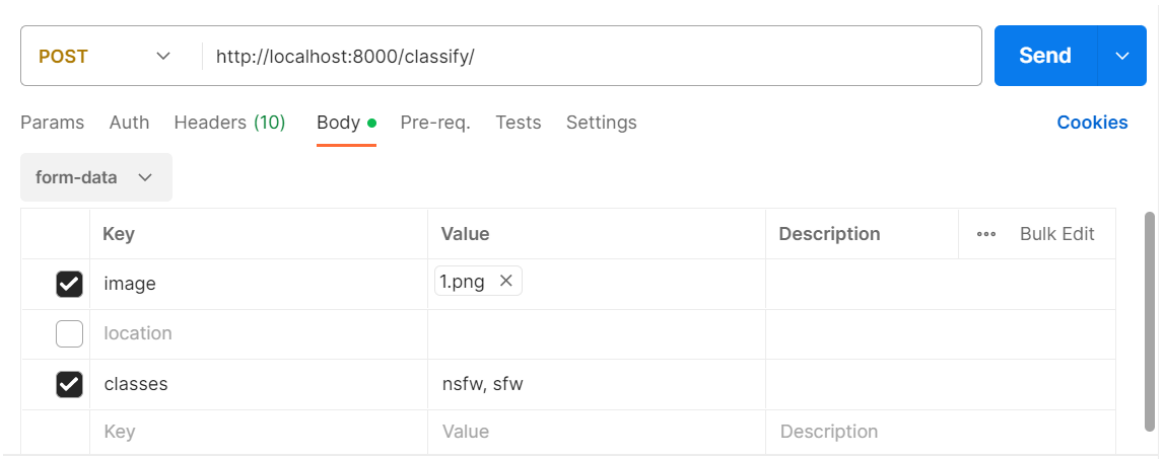
3. **إنشاء طلب جديد:** انقر على زر + لإنشاء علامة تبويب جديدة في Postman. حدد POST من القائمة المنسدلة وأدخل URL للخادم المحلي، الذي يجب أن يكون http://localhost:8000/classify

4. **إعداد الطلب:** في علامة التبويب Body، حدد form-data. هذا يتيح لك إرسال كلا من الملفات والبيانات الإضافية في الطلب. أدخل image في حقل Key، حدد File من القائمة المنسدلة، ثم انقر على Select Files لاختيار ملف صورة من جهاز الكمبيوتر الخاص بك. لإرسال بيانات إضافية، أدخل المفتاح (مثلاً location, classifier, classes) في حقل Key جديد والقيمة المقابلة في حقل Value.

5. **إرسال الطلب:** انقر على زر Send لإرسال الطلب. سيتم عرض الرد من الخادم في الجزء السفلي من النافذة.

يجب التأكد من استبدال المفاتيح والقيم بتلك التي يتوقعها API الخاص بك. على سبيل المثال، إذا كان API الخاص بك يتوقع مفتاح location في جسم الطلب، يجب أن تدخل location في حقل Key وURL الصورة في حقل Value.

يرجى ملاحظة أن الخادم يجب أن يكون قيد التشغيل ويمكن الوصول إليه من الجهاز الذي يعمل عليه Postman. إذا كنت تشغل الخادم وPostman على نفس الجهاز، يمكنك استخدام localhost كاسم المضيف في URL. إذا كانوا على أجهزة مختلفة، استبدل localhost بعنوان IP أو اسم المضيف للجهاز الذي يشغل الخادم.



الفصل الرابع الاختبار والتقييم

4.1 مقدمة

في الفصول السابقة، ناقشنا تكامل نموذج CLIP في التطبيق الخاص بنا. الآن، من الأساسي التركيز على أداء النموذج نفسه لأنه له أكبر تأثير على الأداء. سوف يدخل هذا الفصل في الأساليب والمقاييس المستخدمة لتقييم أداء النموذج في سياق تطبيق تصنيف الصور الخاص بنا.

4.2 تقييم أداء النموذج

يتم تقييم أداء نموذج تصنيف الصور الخاص بنا بناءً على دقته في التنبؤ بالتصنيفات الصحيحة لمجموعة معينة من الصور. هذه العملية التقييمية منهجية وتتضمن العديد من الخطوات الرئيسية، بما في ذلك معالجة مسبقة للصور، توليد التوقعات باستخدام النموذج المدرب، ومقارنة هذه التوقعات مع التصنيفات الفعلية.

4.2.1 تثبيت الاعتماديات

لتثبيت الاعتماديات، ستحتاج إلى استخدام pip، وهو مدير حزم لـ Python والذي يأتي مع تثبيت Python. هنا شرح عن الاعتماديات والأوامر التي ستحتاج لتشغيلها في ال Terminal لتثبيت كل منها:

os: هي وحدة في Python توفر واجهة للتفاعل مع نظام التشغيل. تسمح لك بتنفيذ عمليات مثل التلاعب بالملفات والمجلدات والوصول إلى متغيرات البيئة وغيرها من وظائف نظام التشغيل.

time: هي وحدة في Python توفر وظائف للتعامل مع الوقت والتوقيت. يمكن استخدامها لقياس الوقت المستغرق في تنفيذ البرنامج وللتحكم في الوقت وتنسيق التواريخ والأوقات.

psutil: هي مكتبة للحصول على معلومات حول النظام والموارد المتاحة في .Python توفر واجهة برمجة التطبيقات للوصول إلى معلومات مثل استخدام المعالج والذاكرة والشبكة وحالة العمليات في النظام.

```
pip install psutil
```

sklearn: هي مكتبة مشهورة لتعلم الآلة في .Python توفر مجموعة كبيرة من الأدوات والخوارزميات للتعلم الآلي والتنقيب عن البيانات وتقييم النماذج وإجراء التحليل الإحصائي.

```
pip install scikit-learn
```

transformers: هي مكتبة متخصصة في مجال معالجة اللغة الطبيعية وتطبيقات التعلم العميق. توفر واجهة برمجة التطبيقات لاستخدام النماذج المدربة مسبقًا مثل تصنيف النصوص وترجمة اللغة والتلخيص والترجمة والاستخراج النصي وأكثر من ذلك.

```
pip install transformers
```

torch: هي مكتبة للحوسبة العلمية وتعلم الآلة في .Python توفر بنية تحتية قوية لتنفيذ الحسابات العلمية وتدعم عمليات التحسين والتحليل الرقمي وشبكات العصب الاصطناعية والتعلم العميق.

```
pip install torch
```

Pillow: Pillow (PIL) هو نسخة من PIL (مكتبة الصور في Python). إنه يضيف بعض الميزات الودية للمستخدم مثل تحميل الصور مباشرة.

```
pip install pillow
```

بعد تشغيل هذه الأوامر، يتم تثبيت الاعتماديات المطلوبة لاستخدام الوحدات التي تم استيرادها من هذه المكتبات في الكود.

4.2.2 استيراد المكتبات اللازمة وتحميل النموذج:

تبدأ عملية التقييم باستيراد مكتبات البايثون اللازمة مثل os, psutil, time, PIL (مكتبة معالجة الصور البايثون)، و sklearn.metrics. كما نستورد النموذج المدرب مسبقا CLIPModel و CLIPProcessor من مكتبة transformers. يتم تحميل النموذج CLIPModel باستخدام model_name الذي هو-openai/clip-vit-base patch32.

```
import os
import psutil
import time
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
score, confusion_matrix
from transformers import CLIPProcessor, CLIPModel
import torch
import torch.nn.functional as F
from PIL import Image

model_name = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(model_name)
clip_model = CLIPModel.from_pretrained(model_name)
```

4.2.3 تعريف وظيفة التقييم

تم تعريف الوظيفة evaluate_model للتعامل مع عملية التقييم. تأخذ ثلاث معاملات: image_directory، test_label، و label_list. image_directory هو الدليل الذي يحتوي على الصور الاختبار، test_label هو التصنيف الصحيح لجميع الصور في الدليل، و label_list هو قائمة جميع التصنيفات الممكنة. تقوم الوظيفة بتهيئة قائمة فارغة من predictions ومتغيرات لتخزين إجمالي وقت التنبؤات والذاكرة المستخدمة.

```
def evaluate_model(image_directory, test_label, label_list):
    predictions = []
    total_time = 0
    start_memory = psutil.virtual_memory().used
```

4.2.4 تحميل ومعالجة الصور

يحصل التابع على قائمة بجميع الصور في الدليل وتقوم بتهيئة قائمة من تصنيفات الاختبار وفقًا لـ test_label المقدم. يتم فتح كل صورة في الدليل وتغيير حجمها ومعالجتها إلى أشكال tensors التي يمكن تمريرها إلى النموذج. يتم تسجيل الوقت في بداية التنبؤ.

```
test_images= [os.path.join(image_directory, img) for img in os.listdir(image_directory)
if img.endswith((".png", ".jpg", ".jpeg"))]
    test_labels = [test_label for _ in range(len(test_images))]

    for index, image in enumerate(test_images, start=1):
        # Open and resize the image, then process it with the model's processor
        pil_image = Image.open(image).resize((224, 224))
        inputs = processor(
            text=label_list, images=pil_image, return_tensors="pt", padding=True
        )
```

4.2.5 التنبؤ بالنموذج وقياس الأداء

يقوم النموذج بإجراء التنبؤ استناداً إلى الصورة المعالجة، ويتم تسجيل الوقت في نهاية التنبؤ. يتم إضافة الفرق بين أوقات النهاية والبداية إلى متغير total_time. إذا لم يطابق تنبؤ النموذج test_label، يتم إلحاق مسار الصورة إلى قائمة incorrect_images لتحليل أكثر.

```
    outputs = clip_model(**inputs)
    probs = F.softmax(outputs.logits_per_image, dim=-1).tolist()[0]
    label_probs = dict(zip(label_list, probs))
    end_time = time.time()
    total_time += end_time - start_time
    prediction = max(label_probs, key=label_probs.get)
    predictions.append(prediction)
```

4.2.6 حساب المقاييس

بعد معالجة جميع الصور، تقوم الوظيفة بحساب العديد من المقاييس لقياس أداء النموذج. تشمل هذه Accuracy، Prediction، Recall، معدل F1، الوقت الإجمالي لكل صورة، والذاكرة المستخدمة أثناء العملية.

```
    accuracy = accuracy_score(test_labels, predictions)
    precision = precision_score(test_labels, predictions, average="weighted")
    recall = recall_score(test_labels, predictions, average="weighted")
    f1 = f1_score(test_labels, predictions, average="weighted")
    avg_time_per_image = total_time / len(test_images)
    end_memory = psutil.virtual_memory().used
    memory_used = end_memory - start_memory
```

4.2.7 طباعة نتائج التقييم

أخيرًا، تقوم الوظيفة بطباعة المقاييس المحسوبة، مصفوفة الالتباس التي تظهر توزيع التنبؤات الصحيحة وغير الصحيحة، وقائمة الصور التي تم تصنيفها بشكل غير صحيح.

```
print(f"Model accuracy: {accuracy * 100:.2f}%")
print(f"Model precision: {precision * 100:.2f}%")
print(f"Model recall: {recall * 100:.2f}%")
print(f"Model F1-score: {f1 * 100:.2f}%")
print(f"Average time per image: {avg_time_per_image:.5f} seconds")
print(f"Memory used: {memory_used / (1024**2):.2f} MB")

cm = confusion_matrix(test_labels, predictions, labels=label_list)
print("Confusion matrix:")
print(cm)

if not incorrect_images:
    print("No incorrect images found.")
else: print("Incorrectly classified images:")
    print(incorrect_images)
```

4.2.8 تنفيذ وظيفة التقييم:

يتم ثم استدعاء الوظيفة evaluate_model مع المعاملات المناسبة لبدء عملية التقييم.

```
#Folder containing the test set for a label

image_directory = "/content/test/nsfw"

#The label that the images will be tested against
test_label = "nsfw"

#The list of labels that will be used to test against

label_list = ["nsfw", "sfw"]

evaluate_model(image_directory, test_label, label_list)
```

4.2.8 الكود الكامل

```
import os
import psutil
import time
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, co
nfusion_matrix
from transformers import CLIPProcessor, CLIPModel
import torch
import torch.nn.functional as F
from PIL import Image
import shutil

start_memory = psutil.virtual_memory().used

# Check if a GPU is available and if not, fall back to CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

if torch.cuda.is_available():
    print("Using GPU for Processing")
else:
    print("Using CPU for Processing")

model_name = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(model_name)
clip_model = CLIPModel.from_pretrained(model_name).to(device)

def move_incorrect_images(incorrect_images, target_directory):
    if not os.path.exists(target_directory):
        os.makedirs(target_directory)
    for image in incorrect_images:
        shutil.move(image, target_directory)

def evaluate_model(image_directory, test_label, label_list, threshold):
    # Initialize arrays for predictions and incorrectly classified images
    predictions = []
    total_time = 0
    incorrect_images = []
    unknown_images = []

    # Retrieve all images from the image directory
    test_images = [
        os.path.join(image_directory, img)
        for img in os.listdir(image_directory)
        if img.endswith((".png", ".jpg", ".jpeg"))
    ]

    # Set the labels for all test images to be the test_label
    test_labels = [test_label for _ in range(len(test_images))]

    for index, image in enumerate(test_images, start=1):
        # Open and resize the image, then process it with the model's processor
```

```
        pil_image = Image.open(image).resize((224, 224))
        inputs = processor(
            text=label_list, images=pil_image, return_tensors="pt", padding=True
        )

        # Move inputs to the selected device
        inputs = {name: tensor.to(device) for name, tensor in inputs.items()}

        # Track the start time
        start_time = time.time()

        # Generate model outputs for the inputs
        outputs = clip_model(**inputs)
        # Convert outputs to probabilities
        probs = F.softmax(outputs.logits_per_image, dim=-1).tolist()[0]
        # Create a dictionary of labels and their corresponding probabilities
        label_probs = dict(zip(label_list, probs))

        # Track the end time
        end_time = time.time()
        # Calculate total processing time
        total_time += end_time - start_time

        # Determine the prediction with the highest probability
        prediction = max(label_probs, key=label_probs.get)
        # Add the prediction to the predictions array
        predictions.append(prediction)

        # Display the image being processed
        print(f"Processed image {index} of {len(test_images)}: {image} : Prediction {pr
ediction}")

        # Check if the prediction matches the test label; if not, check the threshold
        if prediction != test_label:
            incorrect_images.append(image)
            continue

        if label_probs[prediction] < threshold:
            unknown_images.append(image)

    # Calculate evaluation metrics
    accuracy = accuracy_score(test_labels, predictions)
    precision = precision_score(test_labels, predictions, average="weighted")
    recall = recall_score(test_labels, predictions, average="weighted")
    f1 = f1_score(test_labels, predictions, average="weighted")
    avg_time_per_image = total_time / len(test_images)
    end_memory = psutil.virtual_memory().used
    memory_used = end_memory - start_memory

    # Display the evaluation metrics and memory usage
    print(f"Model accuracy: {accuracy * 100:.2f}%")
    print(f"Model precision: {precision * 100:.2f}%")
    print(f"Model recall: {recall * 100:.2f}%")
```

```
    print(f"Model F1-score: {f1 * 100:.2f}%")
    print(f"Average time per image: {avg_time_per_image:.5f} seconds")
    print(f"Memory used: {memory_used / (1024**2):.2f} MB")

    # Display the confusion matrix
    cm = confusion_matrix(test_labels, predictions, labels=label_list)
    print("Confusion matrix:")
    print(cm)



    # Display the incorrectly classified images
    if not incorrect_images:
        print("No incorrect images found.")
    else:
        print("Incorrectly classified images:")
        print(incorrect_images)

    # Display the unknown images
    if not unknown_images:
        print("No unknown images found.")
    else:
        print("Unknown images:")
        print(unknown_images)

    # Move incorrectly classified images to a separate directory
    target_directory = image_directory + "/Incorrect"
    if incorrect_images:
        move_incorrect_images(incorrect_images, target_directory)

    # Move unknown images to a separate directory
    target_directory = image_directory + "/Unknown"
    if unknown_images:
        move_incorrect_images(unknown_images, target_directory)


image_directory = "./temp/Incorrect"
test_label = "nsfw"
label_list = ["nsfw", "sfw"]

evaluate_model(image_directory, test_label, label_list, 0.48)
```

4.3 النتائج

تم اختبار تصنيف الصور لـ openai/clip-vit-base-patch32 على وحدة المعالجة المركزية Intel Core i5-1135G7، لم يتم استخدام تسريع وحدة المعالجة الرسومية، الصور مأخوذة من مواقع الويب التي تحتوي على محتوى غير آمن للعمل، وتحتوي على صور آمنة وغير آمنة للعمل وتمت مراجعتها يدويا، هذه هي النتائج من هذه الصور:

NSFW (5,422 Images)

Metric	Result
Model accuracy	96.62%
Model precision	100.00%
Model recall	96.62%
Model F1-score	98.28%
Average time per image	0.10913 seconds
Memory used	470.63 MB

SFW (444 Images)

Metric	Result
Model accuracy	77.25%
Model precision	100.00%
Model recall	77.25%
Model F1-score	87.17%
Average time per image	0.10956 seconds
Memory used	674.94 MB

- الصور الآمنة للعمل تحتوي على الكثير من حالات الحافة حيث لا يمكن للعين البشرية تحديد ما إذا كانت الصورة آمنة للعمل أم لا بدون قواعد واضحة، وهذا ينتج عنه خطأ بشري يمكن أن يؤثر على الدراسة.
- هذه المجموعة الاختبارية محدودة الحجم وهناك حاجة إلى مجموعة اختبار أكبر ولكنها غير متوفرة لذا كان علينا استخدام مسح الويب للحصول على الصور، كل من الصور آمنة للعمل وغير آمنة للعمل مأخوذة من نفس النطاقات لكي تكون الدراسة أكثر دقة.
- استخدام نموذج أكبر مثل openai/clip-vit-large-patch14 سوف يعطي نتائج أفضل، ولكن بطء الاستدلال (3 ثواني للصورة) وزيادة استخدام الذاكرة هي عيوب موجودة.
- الصور التي تقل عن 244 ب 244 أكثر عرضة للتصنيف الخاطئ لقلة وجود عدد كافي من البكسلات للتصنيف.
- اختيار التصنيفات (Labels Text) يؤثر على النتائج لذا يجب اختيار التصنيفات للإنتاج بعناية وبعد الاختبار.

الفصل الخامس التوجهات المستقبلية والخاتمة

5.1 الاعتبارات والاتجاهات المستقبلية

استخدام تصنيف CLIP للرقابة على الصور يقدم حلاً فورياً، ولكنه مؤقت، لمشكلة مستمرة. يخفف هذا النموذج المتاح بسهولة الحاجة إلى التدريب المكلف في البداية ويقدم مرونة مع توسيع قاعدة المستخدمين ومجموعة التدريب. من المرجح أن تقوم عدة إضافات مستقبلية بتعزيز فعالية النظام وتشمل:

1. **المعالجة بالدفعات والفيديو:** تمكين تحميل وتصنيف صور متعددة في وقت واحد وتوسيع الوظائف لتضمن تصنيف الفيديو، من خلال تقييم الإطارات المستخرجة من محتوى الفيديو.

2. **تخصيص النموذج وتمكين استخدام نماذج مختلفة:** السماح للمستخدمين بتحميل بيانات التدريب الخاصة بهم للحصول على نتائج دقيقة وشخصية، وتضمين نماذج مختلفة وخطط مختلفة للاختيار.

3. **توفير الواجهة:** توفير واجهة الويب لتفاعل المستخدم السلس

4. **التكامل بين الخدمات والتصنيف في الوقت الحقيقي:** تيسير التوصيل مع خدمات التخزين السحابي، أو أنظمة إدارة المحتوى، أو منصات وسائل التواصل الاجتماعي لتصنيف الصور مباشرة، ودمج قدرات التصنيف الفوري أو البث المباشر.

5. **التغذية الراجعة، والقدرة على التوسع، وتدابير الأمان:** إقامة آلية تغذية راجعة للمستخدم لتصحيح التصنيفات الخاطئة، مما يساهم في تحسين النموذج. تحسين التطبيق لتحسين القدرة على التوسع من خلال التوازن بين الأحمال، والمعالجة الموزعة، أو تحسين النموذج، إلى جانب التعامل الآمن مع الملفات، والمصادقة، وتحكمات الوصول، والتدقيقات الأمنية الدورية.

5.2 الخاتمة

تطوير تصنيف الصور على أساس FastAPI وCLIP هو عمل قيد التقدم. FastAPI هو إطار عمل ويب عالي الأداء يمكن استخدامه لبناء تطبيقات ويب قابلة للتطوير وموثوقة. CLIP هو نموذج لغوي كبير يمكن استخدامه لتصنيف الصور.

يمكن استخدام مزيج FastAPI وCLIP لبناء تطبيق تصنيف صور قوي. ومع ذلك، لا تزال هناك بعض المجالات التي يمكن تحسين التطبيق فيها.

أحد المجالات التي يمكن تحسين التطبيق فيها هو قدرته على التوسع. التطبيق الحالي غير مصمم للتعامل مع أعداد كبيرة من المستخدمين أو الصور. يمكن تحسين ذلك باستخدام بنية موزعة وتحسين الكود.

المجال الآخر حيث يمكن تحسين التطبيق هو الأمان. التطبيق الحالي ليس آمنًا بما يكفي لاستخدامه في بيئات الإنتاج. يمكن تحسين ذلك من خلال تنفيذ إجراءات الأمان مثل المصادقة والترخيص والتحكم في الوصول.

أخيراً، يمكن تحسين التطبيق من خلال دمجه مع أنظمة أخرى. سيسمح ذلك باستخدام التطبيق في نطاق أوسع من المهام. على سبيل المثال، يمكن دمج التطبيق مع نظام إدارة المحتوى للسماح للمستخدمين بتمييز الصور بالكلمات الرئيسية.

بشكل عام، يعد تطوير تصنيف الصور استنادًا إلى FastAPI وCLIP مجالًا واعدًا للبحث. إن الجمع بين هاتين التقنيتين لديه القدرة على بناء تطبيقات تصنيف صور قوية وقابلة للتطوير. ومع ذلك، لا تزال هناك بعض المجالات التي يمكن تحسين التطبيق فيها. من خلال التركيز على التحسين المستمر والقدرة على التكيف، يمكن للتطبيق أن يستمر في التطور ويصبح أداة قيمة لمهام تصنيف الصور.

المراجع

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied .to document recognition

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep .convolutional neural networks. Advances in neural information processing systems

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large- .scale image recognition

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. .(2015). Going deeper with convolutions

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image .recognition

Vaswani, A., et al. (2017). Attention is all you need. In Advances in neural information .processing systems

.Chen, T., Isola, P., & Zhu, J. Y. (2021). DeiT: Data-efficient image transformers

Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image .recognition at scale

.Radford, A., et al. (2020). Improving language understanding by generative pre-training

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. ."Language models are unsupervised multitask learners." OpenAI Blog 11.3 (2021): 8

Google AI. "Imagen: A Text-to-Image Diffusion Model with High Fidelity and Control." .(2022) Google AI Blog

Dosovitskiy, Alexey, Lucas Beyer, Georg Heigold, Lucas Mnih, Andrew Brock, Aäron .(2020) van den Oord, et al. "An image transformer." arXiv preprint arXiv:2010.11929

Radford, Alec, et al. "Learning transferable visual models from natural language supervision." Proceedings of the European Conference on Computer Vision (ECCV). .2020