المملكة العربية السعودية

وزارة التعليم
Ministry of Education

قررت وزارة التعليم تدريس
هذا الكتاب وطبعه على نفقتها

الذكاء الاصطناعي

التعليم الثانوي - نظام المسارات

السنة الثالثة

طبعة 1447-2025

ⓒ المركز الوطني للمناهج، ١٤٤٦ هـ

المركز الوطني للمناهج
الذكاء الاصطناعي - المرحلة الثانوية - نظام المسارات - السنة الثالثة.
المركز الوطني للمناهج. - الرياض، ١٤٤٦ هـ
٣٣٩ ص ؛ ٢١ x ٢٥,٥ سم

رقم الإيداع : ١٨٧٣٠ / ١٤٤٦
ردمك :١-٠١٧-٥١٤-٦٠٣-٩٧٨

مواد إثرائية وداعمة على "منصة عين الإثرائية"

ien.edu.sa

أعزاءنا المعلمين والمعلمات، والطلاب والطالبات، وأولياء الأمور، وكل مهتم بالتربية والتعليم:
يسعدنا تواصلكم؛ لتطوير الكتاب المدرسي، ومقترحاتكم محل اهتمامنا.

fb.ien.edu.sa

وزارة التعليم
Ministry of Education
2025 - 1447

الناشر: شركة تطوير للخدمات التعليمية

تم النشر بموجب اتفاقية خاصة بين شركة Binary Logic SA وشركة تطوير للخدمات التعليمية
(عقد رقم 2022/0003) للاستخدام في المملكة العربية السعودية

binarylogic

وزارة التعليم
Ministry of Education
2025 - 1447

مقدمة

إن تقدم الدول وتطورها يقاس بمدى قدرتها على الاستثمار في التعليم، ومدى استجابة نظامها التعليمي لمتطلبات العصر ومتغيراته. وحرصًا من وزارة التعليم على ديمومة تطوير أنظمتها التعليمية، واستجابة لرؤية المملكة العربية السعودية 2030 فقد بادرت الوزارة إلى اعتماد نظام «مسارات التعليم الثانوي» بهدف إحداث تغيير فاعل وشامل في المرحلة الثانوية.

إن نظام مسارات التعليم الثانوي يقدم أنموذجًا تعليميًا متميزًا وحديثًا للتعليم الثانوي بالمملكة العربية السعودية يسهم بكفاءة في:

- تعزيز قيم الانتماء لوطننا المملكة العربية السعودية، والولاء لقيادته الرشيدة حفظهم الله، انطلاقًا من عقيدة صافية مستندة على التعاليم الإسلامية السمحة.
- تعزيز قيم المواطنة من خلال التركيز عليها في المواد الدراسية والأنشطة، اتساقًا مع مطالب التنمية المستدامة، والخطط التنموية في المملكة العربية السعودية التي تؤكد على ترسيخ ثنائية القيم والهوية، والقائمة على تعاليم الإسلام والوسطية.
- تأهيل الطلبة بما يتوافق مع التخصصات المستقبلية في الجامعات والكليات أو المهن المطلوبة؛ لضمان اتساق مخرجات التعليم مع متطلبات سوق العمل.
- تمكين الطلبة من متابعة التعليم في المسار المفضل لديهم في مراحل مبكرة، وفق ميولهم وقدراتهم.
- تمكين الطلبة من الالتحاق بالتخصصات العلمية والإدارية النوعية المرتبطة بسوق العمل، ووظائف المستقبل.
- دمج الطلبة في بيئة تعليمية ممتعة ومحفزة داخل المدرسة قائمة على فلسفة بنائية، وممارسات تطبيقية ضمن مناخ تعليمي نشط.
- نقل الطلبة عبر رحلة تعليمية متكاملة بدءًا من المرحلة الابتدائية حتى نهاية المرحلة الثانوية، وتُسهِّل عملية انتقالهم إلى مرحلة ما بعد التعليم العام.
- تزويد الطلبة بالمهارات التقنية والشخصية التي تساعدهم على التعامل مع الحياة، والتجاوب مع متطلبات المرحلة.
- توسيع الفرص أمام الطلبة الخريجين عبر خيارات متنوعة إضافة إلى الجامعات مثل: الحصول على شهادات مهنية، والالتحاق بالكليات التطبيقية، والحصول على دبلومات وظيفية.

ويتكون نظام المسارات من تسعة فصول دراسية تُدرّس في ثلاث سنوات، تتضمن سنة أولى مشتركة يتلقى فيها الطلبة الدروس في مجالات علمية وإنسانية متنوعة، تليها سنتان تخصصيتان، يُسكّن الطلبة بها في مسار عام وأربعة مسارات تخصصية تتسق مع ميولهم وقدراتهم، وهي: المسار الشرعي، مسار إدارة الأعمال، مسار علوم الحاسب والهندسة، مسار الصحة والحياة، وهو ما يجعل هذا النظام هو الأفضل للطلبة من حيث:

- وجود مواد دراسية جديدة تتوافق مع متطلبات الثورة الصناعية الرابعة والخطط التنموية، ورؤية المملكة 2030، تهدف لتنمية مهارات التفكير العليا وحل المشكلات، والمهارات البحثية.
- برامج المجال الاختياري التي تتسق مع احتياجات سوق العمل وميول الطلبة، حيث يُمكّن الطلبة من الالتحاق بمجال اختياري محدد وفق مصفوفة مهارات وظيفية محددة.
- مقياس ميول يضمن تحقيق كفاءة الطلبة وفاعليتهم، ويساعدهم في تحديد اتجاهاتهم وميولهم، وكشف مكامن القوة لديهم، مما يعزز من فرص نجاحهم في المستقبل.
- العمل التطوعي المصمم للطلبة خصيصًا بما يتسق مع فلسفة النشاط في المدارس، ويعد أحد متطلبات التخرج؛ مما يساعد على تعزيز القيم الإنسانية، وبناء المجتمع وتنميته وتماسكه.
- التجسير الذي يمكن الطلبة من الانتقال من مسار إلى آخر وفق آليات محددة.
- حصص الإتقان التي يتم من خلالها تطوير المهارات وتحسين المستوى التحصيلي، من خلال تقديم حصص إتقان إثرائية وعلاجية.

- خيارات التعليم المدمج، والتعلم عن بعد، والذي بُني في نظام المسارات على أسس من المرونة، والملاءمة والتفاعل والفعالية.
- مشروع التخرج الذي يساعد الطلبة على دمج الخبرات النظرية مع الممارسات التطبيقية.
- شهادات مهنية ومهارية تمنح للطلبة بعد إنجازهم مهامَّ محددة، واختبارات معينة بالشراكة مع جهات تخصصية.

وبالتالي فإن مسار علوم الحاسب والهندسة كأحد المسارات المستحدثة في المرحلة الثانوية يسهم في تحقيق أفضل الممارسات عبر الاستثمار في رأس المال البشري، وتحويل الطالب إلى فرد مشارك ومنتج للعلوم والمعارف، مع إكسابه المهارات والخبرات اللازمة لاستكمال دراسته في تخصصات تتناسب مع ميوله وقدراته أو الالتحاق بسوق العمل.

وتعد مادة الذكاء الاصطناعي أحد المواد الرئيسة في مسار علوم الحاسب والهندسة، حيث تسهم في توضيح مفاهيم الذكاء الاصطناعي والتقنيات المرتبطة بها بما يساعد على توظيف هذه التقنيات في عدة مجالات حياتية مثل المدن الذكية والتعليم والزراعة والطب وغيرها من المجالات الاقتصادية المتنوعة. وتهدف المادة إلى تعريف الطالب بأهمية الذكاء الاصطناعي ودوره في الجيل الرابع من الصناعة. وكذلك تركز على اللبنات الأساسية لتقنيات الذكاء الاصطناعي، ثم تتعرَّض بشكل تفصيلي للتطبيقات المتقدمة التي تتعلق بالأنظمة القائمة على القواعد وأنظمة معالجة اللغات الطبيعية. كما تشتمل هذه المادة على مشاريع وتمارين تطبيقية لما يتعلمه الطالب؛ لحل مشاكل واقعية تحاكي مستوياته المعرفية، بتوجيه وإشراف من المعلم.

ويتميز كتاب الذكاء الاصطناعي بأساليب حديثة، تتوافر فيه عناصر الجذب والتشويق، والتي تجعل الطلبة يقبلون على تعلمه والتفاعل معه، من خلال ما يقدمه من تدريبات وأنشطة متنوعة، كما يؤكد هذا الكتاب على جوانب مهمة في تعليم الذكاء الاصطناعي وتعلمه، تتمثل في:

- الترابط الوثيق بين المحتويات والمواقف والمشكلات الحياتية.
- تنوع طرائق عرض المحتوى بصورة جذابة ومشوقة.
- إبراز دور المتعلم في عمليات التعليم والتعلم.
- الاهتمام بترابط محتوياته مما يجعل منه كلًّا متكاملًا.
- الاهتمام بتوظيف التقنيات المناسبة في المواقف المختلفة.
- الاهتمام بتوظيف أساليب متنوعة في تقويم الطلبة بما يتناسب مع الفروق الفردية بينهم.

ولمواكبة التطورات العالمية في هذا المجال، فإن كتاب مادة الذكاء الاصطناعي سوف يوفر للمعلم مجموعة متكاملة من المواد التعليمية المتنوعة التي تراعي الفروق الفردية بين الطلبة، بالإضافة إلى البرمجيات والمواقع التعليمية، التي توفر للطلبة فرصة توظيف التقنيات الحديثة والتواصل المبني على الممارسة؛ مما يؤكد دوره في عملية التعليم والتعلم.

ونحن إذ نقدم هذا الكتاب لأعزائنا الطلبة، نأمل أن يستحوذ على اهتمامهم، ويُلبي متطلباتهم، ويجعل تعلّمهم لهذه المادة أكثر متعة وفائدة.

والله ولي التوفيق

بسم الله الرحمن الرحيم

وزارة التعليم
Ministry of Education
2025 - 1447

الفهرس

الجزء الأول

الجزء الثاني

الجزء الثاني

الوحدة الرابعة
التعرّف على الصور

الوحدة الخامسة
خوارزميات التحسين واتخاذ القرار

الوحدة السادسة
الذكاء الاصطناعي والمجتمع

وزارة التعليم
Ministry of Education
2025 - 1447

وزارة التعليم
Ministry of Education
2025 - 1447

4. التعرّف على الصور

سيتعرّف الطالب في هذه الوحدة على التعلُّم الموجَّه وغير الموجَّه، وكيفية توظيفهما للتعرّف على الصور (Image Recognition) عن طريق إنشاء نموذج وتدريبه؛ ليصبح قادرًا على تصنيف صور لرؤوس الحيوانات أو تجميعها. وسيتعرّف أيضًا على توليد الصور (Image Generation) وكيفية تغييرها، أو إكمال الأجزاء الناقصة فيها مع الحفاظ على واقعيتها.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

> يُعالج الصور معالجة أولية ويستخلص خصائصها.

> يُدرِّب نموذج تعلُّم موجَّه خاص بتصنيف الصور.

> يُعرِّف هيكل الشبكة العصبية.

> يُدرِّب نموذج تعلُّم غير موجَّه خاص بتجميع الصور.

> يُولِّد صورًا بناءً على توجيه نصّي.

> يُكمل الأجزاء الناقصة في صورة مُعطاة بطريقة واقعية.

الأدوات

> مفكّرة جوبيتر (Jupyter Notebook)

> قوقل كولاب (Google Colab)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

التعلُّم الموجَّه لتحليل الصور

التعلُّم الموجَّه في رؤية الحاسب Supervised Learning for Computer Vision

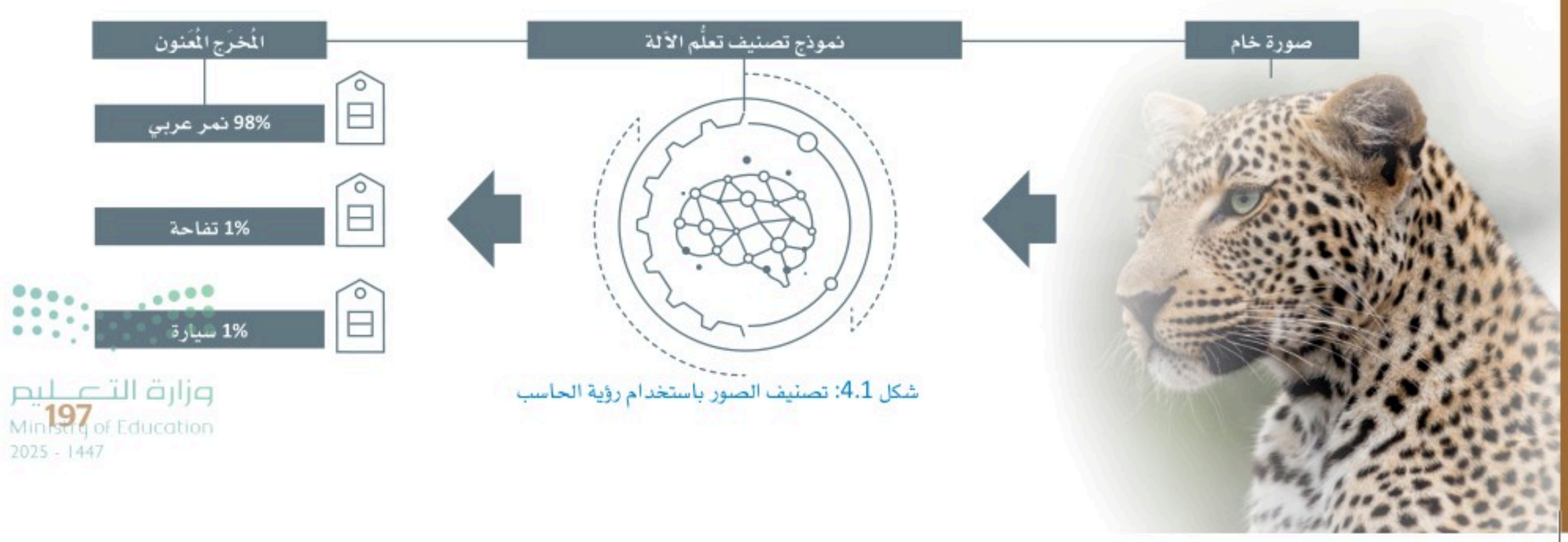
تُعدُّ **رؤية الحاسب** (Computer Vision) مجالًا فرعيًا من مجالات الذكاء الاصطناعي، والذي يُركّز على تعليم أجهزة الحاسب طريقة تفسير العَالَم المرئي وفهمه، ويتضمن استخدام الصور الرقمية ومقاطع الفيديو؛ لتدريب الآلات على التعرّف على المعلومات المرئية وتحليلها مثل: الأشياء والأشخاص والمَشاهد. ويتمثّل الهدف النهائي الذي تسعى رؤية الحاسب إلى تحقيقه في تمكين الآلات من "رؤية" العَالَم كما يراه البشر، واستخدام هذه المعلومات؛ لاتخاذ قرارات، أو للقيام بإجراءات.

هناك مجموعة كبيرة من التطبيقات التي تُستخدم فيها رؤية الحاسب، مثل:

- التصوير الطبي: يمكن أن تساعد رؤية الحاسب الأطباء والمختصين في الرعاية الصحية على تشخيص الأمراض من خلال تحليل الصور الطبية مثل: الأشعّة السينية، والتصوير بالرنين المغناطيسي، والأشعّة المقطعية.
- المركبات ذاتية القيادة: تستخدِم السيارات ذاتية القيادة والطائرات المُسيَّرة رؤية الحاسب للتعرف على إشارات المرور وأشكال الطرق العامة وطرق المشاة والعقبات في الطريق والجو، ولِتمكينها من التنقل بأمان وكفاءة.
- ضبط الجودة: تُستخدم رؤية الحاسب لفحص المنتجات وتحديد عيوب التصنيع، وذلك في مُختلف أنواع الصّناعات، مثل: صناعة السيارات والإلكترونيات والمنسوجات.
- الروبوتية: تُستخدم رؤية الحاسب لمساعدة الروبوتات على التنقل والتفاعل مع بيئتها عن طريق التعرّف على الأشياء والتعامل معها.

يُعدُّ التعلُّم الموجَّه وغير الموجَّه نوعين رئيسين من تعلُّم الآلة يُستخدمان بطريقة شائعة في تطبيقات رؤية الحاسب، ويتضمن كلا النوعين خوارزميات تدريب على مجموعات كبيرة من الصور أو مقاطع الفيديو؛ لكي تتمكن الآلات من التعرّف على المعلومات المرئية وتفسيرها. سبق أن تعرّفت على التعلُّم الموجَّه وغير الموجَّه في الدرسين الأول والثاني من الوحدة الثالثة، وكلاهما طُبِّق في **معالجة اللغات الطبيعية** (NLP) و**توليد اللغات الطبيعية** (NLG)، وسيتم تطبيقهما في هذا الدرس على تحليل الصور.

يتضمَّن التعلُّم غير الموجَّه خوارزميات تدريب على مجموعات بيانات غير مُعَنوَنة - أي لا توجد فيها عناوين أو فئات صريحة -، ثم تتعلّم الخوارزمية تحديد الأنماط المتشابهة في البيانات دون أن تكون لديها أي معرفة مسبقة بالعناوين. على سبيل المثال: يمكن استخدام خوارزمية التعلُّم غير الموجَّه لتجميع الصور المتشابهة معًا بناءً على السمات المشتركة بينها مثل: اللون أو **النقش** (Texture) أو الشكل. وسيتم توضيح التعلُّم غير الموجَّه بالتفصيل في الدرس الثاني.



شكل 4.1: تصنيف الصور باستخدام رؤية الحاسب

في المقابل، يتضمن التعلُّم الموجَّه تدريب الخوارزميات على مجموعات بيانات مُعَنوَنة؛ حيث يُخصص عنوان أو فئة معيّنة لكل صورة أو مقطع فيديو، ثم تقوم الخوارزمية بعد ذلك بالتعرّف على أنماط وخصائص كل عنوان؛ لتتمكن من تصنيف الصور أو مقاطع الفيديو الجديدة بدقة. فعلى سبيل المثال: قد تُدرَّب خوارزمية التعلُّم الموجَّه على التعرّف على سلالات مُختلفة من القطط بناءً على الصور المُعَنوَنة لكل سلالة (انظر الشكل 4.1)، وسيتم التركيز في هذا الدرس على التعلُّم الموجَّه.

تشتمل عملية التعلُّم الموجَّه عادة على أربع خطوات رئيسة وهي: جمع البيانات، وعَنوَنتها، والتدريب عليها، ثم الاختبار. أثناء جمع البيانات ووضع المسميات، تُجمع الصور أو مقاطع الفيديو وتنظّم في مجموعة بيانات، ثم تُعنون كل صورة أو مقطع فيديو بعنوان صنف أو فئة، مثل: eagle (النسر) أو cat (القطّة).

وتستخدِم خوارزمية تعلُّم الآلة أثناء مرحلة التدريب مجموعة البيانات المُعَنوَنة "لتتعلّم" الأنماط والسمات المرتبطة بكل صنف أو فئة، وكلما زادت بيانات التدريب التي تُقدم للخوارزمية أصبحت أكثر دقة في التعرّف على الفئات المُختلفة في مجموعة البيانات، وبالتالي يتحسَّن أداؤها.

وبمجرد أن يُدرَّب النموذج، يتم اختباره على مجموعة منفصلة غير التي تم التدريب عليها من الصور أو مقاطع الفيديو؛ لتقييم أدائه، وتختلف مجموعة الاختبار عن مجموعة التدريب؛ للتأكد من قدرة النموذج على التعميم على البيانات الجديدة. فعلى سبيل المثال: تحتوي البيانات الخاصة بـ cat (القطّة) على خصائص مثل: الوزن واللون والسلالة وما إلى ذلك، وتُقيّم دقة النموذج بناءً على مدى كفاءة أدائه في مجموعة الاختبار.

تشبه العملية السابقة إلى حد كبير العملية المُتّبعة في مهام التعلُّم الموجَّه لأنواع مُختلفة من البيانات مثل النصوص، ولكن البيانات المرئية عادة ما تُعدُّ أكثر صعوبة في التعامل معها من النصّ لأسباب متعددة كما هو موضَّح في الجدول 4.1.

جدول 4.1: تحديَات تصنيف البيانات المرئية

البيانات المرئية عالية الأبعاد	تحتوي الصور على كمية كبيرة من البيانات، مما يجعل معالجتها وتحليلها أكثر صعوبة من البيانات النصيَّة، ففي حين أن العناصر الأساسية للمستند النصيّ هي الكلمات، فإن عناصر الصورة هي وحدات البكسل، وسترى في هذا الفصل أنّ الصورة يمكن أن تتكون من آلاف وحدات البكسل، حتّى الصّغيرة منها.
البيانات المرئية تحتوي على تفاصيل كثيرة ومتنوّعة للغاية	يمكن أن تتأثر الصور بالتفاصيل الكثيرة، والإضاءة، والتشويش، وعوامل أخرى تجعل تصنيفها بدقة عملية صعبة. بالإضافة إلى ذلك، هناك مجموعة واسعة من البيانات المرئية المتنوّعة ذات العديد من العناصر، والمشاهد، والسياقات التي يصعب تصنيفها بدقة.
البيانات المرئية لا تتبع هيكلة محددة	يتبع النصّ بُنية لغوية وقواعد نحويّة عامة، بينما لا تخضع البيانات المرئية لقواعد ثابتة؛ مما يجعل عملية التحليل أكثر تعقيدًا وصعوبة وتكلفة.

نتيجة لهذه التعقيدات يتطلب التصنيف الفعّال للبيانات المرئية أساليب متخصّصة، وتتناول هذه الوحدة التقنيات التي تستخدِم الخصائص الهندسية واللونية للصور، بالإضافة إلى أساليب تعلُّم الآلة المُتقدمة القائمة على الشبكات العصبية.

يوضِّح الدرس الأول كيفية استخدام لغة **البايثون** (Python) في:

- تحميل مجموعة بيانات من الصور المُعَنوَنة.
- تحويل الصور إلى صيغة رقمية يمكن أن تستخدِمها خوارزميات رؤية الحاسب.
- تقسيم البيانات الرقمية إلى مجموعات بيانات للتدريب، ومجموعات بيانات للاختبار.

- تحليل البيانات؛ لاستخراج أنماط وخصائص مفيدة.
- استخدام البيانات المستخلصة؛ لتدريب نماذج التصنيف التي يمكن استخدامها للتنبؤ بعناوين الصور الجديدة.

تحتوي مجموعة البيانات التي ستستخدمها على ألف وسبعمئة وثلاثين (1,730) صورة لوجوه ستّة عشر نوعًا مُختلفًا من الحيوانات، وبالتالي فهي مجموعة مثالية للتعلُّم الموجَّه لتطبيق التقنيات المذكورة سابقًا.

تحميل الصور ومعالجتها الأولية Loading and Preprocessing Images

يستورد المقطع البرمجي التالي مجموعة من المكتبات التي تُستخدم لتحميل الصور من مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) وتحويلها إلى صيغة رقمية:

```
%%capture
import matplotlib.pyplot as plt # used for visualization
from os import listdir # used to list the contents of a directory

!pip install scikit-image # used for image manipulation
from skimage.io import imread # used to read a raw image file (e.g. png or jpg)
from skimage.transform import resize # used to resize images

# used to convert an image to the "unsigned byte" format
from skimage import img_as_ubyte
```

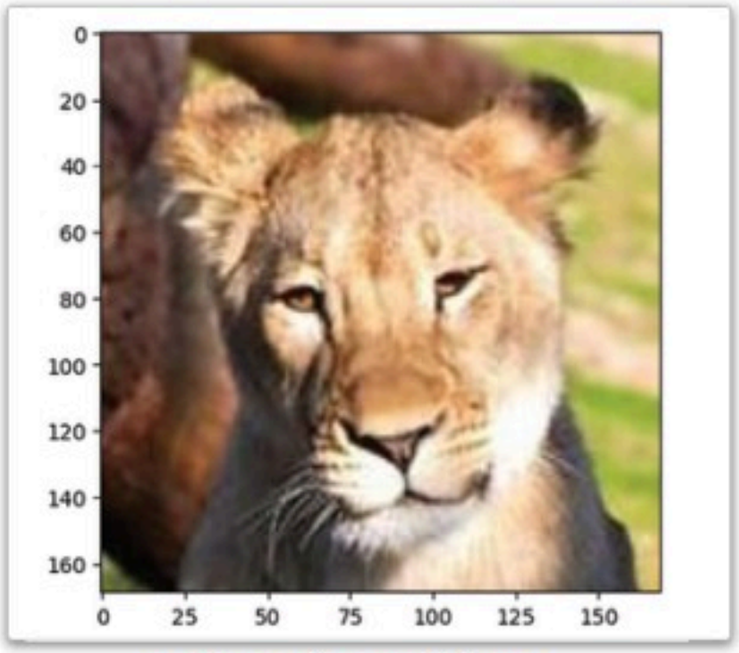
تتطلب خوارزميات التعلُّم الموجَّه أن تكون كل الصور في مجموعة البيانات لها الأبعاد نفسها، ولذلك فإن المقطع البرمجي التالي يقرأ الصور من input_folder (مجلد_ المُدخَلات) ويُغيِّر حجم كل منها بحيث تكون لها أبعاد الطول والعرض نفسها:

```
def resize_images(input_folder:str,
                  width:int,
                  height:int
                 ):

    labels = [] # a list with the label for each image
    resized_images = [] # a list of resized images in np array format
    filenames = [] # a list of the original image file names

    for subfolder in listdir(input_folder): # for each sub folder

        print(subfolder)
        path = input_folder + '/' + subfolder

        for file in listdir(path): # for each image file in this subfolder

            image = imread(path + '/' + file) # reads the image
            resized = img_as_ubyte(resize(image, (width, height))) # resizes the image
            labels.append(subfolder[:-4])  # uses subfolder name without "Head" suffix
            resized_images.append(resized) # stores the resized image
            filenames.append(file)         # stores the filename of this image

    return resized_images, labels, filenames
```

```
resized_images, labels, filenames = resize_images("AnimalFace/Image",
width=100, height=100) # retrieves the images with their labels and resizes them to 100 x 100
```

```
BearHead          EagleHead         PigeonHead
CatHead           ElephantHead      RabbitHead
ChickenHead       LionHead          SheepHead
CowHead           MonkeyHead        TigerHead
DeerHead          Natural           WolfHead
DuckHead          PandaHead
```

هذه هي أسماء المجلدات، وبدون المقطع اللاحق Head (رأس)، تُمثِّل هذه الأسماء عناوين للصور الموجودة داخلها.



شكل 4.2: صورة رأس أسد أصلية

تُنشئ دالة imread() تنسيق ألوان للصورة يُعرف بـ "RGB"، ويُستخدم هذا التنسيق على نطاق واسع؛ لأنه يسمح بتمثيل مجموعة واسعة من الألوان. وفي نظام الألوان RGB، تعني الأحرف R و G و B احتواء التنسيق على ثلاثة مكونات رئيسة للألوان، وهي اللون **الأحمر** (R = Red) واللون **الأخضر** (G = Green) واللون **الأزرق** (B = Blue). يُمثَّل كل بكسل بثلاث قنوات وهي: (قناة للون الأحمر، وقناة للون الأخضر، وقناة للون الأزرق)، كل قناة تحوي **ثمانية بت** (8-bit)، ويمكن أن يأخذ البكسل قيمة بين: 0 و255. يُعرف التنسيق 0-255 أيضًا باسم تنسيق **البايت بدون إشارة** (Unsigned Byte).

يتيح الجمع بين هذه القنوات الثلاث تمثيل مجموعة واسعة من الألوان في البكسل، على سبيل المثال: البكسل ذو القيمة (0، 0، 255) سيكون لونه أحمر بالكامل، والبكسل ذو القيمة (0، 255، 0) سيكون لونه أخضر بالكامل، والبكسل ذو القيمة (255، 0، 0) سيكون لونه أزرق بالكامل، والبكسل ذو القيمة (255، 255، 255) سيكون لونه أبيض، والبكسل ذو القيمة (0، 0، 0) سيكون لونه أسود.

في نظام الألوان RGB، تُرتب قيم البكسل في شبكة ثنائية الأبعاد، تحتوي على صفوف وأعمدة تُمثِّل إحداثيات x وy للبكسلات في الصورة، ويُشار إلى هذه الشبكة باسم **مصفوفة الصور** (Image Matrix). على سبيل المثال، ضع في اعتبارك الصورة الموجودة في الشكل 4.2 والمقطع البرمجي المرتبط بها أدناه:

```
# reads an image file, stores it in a variabe and
# shows it to the user in a window
image = imread('AnimalFace/Image/LionHead/lioni78.jpg')
plt.imshow(image)
image.shape
```

```
(169, 169, 3)
```

تكشف طباعة شكل الصورة عن مصفوفة 169 × 169، بإجمالي: ثمانية وعشرين ألفًا وخمسمئة وواحد وستين (28,561) بكسل، ويمثِّل الرقم 3 في العمود الثالث القنوات الثلاث (أحمر / أخضر / أزرق) لنظام الألوان RGB.

على سبيل المثال، سيطبع المقطع البرمجي التالي قيمة الألوان للبكسل الأول من هذه الصورة:

```
# the pixel at the first column of the first row
print(image[0][0])
```

```
[102  68  66]
```

يؤدي تغيير الحجم إلى تحويل الصور من تنسيق RGB إلى تنسيق **مُستنِد على عدد حقيقي** (Float-Based):

```
resized = resize(image, (100, 100))
print(resized.shape)
print(resized[0][0])
```

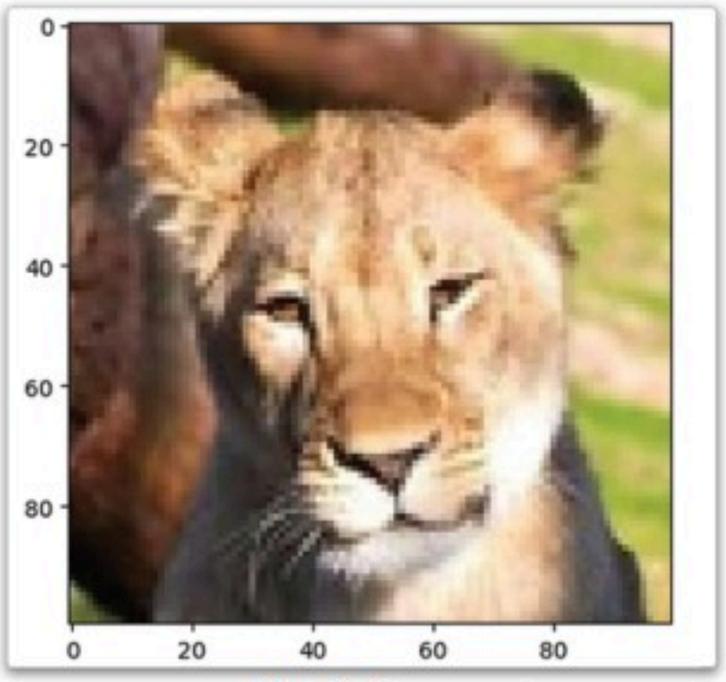
```
(100, 100, 3)
[0.40857161 0.27523827 0.26739514]
```

على الرغم من أن الصورة قد غُيّر حجمها إلى مصفوفة ذات أبعاد 100 × 100، فإن قيم القنوات الثلاث RGB لكل بكسل تم **تسويتها** (Normalized) لتكون ذات قيمة بين 0 و1، ويمكن إعادة تحويلها مرة أخرى إلى تنسيق البايت بدون إشارة من خلال المقطع البرمجي التالي:

```
resized = img_as_ubyte(resized)
print(resized.shape)
print(resized[0][0])
print(image[0][0])
```

```
(100, 100, 3)
[104  70  68]
[102  68  66]
```

تختلف قيم الألوان RGB للبكسل الذي غُيّر حجمه اختلافًا بسيطًا عن القيم الموجودة في الصورة الأصلية، وهو من الآثار الشائعة الناتجة عن تغيير الحجم، وعند طباعة الصورة التي غُيّر حجمها، يتبين أنها أقل وضوحًا، كما يظهر في الشكل 4.3، وهذا ناتج عن ضغط المصفوفة 169 × 169 إلى تنسيق 100 × 100.



شكل 4.3: صورة رأس أسد غُيّر حجمها

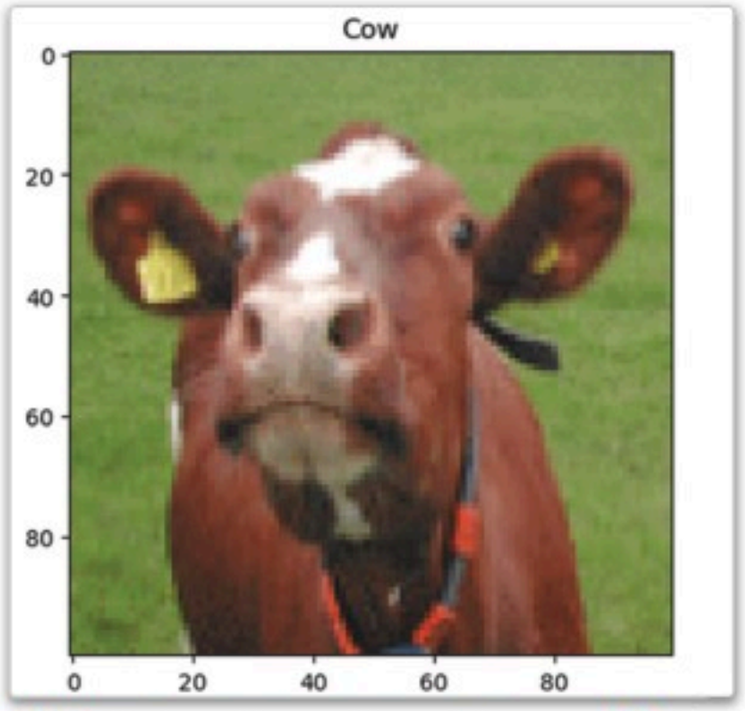
```
# displays the resized image
plt.imshow(resized);
```

قبل بدء التدريب على خوارزميات التعلُّم الموجَّه، من الجيد التحقق مما إذا كانت أي صورة من الصور الموجودة في مجموعة البيانات غير مطابقة للتنسيق (3، 100، 100).

```
violations = [index for index in range(len(resized_images)) if
resized_images[index].shape != (100,100,3)]

violations
```

```
[455, 1587]
```

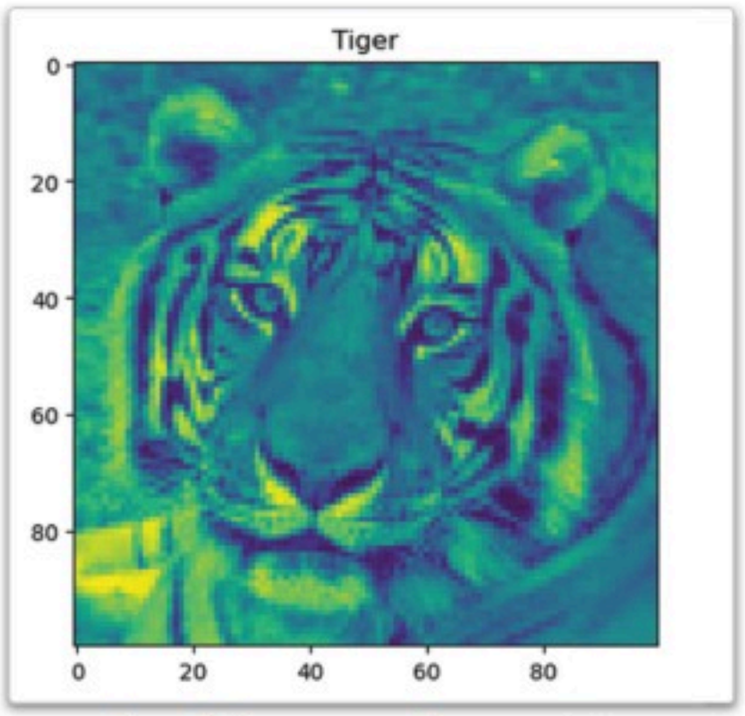
يكشف هذا المقطع البرمجي عن وجود صورتين غير مطابقتين لتلك الصيغة، وهذا غير متوقع؛ لأن دالة resize_image() تمَّ تطبيقها على جميع الصور الموجودة في مجموعة البيانات. يقوم المقطعان البرمجيان التاليان بطباعة هاتين الصورتين، بالإضافة إلى أبعادهما واسمي ملفيهما:



شكل 4.4: صورة بالأحمر والأخضر والأزرق وألفا (RGBA)

```
pos1 = violations[0]
pos2 = violations[1]

print(filenames[pos1])
print(resized_images[pos1].shape)
plt.imshow(resized_images[pos1])
plt.title(labels[pos1])
```

```
cow1.gif
(100, 100, 4)
```

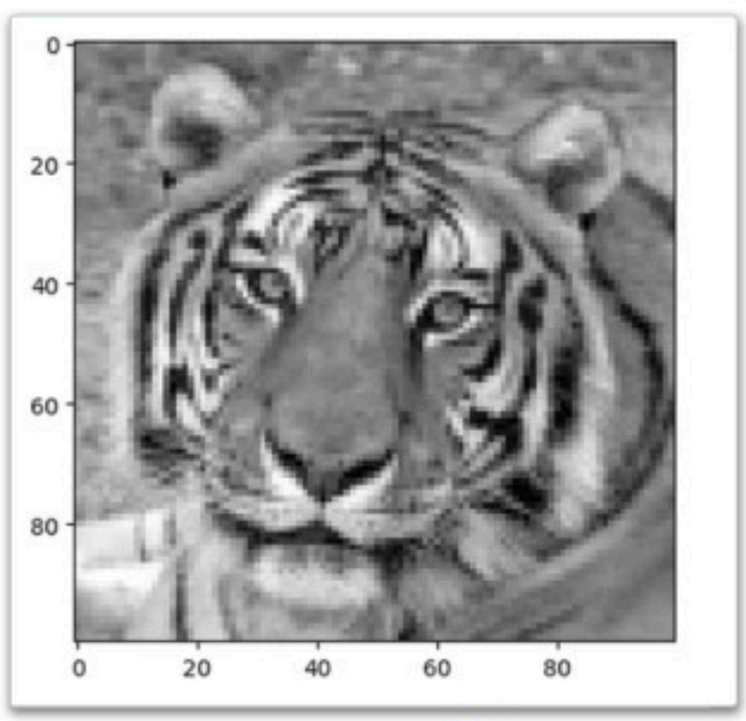


شكل 4.5: صورة تبيّن التنسيق المضلِّل أصفر / أزرق

```
print(filenames[pos2]);
print(resized_images[pos2].shape);
plt.imshow(resized_images[pos2]);
plt.title(labels[pos2]);
```

```
tiger0000000168.jpg
(100, 100)
```

الصورة الأولى: لها شكل ذو أبعاد (4، 100، 100)، ويدلُّ الرقم 4 أنها بتنسيق RGBA بدلًا من تنسيق RGB، وهذا التنسيق يحتوي على قناة إضافية رابعة تدعى قناة **ألفا** (Alpha) التي تُمثِّل شفافية كل بكسل. على سبيل المثال:



شكل 4.6: صورة بتدرج رمادي

```
# prints the first pixel of the RGBA image
# a value of 255 reveals that the pixel is  not transparent
at all.
resized_images[pos1][0][0]
```

```
array([135, 150, 84, 255], dtype=uint8)
```

الصورة الثانية: لها شكل ذو أبعاد (100، 100)، ويدلُّ غياب البُعد الثالث على أن الصورة بتنسيق **تدرج رمادي** (Grayscale) وليست بتنسيق RGB، والتنسيق **المضلِّل أصفر / أزرق** (Misleading Yellow/Blue) المبين سابقًا يعود إلى خريطة لونية تُطبِّقها الدالة imshow بشكل افتراضي على الصور ذات التدرج الرمادي، ويمكن إلغاؤه كما يلي:

```
plt.imshow(resized_images[pos2], cmap = 'gray')
```

صور التدرج الرمادي لها قناة واحدة فقط (بدلًا من قنوات RGB الثلاث)، وقيمة كل بكسل عبارة عن رقم واحد يتراوح من 0 إلى 255، حيث تُمثِّل قيمة البكسل 0 اللون الأسود، بينما تُمثِّل قيمة البكسل 255 اللون الأبيض. على سبيل المثال:

```
resized_images[pos2][0][0]
```

```
100
```

وكاختبار إضافي لجودة البيانات، يقوم المقطع البرمجي التالي بحساب تكرار عنوان كل صورة حيوان في مجموعة البيانات:

```
# used to count the frequency of each element in a list.
from collections import Counter

label_cnt = Counter(labels)
label_cnt
```

```
Counter({'Bear': 101,
         'Cat': 160,
         'Chicken': 100,
         'Cow': 104,
         'Deer': 103,
         'Duck': 103,
         'Eagle': 101,
         'Elephant': 100,
         'Lion': 102,
         'Monkey': 100,
         'Nat': 8,
         'Panda': 119,
         'Pigeon': 115,
         'Rabbit': 100,
         'Sheep': 100,
         'Tiger': 114,
         'Wolf': 100})
```

هنا يمكنك رؤية القيمة المتطرفة وهي فئة Nat (Nature أو الطبيعة)، وتحتوي على ثمانية عناصر فقط مقارنة بالفئات الأخرى.

تحتوي مجموعة البيانات على صور حيوانات وصور أخرى من الطبيعة؛ وذلك بهدف التعرّف على الصور التي تشذ عن صور الحيوانات. يكشف Counter (العداد) عن فئة صغيرة جدًا عنوانها Nat (الطبيعة)، وتحتوي على ثماني صور فقط، وعندما تقوم بكشف سريع يتضح لك أن هذه الفئة ذات **قيم متطرفة** (Outlier) تحتوي على صور لمناظر طبيعية ولا يوجد بها أي وجه لأي حيوان.

يقوم المقطع البرمجي التالي بإزالة صورة RGBA وصورة التدرج الرمادي، وكذلك كل الصور التي تنتمي لفئة Nat (الطبيعة) من قوائم أسماء الملفات، والعناوين، والصور التي غُيّر حجمها.

```
N = len(labels)

resized_images = [resized_images[i] for i in range(N) if i not in violations
and labels[i] != "Nat"]
filenames = [filenames[i] for i in range(N) if i not in violations and
labels[i] != "Nat"]
labels = [labels[i] for i in range(N) if i not in violations and labels[i] !=
"Nat"]
```

تتمثّل الخطوة التالية في تحويل resized_images (الصور _ المُعدَّل حجمها) وقوائم العناوين إلى مصفوفات Numpy (نمباي) حسب ما تتوقعه العديد من خوارزميات رؤية الحاسب. يستخدِم المقطع البرمجي التالي أيضًا المتغيِّرات (X، Y) التي تُستخدم في العادة لتمثيل البيانات والعناوين على التوالي في مهام التعلُّم الموجَّه:

```
import numpy as np
X = np.array(resized_images)
Y = np.array(labels)

X.shape
```

```
(1720, 100, 100, 3)
```

يوضِّح شكل مجموعة بيانات X النهائية اشتمالها على ألف وسبعمئة وعشرين صورة بتنسيق RGB، بناءً على عدد القنوات، وجميعها بأبعاد 100 × 100 (أي عشرة آلاف بكسل). أخيرًا، يمكن استخدام دالة train_test_split() من مكتبة sklearn لتقسيم مجموعة البيانات إلى مجموعة تدريب ومجموعة اختبار.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size = 0.20,   # uses 20% of the data for testing
    shuffle = True,     # to randomly shuffle the data.
    random_state = 42,  # to ensure that data is always shuffled in the same way
)
```

نظرًا لأن مجلدات صور الحيوانات حُمّلت مجلّدًا تلو الآخر، فإن الصور من كل مجلد جُمعت معًا في القوائم السابقة، وقد يؤدي ذلك إلى تضليل العديد من الخوارزميات، خاصة في مجال رؤية الحاسب، وضبط shuffle=True (تفعيل إعادة الترتيب) في المقطع البرمجي السابق يحل هذه المشكلة، وبوجه عام، من الجيد إعادة ترتيب البيانات عشوائيًا قبل إجراء أي تحليل.

التنبؤ بدون هندسة الخصائص Prediction without Feature Engineering

على الرغم من أن الخطوات المتَّبعة في القسم السابق قد حوَّلت البيانات إلى تنسيق رقمي، إلا أنه ليس بالتنسيق القياسي أحادي البُعد الذي تتوقعه العديد من خوارزميات تعلُّم الآلة. على سبيل المثال، وصفتْ الوحدة الثالثة كيف يجب تحويل كل مستند إلى متَّجَه رقمي أحادي البُعد قبل استخدام البيانات في تدريب نماذج تعلُّم الآلة واختبارها، بينما تحتوي كل نقطة بيانات في مجموعة البيانات المرئية هنا على تنسيق ثلاثي الأبعاد.

```
X_train[0].shape
```

```
(100, 100, 3)
```

لذلك يمكن استخدام المقطع البرمجي التالي **لتسطيح** (Flatten) كل صورة في متَّجَه أحادي البُعد، فكل صورة الآن ممثَّلة كمتَّجَه رقمي مسطح قيمته 3 × 100 × 100 = 30,000 قيمة.

```
X_train_flat = np.array([img.flatten() for img in X_train])
X_test_flat = np.array([img.flatten() for img in X_test])
X_train_flat[0].shape
```

```
(30000,)
```

يمكن استخدام هذا التنسيق المسطح مع أي خوارزمية تصنيف قياسية دون بذل أي جهد إضافي لهندسة خصائص تنبؤية أخرى، وسيوضِّح القسم التالي مثالًا على هندسة الخصائص لبيانات صورة، ويستخدِم المقطع البرمجي التالي مُصنِّف **بايز الساذج** (Naive Bayes – NB) الذي استُخدم أيضًا لتصنيف البيانات النصيَّة في الوحدة الثالثة:

```
from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier

model_MNB = MultinomialNB()
model_MNB.fit(X_train_flat,y_train) # fits the model on the flat training data
```

```
MultinomialNB()
```

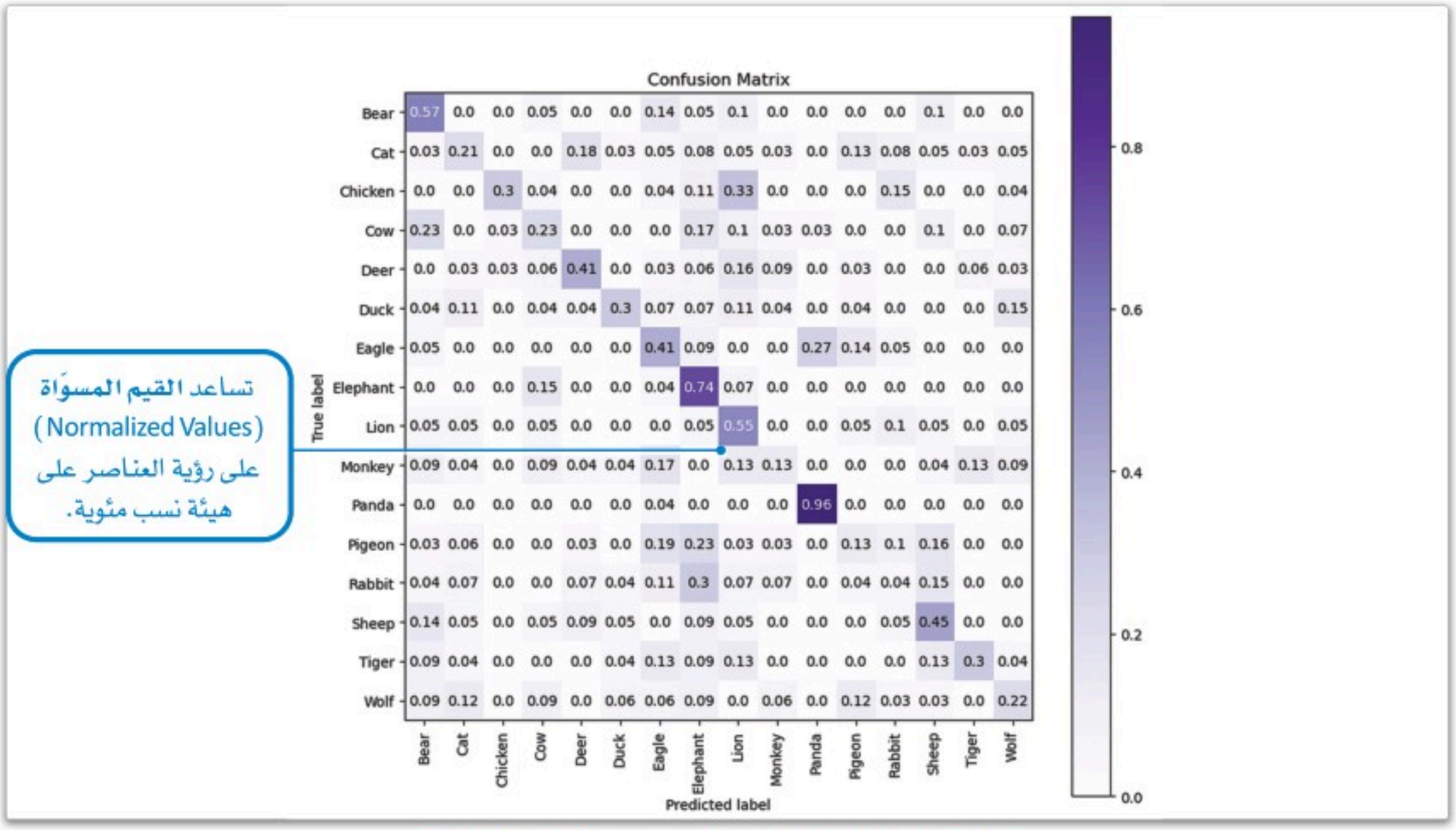
```
from sklearn.metrics import accuracy_score # used to measure the accuracy

pred = model_MNB.predict(X_test_flat) # gets the predictions for the flat test set
accuracy_score(y_test,pred)
```

```
0.36046511627906974
```

يعرض المقطع البرمجي التالي **مصفوفة الدقة** (Confusion Matrix) الخاصة بالنتائج لإعطاء رؤية إضافية:

```
%%capture
!pip install scikit-plot
import scikitplot
```

```
scikitplot.metrics.plot_confusion_matrix(y_test, # actual labels
                                         pred, # predicted labels
                                         title = "Confusion Matrix",
                                         cmap = "Purples",
                                         figsize = (10,10),
                                         x_tick_rotation = 90,
                                         normalize = True # to print percentages
                                         )
```



تساعد القيم المسوّاة (Normalized Values) على رؤية العناصر على هيئة نسب مئوية.

شكل 4.7: مصفوفة الدقة الخاصة بأداء خوارزمية MultinomialNB

تُحقق خوارزمية **بايز الساذجة متعددة الحدود** (MultinomialNB) دقة تقارب 30%، وعلى الرغم من أن هذه النسبة قد تبدو قليلة، إلا أن عليك النظر إليها في ضوء أن مجموعة البيانات تتضمن عشرين عنوانًا مُختلفا. ويعني ذلك أنّه لو افتُرض وجود مجموعة بيانات متوازنة نسبيًا يُغطي فيها كل عنوان 1/20 من البيانات، فإن المُصنِّف العشوائي الذي يُخصص عنوانًا لكل نقطة اختبار بشكل عشوائي، سيحقق دقة تبلغ حوالي 5%، ولذلك ستكون الدقة بنسبة 30% أعلى بست مرات من التخمين العشوائي.

ومع ذلك، كما هو موضَّح في الأقسام التالية، يمكن تحسين هذه الدقة تحسينًا ملحوظًا، وتؤكد مصفوفة الدقة أيضًا أن هناك مجالًا للتحسين. على سبيل المثال، غالبًا ما يخطئ نموذج بايز الساذج ويصنِّف Pigeons (الحَمَام) على أنها Eagles (نسور) أو يصنِّف Wolves (الذئاب) على أنها Cats (قطط). تكمن أسهل طريقة لمحاولة تحسين النتائج في ترك البيانات كما هي، والتجريب باستخدام مُصنِّفات مُختلفة، ومن النماذج التي ثبت أنها تعمل بشكل جيد مع بيانات الصورة المحوَّلة إلى متَّجَهَات نموذج: **مُصنِّف الانحدار التَّدرجي العشوائي** (SGDClassifier) من مكتبة Sklearn، حيث يعمل نموذج SGDClassifier أثناء التدريب على ضبط أوزان النموذج بناءً على بيانات التدريب، والهدف من ذلك يتمثّل في العثور على مجموعة الأوزان التي تقلل من **دالة الخسارة** (Loss Function)، وهي الدالة التي تقيس الفرق بين العناوين المتوقَّعة والعناوين الحقيقية في بيانات التدريب.

يستخدِم المقطع البرمجي التالي مُصنِّف SGDClassifier لتدريب نموذج على مجموعة بيانات مسطحة:

خوارزمية بايز الساذجة متعددة الحدود (MultinomialNB Algorithm):

هي خوارزمية تعلُّم آلة تُستخدم لتصنيف النصوص أو البيانات الأخرى في فئات مُختلفة، وتعتمد على خوارزمية **بايز الساذج** (Naive Bayes) وهي طريقة بسيطة وفعّالة لحل مشكلات التصنيف.

خوارزمية مُصنِّف الانحدار التَّدرجي العشوائي (SGDClassifier Algorithm):

هي خوارزمية تعلُّم آلة تُستخدم في تصنيف البيانات في فئات مُختلفة أو مجموعات، وتعتمد على أسلوب يسمى **الانحدار التَّدرجي العشوائي** (Stochastic Gradient Descent – SGD)، وهي طريقة فعّالة لتحسين الأنواع المتعددة للنماذج وتدريبها، بما فيها المُصنِّفات.

```
from sklearn.linear_model import SGDClassifier

model_sgd = SGDClassifier()
model_sgd.fit(X_train_flat, y_train)
pred=model_sgd.predict(X_test_flat)
accuracy_score(y_test,pred)
```

```
0.46511627906976744
```

يُحقق مصنِّف SGDClassifier دقة أعلى بشكل ملحوظ تزيد عن 46%، على الرغم من تدريبه على البيانات نفسها التي دُرِّب مُصنِّف MultinomialNB عليها، ويدل ذلك على فائدة تجربة خوارزميات تصنيف مُختلفة؛ للعثور على أفضل خوارزمية تتناسب مع أي مجموعة بيانات مُعطاة، ومن المهم فهم نقاط القوة والضعف لكل خوارزمية، فعلى سبيل المثال: من المعروف أن خوارزمية SGDClassifier تعمل بشكل أفضل عندما تُحجّم بيانات الإدخال وتوُحّد الخصائص؛ ولهذا السبب ستستخدِم التحجيم القياسي في نموذجك.

التحجيم القياسي (Standard Scaling):
هو تقنية معالجة أولية تُستخدم في تعلُّم الآلة لتحجيم خصائص مجموعة البيانات بحيث تكون ذات متوسط حسابي صفري وتباين أحادي الوحدة.

يستخدِم المقطع البرمجي التالي أداة StandardScaler (المُحجِّم القياسي) من مكتبة sklearn لتحجيم البيانات:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_flat_scaled = scaler.fit_transform(X_train_flat)
X_test_flat_scaled = scaler.fit_transform(X_test_flat)

print(X_train_flat[0]) # the values of the first image pre-scaling
print(X_train_flat_scaled[0]) # the values of the first image post-scaling
```

```
[144 142 151 ...  76  75  80]
[ 0.33463473  0.27468959  0.61190285 ... -0.65170221 -0.62004162
 -0.26774175]
```

يمكن الآن تدريب نموذج جديد واختباره باستخدام مجموعات البيانات التي تم تحجيمها:

```
model_sgd = SGDClassifier()
model_sgd.fit(X_train_flat_scaled, y_train)
pred=model_sgd.predict(X_test_flat_scaled)
accuracy_score(y_test,pred)
```

```
0.4906976744186046
```

تدل النتائج على وجود تحسُّن بعد التحجيم، ومن المحتمل أن يحدث تحسين إضافي بواسطة تجريب خوارزميات أخرى وضبط متغيِّراتها حتّى تتناسب مع مجموعة البيانات بشكل أفضل.

التنبؤ بانتقاء الخصائص Prediction with Feature Selection

ركَّز القسم السابق على تدريب النماذج عن طريق تسطيح البيانات، في حين سيصف هذا القسم كيفية تحويل البيانات الأصلية لهندسة الخصائص الذكية التي تلتقط الصفات الرئيسة لبيانات الصورة، وعلى وجه التحديد يوضِّح القسم تقنية شائعة تسمى **المُخطَّط التكراري للتدرجات الموجَّهة** (Histogram of Oriented Gradients –HOG).

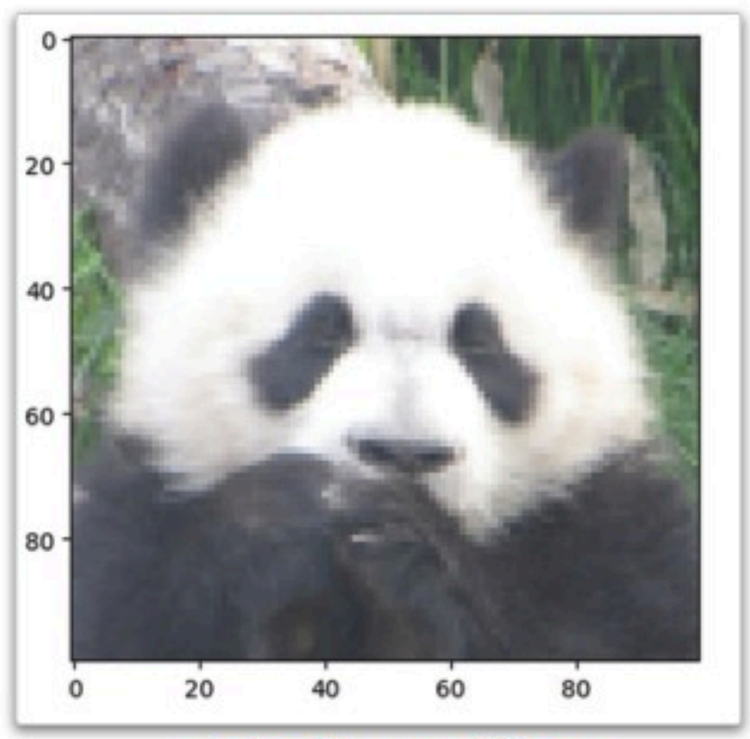
المُخطَّطات التكرارية للتدرجات الموجَّهة (Histogram of Oriented Gradients -HOG):

تقوم المُخطَّطات التكرارية للتدرجات الموجَّهة بتقسيم الصورة إلى أقسام صغيرة وتحلِّل توزيع تغيرات الكثافة في كل قسم حتّى تحدِّد وتفهم شكل الكائن في الصورة.

تتمثَّل الخطوة الأولى في هندسة المُخطَّطات التكرارية للتدرجات الموجَّهة في تحويل الصور من تنسيق RGB إلى صور ذات تدرج رمادي، ويمكن القيام بذلك باستخدام الدالة rgb2gray( ) من مكتبة sckit-image:

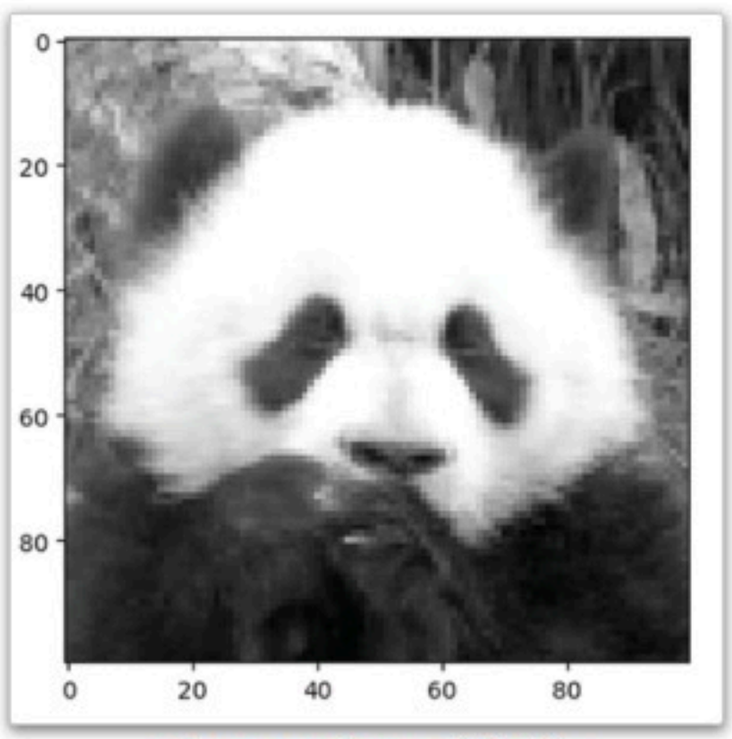
```
from skimage.color import rgb2gray # used to convert a multi-color (rgb) image to grayscale
# converts the training data
X_train_gray = np.array([rgb2gray(img) for img in X_train])
# converts the testing data
X_test_gray = np.array([rgb2gray(img) for img in X_test])
```

```
plt.imshow(X_train[0]);
```



شكل 4.8: صورة بالألوان الأساسية

```
plt.imshow(X_train_gray[0],cmap='gray');
```



شكل 4.9: صورة ذات تدرج رمادي

الشكل الجديد لكل صورة أصبح بتنسيق 100 × 100، بدلًا من التنسيق RGB المُستنِد إلى100 × 100×3:

```
print(X_train[0].shape)
print(X_train_gray[0].shape)
```

```
(100, 100, 3)
(100, 100)
```

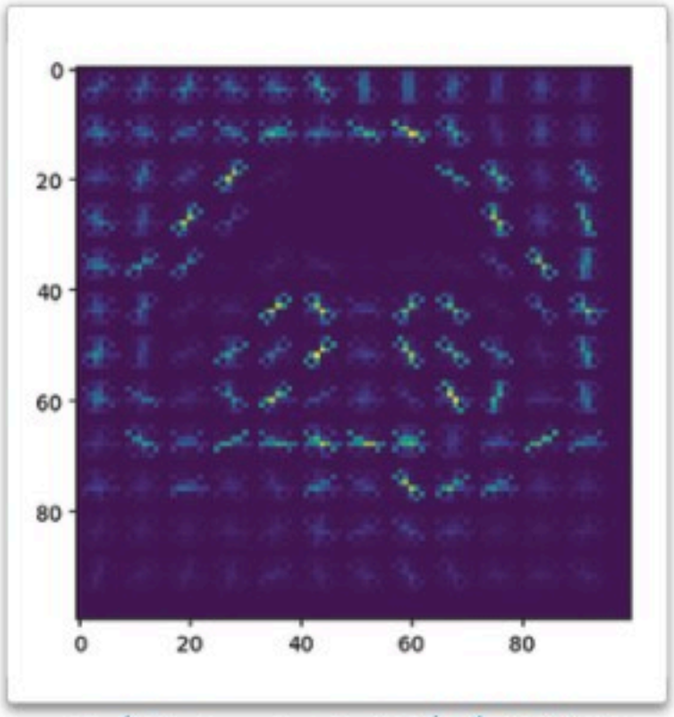
تتمثّل الخطوة التالية في إنشاء خصائص المُخطَّط التكراري للتدرجات الموجَّهة لكل صورة في البيانات، ويمكن تحقيق ذلك من خلال دالة hog( ) من مكتبة sckit-image، ويوضِّح المقطع البرمجي التالي مثالًا على الصورة الأولى في مجموعة بيانات التدريب:

```
from skimage.feature import hog

hog_vector, hog_img = hog(
                  X_train_gray[0],
                  visualize = True
                  )
hog_vector.shape
```

```
(8100,)
```

hog_vector هو متَّجَه أحادي البُعد ذو ثمانية آلاف ومئة قيمة عددية، ويمكن استخدامها لتمثيل الصورة، ويَظهر التمثيل البصري لهذا المتَّجَه باستخدام:

```
plt.imshow(hog_img);
```



شكل 4.10: مُخطَّط تكراري للتدرجات الموجَّهة لصورة

يصوِّر هذا التمثيل الجديد حدود الأشكال الأساسية في الصورة، ويحذف التفاصيل الأخرى ويُركِّز على الأجزاء المفيدة التي يمكنها أن تساعد المُصنِّف على أن يقوم بالتنبؤ، ويطبِّق المقطع البرمجي التالي هذا التغيير على كل الصور في كل من مجموعة التدريب ومجموعة الاختبار:

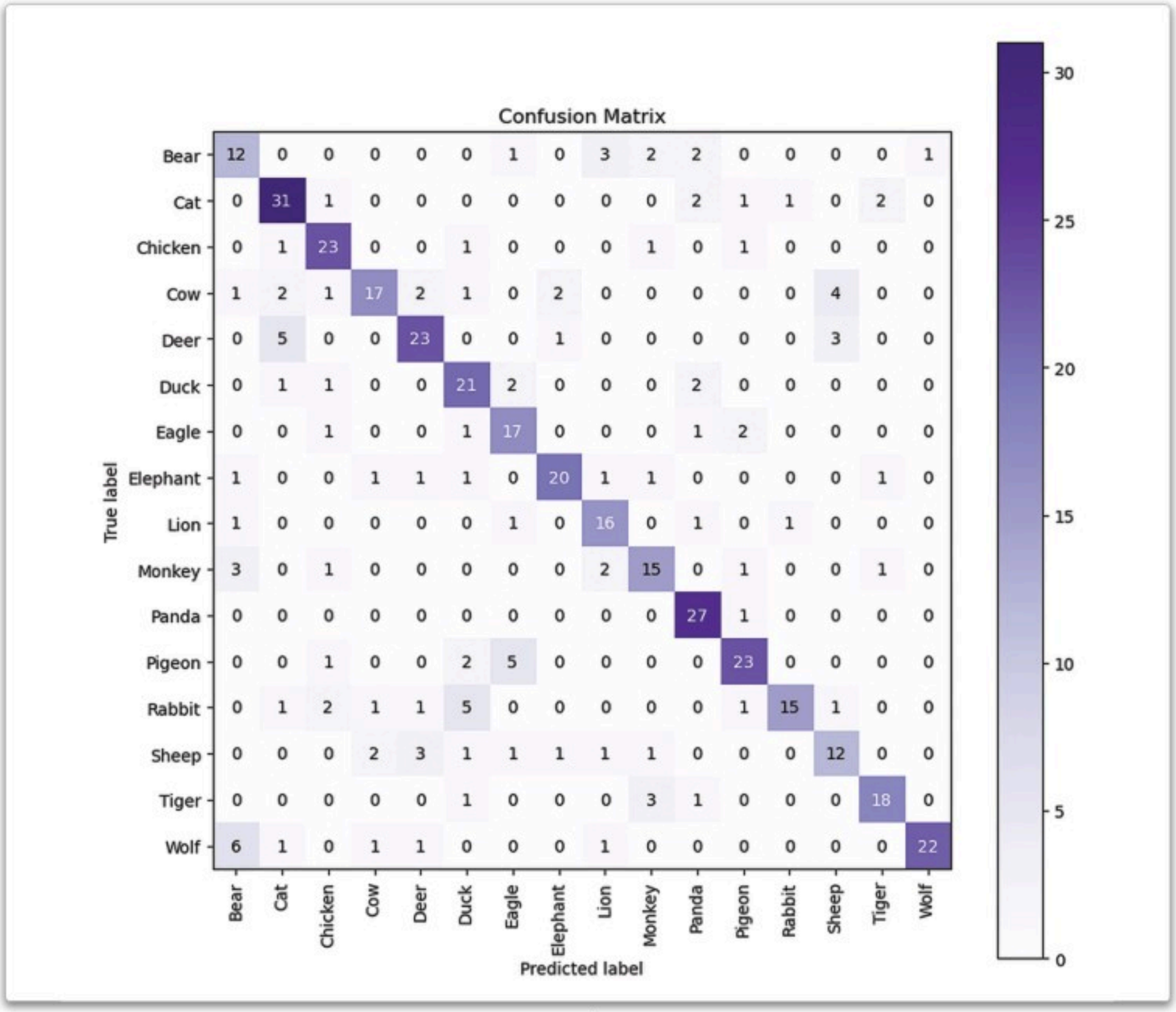
```
X_train_hog = np.array([hog(img) for img in X_train_gray])
X_test_hog = np.array([hog(img) for img in X_test_gray])
```

يمكن الآن تدريب SGDClassifier على هذا التمثيل الجديد:

```
# scales the new data
scaler = StandardScaler()
X_train_hog_scaled = scaler.fit_transform(X_train_hog)
X_test_hog_scaled = scaler.fit_transform(X_test_hog)

# trains a new model
model_sgd = SGDClassifier()
model_sgd.fit(X_train_hog_scaled, y_train)

# tests the model
pred = model_sgd.predict(X_test_hog_scaled)
accuracy_score(y_test,pred)
```

```
0.7418604651162791
```

```
scikitplot.metrics.plot_confusion_matrix(y_test, # actual labels
                                         pred, # predicted labels
                                         title = "Confusion Matrix", # title to use
                                         cmap = "Purples", # color palette to use
                                         figsize = (10,10), # figure size
                                         x_tick_rotation = 90
                                         );
```



شكل 4.11: مصفوفة الدقة لأداء خوارزمية SGDClassifier

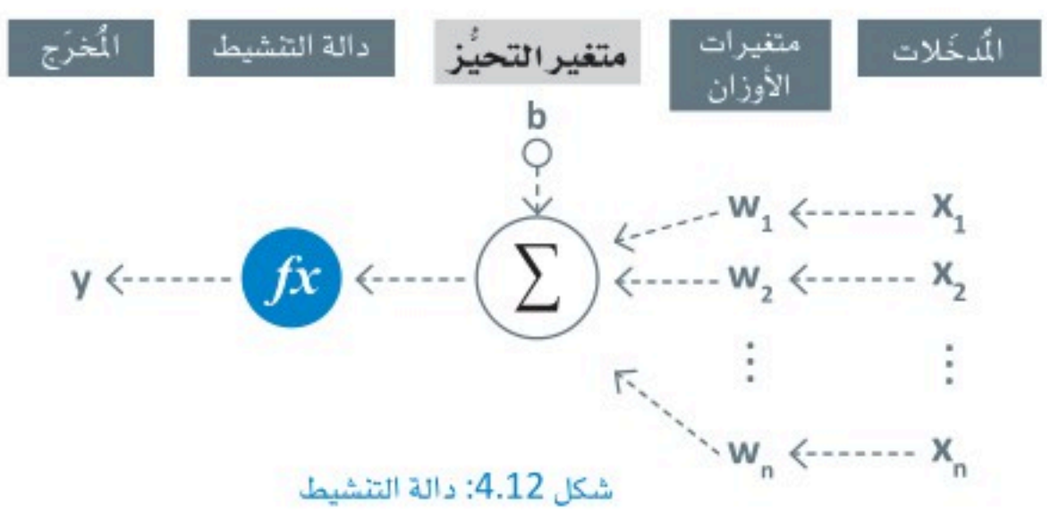
تكشف النتائج الجديدة عن تحسُّن هائل في الدقة التي قفزت لتصل إلى أكثر من 70 %، وتجاوزت بكثير الدقة التي حققها المُصنِّف نفسه على البيانات المسطحة دون القيام بأي هندسة للخصائص، ويتضح التحسُّن أيضًا في مصفوفة الدقة المُحدَّثة التي تشمل عددًا أقل من الأخطاء (التنبؤات الإيجابية الخاطئة)، ويوضِّح ذلك أهمية استخدام تقنيات رؤية الحاسب لهندسة خصائص ذكية تلتقط الصفات المرئية المُختلفة للبيانات.

التنبؤ باستخدام الشبكات العصبية Prediction Using Neural Networks

يوضِّح هذا القسم كيفية استخدام الشبكات العصبية لتصميم مُصنِّفات مخصصة لبيانات الصور، وكيف يمكنها في كثير من الأحيان أن تتفوّق على التقنيات عالية الفعالية مثل: عملية المُخطَّط التكراري للتدرجات الموجَّهة التي وُصفت في القسم السابق، وتُستخدم مكتبة TensorFlow ومكتبة Keras الشهيرتان لهذا الغرض.

مكتبة TensorFlow هي مكتبة منخفضة المستوى تُوفِّر مجموعة واسعة من أدوات تعلُّم الآلة والذكاء الاصطناعي، وتسمح للمستخدِمين بتعريف الحسابات العددية التي تتضمن **متَّجَهات متعددة الأبعاد** (Tensors) ومعالجتها، وهي مصفوفات متعددة الأبعاد من البيانات.من ناحية أخرى، تُعدُّ مكتبة Keras ذات مستوى أعلى وتُوفِّر واجهة أبسط لبناء النماذج وتدريبها، وهي مبنية باستخدام مكتبة TensorFlow (أو مكتبات خلفية أخرى) وتُوفِّر مجموعة من الطبقات والنماذج المعرَّفة مسبقًا والتي يمكن تجميعها بسهولة لبناء نموذج تعلُّم عميق. وصُمِّمت مكتبة Keras لتكون صديقة للمستخدِم وسهلة الاستخدام؛ مما يجعلها خيارًا رائجًا للممارسين.

دوال التنشيط (Activation Functions) هي دوال رياضية تُطَبَّق على مُخرَجات كل خلية عصبية في الشبكة العصبية،كما تتميز بأنها تضيف خصائصَ **غير خطية** (Non-linear) للنموذج وتسمح للشبكة بتعلُّم الأنماط المعقدة في البيانات، ويُعدُّ اختيار دالة التنشيط أمرًا مهمًا ويمكن أن يؤثر على أداء الشبكة، حيث تتلقى الخلايا العصبية المُدخَلات وتعالجها من خلال متغيرات الأوزان والتحيُّزات وتنتج مُخرَجات بناء على دالة التنشيط كما يظهر في الشكل 4.12. تُنشأ الشبكات العصبية من خلال ربط العديد من الخلايا العصبية معًا في طبقات، وتُدرَّب على ضبط متغيرات الأوزان والتحيُّزات وتحسين أدائها بمرور الوقت.



شكل 4.12: دالة التنشيط

يُثبِّت المقطع البرمجي التالي مكتبة tensorflow ومكتبة keras:

```
%%capture
!pip install tensorflow
!pip install keras
```

في الوحدة السابقة، تعرّفت على الخلايا العصبية الاصطناعية وعلى معماريات الشبكات العصبية، وعلى وجه التحديد تعرّفت على نموذج **الكلمة إلى المتَّجَه** (Word2Vec) الذي يَستخدِم طبقة مخفية وطبقة مُخرَجات؛ ليتنبأ بسياق الكلمات لكلمة مُعطاة في جملة. وبعد ذلك تُستخدم مكتبة Keras لإنشاء معمارية عصبية مشابهة للصور. أولًا: تُحوَّل العناوين في y_train إلى تنسيق أعداد صحيحة، طبقًا لمتطلبات مكتبة Keras.

```
# gets the set of all distinct labels
classes=list(set(y_train))
print(classes)
print()

# replaces each label with an integer (its index in the classes lists) for both the training and testing data
y_train_num = np.array([classes.index(label) for label in y_train])
y_test_num = np.array([classes.index(label) for label in y_test])
print()

# example:
print(y_train[:5]) # first 5 labels
print(y_train_num[:5]) # first 5 labels in integer format
```

```
['Elephant', 'Duck', 'Monkey', 'Cow', 'Sheep', 'Wolf', 'Tiger', 'Deer',
'Cat', 'Lion', 'Rabbit', 'Panda', 'Pigeon', 'Chicken', 'Eagle', 'Bear']

['Panda' 'Pigeon' 'Monkey' 'Panda' 'Sheep']
[11 12 2 11 4]
```

ويمكن الآن استخدام أداة Sequential (التتابع) من مكتبة Keras لبناء شبكة عصبية في شكل طبقات متتابعة.

```
from keras.models import Sequential # used to build neural networks as sequences of layers
# every neuron in a dense layer is connected to every other neuron in the previous layer.
from keras.layers import Dense

# builds a sequential stack of layers
model = Sequential()
# adds a dense hidden layer with 200 neurons, and the ReLU activation function.
model.add(Dense(200,input_shape = (X_train_hog.shape[1],), activation='relu'))
# adds a dense output layer and the softmax activation function.
model.add(Dense(len(classes), activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 200)               1620200

 dense_1 (Dense)             (None, 16)                3216

=================================================================
Total params: 1,623,416
Trainable params: 1,623,416
Non-trainable params: 0
_________________________________________________________________
```

عدد الخلايا العصبية في الطبقة المخفية يعتمد على الخيار الذي يُتخذ عند التصميم، وعدد الفئات يحدِّد عدد الخلايا العصبية في طبقة المُخرَجات.

يكشف ملخص النموذج عن العدد الإجمالي للمتغيِّرات التي يجب أن يتعلّمها النموذج من خلال ضبطها على بيانات التدريب، وبما أن المُدخَلات تحتوي على ثمانية آلاف ومئة (8,100) مُدخَل، وهي أبعاد صور المُخطَّط التكراري للتدرجات الموجَّهة X_train_hog وتحتوي الطبقة المخفية على مئتي خلية عصبية، وهي طبقة كثيفة متصلة بالمُدخَلات اتصالًا كاملًا، فإن المجموع 8,100 × 200 = 1,620,000 وصلة موزونة يجب تعلُّم أوزانها (متغيِّراتها). تمت إضافة مئتي متغيِّر **تحيُّز** (Bias) إضافي، بواقع متغيِّر لكل خلية عصبية في الطبقة المخفية، ومتغيِّر التحيُّز هو قيمة تُضاف إلى مُدخَلات كل خلية عصبية في الشبكة العصبية، وتُستخدم لتوجيه دالة تنشيط الخلايا العصبية إلى الجانب السلبي أو الإيجابي، مما يسمح للشبكة بنمذجة علاقات أكثر تعقيدًا بين بيانات المُدخَلات وعناوين المُخرَجات.

وبما أن طبقة المُخرَجات تحتوي على ستّ عشرة خلية عصبية متصلة بالكامل بمئتي خلية عصبية موجودة في الطبقة المخفية، فإن مجموع الوصلات الموزونة يبلغ 16 × 200 = 3,200. ويُضاف ستة عشر متغيِّر تحيُّز إضافي، بواقع متغيِّر واحد لكل خلية عصبية في طبقة المُخرَجات، ويُستخدم السطر البرمجي التالي **لتجميع** (Compile) النموذج:

```
# compiling the  model
model.compile(loss = 'sparse_categorical_crossentropy', metrics =
['accuracy'], optimizer = 'adam')
```

تُستخدم دالة إعداد النموذج الذكي في مكتبة Keras والمعروفة **بالتجميع** (model.compile()) في عملية تحديد الخصائص الأساسية للنموذج الذكي وإعداده للتدريب والتحقق والتنبؤ، وتتخذ ثلاثة مُعامِلات رئيسة كما هو موضَّح في الجدول 4.2.

جدول 4.2: مُعامِلات طريقة التجميع

الخسارة (loss)	هي الدالة التي تُستخدم لتقييم الخطأ في النموذج أثناء التدريب، وتقيس مدى تطابق تنبؤات النموذج مع العناوين الحقيقية لمجموعة معيّنة من بيانات المُدخَلات. الهدف من التدريب تقليل دالة الخسارة مما يتضمن في العادة تعديل أوزان النموذج ومقدار التحيُّز، وفي هذه الحالة تكون دالة الخسارة هي: sparse_categorical_crossentropy وهي دالة خسارة مناسبة لمهام التصنيف متعدِّدة الفئات؛ حيث تكون العناوين أعدادًا صحيحة كما في y_train_num.
المقاييس (metrics)	هي قائمة المقاييس المستخدَمة لتقييم النموذج أثناء التدريب والاختبار، وتُحسب هذه المقاييس باستخدام مُخرَجات النموذج والعناوين الحقيقية، ويمكن استخدامها لمراقبة أداء النموذج وتحديد المجالات التي يمكن تحسينه فيها. مقياس **الدقة** (Accuracy) هو مقياس شائع لمهام التصنيف يقيس نسبة التنبؤات الصحيحة التي قام بها النموذج.
المُحسّن (optimizer)	هو خوارزمية التحسين التي تُستخدم في ضبط أوزان النموذج ومقدار التحيُّز أثناء التدريب. ويستخدِم المحسِّن دالة الخسارة والمقاييس لإرشاد عملية التدريب، ويقوم بضبط متغيِّرات النموذج في محاولة لتقليل الخسارة وزيادة أداء النموذج إلى الحد الأقصى. وفي هذه الحالة فقد تم استخدام المُحسِّن adam الذي يُعدُّ خوارزمية شائعة لتدريب الشبكات العصبية.

وأخيرًا، تُستخدم دالة fit ( ) لتدريب النموذج على البيانات المتاحة.

```
model.fit(X_train_hog, # training data
          y_train_num, # labels in integer format
          batch_size = 80, # number of samples processed per batch
          epochs = 40, # number of iterations over the whole dataset
         )
```

```
Epoch 1/40
17/17 [==============================] - 1s 16ms/step - loss: 2.2260 - accuracy: 0.3333
Epoch 2/40
17/17 [==============================] - 0s 15ms/step - loss: 1.1182 - accuracy: 0.7256
Epoch 3/40
17/17 [==============================] - 0s 15ms/step - loss: 0.7198 - accuracy: 0.8155
Epoch 4/40
17/17 [==============================] - 0s 15ms/step - loss: 0.4978 - accuracy: 0.9031
Epoch 5/40
17/17 [==============================] - 0s 16ms/step - loss: 0.3676 - accuracy: 0.9388
...
Epoch 36/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0085 - accuracy: 1.0000
Epoch 37/40
17/17 [==============================] - 0s 21ms/step - loss: 0.0080 - accuracy: 1.0000
Epoch 38/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0076 - accuracy: 1.0000
Epoch 39/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0073 - accuracy: 1.0000
Epoch 40/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0071 - accuracy: 1.0000
```

تُستخدم دالة ( ) fit لتدريب نموذج على مجموعة معيّنة من بيانات الإدخال والعناوين، وتتخذ أربع مُعامِلات رئيسة، كما هو موضَّح في الجدول 4.3.

جدول 4.3: مُعامِلات طريقة fit

X_train_hog	هو مُعامِل بيانات الإدخال المستخدَمة لتدريب النموذج، وتتكون من البيانات المحوَّلة عن طريق المُخطَّط التكراري للتدرجات الموجَّهة التي استُخدمت أيضًا لتدريب أحدث إصدار من خوارزمية SGDClassifier في القسم السابق.
y_train_num	هو مُعامِل يتضمّن عنوانًا لكل صورة بتنسيق أعداد صحيحة.
batch_size	هو عدد العينات التي تمت معالجتها في كل دُفعة أثناء التدريب، ويقوم النموذج بتحديث أوزانه ومقدار التحيُّز بعد كل دُفعة، ويمكن أن يؤثر حجم الدُفعة على سرعة عملية التدريب، واستقراراها، كما يمكن أن تؤدي أحجام الدُفعات الأكبر إلى تدريب أسرع، ولكنها قد تكون أكثر تكلفة من الناحية الحسابية وقد تؤدي إلى تدرجات أقل استقرارًا.
epochs	هو عدد المرات التي يتكرر فيها تدريب النموذج باستخدام مجموعة البيانات بأكملها، وتتكون **الفترة** (Epoch) من مرور واحد عبر مجموعة البيانات بأكملها. ويقوم النموذج بتحديث أوزانه ومقدار التحيُّز بعد كل دورة، كما يمكن أن يؤثر عدد الفترات على قدرة النموذج على التعلُّم والتعميم على البيانات الجديدة، والفترة متغيِّر مهم يجب اختياره بعناية، وفي هذه الحالة يُدرَّب النموذج على أربعين فترة.

ويمكن الآن استخدام نموذج التدريب للتنبؤ بعناوين الصور في مجموعة الاختبار.

```
pred = model.predict(X_test_hog)
pred[0] # prints the predictions for the first image
```

```
14/14 [==============================] - 0s 2ms/step

array([4.79123509e-03, 9.79321003e-01, 8.39506648e-03, 1.97884417e-03,
       7.83501855e-06, 3.50346789e-04, 3.45465224e-07, 1.19854585e-05,
       4.41945267e-05, 4.11721296e-04, 1.27362555e-05, 9.83431892e-06,
       1.97038025e-04, 2.34744814e-03, 5.49758552e-04, 1.57057808e-03],
      dtype=float32)
```

بينما تُظهر دالة ( )predict من مكتبة sklearn العنوان الأكثر احتمالًا الذي يتنبأ به المُصنِّف، تُظهر دالة ( )predict في مكتبة Keras احتمالات كل العناوين المُرشَّحة. في هذه الحالة، يمكن استخدام دالة ( )np.argmax لإظهار مؤشر العنوان الأكثر احتمالًا.

```
# index of the class with the highest predicted probability.
print(np.argmax(pred[0]))
# name of this class
print(classes[np.argmax(pred[0])])
# uses axis=1 to find the index of the max value per row
accuracy_score(y_test_num,np.argmax(pred, axis=1))
```

```
1
Duck
0.7529021558872305
```

تحقق هذه الشبكة العصبية البسيطة دقة تبلغ حوالي 75%، وهي دقة مشابهة لدقة SGDClassifier، ولكن ميزة المعماريات العصبية تنبع من براعتها، وهو ما يسمح لك بتجربة معماريات مُختلفة للعثور على أفضل ما يناسب مجموعة بياناتك. تم تحقيق هذه الدقة من خلال معمارية بسيطة تضمنت طبقة مخفية واحدة تحتوي على مئتي خلية عصبية، وإضافة طبقات إضافية تجعل الشبكة أعمق، بينما تؤدي إضافة المزيد من الخلايا العصبية لكل طبقة إلى جعلها أوسع، ويُعدُّ اختيار عدد الطبقات وعدد الخلايا العصبية لكل طبقة عناصر مهمة لتصميم الشبكة العصبية، ولها تأثير كبير على أدائها، ولكنها ليست الطريقة الوحيدة لتحسين الأداء، وفي بعض الحالات قد يكون استخدام نوع مُختلف من معمارية الشبكة العصبية أكثر فاعلية.

التنبؤ باستخدام الشبكات العصبية الترشيحية

Prediction Using Convolutional Neural Networks

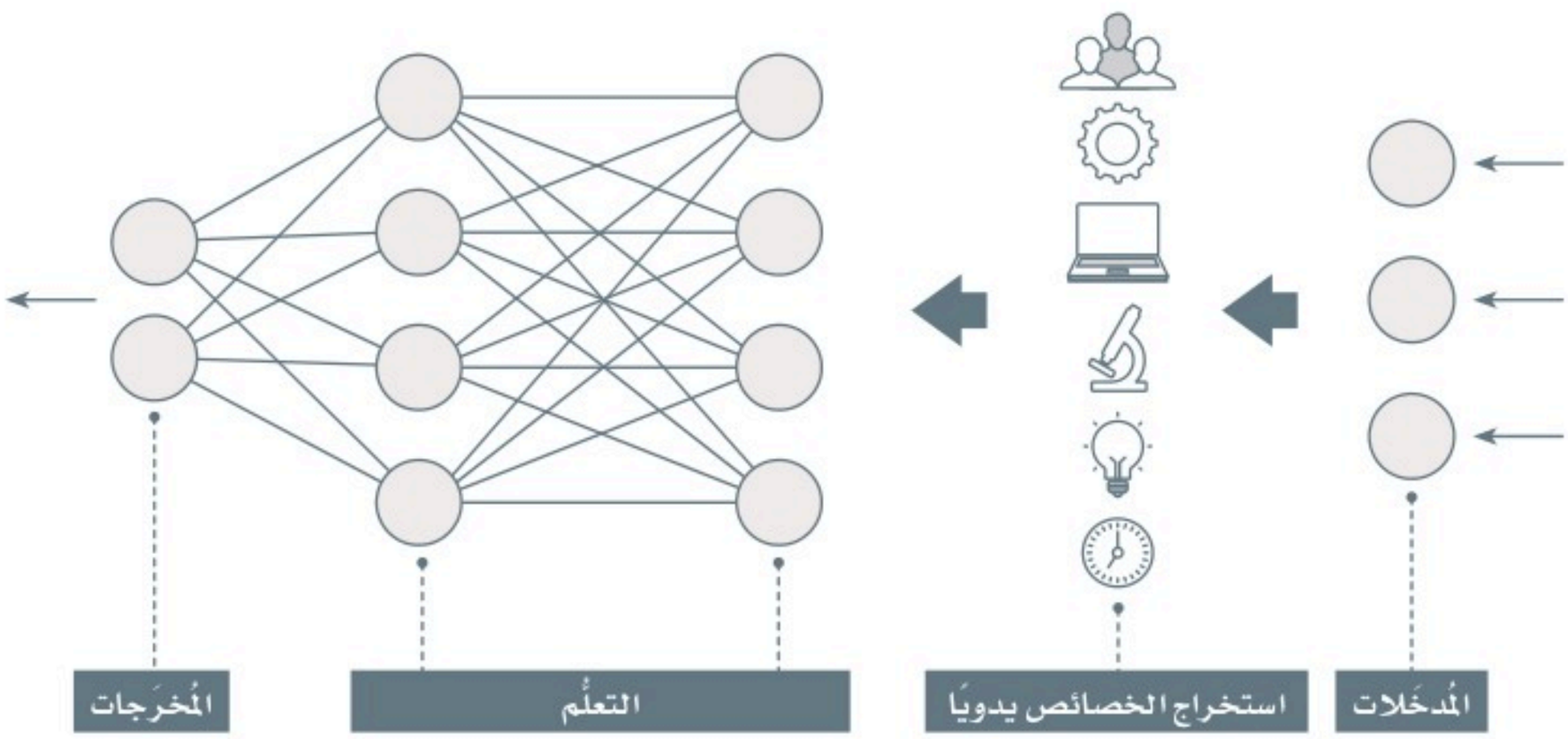
أحد هذه الأنواع من المعماريات التي تناسب تصنيف الصور بشكلٍ جيّد يتمثّل في **الشبكة العصبية الترشيحية** (Convolutional Neural Network –CNN)، وبما أن الشبكة العصبية الترشيحية تعالج بيانات الإدخال، فإنها تقوم باستمرار بضبط متغيِّرات الفلاتر المرشَّحَة لاكتشاف الأنماط بناءً على البيانات التي تراها؛ حتّى تتمكن بشكل أفضل من اكتشاف الخصائص المهمة، ثم تنقل مُخرَجات كل طبقة إلى الطبقة التالية التي يُكتشف فيها خصائص أكثر تعقيدًا إلى أن تُنتج المُخرَجات النهائية.

على الرغم من فوائد الشبكات العصبية المعقدة مثل: الشبكات العصبية الترشيحية إلا أنه من المهم ملاحظة ما يلي:

- تكمن قوة الشبكات العصبية الترشيحية في قدرتها على أن تستخرج الخصائص المهمة ذات الصلة من الصور بشكل تلقائي، دون الحاجة إلى **هندسة الخصائص اليدوية** (Manual Feature Engineering).
- تحتوي المعماريات العصبية الأكثر تعقيدًا على المزيد من المتغيِّرات التي يجب تعلُّمها من البيانات أثناء التدريب، ويتطلب ذلك مجموعة بيانات تدريب أكبر قد لا تكون متاحة في بعض الحالات، وفي مثل هذه الحالات من غير المحتمل أن يكون إنشاء معمارية معقدة للغاية أمرًا فعَّالًا.
- على الرغم من أن الشبكات العصبية قد حققت بالفعل نتائج مبهرة في معالجة الصور والمهام الأخرى، إلا أنها لا تضمن تقديم أفضل أداء لجميع المشكلات ومجموعات البيانات.
- حتّى لو كانت معمارية الشبكة العصبية أفضل حل ممكن لمُهِمَّة محددة، فقد يستغرق الأمر كثيرًا من الوقت والجهد والموارد الحاسوبية لتجربة خيارات مُختلفة إلى أن يتم العثور على هذه المعمارية. لذلك من الأفضل البدء بنماذج أبسط (لكنها لا تزال فعّالة)، مثل: نموذج SGDClassifier وغيره من النماذج الأخرى الكثيرة المتوفرة في المكتبات مثل: مكتبة sklearn، وبمجرد حصولك على تنبُّؤ أفضل لمجموعة البيانات ووصولك إلى النقطة التي لا يمكن فيها تحسين هذه النماذج أكثر من ذلك، فإن التجريب على المعماريات العصبية الأخرى يُعدُّ خطوة ممتازة.

الشبكة العصبية الترشيحية (Convolutional Neural Network -CNN):

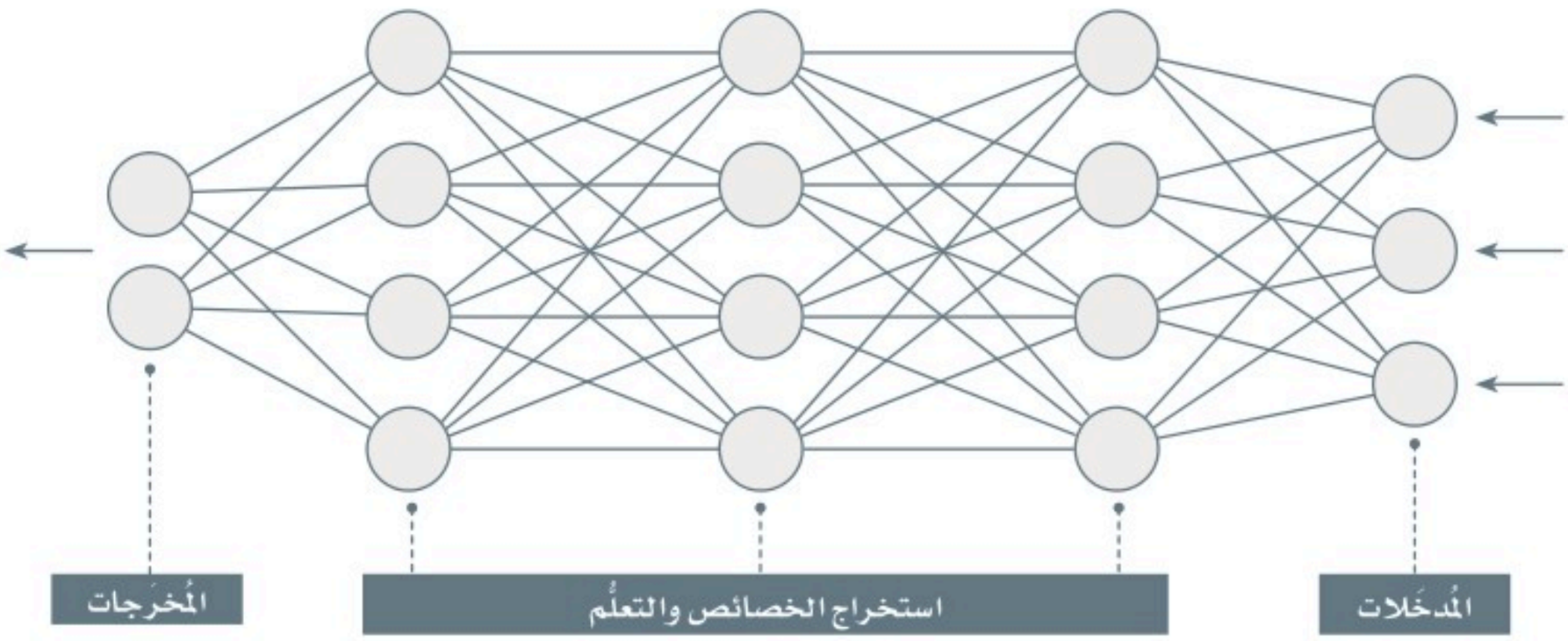
هي شبكات عصبية عميقة تتعلّم تلقائيًا تسلسل الخصائص من البيانات الخام مثل الصور، عن طريق تطبيق سلسلة من الفلاتر الترشيحية على بيانات الإدخال، التي يتم تصميمها بحيث تكتشف أنماطًا أو خصائصَ محدَّدة.



شكل 4.13: شبكة عصبية ذات هندسة خصائص يدوية

معلومة

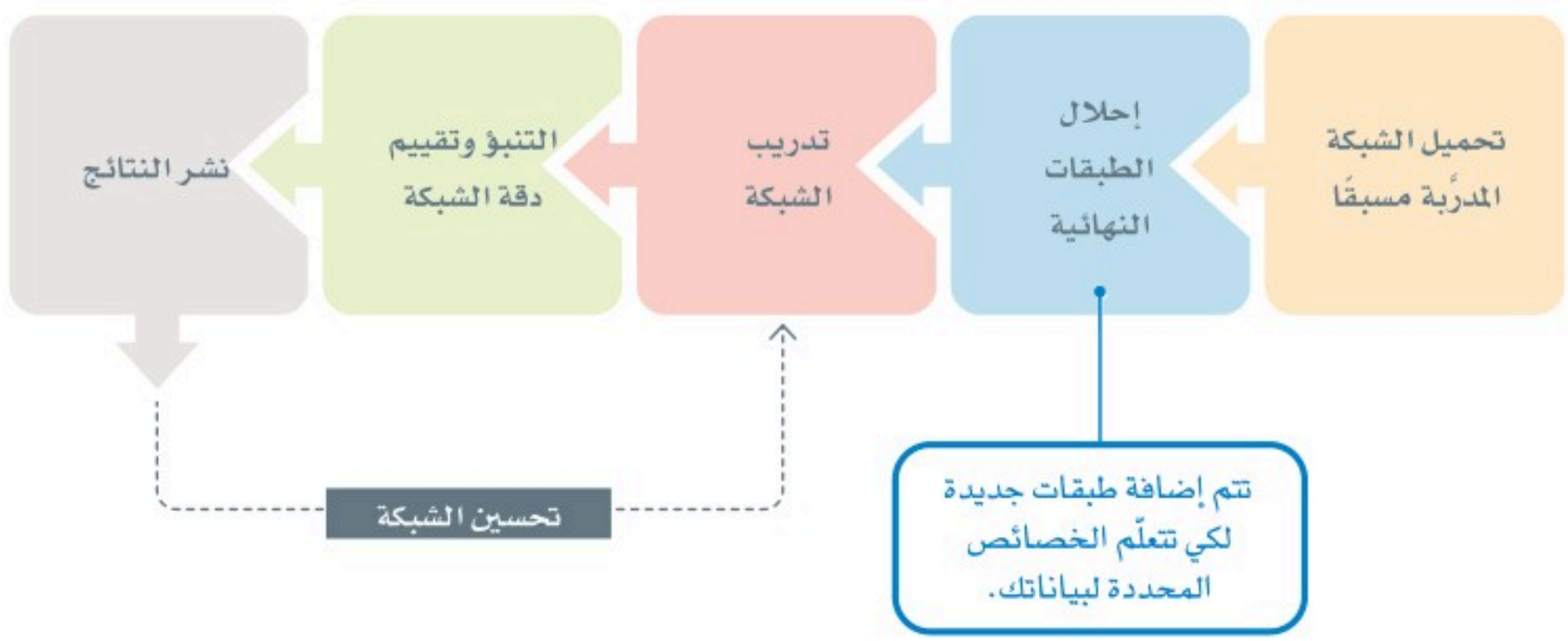
من المزايا الأساسية للشبكات العصبية الترشيحية أنها جيدة جدًا في التعلُّم من كميات كبيرة من البيانات، ويمكنها في العادة أن تحقق مستويات عليا في دقة المهام مثل: تصنيف الصور دون الحاجة إلى هندسة الخصائص اليدوية مثل: المُخطَّط التكراري للتدرجات الموجَّهة.



شكل 4.14: شبكة عصبية ترشيحية من دون هندسة الخصائص اليدوية

التعلُّم المنقول Transfer Learning

التعلُّم المنقول هو عملية يُعاد فيها استخدام شبكة عصبية مُدرَّبة مسبقًا في حل مُهِمَّة جديدة. في سياق الشبكات العصبية الترشيحية يتضمن التعلُّم المنقول أخذ نموذج مدرَّب مسبقًا على مجموعة بيانات كبيرة وتكييفه على مجموعة بيانات أو مُهِمَّة جديدة، فبدلًا من البدء من نقطة الصفر، يتيح التعلُّم المنقول استخدام النماذج المدرَّبة مسبقًا، أي التي تعلّمت بالفعل خصائص مهمة مثل: الحواف، والأشكال، والنقوش من مجموعة بيانات التدريب.



شكل 4.15: إعادة استخدام الشبكة المدرَّبة مسبقًا

تمرينات

1 ما تحديّات تصنيف البيانات المرئية؟

2 لديك مصفوفتا قيم Numpy، وهما مصفوفة X_train ومصفوفة Y_train. كل صف في مصفوفة X_train شكله (3،100، 100) يمثّل صورة بأبعاد 100x100 وبتنسيق RGB. والصف n في المصفوفة Y_train يمثّل تسمية صورة n في مصفوفة X_train. أكمل المقطع البرمجي التالي، بحيث يُسطّح X_train ثم يُدرِّب النموذج MultinomialNB على مجموعة البيانات هذه:

```
from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier from sklearn

X_train_flat = np.array(_____________)

model_MNB = MultinomialNB() # new Naive Bayes model

model_MNB.fit(_________________, _________________) # fits model on the flat training data
```

3 صِف باختصار طريقة عمل الشبكات العصبية الترشيحية وإحدى مميزاتها الرئيسة.

4 لديك مصفوفتا قيم Numpy، وهما مصفوفة X_train ومصفوفة Y_train. كل صف في مصفوفة X_train شكله (3،100،100) يمثّل صورة بأبعاد 100x100 وبتنسيق RGB. والصف n في المصفوفة Y_train يمثّل تسمية صورة n في مصفوفة X_train. أكمل المقطع البرمجي التالي، بحيث يطبّق تحويلات المُخطَّط التكراري للتدرجات الموجَّهة ثم يستخدِم البيانات المحوَّلة في تدريب نموذج:

```
from skimage.color import _____________   # used to convert a multi-color (rgb) image to grayscale

from sklearn._____________ import StandardScaler # used to scale the data

from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier from sklearn

X_train_gray = np.array([_________________(img) for img in X_train]) # converts training data

X_train_hog = ___________________

scaler = StandardScaler()

X_train_hog_scaled = ___________________.fit_transform(X_train_hog)

model_MNB = MultinomialNB()

model_MNB.fit(X_train_flat_scaled, ___________________)
```

5 اذكر بعض عيوب الشبكات العصبية الترشيحية.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

التعلُّم غير الموجَّه لتحليل الصور

فهم محتوى الصور
Understanding Image Content

في سياق رؤية الحاسب يُستخدم التعلُّم غير الموجَّه في مجموعة متنوّعة من المهام مثل: **تقطيع** أو **تجزئة الصورة** (Image Segmentation)، و**تقطيع الفيديو** (Video Segmentation)، و**اكتشاف العناصر الشاذّة** (Anomaly Detection)، ومن الاستخدامات الرئيسة الأخرى للتعلُّم غير الموجَّه: **البحث عن الصورة** (Image Search) ويتضمن البحث في قاعدة بيانات كبيرة من الصور للعثور على الصورة المشابهة للصورة المطلوبة.

تتمثّل الخطوة الأولى لبناء محرك بحث لبيانات صورة في تحديد **دالة التشابه** (Similarity Function) والتي يمكنها تقييم التشابه بين صورتين بناءً على خصائصهما المرئية مثل: الحدود، أو النقش، أو الشكل. وبمجرد أن يُرسِل المستخدِم صورة جديدة ليستعلم عنها، يقوم محرك البحث بالاطلاع على جميع الصور الموجودة في قاعدة البيانات المتاحة، ويعثر على الصور التي بها أعلى درجة تشابه، ويُظهرها للمستخدِم.

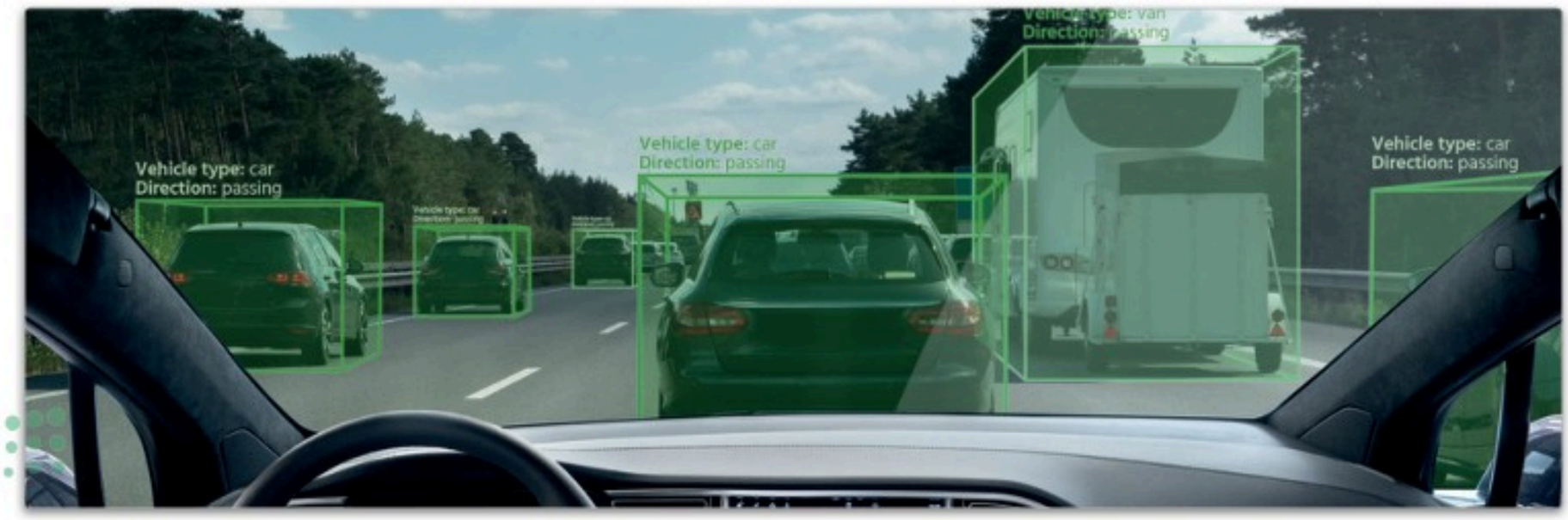
وهناك طريقة بديلة تتمثّل في استخدام دالة التشابه لفصل الصور في عناقيد؛ بحيث يتكون كل عنقود من صور متشابهة بصريًا مع بعضها، ثم يُمَثَّل كل عنقود من خلال **بؤرة تجميع** (Centroid): وهي صورة تقع في مركز العنقود وتمتلك أصغر مسافة عامة (أي اختلاف) من الصور الأخرى في العنقود. وبمجرد أن يُرسِل المستخدِم صورة جديدة للاستعلام عنها، فإن محرك البحث سينتقل إلى جميع العناقيد ويختار العنقود الذي تكون بؤرة تجميعه أكثر تشابهًا مع الصورة المطلوبة من المستخدِم لتظهر له صور العنقود المحددة، ويوضِّح الشكل 4.16 مثالًا على هذا.

اكتشاف العناصر الشاذّة (Anomaly Detection):

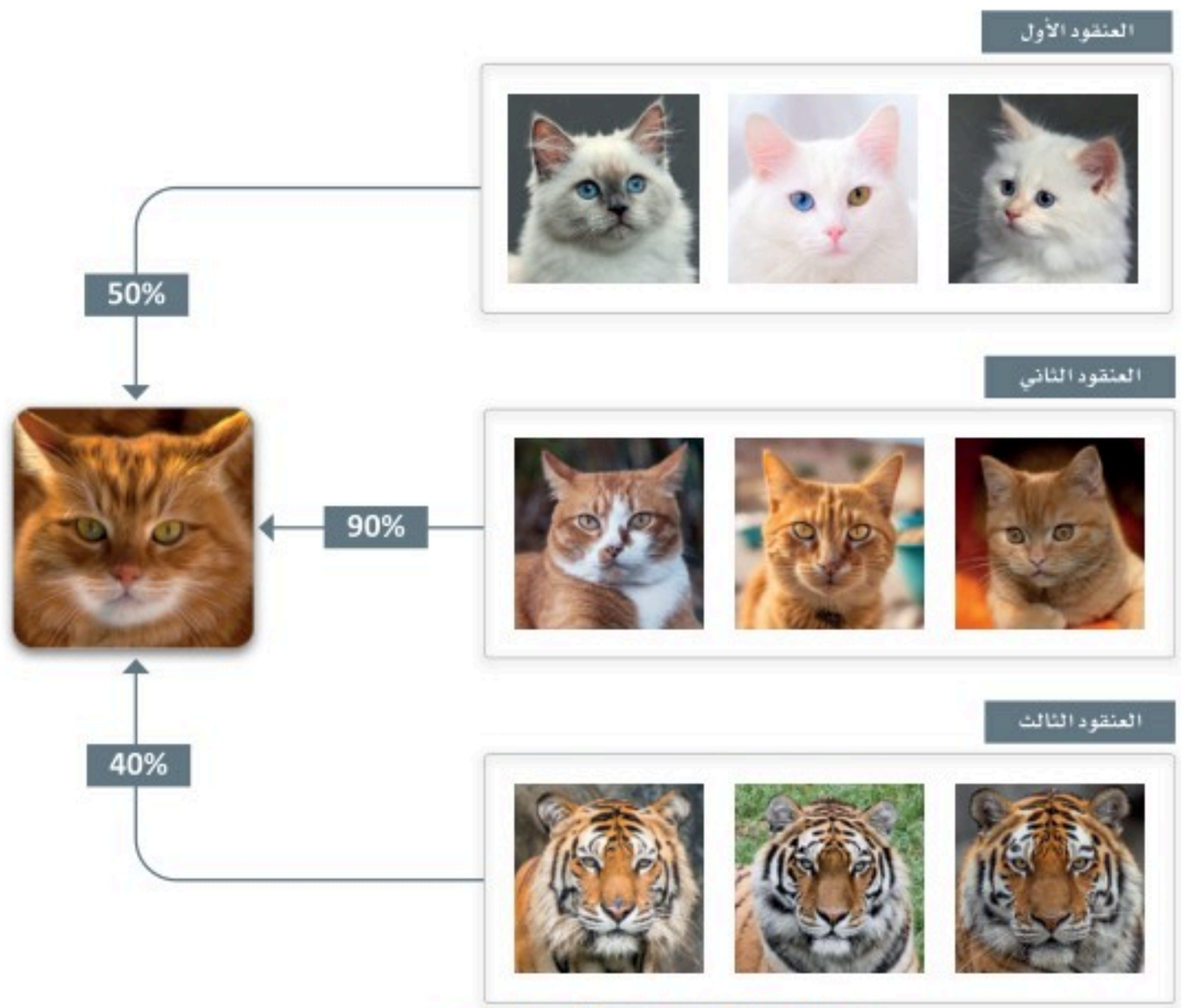
هي عملية تُستخدم لتحديد الأنماط أو الأحداث أو نقاط البيانات الشاذّة أو غير الطبيعية داخل مجموعة البيانات، وتهدف إلى الكشف عن الحالات الغريبة التي تختلف عن المعيار وقد تحتاج إلى استقصاء إضافي.

تقطيع الصورة (Image Segmentation):

هي عملية تقسيم الصورة إلى أجزاء أو مناطق متعددة تتقاسم خصائص بصرية مشتركة، وتهدف إلى تجزئة الصورة إلى أجزاء مترابطة، وذات مغزى يمكن استخدامها في القيام بتحليل إضافي.



شكل 4.16: رؤية مركبة ذاتية القيادة من خلال تقطيع الصورة



شكل 4.17: عناقيد التعرّف على الصور

في المثال الموضَّح في الشكل 4.17، تحتوي صورة البحث على تشابه بنسبة: 40% و50% و90% مع بُؤر التجميع لعناقيد الصور الثلاث على التوالي، ويُفترض أن تكون نسبة التشابه بين 0% و 100%، وحصل العنقود الثاني على أعلى نسبة تشابه؛ إذ أنه يشتمل على قطط من نفس سلالة ولون القطّة المحددة في صورة البحث، كما أن نتائج العنقودين الأول والثالث متقاربة (40% و50%)؛ إذ يتشابه العنقودان مع صورة البحث بطرائق مُختلفة، أما العنقود الأول فيتضمن قططًا يختلف نمط ألوانها تمامًا عن المطلوب، وبالرغم من أن العنقود الثالث يمثِّل نوعًا مُختلفًا من الحيوانات وهو النمر، فإن نمط اللون مشابه لصورة البحث.

تُشبه عملية تجميع البيانات المرئية في عناقيد، عملية تجميع البيانات الرقمية أو النصيَّة، ومع ذلك تتطلب الطبيعة الفريدة للبيانات المرئية طرائق متخصّصة؛ لتقييم التشابه البصري، وبالرغم من أن الأساليب الأقدم كانت تعتمد على خصائص مصنوعة يدويًا، فقد أدت التطورات الحديثة في التعلُّم العميق إلى تطوير نماذج قوية يمكنها تلقائيًا أن تتعلّم خصائص متطورة من البيانات المرئية غير المُعَنوَنة.

يستخدِم هذا الدرس مُهِمَّة خاصة بتجميع الصور؛ لتوضيح كيف يمكن أن يؤدي استخدام خصائص أكثر تعقيدًا إلى تقديم نتائج أفضل بشكل ملحوظ، وسيوضِّح هذا الدرس -تحديدًا- ثلاث طرائق مُختلفة:

- تسطيح البيانات الأصلية وتجميعها بدون أي هندسة للخصائص.
- تحويل البيانات باستخدام **واصف الخصائص** (Feature Descriptor) الذي يعتمد على **المُخطَّط التكراري للتدرجات الموجَّهة** (HOG) - تعرّفت عليه في الدرس السابق- ثم تجميع البيانات المحوَّلة.
- استخدام نموذج الشبكة العصبية؛ لتجميع البيانات الأصلية في مجموعات عنقودية بدون هندسة الخصائص.

مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) التي استُخدمت في الدرس السابق وستُستخدم في هذا الدرس أيضًا؛ لتقييم التقنيات المتنوّعة لتجميع الصور، وتم تصميم هذه المجموعة في الأصل لمهام التصنيف، وتتضمن العنوان الحقيقي (نوع الحيوان الفعلي) لكل صورة. وفي هذا الدرس، ستُستخدم هذه العناوين فقط للتحقق من صحتها، ولن تُستخدم لتجميع الصور. يجب أن يكون أي أسلوب تجميع أسلوبًا فعّالًا وقادرًا على تجميع الصور مع العنوان نفسه، وفي العنقود نفسه، وعلى فصل الصور ذات العناوين المُختلفة، ووضعها في عناقيد مُتباينة.

تحميل الصور ومعالجتها أوليًا Loading and Preprocessing Images

يستورد المقطع البرمجي التالي المكتبات التي ستُستخدم لتحميل الصور ومعالجتها أوليًا:

```
%%capture
import matplotlib.pyplot as plt
from os import listdiry

!pip install scikit-image
from skimage.io import imread
from skimage.transform import resize
from skimage import img_as_ubyte

# a palette of 10 colors that will be used to visualize the clusters.
color_palette = ['blue','green','red','yellow','gray','purple','orange',
'pink','black','brown']
```

تقرأ الدالة التالية صور مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) من input_folder (مجلد_المُدخَلات) الخاص بها، وتُعدِّل حجم كل منها بحيث تكون لها أبعاد الطول والعرض نفسها، ثم تقوم بتحسين دالة resize_images( ) من الدرس السابق بالسماح للمستخدِم بأن يحدِّد قائمة فئات الحيوانات التي يجب أن تؤخذ بالاعتبار، كما أنها تستخدِم سطرًا واحدًا من المقطع البرمجي بلغة البايثون؛ لكي تقرأ كل صورة وتعدِّل حجمها وتخزِّنها:

```
def resize_images_v2(input_folder:str,
                     width:int,
                     height:int,
                     labels_to_keep:list
                    ):
    labels = []          # a list with the label for each image
    resized_images = [] # a list of resized images in np array format
    filenames = []       # a list of the original image file names

    for subfolder in listdir(input_folder):

        print(subfolder)
        path = input_folder + '/' + subfolder

        for file in listdir(path):

            label=subfolder[:-4] # uses the subfolder name without the "Head" suffix
            if label not in labels_to_keep: continue
            labels.append(label) # appends the label
            #loads, resizes, preprocesses, and stores the image.
            resized_images.append(img_as_ubyte(resize(imread(path+'/'+file),
(width, height))))
            filenames.append(file)

    return resized_images,labels,filenames
```

البيانات غير المُنظّمة (Unstructured Data) متنوّعة، ويمكن أن تحتاج إلى كثير من الوقت والموارد الحاسوبية، ويُعدُّ هذا صحيحًا بشكلٍ خاص عند معالجتها عن طريق أساليب تعلُّم عميقة ومعقدة، كما سيُنفذ لاحقًا في هذا الدرس، ولتقليل الوقت الحسابي يتم تطبيق دالة resize_images_v2( ) على مجموعة فرعية من الصور من فئات الحيوانات:

```
resized_images,labels,filenames=resize_images_v2(
            "AnimalFace/Image",
            width = 224,
            height = 224,
            labels_to_keep=['Lion', 'Chicken', 'Duck', 'Rabbit', 'Deer',
'Cat', 'Wolf', 'Bear', 'Pigeon', 'Eagle']
            )
```

```
BearHead
CatHead
ChickenHead
CowHead
DeerHead
DuckHead
EagleHead
ElephantHead
LionHead
MonkeyHead
Natural
PandaHead
PigeonHead
RabbitHead
SheepHead
TigerHead
WolfHead
```

هذه العناوين العشرة التي سيتم استخدامها.

يمكنك بسهولة تعديل المتغيِّر labels_to_keep (العناوين _ المحتفظ بها)؛ للتركيز على فئات معيّنة، وستلاحظ أن عرض الصور وارتفاعها تم ضبطهما على 224 × 224، بدلًا من الشكل 100 × 100 الذي استُخدم في الدرس السابق؛ لأن إحدى طرائق التجميع القائمة على التعلُّم العميق -الواردة في هذا الدرس- تتطلب أن تكون للصور هذه الأبعاد، ولذا اعتُمد الشكل 224 × 224؛ لضمان منح حق الوصول لجميع الطرائق إلى المُدخَلات نفسها.

كما ذُكِر في الدرس السابق فإن القوائم الأصلية: resized_images (الصور _المُعدَّل حجمها)، وlabels (العناوين)، وfilenames (أسماء الملفات) تشتمل على الصور التي تنتمي لكل فئة مُجمَّعة معًا. على سبيل المثال: تظهر جميع صور Lion (الأسد) معًا في بداية القائمة المُعدَّل حجمها، وقد يُضلِل ذلك العديد من الخوارزميات، خاصة في مجال رؤية الحاسب، وطالما أنه يمكن فهرسة الصور عشوائيًا لكل قائمة من القوائم الثلاث، فمن المهم التأكد من استخدام الترتيب العشوائي نفسه لهذه القوائم. وبخلاف ذلك، من المستحيل العثور على العنوان الصحيح لصورة معيّنة أو اسم الملف الصحيح لها.

في الدرس السابق، تم إجراء **إعادة الترتيب** (Shuffling) باستخدام الدالة train_test_split( )، وبما أن هذه الدالة غير قابلة للتطبيق على مهام التجميع، فستستخدِم المقطع البرمجي التالي لإعادة الترتيب:

```
import random

#connects the three lists together, so that they are shuffled in the same order
connected = list(zip(resized_images,labels,filenames))
random.shuffle(connected)
# disconnects the three lists
resized_images,labels,filenames= zip(*connected)
```

تتمثّل الخطوة التالية في تحويل قائمتي resized_images (الصور_المُعدَّل حجمها)، وlabels (العناوين) إلى مصفوفات numpy، وكما هو الحال في الدرس السابق يُستخدم الاسمان المتغيِّران القياسيان (X،Y) لتمثيل البيانات والعناوين:

```
import numpy as np # used for numeric computations
X = np.array(resized_images)
Y = np.array(labels)

X.shape
```

```
(1085, 224, 224, 3)
```

يتحقق شكل البيانات من أنها تشمل 1,085 صورة، كل صورة منها ذات أبعاد 224 × 224، وذات ثلاث قنوات ألوان RGB.

التجميع بدون هندسة الخصائص Clustering without Feature Engineering

ستركز محاولة التجميع الأولى على القيام بتسطيح الصور؛ لتحويل كل منها إلى متَّجَه أحادي البُعد أرقامه 224 × 224 × 3 = 150,528 رقمًا.

وعلى غرار خوارزميات التصنيف التي تم توضيحها في الدرس السابق، فإن معظم خوارزميات التجميع تتطلب هذا النوع من التنسيق المتَّجَهي.

```
X_flat = np.array([img.flatten() for img in X])
X_flat[0].shape
```

```
(150528,)
```

```
X_flat[0] # prints the first flat image
```

```
array([107, 146, 102, ...,  91,  86, 108], dtype=uint8)
```

كل قيمة عددية في هذا التنسيق المسطح ذات قيمة ألوان RGB تتراوح بين 0 و255، وفي الدرس السابق، تمّ توضيح أن التحجيم القياسي والتسوية يؤديان أحيانًا إلى تحسين نتائج بعض خوارزميات التعلُّم الآلي.

يمكن استخدام المقطع البرمجي التالي لتسوية القيم وجعلها ما بين 0 و1:

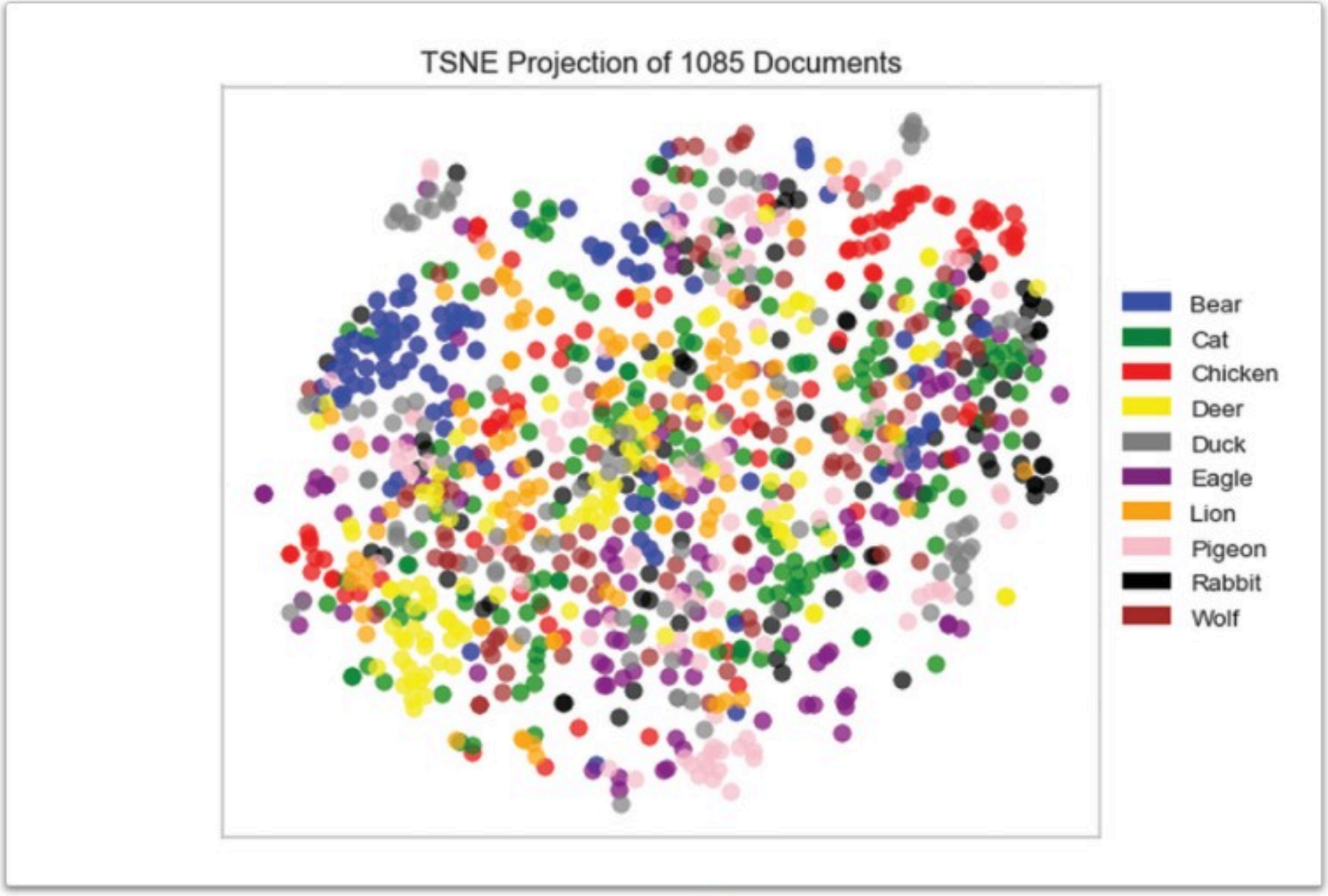
```
X_norm = X_flat / 255
X_norm[0]
```

```
array([0.41960784, 0.57254902, 0.4       , ..., 0.35686275, 0.3372549 ,
       0.42352941])
```

يمكن الآن تصوير البيانات بصريًا باستخدام أداة TSNEVisualizer المألوفة من مكتبة yellowbrick، وتم استخدام هذه الأداة أيضًا في الدرس الثاني من الوحدة الثالثة؛ لتصوير العناقيد بصريًا في البيانات النصيَّة.

```
%%capture
!pip install yellowbrick
from yellowbrick.text import TSNEVisualizer
```

```
tsne = TSNEVisualizer(colors = color_palette) # initializes the tool
tsne.fit(X_norm, y) # uses TSNE to reduce the data to 2 dimensions
tsne.show();
```



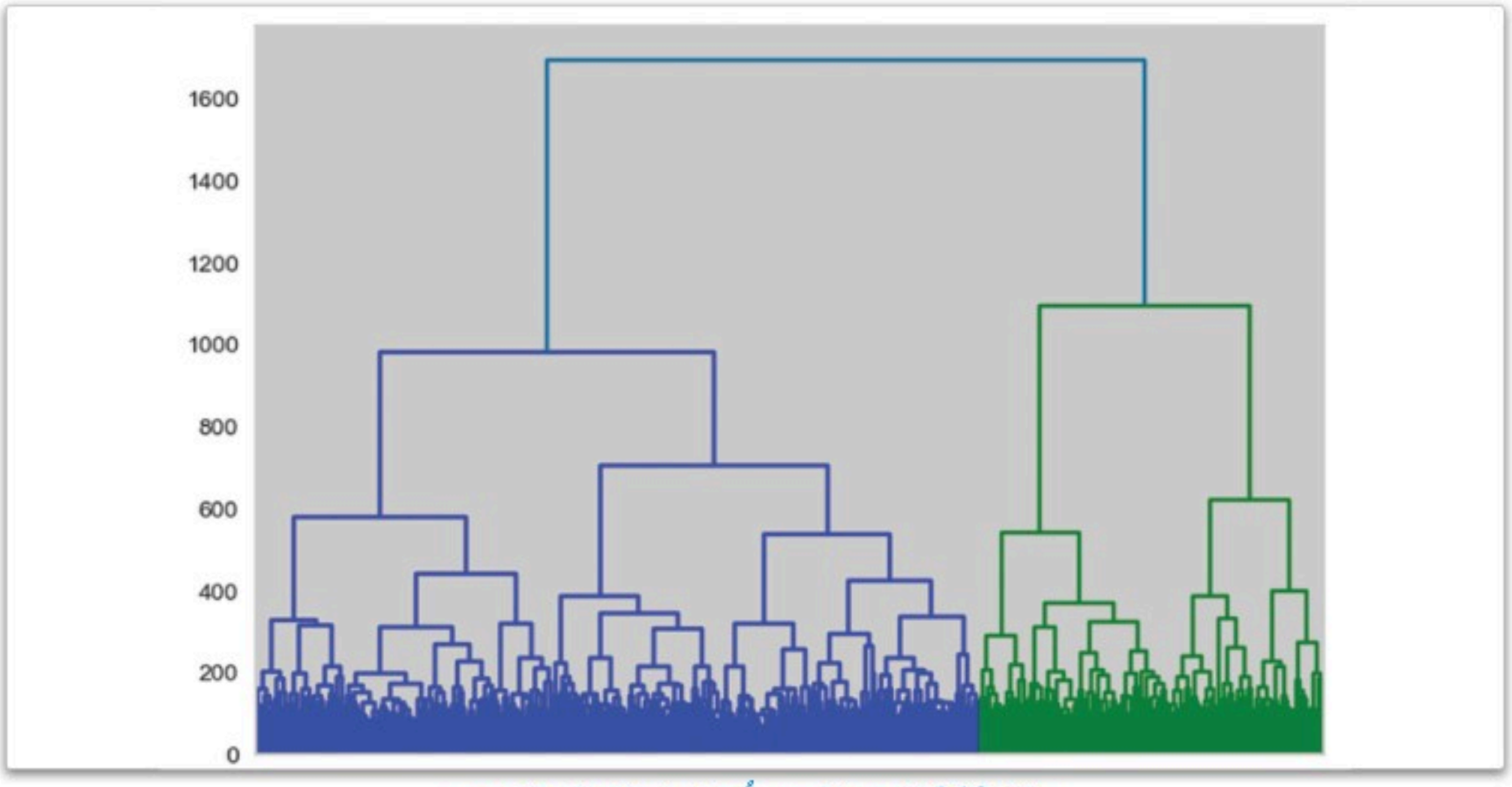
شكل 4.18: تصوير العناقيد

التصوير التمهيدي هذا ليس كما هو متوقَّع، فيبدو أن فئات الحيوانات المُختلفة مختلطة ببعضها، دون تمييز واضح بينها ودون عناقيد واضحة لها، ويدل ذلك على أن مجرد القيام بتسطيح بيانات الصورة الأصلية من المحتمل ألا يؤدي إلى نتائج ذات جودة عالية.

بعد ذلك، ستُستخدم خوارزمية **التجميع التكتلي** (Agglomerative Clustering) نفسها التي استُخدمت في الدرس الثاني من الوحدة الثالثة؛ لتجميع البيانات في متغيِّر X_norm، ويستورد المقطع البرمجي التالي مجموعة الأدوات المطلوبة، ويصوِّر الرسم الشجري لمجموعة البيانات:

```
from sklearn.cluster import AgglomerativeClustering # used for agglomerative clustering
import scipy.cluster.hierarchy as hierarchy

hierarchy.set_link_color_palette(color_palette) # sets the color palette
plt.figure()

# iteratively merges points and clusters until all points belong to a single cluster
linkage_flat = hierarchy.linkage(X_norm, method = 'ward')
hierarchy.dendrogram(linkage_flat)
plt.show()
```

ward (وارد) عبارة عن طريقة ربط تُستخدم في التجميع التكتلي الهرمي.



شكل 4.19: الرسم الشجري يُصنف البيانات إلى عنقودين

يكشف الرسم الشجري عنقودين كبيرين يمكن تقسيمهما إلى عناقيد أصغر، ويُستخدم المقطع البرمجي التالي أداة AgglomerativeClustering (التجميع التكتلي)؛ لإنشاء عشرة عناقيد، وهو العدد الفعلي للعناقيد الموجودة في البيانات:

```
AC = AgglomerativeClustering(linkage = 'ward',n_clusters = 10)
AC.fit(X_norm) # applies the tool to the data

pred = AC.labels_ # gets the cluster labels

pred
```

```
array([9, 6, 3, ..., 4, 4, 3], dtype=int64)
```

وأخيرًا، تُستخدم مؤشرات: Homogeneity (التجانس)، وCompleteness (الاكتمال)، وAdjusted Rand (راند المُعدَّل) وكلها تعرّفت عليها في الدرس الثاني من الوحدة الثالثة؛ لتقييم جودة العناقيد الناتجة.

```
from sklearn.metrics import homogeneity_score, adjusted_rand_score,
completeness_score

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.09868725008128477

Adjusted Rand score: 0.038254515908926826

Completeness score: 0.101897123096584
```

كما سبق توضيحه بالتفصيل في الدرس الثاني من الوحدة الثالثة، فإن مؤشري التجانس والاكتمال يأخذان قيمًا بين 0 و1، وترتفع قيمة مؤشر التجانس إلى أقصى حد عندما يكون لجميع نقاط العنقود الواحد العنوان الحقيقي الأساسي نفسه، كما ترتفع قيمة مؤشر الاكتمال إلى الحد الأقصى عندما تنتمي جميع نقاط البيانات التي تحمل العنوان الحقيقي الأساسي نفسه إلى العنقود نفسه، وأخيرًا يأخذ مؤشر راند المُعدَّل قيمًا بين 0.5- و1.0، وترتفع إلى الحد الأقصى عندما تكون جميع نقاط البيانات التي لها العنوان نفسه في العنقود نفسه، وتكون جميع النقاط ذات العناوين المُختلفة في عناقيد متباينة، وكما هو متوقَّع تفشل الخوارزمية بعد تصوير البيانات في العثور على عناقيد عالية الجودة تتطابق مع فئات الحيوانات الفعلية، حيث أن قيم المؤشرات الثلاث منخفضة للغاية، وعلى الرغم من أن مجرد القيام بتسطيح البيانات كان كافيًا للحصول على نتائج معقولة لتصنيف الصور، إلا أن تجميع الصور في عناقيد يُمثِّل مشكلة أكثر صعوبة.

التجميع بانتقاء الخصائص Clustering with Feature Selection

في الدرس السابق تم توضيح أنّ استخدام تحويل **المُخطَّط التكراري للتدرجات الموجَّهة** (HOG) لتحويل بيانات الصور إلى صيغة أكثر دلالة يؤدي إلى إنجاز أعلى بشكل ملحوظ في تصنيف الصور، وسيُطبَّق التحويل نفسه لاختبار ما إذا كان بإمكانه أيضًا تحسين نتائج مهام تجميع الصور.

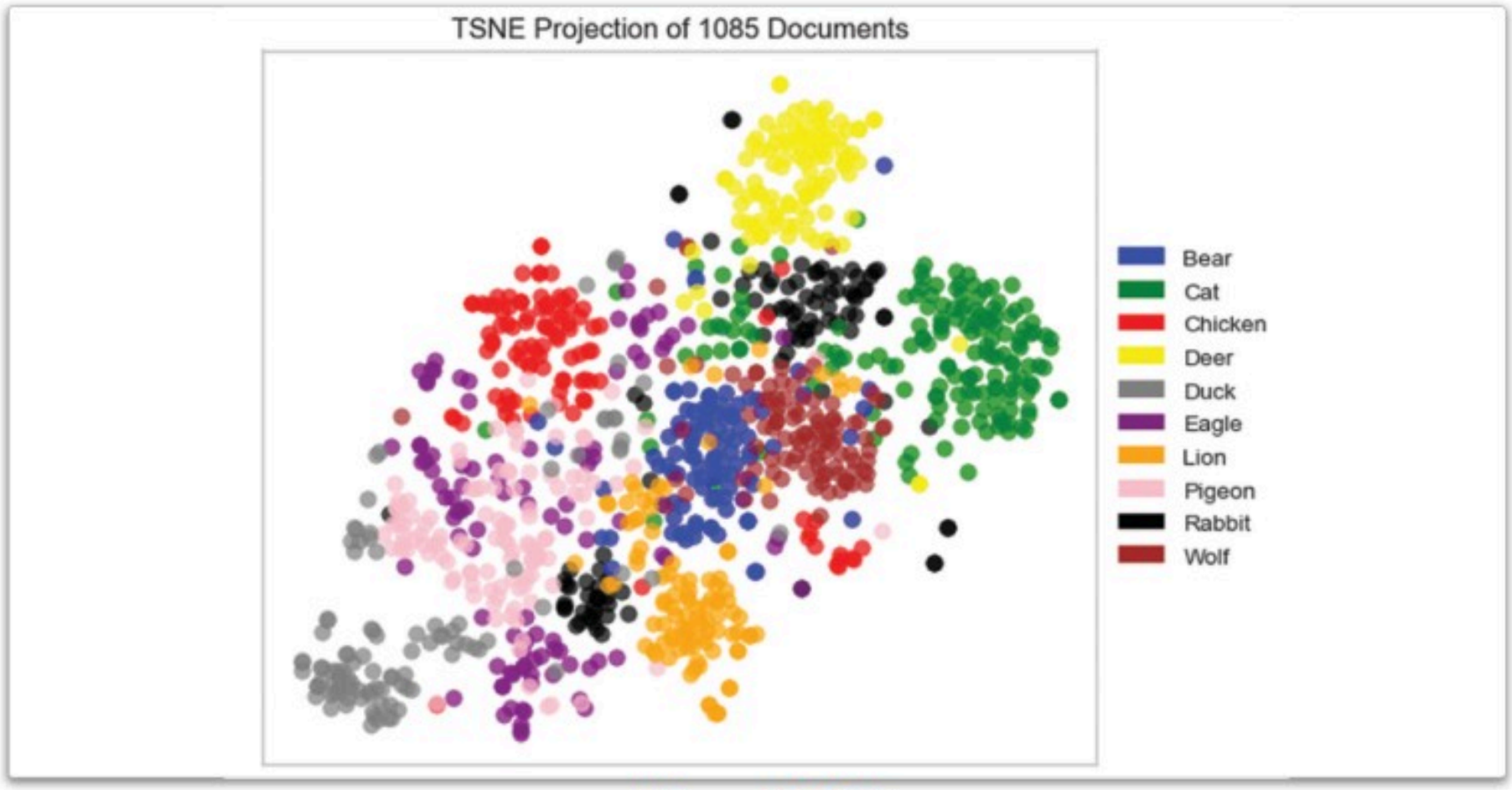
```
from skimage.color import rgb2gray
from skimage.feature import hog
# converts the list of resized images to an array of grayscale images
X_gray = np.array([rgb2gray(img) for img in resized_images])
# computes the HOG features for each grayscale image in the array
X_hog = np.array([hog(img) for img in X_gray])
X_hog.shape
```

```
(1085, 54756)
```

يكشف شكل البيانات المحوَّلة أن كل صورة تُمثَّل الآن على هيئة متَّجَه بقيمة عددية هي: أربعة وخمسون ألفًا وسبعمئة وستة وخمسون (54,756).

يستخدِم المقطع البرمجي التالي أداة TSNEVisualizer لتصوير هذا التنسيق الجديد:

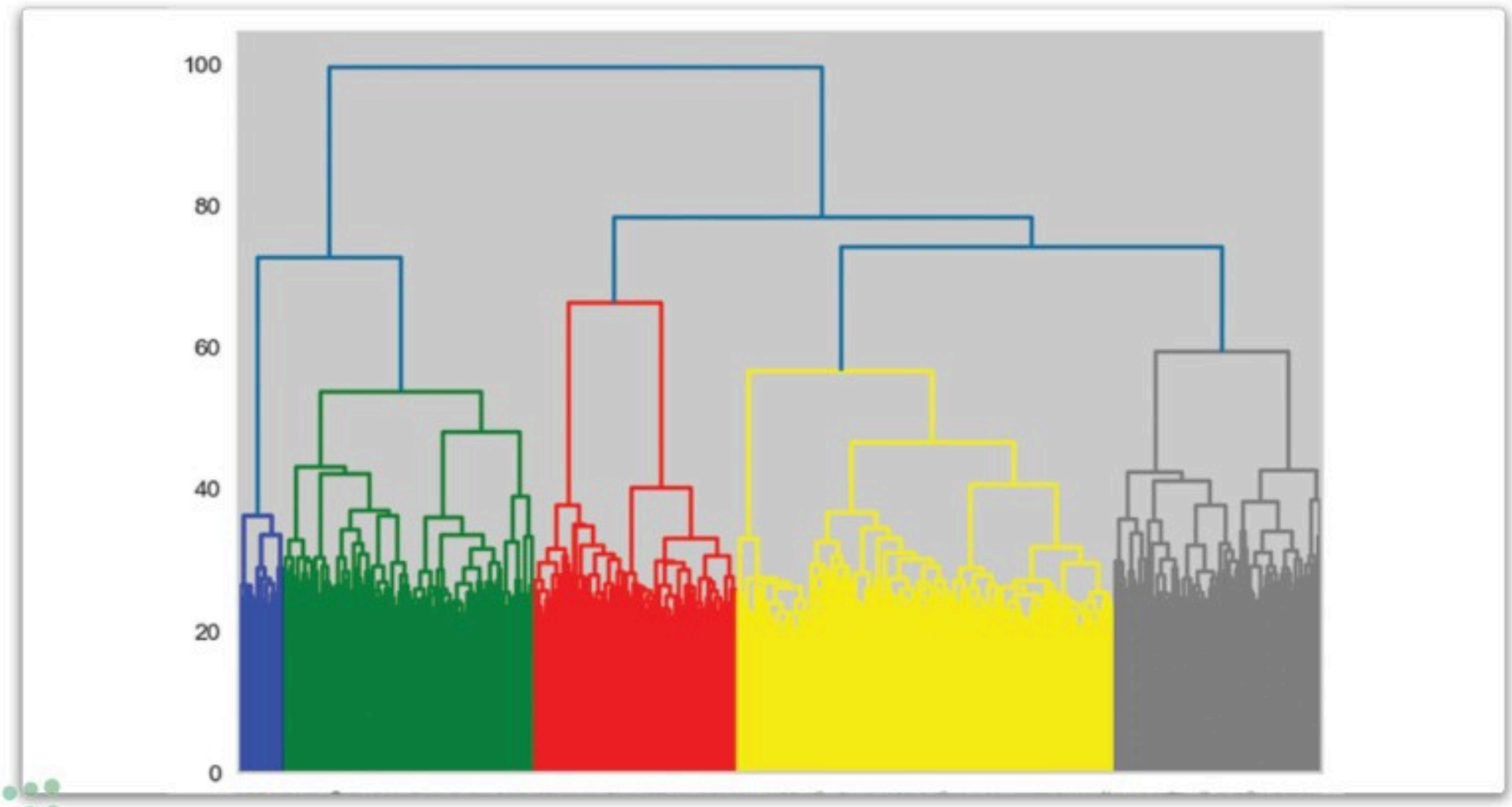
```
tsne = TSNEVisualizer(colors = color_palette)
tsne.fit(X_hog, y)
tsne.show();
```



شكل 4.20: تصوير العناقيد

يُعدُّ هذا التصوير أكثر مصداقية من الذي تم إنتاجه للبيانات غير المحوَّلة، وعلى الرغم من وجود بعض الشوائب، فإن الشكل يُظهر عناقيد واضحة ومفصولة جيدًا، ويمكن الآن حساب الرسم الشجري لمجموعة البيانات هذه.

```
plt.figure()
linkage_2 = hierarchy.linkage(X_hog,method = 'ward')
hierarchy.dendrogram(linkage_2)
plt.show()
```



شكل 4.21: الرسم الشجري لفئات وجوه الحيوانات المُختلفة باستخدام مُخطَّط تكراري للتدرجات الموجَّهة (HOG)

يقترح الرسم الشجري خمسة عناقيد، وهو بالضبط نصف العدد الصحيح البالغ عشرة عناقيد. يتبنى المقطع البرمجي التالي هذا الاقتراح ويطبِّق أداة AgglomerativeClustering (التجميع التكتلي) ويُظهر نتائج المؤشرات الثلاثة:

```
AC = AgglomerativeClustering(linkage = 'ward', n_clusters = 5)
AC.fit(X_hog)
pred = AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.4046340612330986

Adjusted Rand score: 0.29990205334627734

Completeness score: 0.6306921317302154
```

تكشف النتائج أنه على الرغم من أن عدد العناقيد التي تم استخدامها كان أقل بكثير من العدد الصحيح، إلا أن النتائج أفضل بكثير من النتائج التي ظهرت عند استخدام الرقم الصحيح على البيانات غير المحوَّلة.

ويوضِّح ذلك ذكاء التحويل بواسطة المُخطَّط التكراري للتدرجات الموجَّهة، ويُثبت أنه يمكن أن يؤدي إلى تحسينات رائعة في الأداء لكل من مهام التعلُّم الموجَّه ومهام التعلُّم غير الموجَّه في رؤية الحاسب، ولإكمال التحليل يُعيد المقطع البرمجي التالي تجميع البيانات المحوَّلة بالعدد الصحيح للعناقيد:

```
AC = AgglomerativeClustering(linkage = 'ward', n_clusters = 10)
AC.fit(X_hog)
pred = AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.5720932612704411

Adjusted Rand score: 0.41243540297103065

Completeness score: 0.617016965322667
```

وكما هو متوقَّع، زادت قيم المؤشرات بشكل عام، فعلى سبيل المثال تجاوز كل من التجانس والاكتمال الآن 0.55، مما يدل على أن الخوارزمية تقوم بعمل أفضل فيما يتعلق بكل من: وضع الحيوانات التي تنتمي لفئة واحدة في العنقود نفسه، وإنشاء عناقيد نقية (Pure) تتكون في الغالب من فئة الحيوان نفسِه.

التجميع باستخدام الشبكات العصبية Clustering Using Neural Networks

أحدث استخدام نماذج التعلُّم العميق (الشبكات العصبية العميقة ذات الطبقات المتعددة) ثورة في مجال تجميع الصور من خلال توفير خوارزميات قوية وعالية الدقة، ويمكنها تجميع الصور المتشابهة معًا تلقائيًا دون الحاجة إلى هندسة الخصائص. تعتمد العديد من الطرائق التقليدية لتجميع الصور على خاصية **المستخرِجات** (Extractors) لاستخراج معلومات ذات مغزى من صورة ما، واستخدام هذه المعلومات لتجميع الصور المتشابهة معًا، ويمكن أن تستغرق هذه العملية وقتًا طويلًا وتتطلب خبرة في المجال لتصميم خاصية المستخرِجات بخصائص فعّالة. بالإضافة إلى ذلك -وكما تم التوضيح في الدرس السابق- على الرغم من أن خاصية **الواصِفات** (Descriptors) مثل: تحويل المُخطَّط التكراري للتدرجات الموجَّهة يمكنها بالفعل تحسين النتائج، إلا أنها بعيدة كل البُعد عن الكمال، وبالتأكيد يوجد مجال للتحسين. من ناحية أخرى، يتمتع التعلُّم العميق بالقدرة على تعلُّم تمثيلات الخصائص من البيانات الخام تلقائيًا، ويتيح ذلك لطرائق التعلُّم العميق معرفة الخصائص شديدة التمايز التي تلتقط الأنماط الهامة وراء البيانات، مما يؤدي إلى تجميع أكثر دقة وقوة، ولتحقيق ذلك تُستخدم عدة طبقات مُختلفة في الشبكة العصبية بما فيها:

- **الطبقات الكثيفة** (Dense Layers)
- **طبقات التجميع** (Pooling Layers)
- **طبقات الإقصاء** (Dropout Layers)

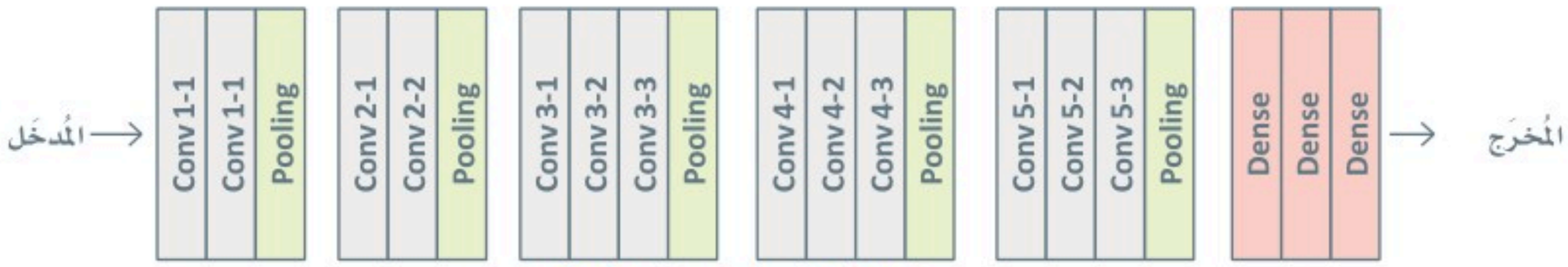
في الشبكة العصبية في الدرس الأول من الوحدة الثالثة، تم استخدام طبقة مخفية مكونة من ثلاث مئة خلية عصبية من نموذج **الكلمة إلى المتَّجَه** (Word2Vec)؛ لتمثيل كل كلمة، وفي تلك الحالة دُرِّب نموذج الكلمة إلى المتَّجَه مسبقًا على مجموعة بيانات كبيرة جدًا تحتوي على ملايين الأخبار من **أخبار قوقل** (Google News). تُعدُّ نماذج الشبكات العصبية المدرَّبة مسبقًا شائعة أيضًا في مجال رؤية الحاسب، ومن الأمثلة المعهودة على ذلك نموذج VGG16 الذي يشيع استخدامه في مهام التعرّف على الصور، ويتبع نموذج VGG16 معمارية عميقة قائمة على الشبكات العصبية الترشيحية يوجد بها ست عشرة طبقة، ويُعدُّ نموذجًا موجَّهًا دُرِّب على مجموعة بيانات كبيرة من الصور المُعَنوَنة تسمى **شبكة الصور** (ImageNet)، ومع ذلك، تتكون مجموعة بيانات التدريب الخاصة بنموذج VGG16 من ملايين الصور ومئات العناوين المُختلفة، مما يحسِّن بشكل كبير من قدرة النموذج على فهم الأجزاء المُختلفة من الصورة، وعلى غرار الشبكة العصبية الترشيحية البسيطة الموضَّحة في الشكل 4.22، ويستخدِم نموذج VGG16 أيضًا طبقة كثيفة نهائية تحتوي على أربعة آلاف وستة وتسعين خلية عصبية لتمثيل كل صورة قبل إدخالها في **طبقة المُخرَج** (Output Layer)، ويوضِّح هذا القسم كيف يمكن تكييف نموذج VGG16 لتجميع الصور، على الرغم من أنه صُمِّم في الأصل لتصنيف الصور:

1. حمِّل النموذج VGG16 الذي دُرِّب مسبقًا.
2. احذف طبقة المُخرَج من النموذج، فذلك يجعل الطبقة الأخيرة الكثيفة هي طبقة المُخرَج الجديدة.
3. استخدم **النموذج المقتطع** (Truncated Model) - النموذج السابق الذي اقتُطعت الطبقة الأخيرة منه-؛ لتحويل كل صورة في مجموعة بيانات Animal Faces (وجوه الحيوانات) إلى متَّجَه عددي له أربعُ آلاف وستٌ وتسعون قيمة.
4. استخدم التجميع التكتلي؛ لتجميع المتَّجَهات الناتجة عن ذلك.

الطبقة الكثيفة (Dense Layer):
هي طبقة في الشبكات العصبية ترتبط فيها كل العُقد التي في الطبقة السابقة بكل العُقد التي في الطبقة الحالية، حيث يتم تمرير الإشارات من العُقد في الطبقة السابقة في الشبكة إلى العُقد في الطبقة الحالية بواسطة وزنية محدَّدة، وتُطبَّق **دالَّة التنشيط** (Activation Function) على الإشارات المرسَلة إلى الطبقة الكثيفة لتوليد نتائج الإخراج النهائية.

طبقة التجميع (Pooling Layer):
هي طبقة في الشبكات العصبية تُستخدم لتقليل الأبعاد الفراغية لبيانات المُدخَلات.

طبقة الإقصاء (Dropout Layer):
هي طريقة تنظيم تُستخدم لمنع فرط التخصيص في نموذج لمجموعة بيانات في الشبكات العصبية عن طريق إقصاء عُقد موجودة في الطبقة خلال كلّ دورة تدريب.



شكل 4.22: معمارية نموذج VGG16

يمكن استخدام مكتبة TensorFlow ومكتبة Keras اللتين تعرّفت عليهما في الدرس السابق للوصول إلى نموذج VGG16 واقتطاعه، وتتمثّل الخطوة الأولى في استيراد جميع الأدوات المطلوبة:

```
from keras.applications.vgg16 import VGG16  # used to access the pre-trained VGG16 model
from keras.models import Model

model = VGG16() # loads the pretrained VGG16 model
# removes the output layer
model = Model(inputs = model.inputs, outputs = model.layers[-2].output)
```

يحذف الطبقة الأخيرة من المُخرَج.

يطبِّق المقطع البرمجي التالي المعالجة الأولية الأساسية نفسها التي يتطلبها نموذج VGG16 مثل: تحجيم قيم ألوان RGB لتكون بين 0 و1:

```
from keras.applications.vgg16 import preprocess_input
X_prep = preprocess_input(X)
X_prep.shape
```

```
(1085, 224, 224, 3)
```

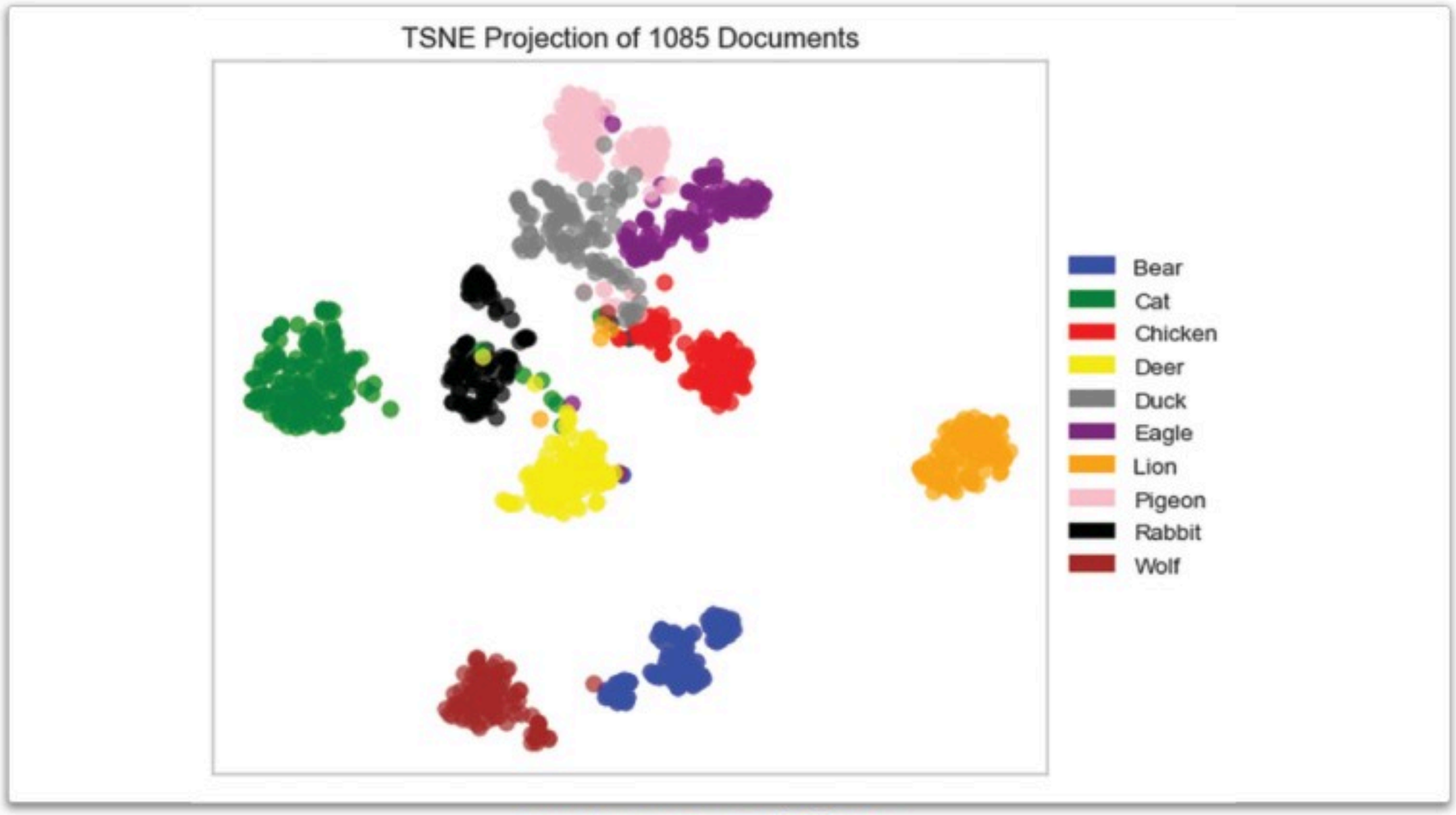
لاحظ أن شكل البيانات يظل كما هو، أي: ألفٌ وخمسٌ وثمانون صورة، كل صورة منها أبعادها 224 × 224، وثلاث قنوات ألوان RGB، وبعد ذلك يمكن استخدام النموذج المقتطع لتحويل كل صورة إلى متَّجه مكّون من 4,096 عدد.

```
X_VGG16 = model.predict(X_prep, use_multiprocessing = True)
X_VGG16.shape
```

```
34/34 [==============================] - 57s 2s/step

(1085, 4096)
```

يُضبط متغيِّر المعالجة المتعددة multiprocessing=True (تفعيل المعالجة المتعددة) لتسريع العملية من خلال حساب المتَّجهات للصور المتعددة بالتوازي، وقبل إكمال خطوة التجميع يُستخدم المقطع البرمجي التالي لتصوير **البيانات المتَّجِهة** (Vectorized Data):

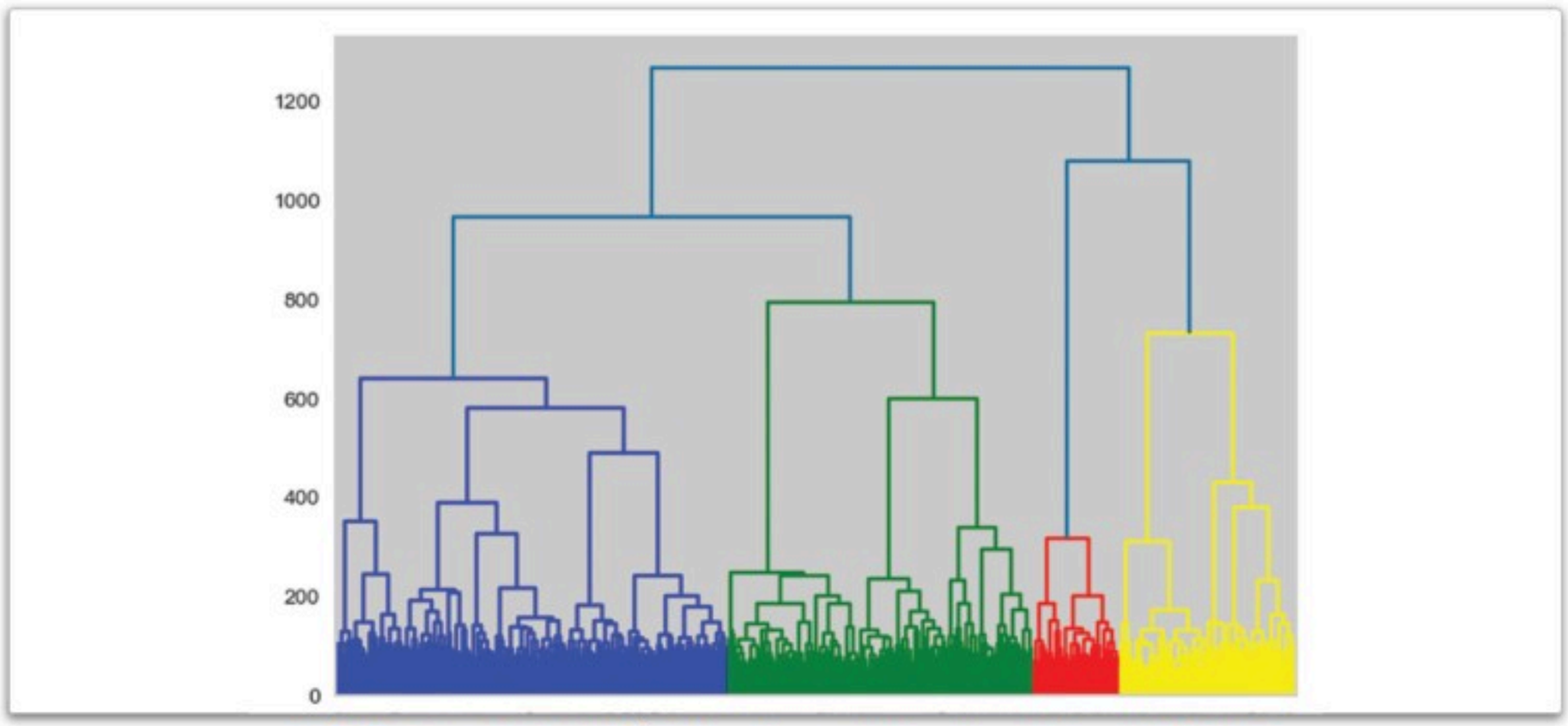
```
tsne = TSNEVisualizer(colors = color_palette)
tsne.fit(X_VGG16, labels)
tsne.show();
```



شكل 4.23: تصوير العناقيد المتشابهه

تُعدُّ النتائج مذهلة؛ لأن التصوير الجديد يكشف عناقيد مفصولة عن بعضها بوضوح وتكاد تكون كاملة، كما أن الفصل هنا أفضل بكثير من الفصل الذي كان في البيانات التي حُوِّلت بواسطة المُخطَّط التكراري للتدرجات الموجَّهة.

```
linkage_3 = hierarchy.linkage(X_VGG16, method = 'ward')
plt.figure()
hierarchy.dendrogram(linkage_3)
plt.show()
```



شكل 4.24: الرسم الشجري الهرمي لفئات وجوه الحيوانات المُختلفة باستخدام نموذج VGG16

يقترح الرسم الشجري أربعة عناقيد، وفي هذه الحالة يمكن للممارس أن يتجاهل الاقتراح بسهولة، ويتّبع التصوير السابق بدلًا منه والذي يبيّن بوضوح وجود عشرة عناقيد.

يستخدِم المقطع البرمجي التالي التجميع التكتلي ويوضِّح قيم المؤشرات لكل من العناقيد الأربعة والعناقيد العشرة:

```
AC = AgglomerativeClustering(linkage = 'ward',n_clusters = 4)
AC.fit(X_VGG16)
pred=AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.504687456015823

Adjusted Rand score: 0.37265351562538257

Completeness score: 0.9193141240200559
```

```
AC = AgglomerativeClustering(linkage='ward',n_clusters = 10)
AC.fit(X_VGG16)
pred=AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.8403973102506642

Adjusted Rand score: 0.766734821176714

Completeness score: 0.8509145102288217
```

تثبت النتائج صحة الأدلة التي قدمها التصوير، وتؤدي التحولات التي أنتجها نموذج VGG16 إلى نتائج مذهلة إلى حد كبير لكل من العناقيد الأربعة والعناقيد العشرة. في الواقع، ظهرت نتائج شبه مثالية لجميع المؤشرات الثلاثة عند استخدام عشرة عناقيد، مما يثبت أن النتائج غالبًا تتوافق تمامًا مع فئات الحيوانات في مجموعة البيانات.

يُعدُّ نموذج VGG16 من أقدم نماذج الشبكات العصبية الترشيحية عالية الذكاء المدرَّبة مسبقًا لغرض استخدامها في تطبيقات رؤية الحاسب، ومع ذلك نُشرت العديد من نماذج الشبكات العصبية الترشيحية الذكية الأخرى المدرَّبة مسبقًا والتي تجاوز أداؤها أداء نموذج VGG16.

تمرينات

1 اذكر الميزة التي تتمتع بها تقنيات التعلُّم غير الموجَّه مقارنة بتقنيات التعلُّم الموجَّه في تحليل الصور.

2 لديك مصفوفة قيم موحدة X_flat تشمل صورًا مُسطحة، وكل صف في المصفوفة يمثِّل صورة مسطحة مُختلفة على هيئة متتالية من الأعداد الصحيحة تتراوح بين 0 و255. أكمل المقطع البرمجي التالي، بحيث يستخدِم التجميع التكتلي في تصنيف الصور التي من X_flat إلى خمسة عناقيد مُختلفة:

```
from _____________ import AgglomerativeClustering # used for agglomerative clustering

AC = AgglomerativeClustering(linkage='ward', _____________)

X_norm = _________________ # normalizes the data

AC.fit(X_norm) # applies the tool to the data

pred = AC._________________ # gets the cluster labels
```

3 عدِّد بعض مزايا استخدام التعلُّم العميق التي يمتاز بها على طرائق تجميع الصور التقليدية.

4 لديك مصفوفة قيم موحدة X_flat تشمل صورًا مسطحة، وكل صف في المصفوفة يمثِّل صورة مسطحة مُختلفة على هيئة متتالية من الأعداد الصحيحةتتراوح بين 0 و255. أكمل المقطع البرمجي التالي، بحيث يستخدِم طريقة وارد (ward) لإنشاء وتصوير رسم شجري للصور في هذه المصفوفة:

```
import scipy.cluster.hierarchy as hierarchy # visualizes and supports hierarchical clustering tasks

import ________________ as plt

X_norm = ________________ # normalizes the data

plt.figure() # creates a new empty figure

linkage_flat=hierarchy.linkage(_______________, method='_______________')

hierarchy._______________(linkage_flat)

plt.show() #shows the figure
```

5 صِف الطريقة التي يُطبَّق بها التجميع بالشبكات العصبية في تحليل الصور.

رابط الدرس الرقمي

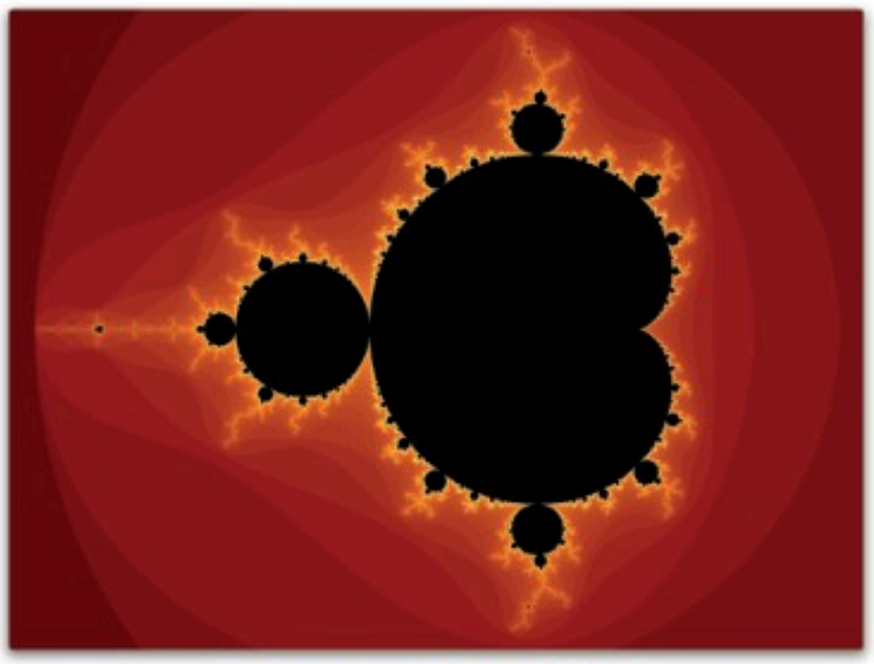
www.ien.edu.sa

الدرس الثالث

توليد البيانات المرئية

استخدام الذكاء الاصطناعي في توليد الصور Using AI to Generate Images

بينما ركزت خوارزميات رؤية الحاسب التي تم توضيحها في الدرسين السابقين من هذه الوحدة على فهم الجوانب المُختلفة لصورة معيّنة، يُركّز مجال **توليد الصور** (Image Generation) في هذا الدرس على إنشاء صور جديدة. فمجال **توليد الصور** (Image Generation) له تاريخ طويل يعود إلى الخمسينيات والستينيات من القرن العشرين، عندما بدأ الباحثون لأول مرة في إجراء تجارب على معادلات رياضية لإنشاء الصور، وفي عصرنا الحالي نما هذا المجال ليشمل مجموعة واسعة من التقنيات. يُعدُّ استخدام **الفراكتلات** (Fractals) من أقدم وأشهر تقنيات إنشاء الصور، والفراكتل هو شكل أو نمط هندسي مشابه لذاته، مما يعني أنه يبدو متشابهًا عند تكبيره بمقاييس مُختلفة، وأشهر فراكتل هو الذي يضم مجموعة **ماندلبروت** (Mandelbrot) الموضَّح في الشكل 4.25.



شكل 4.25: فراكتل ماندلبروت

في أواخر القرن العشرين، بدأ الباحثون في استكشاف أساليب أكثر تقدمًا لتوليد الصور مثل الشبكات العصبية.

يُعدُّ **إنشاء صورة من نصّ** (Text-to-Image Synthesis) من أكثر التقنيات شيوعًا لإنشاء الصور باستخدام الشبكات العصبية، وتتضمن هذه التقنية تدريب شبكة عصبية على توليد صور من أوصاف نصيَّة، فتُدرَّب الشبكة العصبية على مجموعة بيانات من الصور والأوصاف النصيَّة المرتبطة بها. وتتعلّم الشبكة ربط كلمات أو عبارات معيّنة بخصائص معيّنة للصورة مثل: شكل العنصر أو لونه، وبمجرد أن تُدرَّب الشبكة يصبح من الممكن استخدامها في إنشاء صور جديدة بناءً على الأوصاف الواردة في النص، وتُستخدم هذه التقنية في إنشاء مجموعة واسعة من الصور تتراوح ما بين العناصر البسيطة إلى المشاهد المعقدة.

وهناك تقنية أخرى لتوليد الصورة تتمثّل في **إنشاء صورة من صورة** (Image-to-Image Synthesis)، وتتضمن هذه التقنية تدريب شبكة عصبية على مجموعة بيانات من الصور؛ لتتعلّم التعرّف على الخصائص الفريدة للصورة حتّى تولّد صورًا جديدة مشابهة للصورة الموجودة، ولكن مع وجود اختلافات. في الآونة الأخيرة استكشف الباحثون **إنشاء صورة من صورة بالاسترشاد بنصّ** (Text-Guided Image-to-Image Synthesis)، مما يجمع بين نقاط القوة في طرائق إنشاء صورة من نصّ، وطرائق إنشاء صورة من صورة من خلال السماح للمستخدِم بتوجيه عملية الإنشاء باستخدام **توجيهات نصيَّة** (Text Prompts)، وتُستخدم هذه التقنية في توليد صور عالية الجودة تتوافق مع التوجيه النصيّ، وتكون في الوقت ذاته مشابهة بصريًا للصورة الطبيعية.

وأخيرًا، هناك تقنية أخرى من أحدث التقنيات في هذا المجال تتمثّل في **رسم صورة بالاسترشاد بنصّ** (Text-Guided Image-Inpainting)، ويُركِّز على ملء الأجزاء المفقودة أو التالفة من الصورة بناءً على وصف نصيٍّ معين، ويقدِّم الوصف النصيّ معلومات عن الشكل الذي يجب أن تبدو عليه الأجزاء المفقودة أو التالفة من الصورة، والهدف من خوارزمية الرسم هذه أن تُستخدم المعلومات؛ لإنشاء صورة واقعية ومترابطة. يقدِّم هذا الدرس أمثلة عملية على توليد الصور من خلال: إنشاء صورة من نصّ، وإنشاء صورة من صورة بالاسترشاد بنصّ، ورسم صور بالاسترشاد بنصّ.

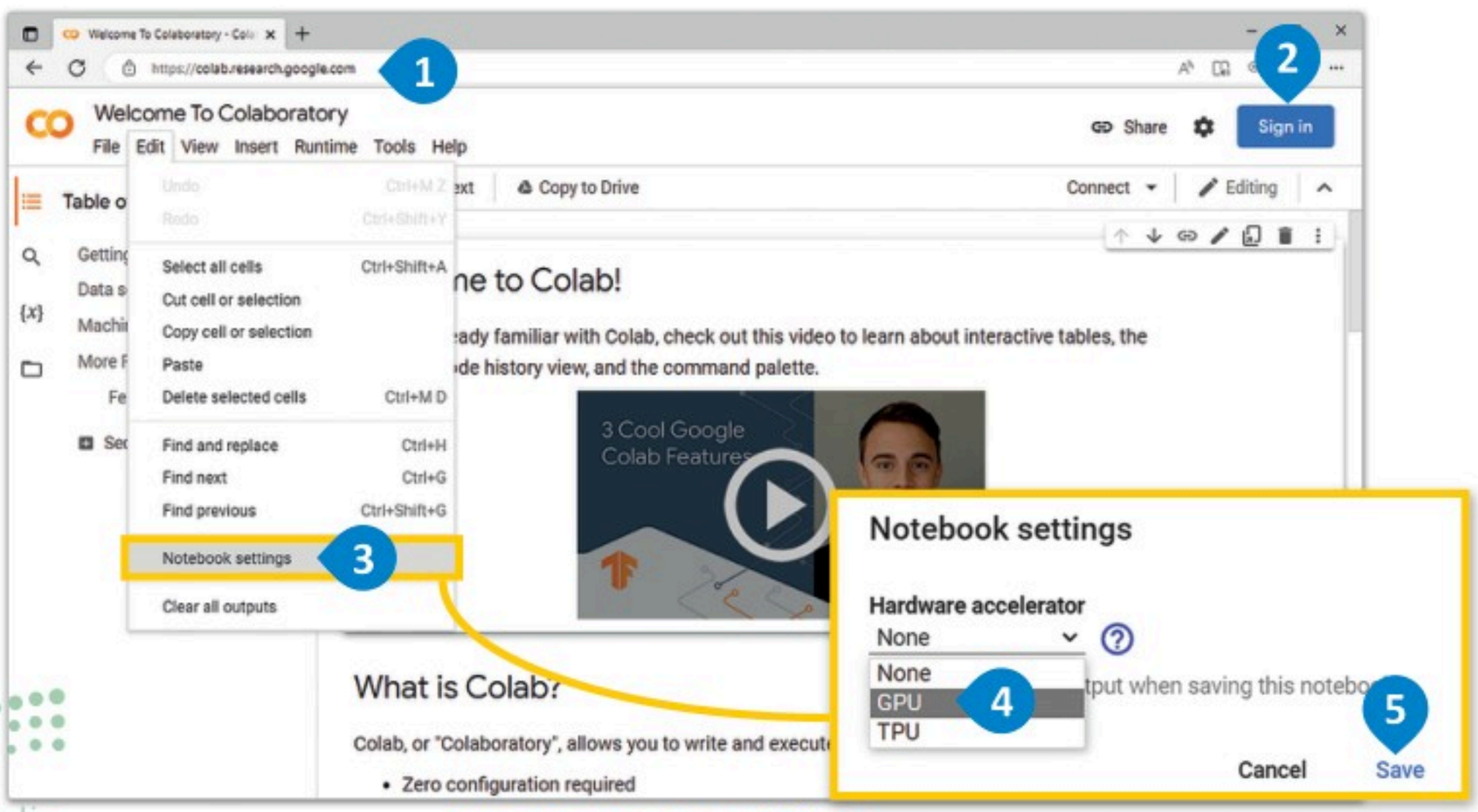
توليد الصور والموارد الحاسوبية
Image Generation and Computational Resources

وحدة معالجة الرسومات (Graphics Processing Unit - GPU):
هي نوع خاص من أنواع المعالِجات مصمَّم للتعامل مع كميات كبيرة من العمليات الحسابية المطلوبة لمعالجة الصور والفيديوهات.

إنشاء الصور مُهمَّة مكلِّفة من الناحية الحاسوبيّة؛ لأنها تتضمن استخدام خوارزميات معقَدة تتطلب قدرات عالية من قوة المعالجة، وعادةً تتضمن هذه الخوارزميات معالجة كميات كبيرة من البيانات مثل: نماذج ثلاثية الأبعاد، والنقوش، ومعلومات الإضاءة، مما يمكن أن يؤدي أيضًا إلى زيادة المتطلبات الحاسوبية للمُهِمَّة. يُعدُّ استخدام **وحدات معالجة الرسومات** (Graphics Processing Units – GPUs) أحد التقنيات الرئيسة التي تُستخدم لتسريع توليد الصور. وعلى عكس **وحدة المعالجة المركزية** (Central Processing Unit – CPU) التقليدية المُصمَّمة للتعامل مع مجموعة واسعة من المهام، تم تحسين وحدة معالجة الرسومات حتّى تتناسب مع أنواع العمليات الحسابية المطلوبة لمعالجة الصور والمهام الأخرى المتعلقة بالرسومات، مما يجعلها أكثر كفاءة في التعامل مع كميات كبيرة من البيانات وإجراء عمليات حسابية معقدة، ويُعدُّ هذا سببًا في استخدامها عادة في توليد الصور والمهام الأخرى المكلِّفة حاسوبيًّا. يوضِّح هذا الدرس كيف يمكنك استخدام منصة **قوقل كولاب** (Google Colab) الشهيرة للوصول إلى بُنية تحتية قوية قائمة على وحدة معالجة الرسومات دون أي تكلفة، وذلك باستخدام حساب عادي على قوقل، وقوقل كولاب هو منصة مجانية تعتمد على التقنية السحابية، وتتيح للمستخدِمين كتابة المقاطع البرمجية، وتنفيذها، وإجراء التجارب، وتدريب النماذج في بيئة **مفكرة جوبيتر** (Jupyter Notebook).

للوصول إلى منصة قوقل كولاب:

> اذهب إلى: https://colab.research.google.com. (1)
> سجِّل الدخول بحساب Google (قوقل) الخاص بك. (2)
> اضغط على Edit (تحرير)، ثم Notebook settings (إعدادات المفكرة). (3)
> اختر GPU (وحدة معالجة الرسومات)، (4) ثم اضغط على Save (حفظ). (5)



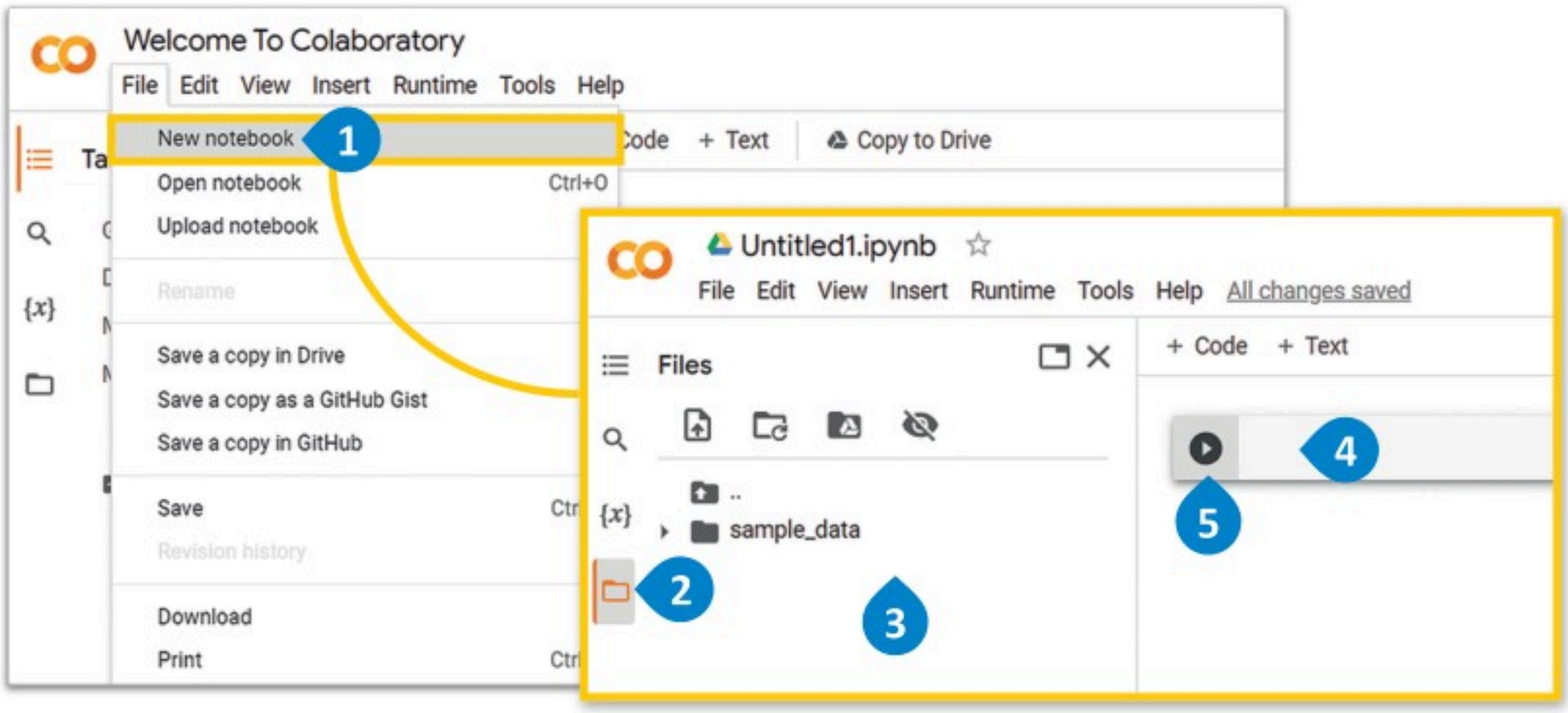
شكل 4.26: الوصول إلى منصة قوقل كولاب

لاستخدام مفكرة البايثون:

> اضغط على File (ملف)، ثم على New notebook (مفكرة جديدة). 1

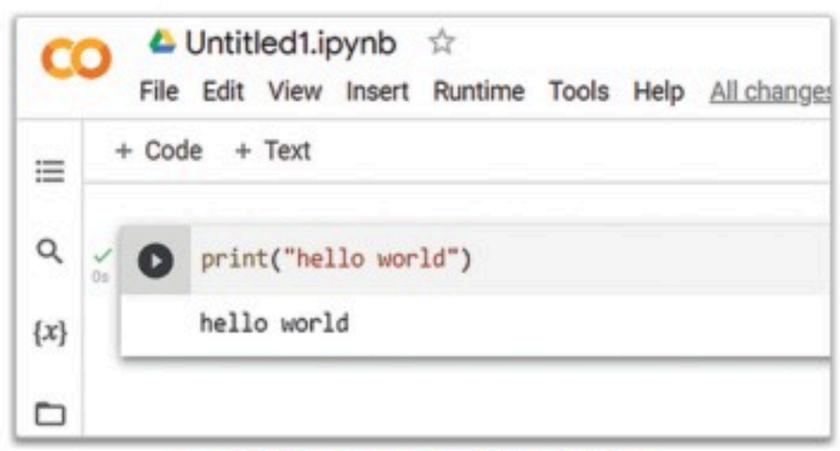
> اضغط على Files (ملفات)، 2 وفي المنطقة المجاورة التي ستظهر لك اسحب وأفلت images (الصور) التي ستستخدمها في الدرس. 3

> يمكنك الآن كتابة مقطعك البرمجي بلغة البايثون داخل خلية المقطع البرمجي، 4 ثم شغِّله من خلال الضغط على الزر الموجود بجانب خلية المقطع البرمجي. 5



تعمل بيئة قوقل كولاب بشكل مشابه لعمل مفكرة جوبيتر، وفيما يلي تجد مثال Hello World (مرحبًا بالعالَم) التقليدي:

خوارزميات توليد الصور (Image Generation) التي وصفناها في هذا الفصل مصممة بطريقة تجعلها إبداعيَة وبالتالي فهي ليست ثابتة، مما يعني أنه من غير المضمون أن تقوم دائما بتوليد الصورة نفسها للمُدخَلات نفسها. وعليه، فإن الصور المولَّدَة المدرجة في هذا الفصل مجرد أمثلة على الصور التي يمكن توليدها باستخدام المقطع البرمجي.



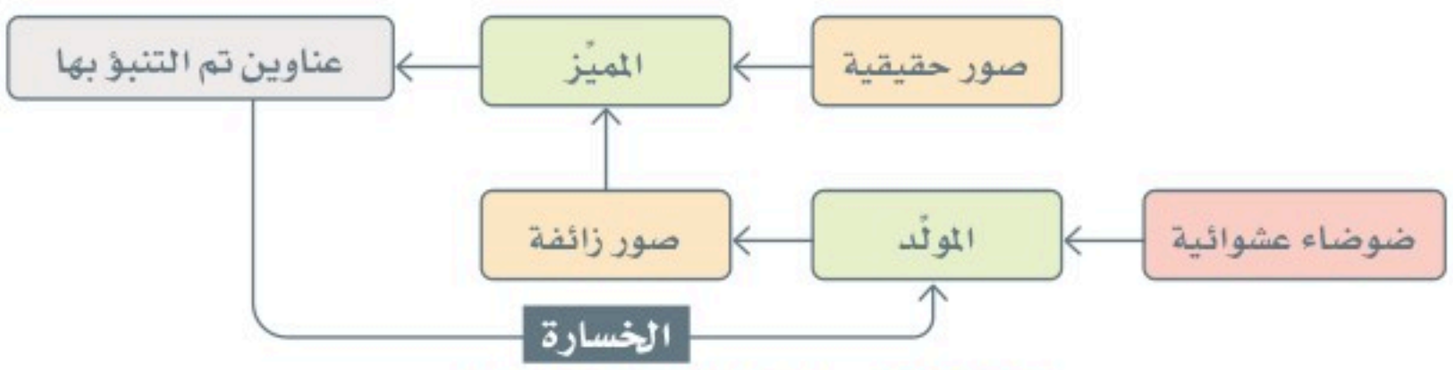
شكل 4.27: استخدام مفكرة البايثون

نماذج الانتشار والشبكة التوليدية التنافسية
Diffusion Models and Generative Adversarial Networks

في السنوات الأخيرة شهد مجال توليد الصور تُقدمًا كبيرًا مع تطوير أساليب ونماذج مُختلفة يمكنها توليد صور واقعية وعالية الجودة من مصادر مُختلفة للمعلومات، وهناك تقنيتان من أكثر التقنيات شيوعًا واستخدامًا على نطاق واسع لتوليد الصور هما: **الشبكة التوليدية التنافسية** (GANs)، ونموذج **الانتشار المستقر** (Stable Diffusion). ستتعرف في هذا القسم على المفاهيم والأساليب الرئيسة الخاصة بالشبكة التوليدية التنافسية ونموذج الانتشار المستقر، كما سيتم تقديم نظرة عامة على تطبيقاتها في توليد الصور، وسيتم مناقشة أوجه التشابه والاختلاف بينهما، ومزايا كل تقنية وعيوبها.

توليد الصور بالشبكة التوليدية التنافسية
Generating Images with Generative Adversarial Networks (GANs)

الشبكة التوليدية التنافسية هي فئة من النماذج التوليدية التي تتكون من مكونين رئيسين وهما: **المولِّد** (Generator) **والمميِّز** (Discriminator)، حيث يقوم المولِّد بتوليد صور زائفة، بينما يحاول المميِّز تمييز الصور المولَّدة من الصور الحقيقية، ويُدرَّب هذان المكوِّنان تدريبًا تنافسيًّا، إذ يحاول المولِّد أن "يخدع" المميِّز، ويحاول المميِّز أن يصبح أفضل في اكتشاف الصور الزائفة. تتمثَّل إحدى المزايا الرئيسة للشبكة التوليدية التنافسية في قدرتها على توليد صور عالية الجودة وواقعية يصعب تمييزها عن الصور الحقيقية، ولكن يوجد بها أيضًا بعض القيود مثل: **عدم التقارب** (Non-convergence) أو بعبارة أخرى، فشل شبكتي المولِّد والمميِّز في التحسن مع مرور الوقت، و**نقص التنوع** (Mode Collapse) في المُخرَجات، حيث ينتج النموذج نفس المُخرَجات المتشابهة مرارًا وتكرارًا بغض النظر عن المُدخَلات.



يُطبَّق المولِّد والمميِّز في الشبكة التوليدية التنافسية في العادة باستخدام الشبكات العصبية الترشيحية (CNNs) أو أي معمارية مشابهة.

شكل 4.28: معمارية الشبكة التوليدية التنافسية

توليد الصور بالانتشار المستقر Generating Images with Stable Diffusion

الانتشار المستقر هو نموذج تعلُّم عميق لتوليد صورة من نصّ، وتتكون هذه الطريقة من مكونين رئيسين: **مُرمِّز النصّ** (Text Encoder)، و**مفكِّك الترميز المرئي** (Visual Decoder). ويُدرَّب مُرمِّز النصّ ومفكِّك الترميز المرئي معًا على مجموعة بيانات مكونة من بيانات نصوص وبيانات صور مقترنة ببعضها؛ حيث يقترن كل مُدخَل نصّي بصورة مقابِلة أو أكثر. مُرمِّز النصّ هو شبكة عصبية تأخذ مُدخَلات نصيَّة مثل: جملة أو فقرة وتحوِّلها إلى **تضمين** (Embedding)، والتضمين هو متَّجَه عددي له عدد ثابت من القيم، ويلتقط تمثيل التضمين هذا معنى النصّ المُدخَل. يتم استخدام نهج مشابه في نموذج **الكلمة إلى المتَّجَه** (Word2Vec) ونموذج **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) اللذين تم توضيحهما في الوحدة الثالثة، حيث يولِّدان تضمينات للكلمات والجمل الفردية على الترتيب. ويُمرِّر بعد ذلك **تضمين النصّ** (Text Embedding) الذي أنشأه المُرمِّز عبر مفكِّك الترميز المرئي لتوليد صورة، ومفكِّك الترميز المرئي هو أيضًا نوع من الشبكات العصبية ويُنفذ عادةً باستخدام **شبكة عصبية ترشيحية** (CNN) أو معمارية مشابهة، وتُقارن الصورة المولَّدة بالصورة الحقيقية المقابلة الموجودة في مجموعة البيانات، ويُستخدم الفرق بينهما لحساب **الخسارة** (Loss)، ثم تُستخدم الخسارة لتحديث متغيِّرات مُرمِّز النصّ ومفكِّك الترميز المرئي؛ لتقليل الاختلاف بين الصور التي وُلِّدت والصور الحقيقية.

جدول 4.4: عملية تدريب الانتشار المستقر

1. مرِّر المُدخَلات النصيَّة عبر مُرمِّز النصّ للحصول على تضمين النصّ.
2. مرِّر تضمين النصّ عبر مفكِّك الترميز المرئي لتوليد صورة.
3. احسب الخسارة (الاختلاف) بين الصورة المولَّدة والصورة الحقيقية المقابلة لها الموجودة في مجموعة البيانات.
4. استخدِم الخسارة؛ لتحديث متغيِّرات مُرمِّز النصّ ومفكِّك الترميز المرئي، وعندما يكون المستوى عاليًا يتضمن ذلك **مكافأة** (Rewarding) الخلايا العصبية التي ساعدت على تقليل الخسارة و**معاقبة** (Punishing) الخلايا العصبية التي ساهمت في زيادتها.
5. كرِّر الخطوات المذكورة سابقًا مع أزواج متعددة من النصوص والصور في مجموعة البيانات.

حقَّق كلٌّ من نموذج الشبكة التوليدية التنافسية ونموذج الانتشار المستقر نتائج مبهرة في مجال توليد الصور، ويُركِّز الجزء المتبقي من هذا الدرس على تقديم أمثلة عملية بلغة البايثون على النهج **القائم على الانتشار** (Diffusion-Based) والذي يُعدُّ حاليًا أحدث ما توصلت إليه التقنية. كما تم التوضيح من قبل، يُعدُّ توليد الصور مُهمَّة مكلِّفة حاسوبيًّا، ولذلك نوصيك بشدة بأن تطبق جميع أمثلة البايثون على نظام قوقل كولاب الأساسي أو أي بنية أساسية مُختلفة تدعمها وحدة معالجة رسومات يكون لديك حق الوصول إليها.

يستخدِم هذا الفصل مكتبة diffusers التي تُعدُّ حاليًا أفضل مكتبة مفتوحة المصدر للنماذج القائمة على الانتشار، ويقوم المقطع البرمجي التالي بتثبيت المكتبة، وكذلك بعض المكتبات الإضافية المطلوبة:

```
%%capture
!pip install diffusers
!pip install transformers
!pip install accelerate

import matplotlib.pyplot as plt
from PIL import Image # used to represent images
```

توليد الصورة من نصّ Text-to-Image Generation

يوضِّح هذا القسم الطريقة التي يمكن بها استخدام مكتبة diffusers لتوليد صور تعتمد على التوجيه النصيّ الذي يقدمه المستخدِم، وتُستخدم الأمثلة الواردة في هذا القسم نموذج stable-diffusion-v1-4 (الانتشار-المستقر- الإصدار 1-4)، وهو نموذج شائع مُدرَّب مسبقًا لتوليد الصورة من نصّ.

```
# a tool used to generate images using stable diffusion
from diffusers import DiffusionPipeline
generator = DiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4")
# specifies what GPUs should be used for this generation
generator.to("cuda")

image = generator("A photo of a white lion in the jungle.").images[0]
plt.imshow(image);
```

يستجيب النموذج للتوجيه A photo of a white lion in the jungle (صورة أسد أبيض في الغابة) بصورة مبهرة وواقعية جدًا، كما هو موضَّح في الشكل 4.29، ويُعدُّ التجريب باستخدام التوجيهات الإبداعية هو أفضل طريقة لاكتساب الخبرة وفهم قدرات هذا النهج ونقاط ضعفه.

شكل 4.29: صورة مولَّدة لأسد أبيض في الغابة

معلومة

معمارية أجهزة الحاسب الموحد (Compute Unified Device Architecture - CUDA) هي منصة حوسبة موازية تتيح استخدام وحدات معالجة الرسومات (GPUs).

يضيف **التوجيه** (Prompt) التالي بُعدًا إضافيًّا لعملية التوليد، إذ يطلب أن يُرسم أسد أبيض بطريقة **بابلو بيكاسو** (Pablo Picasso)، وهو من أشهر الرسامين في القرن العشرين.

```
image = generator("A painting of a white lion in the style of Picasso.").
images[0]
plt.imshow(image);
```

شكل 4.30: صورة مولَّدة لأسد على نمط بيكاسو

ومرة أخرى، النتائج مبهرة وتُظهر الإبداع في عملية الانتشار المستقر، فالصورة الناتجة عن العملية هي في الواقع صورة أسد أبيض. ولكن على عكس التوجيه السابق، يؤدي التوجيه الجديد إلى صور تشبه الرسم بدلًا من أن تشبه الصور الفوتوغرافية، بالإضافة إلى ذلك، فإن أسلوب اللوحة يشبه بالفعل وبشكل ملحوظ أسلوب بابلو بيكاسو.

توليد صورة من صورة من خلال الاسترشاد بنصّ
Image-to-Image Generation with Text Guidance

يستخدِم المثال التالي مكتبة diffusers لتوليد صورة بناءً على مُدخَلين هما: صورة موجودة تعمل كأساس للصورة الجديدة التي سيتم إنشاؤها، وتوجيه نصيّ يصف الشكل الذي يجب أن تبدو عليه الصورة المنتَجة. بما أن مُهِمَّة تحويل النصّ إلى الصورة الموضَّحة في القسم السابق كانت محدودة فقط بتوجيه نصيّ، فيجب أن تضمن المُهِمَّة الجديدة أن تكون الصورة الجديدة مشابهة للصورة الأصلية، ومُمثِّلةً بشكلٍ دقيق للوصف الوارد في التوجيه النصيّ.

```
# pipeline used for image to image generation with stable diffusion
from diffusers import StableDiffusionImg2ImgPipeline
# loads a pretrained generator model
generator = StableDiffusionImg2ImgPipeline.from_pretrained("runwayml/stable-
diffusion-v1-5")
# moves the generator model to the GPU (CUDA) for faster processing
generator.to("cuda")

init_image = Image.open("landscape.jpg")
init_image.thumbnail((768, 768)) # resizes the image to prepare it as input of the model
plt.imshow(init_image);
```

المثال الموجود في الشكل 4.31 يستخدِم النموذج المدرَّب مسبقًا stable-diffusion-v1-4 المناسب لتوليد صورة من صورة من خلال التوجيه النصي.

شكل 4.31: صورة المنظر الطبيعي الأصلية

```
# a detailed prompt describing the desired visual
# for the produced image
prompt = "A realistic mountain
landscape with a large castle."
image = generator(prompt=prompt,
image = init_image, strength=0.75).
images[0]
plt.imshow(image);
```

شكل 4.32: صورة منظر طبيعي مولَّدة بقوة = 0.75

في الواقع، يولِّد النموذج صورة مستجيبةً للتوجيه النصيّ ومشابهة بصريًا للصورة الأصلية، ويُستخدم متغيِّر strength (القوة) للتحكم في الاختلاف البصري بين الصورة الأصلية والصورة الجديدة، ويتخذ المتغيِّر قيمًا بين 0 و1، وتسمح القيم الأعلى للنموذج بأن يكون أكثر مرونة وأقل تقيُّدًا بالصورة الأصلية. على سبيل المثال، يُستخدم المقطع البرمجي التالي لنفس prompt (التوجيه) من خلال ضبط المتغيِّر strength ليساوي 1.

```
# generate a new image based on the prompt and the
# initial image using the generator model
image = generator(prompt=prompt,
image = init_image, strength=1).images[0]
plt.imshow(image);
```

شكل 4.33: صورة منظر طبيعي مولَّدة بقوة = 1

تؤكد الصورة الناتجة في شكل 4.33 أن زيادة قيمة متغيِّر القوة تؤدي إلى شكل بصري أفضل بالإرشاد الوارد في التوجيه النصي، ولكنه أيضًا أقل تشابهًا إلى حد كبير مع الصورة المُدخَلة.

وهذا مثال نموذجي آخر، يتضح مُخرَجه في الشكل 4.34.

```
init_image = Image.open("cat_1.jpg")
init_image.thumbnail((768, 768))
plt.imshow(init_image);
```

شكل 4.34: صورة القطّة الأصلية

وسيُستخدم المقطع البرمجي التالي لتحويل هذه الصورة إلى صورة tiger (نمر):

```
prompt = "A photo of a tiger"
image = generator(prompt=prompt, image=init_image, strength=0.5).images[0]
plt.imshow(image);
```

تتقيد المحاولة الأولى بقيمة المتغيِّر strength، مما أدّى إلى صورة تبدو وكأنها مزيج بين النمر والقطّة الموجودة في الصورة الأصلية، كما هو موضَّح في الشكل 4.35، وتَدُل الصورة الجديدة على أن الخوارزمية لم تكن لديها القوة الكافية لتحويل وجه القطّة تحويلًا صحيحًا إلى وجه نمر، وتظل الخلفية مشابهة جدًا لخلفية الصورة الأصلية.

بعد ذلك، تتم زيادة المتغيِّر strength للسماح للنموذج بالابتعاد عن الصورة الأصلية والاقتراب أكثر من التوجيه النصيّ.

شكل 4.35: صورة نمر مولَّدة بقوة = 0.5

```
image = generator(prompt=prompt,
image = init_image, strength=0.75).
images[0]
plt.imshow(image);
```

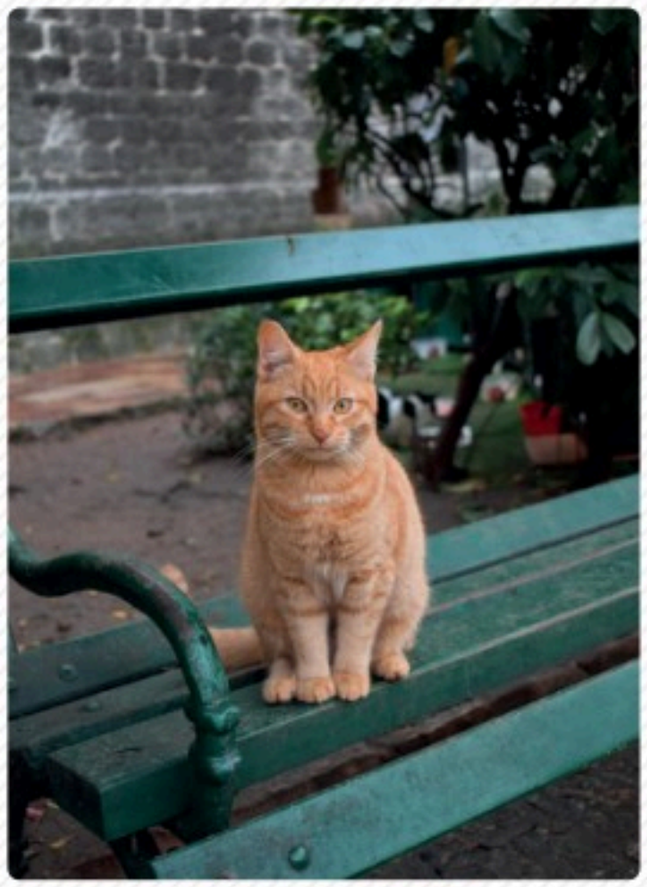
في الواقع، الصورة الجديدة المعروضة هي صورة نمر، ولكن لاحظ أن البيئة المحيطة بالحيوان ووضعية جلوسه وزواياه تظل شديدة الشبه بالصورة الأصلية، ويَدُل ذلك على أن النموذج ما زال واعيًا بالصورة الأصلية وحاولَ أن يحافظ على عناصر كان لا بد ألا تُغير؛ حتّى يقترب أكثر من التوجيه النصيّ.

شكل 4.36: صورة النمر مولَّدة بقوة = 0.75

رسم صورة بالاسترشاد بنصّ Text-Guided Image-Inpainting

يُركِّز المثال التالي على استخدام نموذج الانتشار المستقر لاستبدال شكل بصري جديد يصفه التوجيه النصيّ بأجزاء محددة من صورة معيّنة، ويُستخدم لهذا الغرض النموذج المدرَّب مسبقًا stable-diffusion-inpainting (رسم -الانتشار- المستقر)، ويقوم المقطع البرمجي التالي بتحميل صورة قطّة على مقعد، وهناك **قناع** (Mask) يعزل الأجزاء المحددة من الصورة التي تغطيها القطّة:

```
# tool used for text-guided image in-painting
from diffusers import StableDiffusionInpaintPipeline
init_image = Image.open("cat_on_bench.png").resize((512, 512))
plt.imshow(init_image);
mask_image = Image.open("cat_mask.jpg").resize((512, 512))
plt.imshow(mask_image);
```



شكل 4.37: صورة القطّة الأصلية

شكل 4.38: قناع صورة القطّة

القناع (Mask) هو صورة بسيطة بالأبيض والأسود لها نفس أبعاد الصورة الأصلية بالضبط، والأجزاء التي استُبدلت في الصورة الجديدة تُميز باللون الأبيض، في حين أن الأجزاء الأخرى من القناع سوداء. بعد ذلك، يتم تحميل النموذج المدرَّب مسبقًا، ويتم إنشاء prompt (التوجيه) لكي توضع صورة رائد الفضاء مكان القطة التي في الصورة الأصلية، كما يظهر في الشكل 4.39.

```
generator = StableDiffusionInpaintPipeline.from_pretrained("runwayml/stable-
diffusion-inpainting")
generator = generator.to("cuda")

prompt = "A photo of an astronaut"
image = generator(prompt=prompt, image=init_image, mask_image=mask_image).
images[0]
plt.imshow(image);
```

نجحت الصورة الجديدة في أن تظهر صورة واقعية للغاية لرائد الفضاء الذي وضعته مكان القطة التي كانت في الصورة الأصلية، كما يمتزج هذا الشكل البصري بسلاسة مع عناصر الخلفية والإضاءة في الصورة.

في الواقع، حتّى لو كان القناع أبسط وأقل دقة، يمكن إنتاج بديل واقعي. لاحظ صورة المُدخَل والقناع التاليين:

شكل 4.39: صورة رائد فضاء مولَّدة

```
init_image = Image.open("desk.jpg").resize((512, 512))
plt.imshow(init_image);
mask_image = Image.open("desk_mask.jpg").resize((512, 512))
plt.imshow(mask_image);
```

شكل 4.40: صورة المكتب الأصلية

شكل 4.41: قناع صورة المكتب

في هذا المثال، يغطي القناع جهاز الحاسب المحمول الموجود في وسط الصورة، ثم يُستخدم prompt (التوجيه) التالي والمقطع البرمجي ليتم وضع صورة الكتاب مكان جهاز الحاسب المحمول الموجود في الصورة الأصلية:

```
prompt = "A photo of a book"
image = generator(prompt=prompt, image=init_image, mask_image=mask_image).
images[0]
plt.imshow(image);
```

على الرغم من أن prompt (التوجيه) طلب إدخال كائن (كتاب) يختلف اختلافًا كبيرًا عن الكائن الذي استُبدل وهو (جهاز الحاسب المحمول)، فقد قام النموذج بعمل جيد في مزج الأشكال والألوان؛ لإنشاء شكل بصري دقيق، ومع التُقدم المستمر في تقنيات تعلُّم الآلة ورسومات الحاسب، من المحتمل أن تُنشِئَ صورًا أكثر إبهارا وأكثر واقعية في المستقبل.

شكل 4.42: صورة مكتب مولَّدة وعليها كتاب

تمرينات

1 صِف باختصار عملية رسم صورة بالاسترشاد بنصّ.

2 صِف عملية تدريب نماذج الانتشار المستقر.

3 صِف المولِّد والمميِّز في الشبكة التوليدية التنافسية.

4 استخدِم أداة DiffusionPipeline من مكتبة diffusers لإنشاء صورة لحيوانك المفضل وهو يأكل طعامك المفضل. يمكنك استخدام منصة قوقل كولاب في هذه المُهِمَّة.

5 استخدِم أداة StableDiffusion2ImagePipleline من مكتبة diffusers لتحويل الحيوان في الصورة المرسومة في التمرين السابق إلى حيوان آخر من اختيارك. يمكنك استخدام منصة قوقل كولاب في هذه المُهِمَّة.

المشروع

لا تستجيب كل مجموعة بيانات بالطريقة نفسها للتدريب بكل خوارزميات التصنيف، ولكي تحصل على أفضل النتائج لمجموعة بياناتك عليك أن تجرِّب استخدام خوارزميات مُختلفة، وتُقدم لك مكتبة Sklearn في البايثون مجموعة متنوّعة من الخوارزميات التي يمكنك تجريبها، بما فيها الخوارزميات التالية:

> من sklearn.ensemble.forest استورد خوارزمية RandomForestClassifier.

> من sklearn.naive_bayes استورد خوارزمية GaussianNB.

>من sklearn.svm استورد خوارزمية SVC.

1 استخدِم مجموعة تدريب وجوه الحيوانات لتدريب نموذج يحقق أكبر دقة ممكنة على مجموعة الاختبار.

2 استبدِل خوارزمية SGDClassifier بكل من الخوارزميات المذكورة أعلاه (RandomForestClassifier، GaussianNB، SVC) وحاول أن تحدِّد أفضلها.

3 أعِد تشغيل مفكرتك بعد كل عملية استبدال لحساب دقة كل نموذج جديد تجرِّبه.

4 أنشئ تقريرًا يقارن دقة كل النماذج التي جرّبتَها وحدِّد النموذج الذي حقق أفضل دقة.

ماذا تعلّمت

- إعداد الصور للتعرُّف عليها.
- استخدام المكتبات والدوال لإنشاء نماذج التعلُّم الموجَّه لتصنيف الصور.
- وصف طريقة تركيب الشبكات العصبية.
- استخدام المكتبات والدوال لإنشاء نماذج التعلُّم غير الموجَّه لعنقدة الصور.
- إنشاء الصور من خلال توفير التوجيه النصيّ.
- إكمال الأجزاء الناقصة لصورة ببيانات واقعية.

المصطلحات الرئيسة

صورة	Image
توليد الصور	Image Generation
المعالجة الأوليَّة للصور	Image Preprocessing
طبقة الشبكة	Network Layer
التعرّف	Recognition
الانتشار المستقر	Stable Diffusion
تحجيم قياسي	Standard Scaling
بيانات مرئية	Visual Data

رؤية الحاسب	Computer Vision
الشبكة العصبية الترشيحية	Convolutional Neural Network - CNN
نموذج الانتشار	Diffusion Model
هندسة الخصائص	Feature Engineering
انتقاء الخصائص	Feature Selection
الشبكة التوليدية التنافسية	Generative Adversarial Network - GAN
مُخطَّط تكراري للتدرجات الموجَّهة	Histogram of Oriented Gradients - HOG

5. خوارزميات التحسين واتخاذ القرار

سيتعرّف الطالب في هذه الوحدة على عدة خوارزميات وتقنيات تساعده في إيجاد أكثر الحلول كفاءة لمشكلات التحسين المعقدة، كما سيتعلّم طريقة عمل خوارزميات التحسين، وخوارزميات اتخاذ القرار، وطريقة تطبيقها لحلّ مشكلات متعلقة بالعالم الواقعي ترتبط بتخصيص الموارد والجدولة وتحسين المسارات.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُصنِّف طرائق التحسين لمعالجة مشكلات معقدة.
- يَصِف خوارزميات اتخاذ القرار المُختلفة.
- يَستخدِم البايثون لحلّ مشكلات تخصيص الموارد المتعلقة بفِرق العمل.
- يَحلّ مشكلات الجدولة باستخدام خوارزميات التحسين.
- يَستخدِم البايثون لحلّ مشكلات الجدولة.
- يَستخدِم البرمجة الرياضية لحلّ مشكلات التحسين.
- يُعرِّف مشكلة حقيبة الظهر (Knapsack Problem).
- يُعرِّف مشكلة البائع المُتجوّل (Traveling Salesman Problem).

الأدوات

- مفكَرة جوبيتر (Jupyter Notebook)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

مشكلة تخصيص الموارد

خوارزميات التحسين في الذكاء الاصطناعي
Optimization Algorithms in AI

يُستخدم الذكاء الاصطناعي في مُختلف الصناعات لاتخاذ قرارات تتسم بالكفاءة والدقة، ويُعدُّ استخدام خوارزميات تعلُّم الآلة إحدى طرائق الذكاء الاصطناعي المُستخدَمة في اتخاذ القرارات. وكما تعلّمت في الوحدة السابقة، فإن خوارزميات تعلُّم الآلة تقوم بتمكين الذكاء الاصطناعي من التعلُّم بواسطة البيانات ومن ثَمّ القيام بالتنبؤات أو تقديم التوصيات. على سبيل المثال، في مجال الرعاية الصحية، يُمكن استخدام الذكاء الاصطناعي للتنبؤ بنتائج المرضى والتوصية بخُطط علاجية بناءً على البيانات التي جُمعت من حالات مماثلة. وفي مجال التمويل، يُمكن استخدام الذكاء الاصطناعي في اتخاذ قرارات استثمارية بواسطة تحليل مجموعات كبيرة من البيانات المالية وتحديد الأنماط التي تبيّن المخاطر أو الفرص المحتملة. وعلى الرغم من أن خوارزميات تعلُّم الآلة تحظى بشعبية متزايدة إلا أنها ليست النوع الوحيد من خوارزميات الذكاء الاصطناعي التي يُمكن استخدامها في اتخاذ القرارات، فهناك طريقة أخرى تتمثل في استخدام خوارزميات التحسين التي تُستعمَل بوجه عام لإيجاد أفضل حلّ لمشكلة محدَّدة بناءً على قيود وأهداف معيّنة.

يهدف التحسين إلى تحقيق التصميم الأفضل بالنسبة لمجموعة من المعايير أو القيود ذات الأولوية، وتشمل تعزيز عوامل معيّنة مثل: الإنتاجية، والموثوقية، وطول العمر، والكفاءة، وفي الوقت نفسه تقليل عوامل أخرى مثل: التكاليف، والفاقد، والتوقف عن العمل، والأخطاء.

القيود (Constraints):
هي بمثابة شروط تقيِّد الحل، مثل الحد الأقصى لوزن الطرد الذي يمكن شحنه.

الدوال الموضوعية (Objective Functions):
هي معايير تحدِّد مدى اقتراب الحل المقدَّم من النتائج المطلوبة، مثل تقليل مسافة السفر لشاحنة توصيل.

مشكلات التخصيص Allocation Problems

تُعدُّ مشكلات التخصيص من مشكلات التحسين الشائعة؛ ففيها يتم تخصيص مجموعة من الموارد مثل: العمّال، أو الآلات، أو الأموال لمجموعة من المهام أو المشاريع بأعلى كفاءة ممكنة، وتنشأ هذه المشكلات في مجموعة واسعة من المجالات بما فيها التصنيع والخدمات اللوجستية وإدارة المشاريع والتمويل، ويُمكن صياغتها بطرائق مُختلفة بناءً على قيودها وأهدافها. في هذا الدرس ستتعرّف على مشكلات التخصيص وخوارزميات التحسين المستخدَمة لحلّها.

شكل 5.1: استخدام خوارزميات التحسين في مستودع

بعد ذلك، ستشاهد عددًا من الأمثلة، ولكل مثال منها قيود ودوال موضوعية خاصة به.

	القيود	الدوال الموضوعية
شركات النقل	- وضع أُطر زمنية للتوصيل؛ لضمان توصيل الطرود وفق إطار زمني محدَّد. - توفير سِعة مركبات التوصيل؛ لضمان استخدام المركبة المناسبة لكل عملية توصيل، ومقدرتها على حمل الكمية اللازمة من الطرود. - توفير السائقين والموظفين، ومراعاة تقسيم أوقات عملهم؛ لضمان كفاءة العمل، وعدم تكليفهم بأعمال فوق قدرتهم.	- **تقليل** (Minimizing) وقت التوصيل ومسافة السفر؛ لخفض التكلفة وتحسين الكفاءة. - **زيادة** (Maximizing) عدد الطرود في كل مركبة؛ لتقليل عدد الرحلات اللازمة. - **زيادة** (Maximizing) رضا العملاء من خلال توصيل الطرود في وقت محدَّد وفق إطار زمني محدَّد.
جدولة خطوط الطيران	- تَوَفُّر الطائرات وجداول الصيانة؛ لضمان إجراء الصيانة الجيدة لها، ومدى جاهزيّتها للرحلات. - قيود مراقبة الحركة الجوية؛ لتجنُّب التأخير وتقليل استهلاك الوقود. - مراعاة حاجة المسافر وتفضيلاته؛ لجدولة رحلات الطيران الأنسب للمسافرين.	- **تقليل** (Minimizing) تأخر رحلات الطيران أو إلغائها؛ لزيادة رضا العملاء. - **زيادة** (Maximizing) استغلال الطائرات؛ لتقليل التكاليف وتحسين الكفاءة. - **زيادة** (Maximizing) الإيرادات من خلال عمل عروض خاصة على رحلات الطيران عالية الطلب، وتعديل أسعار التذاكر بناءً على الطلب.
المصنِّعون	- سعة الإنتاج والمهلة الزمنية؛ لضمان تصنيع المُنتَجات في الوقت المناسب. - توفير المواد وسعة التخزين؛ لتجنُّب نفاد المخزون أو تكدسه. - تقلّبات الطلب؛ لتعديل جداول الإنتاج بناء على التغيرات في طلبات العملاء.	- **تقليل** (Minimizing) تكاليف الإنتاج من خلال تحسين استخدام الموارد وتقليل الفاقد. - **زيادة** (Maximizing) كفاءة الإنتاج من خلال جدولة دورات الإنتاج؛ لتقليل أوقات التجهيز والتبديل. - **زيادة** (Maximizing) رضا العملاء من خلال ضمان توفير المُنتَجات عند الحاجة إليها.
إدارة المخزون في الشركات	- سعة تخزين محدودة تتطلب إدارة دقيقة لمستويات المخزون. - فترات مهلة التسليم وتنوّعها، التي تؤثر على مقدار المخزون الذي يجب الاحتفاظ به في أي وقت. - توفير ميزانية؛ لشراء مخزون.	- **زيادة** (Maximizing) الربح من خلال ضمان وجود مستويات كافية من مخزون السلع ذات هامش الربح العالي. - **تقليل** (Minimizing) تكاليف التخزين من خلال تحسين مستويات المخزون بناءً على توقُّعات الطلب. - **زيادة** (Maximizing) رضا العملاء من خلال ضمان توفُّر المُنتَجات المناسبة في الوقت المناسب وفي المكان المناسب، وبتقليل نفاد المخزون والتأخير والمشكلات الأخرى التي قد تؤثر على تجربة العملاء.
شركات الطاقة	- مراعاة الطلب على الكهرباء وتقلّباته. - توفُّر المواد الخام وموارد الطاقة الضرورية. - قيود النقل والتوزيع مثل: سعة الشبكة والمسافة بين مصانع توليد الطاقة والمستهلكين.	- **تقليل** (Minimizing) تكلفة توليد الكهرباء وتوزيعها من خلال تحسين استخدام الموارد. - **تقليل** (Minimizing) هدر الطاقة وفشل الخدمات.

يُمكن نمذجة كل التطبيقات الواردة سابقًا في صورة مشكلات معقدة لها عدد كبير من الحلول المُمكنة. على سبيل المثال، فكِّر في مشكلة تخصيص الموارد المعهودة التي تركِّز على تشكيل فريق، حيث تنشأ المشكلة عندما يكون لديك:

- مجموعة كبيرة من العمّال يمتلكون مهارات مُختلفة.
- مُهِمَّة تتطلب مجموعةً فرعية محدَّدة من المهارات لأجل إكمالها.

ويتمثَّل الهدف في تكوين فريق بأقل عدد ممكن من العمّال، مع الالتزام في الوقت نفسه **بالقيد** (Constraint) الذي ينصّ على توفّر جميع المهارات المطلوبة في أعضاء الفريق؛ لأداء المُهِمَّة.

على سبيل المثال، تخيل سيناريو بسيطًا يوجد فيه خمسة عمال:

تتطلب المُهِمَّة المراد إنجازها كل المهارات: م1، م2، م3، م4، م5، م6. يتمثَّل الحلّ القائم على **القوة المُفرطة** (Brute Force) في أخذ كل فِرق العمّال المُمكنة في الاعتبار، والتركيز على الفِرق التي تتوفّر فيها جميع المهارات المطلوبة، واختيار الفريق الأقل عددًا، وعلى افتراض أن كل فريق يتكون من شخص واحد على الأقل، فيُمكنك أن تُشكِّل واحدًا وثلاثين فريقًا مختلفًا يتكون كل منهم من خمسة عمّال.

القوة المُفرطة (Brute-force):
هي طريقة من طرائق حلّ المشكلات تتضمن التجريب المنهجي لجميع الحلول الممكنة للمشكلة بهدف الوصول إلى الحلّ الأمثل، بغضّ النظر عن التكلفة الحاسوبية.

- بالنسبة للفريق المُكوَّن من عامل واحد، هناك خمس طرائق لاختيار عامل واحد من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من عاملين اثنين، هناك عشر طرائق لاختيار عاملين من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من ثلاثة عمّال، هناك عشر طرائق لاختيار ثلاثة عمّال من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من أربعة عمّال، هناك خمس طرائق لاختيار أربعة عمّال من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من خمسة عمّال، هناك طريقة واحدة لاختيار كل العمّال الخمسة.

العدد الإجمالي للفِرق المُختلفة التي يُمكنك تكوينها هو: 31 = 1 + 5 + 10 + 10 + 5، ويُمكن حساب العدد أيضًا وفقًا للمعادلة: $2^5 - 1$.

يكشف تقييم كل الفِرق الإحدى والثلاثين عن أفضل حلّ ممكن يتمثَّل في تكوين فريق يشمل العمّال: الأول والرابع والخامس، وسيغطي هذا الفريق كل المهارات الست المطلوبة، وسيشمل الفريق ثلاثة عمّال، ولا يُمكن تغطية كل المهارات بفريق يشتمل على عدد عمّال أقل من ذلك، مما يجعل هذا الحلّ هو **الحلّ الأمثل** (Optimal Solution).

وهناك حلّ آخر يتمثَّل في تكوين فريق يشمل العمّال: الأول والثاني والثالث والخامس، وعلى الرغم من أن هذا الفريق يغطي كل المهارات الست، إلا أنه يتطلب أيضًا عمّالًا أكثر، مما يجعل هذا الحلّ ممكنًا، ولكنه ليس الحلّ الأمثل.

الطبيعة الخاصّة بأسلوب القوة المُفرطة تضمن دائمًا إيجاد الحلّ الأمثل، متى أمكن ذلك، ولكنّ فحص كل الفِرق المُمكنة يُعدُّ عملية مكلّفة حاسوبيًا، فمثلًا:

- إذا كان لديك ستة عمّال، فسيكون عدد الفِرق المُمكنة: $2^6 - 1 = 63$.
- إذا كان لديك عشرة عمّال، فسيكون عدد الفِرق المُمكنة: $2^{10} - 1 = 1,023$.
- إذا كان لديك خمسة عشر عاملًا، فسيكون عدد الفِرق المُمكنة: $2^{15} - 1 = 32,767$.
- إذا كان لديك عشرون عاملًا، فسيكون عدد الفِرق المُمكنة: $2^{20} - 1 = 1,048,575$.
- إذا كان لديك خمسون عاملًا، فسيكون عدد الفِرق المُمكنة: $2^{50} - 1 = 1,125,899,906,842,623$.

من الواضح في مثل هذه المواقف أن حصر عدد الفِرق لكل الحلول المُمكنة ليس خيارًا عمليًّا، ولذلك تم اقتراح طرائق تحسين أخرى لمعالجة المشكلات المعقدة عن طريق البحث في خيارات الحلول المُمكنة بأسلوب أكثر كفاءة من أسلوب القوة المُفرطة، ويُمكن بوجه عام تصنيف هذه الطرائق في ثلاث فئات:

- **طرائق الاستدلال** (Heuristic Methods)
- **البرمجة القيدية** (Constraint Programming)
- **البرمجة الرياضية** (Mathematical Programming)

حتى بالنسبة لعدد معتدل من 50 عاملًا، فإن عدد الفرق المحتملة يتضخم إلى أكثر من كوادريليون (10^{15}=Quadrillion).

الحلّ الأمثل Optimal Solution

من الممكن أن تكون هناك العديد من الحلول المُثلى، كأن يكون لديك عدة فِرق تشمل ثلاثة عمّال وبإمكانها أن تستوفي كل المهارات المطلوبة، كما أنه من الممكن ألا يوجد حلّ لبعض المشكلات، على سبيل المثال: إذا كانت المُهِمَّة تتطلب المهارة السابعة وهي لا تتُوفّر في أي عامل من العمّال، فلن يكون هناك حلّ للمشكلة.

طرائق الاستدلال (Heuristic Methods)

تقوم **طرائق الاستدلال** (Heuristic Methods – HM) في العادة على التجربة، أو البديهة، أو الفطرة السليمة، وليس على التحليل الرياضي الدقيق، ويُمكن استخدامها لإيجاد حلول جيدة بشكل سريع، ولكنها لا تضمن الوصول إلى الحلّ الأمثل (أفضل حل يمكن الحصول عليه)، ومن الأمثلة على الخوارزميات الاستدلالية: **الخوارزميات الجشعة** (Greedy Algorithms)، و**محاكاة التلدين** (Simulated Annealing)، و**الخوارزميات الجينية** (Genetic Algorithms)، و**تحسين مستعمرة النمل** (Ant Colony Optimization). تُستخدم هذه الطرائق في العادة لحلّ المشكلات المعقدة التي تستغرق وقتًا حاسوبيًا طويلًا جدًّا، ولكن لا يُمكنها إيجاد حلول دقيقة، وستتعلّم في الدروس القادمة المزيد عن هذه الخوارزميات.

+ الإيجابيات

تتميز الاستدلالات بالكفاءة الحاسوبية، ويُمكنها أن تتناول المشكلات المعقدة، كما يُمكنها أن تجد حلولًا ذات جودة عالية إذا استُخدمت لها استدلالات معقولة.

- السلبيات

لا تضمن الوصول إلى الحلّ الأمثل، كما أن بعض الاستدلالات تتطلب ضبطًا كبيرًا حتى تُؤدي إلى نتائج جيدة.

البرمجة القيدية (Constraint Programming)

البرمجة القيدية (Constraint Programming – CP) تحلّ مشكلات التحسين عن طريق نمذجة القيود وإيجاد حلّ يخضع لجميع القيود، وهذا الأسلوب مفيد بشكل خاص في المشكلات التي بها عدد كبير من القيود أو التي تتطلب تحسين عدة أهداف.

+ الإيجابيات

يُمكن للبرمجة القيدية أن تتعامل مع قيود معقدة وأن تجد أفضل الحلول.

- السلبيات

يُمكن أن تكون هذه الطرائق مكلِّفة حاسوبيًا في المشكلات الكبيرة.

البرمجة الرياضية (Mathematical Programming)

البرمجة الرياضية (Mathematical Programming – MP) هي مجموعة من التقنيات التي تَستخدم نماذج رياضية؛ لحلّ مشكلات التحسين، وتشمل: **البرمجة الخطية** (Linear Programming)، **والبرمجة الرباعية** (Quadratic Programming)، **والبرمجة غير الخطية** (Nonlinear Programming) **وبرمجة الأعداد الصحيحة المختلطة** (Mixed-Integer Programming)، وتُستخدم هذه التقنيات على نطاق واسع في الكثير من المجالات؛ بما فيها علم الاقتصاد والهندسة وعمليات البحث. تلعب أساليب البرمجة الرياضية دورًا مهمًّا في **التعلُّم العميق** (Deep Learning)، وتمتلك نماذج التعلُّم العميق عددًا كبيرًا من المُعامِلات التي تحتاج أن تتعلّم من البيانات، حيث تُستخدم خوارزميات التحسين لتعديل مُعامِلات النموذج من أجل تقليل دالة التكلفة التي تقيس الفَرق بين مُخرَجات النموذج المتنبّأ بها والمُخرَجات الصحيحة. تم تطوير العديد من خوارزميات التحسين الخاصة بنماذج التعلُّم العميق مثل: خوارزمية آدم (Adam)، وخوارزمية **الاشتقاق التكيُّفي** (AdaGrad)، وخوارزمية **نشر متوسط الجذر التربيعي** (RMSprop).

+ الإيجابيات

تتعامل البرمجة الرياضية مع مجموعة واسعة من مشكلات التحسين وهي غالبًا تضمن الوصول إلى الحلّ الأمثل.

- السلبيات

يُعدُّ كلٌّ من التكلفة الحاسوبية للمشكلات الكبيرة وتعقيد إنشاء الصيغة الرياضية المناسبة مرتفعَين بالنسبة لمشكلات العالم الواقعي المعقدة.

مثال عملي: تحسين مشكلة تشكيل الفريق
A Working Example: Optimization for the Team-Formation Problem

سيوضِّح هذا الدرس استخدام **خوارزمية القوة المُفرطة** (Brute-Force Algorithm)، **والخوارزمية الاستدلالية الجشعة** (Greedy Heuristic Algorithm) لحلّ مشكلة اتخاذ القرار المُرتكزة على مشكلة تخصيص الموارد القائمة على الفريق والتي تم وصفها سابقًا، بعد ذلك ستتم مقارنة نتائج هاتين الخوارزميتين.

يُمكن استخدام الدالة التالية لإنشاء أمثلة عشوائية لمشكلة تشكيل الفِرق، وتسمح هذه الدالة للمستخدِم أن يُحدِّد أربعة مُعامِلات هي: العدد الإجمالي للمهارات التي يجب أن تؤخذ بعين الاعتبار، والعدد الإجمالي للعمّال المتوفّرين، وعدد المهارات التي يجب أن تتوفّر في أعضاء الفريق بشكل جماعي حتى ينجزوا المُهِمَّة، والعدد الأقصى للمهارات التي يُمكن أن يمتلكها كل عامل.

وبعد ذلك، تقوم الدالة بإنشاء وإظهار مجموعة عمّال لديهم عدة مهارات مُختلفة، وعدة مهارات مطلوبة، وتَستخدم هذه الدالة المكتبة الشهيرة Random التي يُمكن استخدامها في إخراج عيّنة أعداد عشوائية من مجموعة أعداد معيّنة أو عناصر عشوائية من قائمة معيّنة.

الخوارزمية الاستدلالية الجشعة (Greedy Heuristic Algorithm):

هي أسلوب استدلالي لحلّ المشكلات، وفيه تقوم الخوارزمية ببِناء الحلّ خطوةً خطوةً، وتختار الخيار الأمثل محلّيًّا في كل مرحلة، حتى تصل في النهاية إلى حلّ شامل ونهائي.

```
import random

def create_problem_instance(skill_number, # total number of skills
                            worker_number, # total number of workers
                            required_skill_number, # number of skills the team has to cover
                            max_skills_per_worker # max number of skills per worker
                              ):
```

```python
    # creates the global list of skills s1, s2, s3, ...
    skills = ['s' + str(i) for i in range(1, skill_number+1)]

    worker_skills = dict() # dictionary that maps each worker to their set of skills

    for i in range(1, worker_number+1): # for each worker

        # makes a worker id (w1, w2, w3, ...)
        worker_id = 'w' + str(i)

        # randomly decides the number of skills that this worker should have (at least 1)
        my_skill_number = random.randint(1, max_skills_per_worker)

        # samples the decided number of skills
        my_skills = set(random.sample(skills, my_skill_number))

        # remembers the skill sampled for this worker
        worker_skills[worker_id] = my_skills

    # randomly samples the set of required skills that the team has to cover
    required_skills = set(random.sample(skills, required_skill_number))

    # returns the worker and required skills
    return {'worker_skills':worker_skills, 'required_skills':required_skills}
```

ستقوم الآن باختبار الدالة الواردة سابقًا من خلال إنشاء نسخة من مشكلة معطياتها كالتالي: عشر مهارات إجمالية، وستة عمّال، وتتطلب خمس مهارات كحدٍّ أقصى لكل عامل.

شكل 5.2: رسم توضيحي للمثال الخاص بالمشكلة

بسبب الطبيعة العشوائية للدالة، ستحصل على نسخة مختلفة من المشكلة في كل مرة تقوم فيها بتشغيل هذا المقطع البرمجي.

```python
# the following code represents the above test
sample_problem = create_problem_instance(10, 6, 5, 5)

# prints the skills for each worker
for worker_id in sample_problem['worker_skills']:
    print(worker_id, sample_problem['worker_skills'][worker_id])

print()

# prints the required skills that the team has to cover
print('Required Skills:', sample_problem['required_skills'])
```

```
w1 {'s10'}
w2 {'s2', 's8', 's5', 's6'}
w3 {'s7', 's2', 's4', 's5', 's1'}
w4 {'s9', 's4'}
w5 {'s7', 's4'}
w6 {'s7', 's10'}

Required Skills: {'s6', 's8', 's7', 's5', 's9'}
```

تتمثل الخطوة التالية في إنشاء **خوارزمية حلّ** (Solver)، وهي خوارزمية تحسين يُمكنها أن تحدِّد أقل عدد ممكن لفريق العمّال الذي يُمكن اعتماده لإستيفاء كل المهارات المطلوبة.

اتخاذ القرار بخوارزمية القوة المُفرطة

Decision Making with a Brute-Force Algorithm

ستُطبِّق أول خوارزمية حلّ أسلوب القوة المُفرطة الذي يعتمد على التعداد الشامل لكل الفِرق المُمكنة وأخذها بعين الاعتبار، وسَتَستخدم هذه الخوارزمية أدوات combinations (توافيق) من وحدة itertools؛ لتوليد كل الفِرق المُمكنة ذات العدد المحدَّد.

سيتم توضيح الأداة بالمثال البسيط أدناه:

```python
# used to generate all possible combinations in a given list of elements
from itertools import combinations

L = ['w1', 'w2', 'w3', 'w4']

print('pairs', list(combinations(L, 2))) # all possible pairs
print('triplets', list(combinations(L, 3))) # all possible triplets
```

```
pairs [('w1', 'w2'), ('w1', 'w3'), ('w1', 'w4'), ('w2', 'w3'), ('w2',
'w4'), ('w3', 'w4')]
triplets [('w1', 'w2', 'w3'), ('w1', 'w2', 'w4'), ('w1', 'w3', 'w4'),
('w2', 'w3', 'w4')]
```

بعد ذلك، يُمكن إنشاء الدالة التالية لحلّ مشكلة تكوين الفريق بأسلوب القوة المُفرطة، وهذه الخوارزمية تأخذ بعين الاعتبار جميع أحجام الفرق الممكنة، وتنشىء الفِرق بناءً على الأعداد الممكنة، ثم تحصر الفِرق التي تستوفي كل المهارات المطلوبة وتحدِّد الفريق الأقل عددًا:

```
def brute_force_solver(problem):

    worker_skills = problem['worker_skills']
    required_skills = problem['required_skills']

    worker_ids = list(worker_skills.keys()) # gets the ids of all the workers
    worker_num = len(worker_ids) # total number of workers
    all_possible_teams = [] # remembers all possible teams
    best_team = None # remembers the best (smallest) team found so far

    #for each possible team size (singles, pairs, triplets, ...)
    for team_size in range(1, worker_num+1):

        # creates all possible teams of this size
        teams = combinations(worker_ids, team_size)
        for team in teams: # for each team of this size

            skill_union = set() # union of skills covered by all members of this team
            for worker_id in team: # for each team member
                # adds their skills to the union
                skill_union.update(worker_skills[worker_id])

            # if all the required skills are included in the union
            if required_skills.issubset(skill_union):

                # if this is the first team that covers all required skills
                # or this team is smaller than the best one or
                if best_team == None or len(team) < len(best_team):
                    best_team = team # makes this team the best one

    return best_team # returns the best solution
```

من الممكن ألا يكون هناك حلّ لنسخة المشكلة الواردة، فإذا كانت مجموعة المهارات المطلوبة تشمل مهارة لا يمتلكها أي عامل من العمّال المتواجدين، فلن تجد طريقة لإنشاء فريق يغطي كل المهارات، وفي مثل هذه الحالات ستُظهر الخوارزمية المذكورة سابقًا النتيجة بعدم وجود حلّ.

يُمكنك الآن استخدام المقطع البرمجي التالي لاختبار خوارزمية الحلّ بالقوة المُفرطة وفقًا للمثال الذي تم إنشاؤه سابقًا:

```
# uses the brute-force solver to find the best team for the sample problem
best_team = brute_force_solver(sample_problem)
print(best_team)
```

```
('w2', 'w3', 'w4')
```

من المؤكد أن خوارزمية الحلّ بالقوة المُفرطة ستجد أفضل حلّ ممكن، أي: أقلّ الفِرق عددًا طالما أن هناك حلٌّ ممكنٌ، ولكن كما تم مناقشته في بداية هذا الدرس فإن طبيعة الخوارزمية الشمولية تُؤدي إلى زيادة هائلة في التكلفة الحاسوبية كلما زاد حجم المشكلة.

يُمكن توضيح ذلك من خلال إنشاء نُسخ لمشكلات متعددة من حيث تزايد عدد العمّال، ويُمكن استخدام المقطع البرمجي التالي لتوليد نُسخ متنوعة من مشكلة تكوين الفريق، حيث يتنوع عدد العمّال ليكون: 5 و10 و15 و20، ثم يتم توليد 100 نسخة بعدد العمّال، وتشمل كل النُسخ المهارات الإجمالية العشر، والمهارات الثمان المطلوبة، والخمس مهارات كحدٍّ أقصى لكل عامل:

```
problems_with_5_workers = [] # 5 workers
problems_with_10_workers = [] # 10 workers
problems_with_15_workers = [] # 15 workers
problems_with_20_workers = [] # 20 workers

for i in range(100): # repeat 100 times

    problems_with_5_workers.append(create_problem_instance(10, 5, 8, 5))
    problems_with_10_workers.append(create_problem_instance(10, 10, 8, 5))
    problems_with_15_workers.append(create_problem_instance(10, 15, 8, 5))
    problems_with_20_workers.append(create_problem_instance(10, 20, 8, 5))
```

تَقبل الدالة التالية قائمة بنُسخ المشكلة وخوارزمية الحلّ بالقوة المُفرطة، وتُستخدم هذه الخوارزمية لإجراء العمليات الحسابية ثم استخراج الحلّ لجميع النُسخ، كما أنها تُسجل الوقت الإجمالي المطلوب (بالثواني) لحساب الحلول وكذلك العدد الإجمالي للنُسخ التي يُمكن إيجاد حلّ منها:

```
import time

def gets_solutions(problems,solver):

    total_seconds = 0 # total seconds required to solve all problems with this solver
    total_solved = 0 # total number of problems for which the solver found a solution
    solutions = [] # solutions returned by the solver

    for problem in problems:

        start_time = time.time() # starts the timer
        best_team = solver(problem) # computes the solution
        end_time = time.time() # stops the timer
        solutions.append(best_team) # rememberσ the solution
        total_seconds += end_time-start_time # computes total elapsed time

        if best_team != None: # if the best team is a valid team
            total_solved += 1
    print("Solved {} problems in {} seconds".format(total_solved,
                                                    total_seconds))

    return solutions
```

يستخدِم المقطع البرمجي التالي هذه الدالة وخوارزمية الحلّ بالقوة المُفرطة لحساب الحلول المُمكنة لمجموعات البيانات التي تم إنشاؤها سابقًا والمُكوَّنة من 5-workers (خمسة _عمّال)، و10-workers (عشرة _عمّال)، و15-workers (خمسة عشر _عاملًا)، و20-workers (عشرين _عاملًا):

```
brute_solutions_5 = gets_solutions(problems_with_5_workers,
           solver = brute_force_solver)

brute_solutions_10 = gets_solutions(problems_with_10_workers,
           solver = brute_force_solver)

brute_solutions_15 = gets_solutions(problems_with_15_workers,
           solver = brute_force_solver)

brute_solutions_20 = gets_solutions(problems_with_20_workers,
           solver = brute_force_solver)
```

```
Solved 23 problems in 0.0019948482513427734 seconds
Solved 80 problems in 0.06984829902648926 seconds
Solved 94 problems in 2.754629373550415 seconds
Solved 99 problems in 109.11902689933777 seconds
```

على الرغم من أن الأعداد المطلوبة سُجلت بواسطة الدالة gets_solutions() إلا أنها ستكون متفاوتة نظرًا للطبيعة العشوائية لمجموعات البيانات، وسيكون هناك نمطان ثابتان على الدوام هما:

- زيادة عدد العمّال تُؤدي إلى عدد أكبر من نُسخ المشكلات التي من الممكن إيجاد حلّ لها، وهذا النمط من الحلول معقول ومتوقَّع؛ لأن وجود عدد كبير من العمّال يزيد من احتمال وجود عاملٍ واحدٍ على الأقل يمتلك مهارة واحدة مطلوبة ضمن مجموعة العمّال المتاحة.
- زيادة عدد العمّال يؤدي إلى زيادة كبيرة (أُسِّيَّة) في الزمن الحاسوبي، وهذا متوقع حسب التحليل الذي تم إجراؤه في بداية هذا الدرس، وبالنسبة لمجموع العمّال ممن هم بعدد: خمسة، وعشرة، وخمسة عشر، وعشرون عاملًا، فإن عدد الفِرق المُمكنة يساوي: 31، 1023، 32767، و1048575 على الترتيب.

بصفة عامة، وبالنظر إلى عدد العمّال المُعطى N، فإن عدد الفِرق المُمكنة يساوي 2^N-1، وهذا العدد سيصبح كبيرًا لتقييمه حتى بالنسبة للقيم الصغيرة لـ N. كذلك بالنسبة لأي مشكلة بسيطة بها قيد واحد (يغطي جميع المهارات المطلوبة) وهدف واحد (تقليل حجم الفريق)، فإن القوة المُفرطة قابلة للتطبيق فقط على مجموعات البيانات الصغيرة جدًا، وذلك بالتأكيد ليس حلًا عمليًّا لأي من مشكلات التحسين المعقدة التي نواجها في الواقع والتي أشرنا إليها في بداية هذا الدرس.

اتخاذ القرار باستخدام خوارزمية استدلالية جشعة
Decision Making with a Greedy Heuristic Algorithm

تتعامل الدالة التالية مع هذا القيد بواسطة تنفيذ خوارزمية تحسين تعتمد على الأسلوب الاستدلالي الجشع، حيث تقوم الخوارزمية تدريجيًا بتكوين الفريق عن طريق إضافة عضو واحد في كل مرة، فالعضو الذي أضيف مؤخرًا يكون دائمًا هو العضو الذي يمتلك معظم المهارات التي لم توجد في سابقه، وتستمر العملية حتى تستوفي جميع المهارات المطلوبة.

الدالة الاستدلالية الجشعة (Greedy Heuristic) المُستخدَمة في هذا المثال هي معيار لاختيار عامل يتوفر فيه أكبر عدد من المهارات التي تُستوفى في الفريق إلى الآن، ويمكن استخدام دالة استدلالية أخرى، مبنية على إضافة العامل الذي يتوفر فيه العدد الأكبر من المهارات أولًا.

```python
def greedy_solver(problem):

    worker_skills = problem['worker_skills']
    required_skills = problem['required_skills']

    # skills that still have not been covered
    uncovered_required_skills = required_skills.copy()
    best_team = []
    # remembers only the skills of each worker that are required but haven't been covered yet
    uncovered_worker_skills = {}

    for worker_id in worker_skills:

        # remembers only the required uncovered skills that this worker has
        uncovered_worker_skills[worker_id] = worker_skills[worker_id].
intersection(uncovered_required_skills)

    # while there are still required skills to cover
    while len(uncovered_required_skills) > 0:

        best_worker_id = None  # the best worker to add next
        # number of uncovered skills required for the best worker to cover
        best_new_coverage = 0

        for worker_id in uncovered_worker_skills:

            # uncovered required skills that this worker can cover
            my_uncovered_skills = uncovered_worker_skills[worker_id]

            # if this worker can cover more uncovered required skills than the best worker so far
            if  len(my_uncovered_skills) > best_new_coverage:
                best_worker_id=worker_id  # makes this worker the best worker
                best_new_coverage=len(my_uncovered_skills)

        if best_worker_id != None:  # if a best worker was found
            best_team.append(best_worker_id)  # adds the worker to the solution

            #removes the best worker's skills from the skills to be covered
            uncovered_required_skills = uncovered_required_skills -
                                        uncovered_worker_skills[best_worker_id]

            for worker_id in uncovered_worker_skills:

                # remembers only the required uncovered skills that this worker has
                uncovered_worker_skills[worker_id] =
uncovered_worker_skills[worker_id].intersection(uncovered_required_skills)

        else:  # no best worker has been found and some required skills are still uncovered
            return None  # no solution could be found

    return best_team
```

تُظهر الدالة intersections( ) مجموعة جديدة تحتوي فقط على المهارات المشتركة من جميع مهارات العمّال الموجودة في worker_skills، والمهارات المطلوبة التي لم تُستوفَ في uncovered_worker_skills.

لا تأخذ خوارزمية الحلّ الجشعة كل الفِرق المُمكنة بعين الاعتبار ولا تضمن إيجاد الحلّ الأمثل، ولكنها كما هو موضَّح أدناه أسرع بكثير من خوارزمية الحلّ التي تعتمد على القوة المُفرطة، ومع ذلك يُمكنها أن تُنتج حلولًا جيدة، هي في الغالب حلولٌ مثلى، ومن المؤكد أن تجد هذه الطريقة حلًّا إذا كان موجودًا.

يستخدِم المقطع البرمجي التالي خوارزمية الحلّ الجشعة لحساب حلول مجموعات البيانات: 5-workers (خمسة _عمّال)، و10-workers (عشرة _عمّال)، و15-workers (خمسة عشر _عاملًا)، و20-workers (عشرين _عاملًا) التي تم استخدامها سابقًا لتقييم خوارزمية الحلّ بالقوة المُفرطة:

```
greedy_solutions_5 = gets_solutions(problems_with_5_workers,
           solver = greedy_solver)

greedy_solutions_10 = gets_solutions(problems_with_10_workers,
           solver = greedy_solver)

greedy_solutions_15 = gets_solutions(problems_with_15_workers,
           solver = greedy_solver)

greedy_solutions_20 = gets_solutions(problems_with_20_workers,
           solver = greedy_solver)
```

```
Solved 23 problems in 0.0009970664978027344 seconds
Solved 80 problems in 0.000997304916381836 seconds
Solved 94 problems in 0.001995086669921875 seconds
Solved 99 problems in 0.0019943714141845703 seconds
```

والآن يتضح الفَرق في السرعة بين الخوارزميتين؛ حيث يُمكن تطبيق خوارزمية الحلّ الجشعة على النُسخ المتعلقة بالمشكلات الكبيرة جدًا، كما في المثال التالي:

```
# creates 100 problem instances of a team formation problem with 1000 workers
problems_with_1000_workers = []

for i in range(100): # repeats 100 times
    problems_with_1000_workers.append(create_problem_instance(10, 1000, 8, 5))

# solves the 100-worker problems using the greedy solver
greedy_solutions_1000 = gets_solutions(problems_with_1000_workers,
           solver = greedy_solver)
```

```
Solved 100 problems in 0.09574556350708008 seconds
```

مقارنة الخوارزميات Comparing the Algorithms

بعد أن تم توضيح ميزة السرعة لخوارزمية الحلّ الاستدلالية الجشعة، تتمثّل الخطوة التالية في التحقق من جودة الحلول التي تُنتجها، حيث تَقبل الدالة التالية الحلول التي أنتجتها الخوارزمية الجشعة وخوارزمية القوة المُفرطة على نفس مجموعة نُسخ المشكلات، ثم تبيّن النِّسب المئوية للنُسخ التي تقوم كلتا الخوارزميتين بذكر الحلّ الأمثل لها (الفريق الأقل عددًا):

```
def compare(brute_solutions,greedy_solutions):
    total_solved = 0
    same_size = 0

    for i in range(len(brute_solutions)):

        if brute_solutions[i] != None: # if a solution was found
            total_solved += 1

            # if the solvers reported a solution of the same size
            if len(brute_solutions[i]) == len(greedy_solutions[i]):
                same_size += 1

    return round(same_size / total_solved, 2)
```

يُمكن الآن استخدام الدالة compare() لمقارنة فاعلية الخوارزميتين المطبقتين على: الخمسة عمّال، والعشرة عمّال، والخمسة عشر عاملًا، والعشرين عاملًا.

```
print(compare(brute_solutions_5,greedy_solutions_5))
print(compare(brute_solutions_10,greedy_solutions_10))
print(compare(brute_solutions_15,greedy_solutions_15))
print(compare(brute_solutions_20,greedy_solutions_20))
```

```
1.0
0.82
0.88
0.85
```

توضِّح النتائج أن الخوارزمية الاستدلالية الجشعة يُمكنها أن تجد باستمرار الحلّ الأمثل لحوالي 80 % أو أكثر من كل نُسخ المشكلات القابلة للحلّ. وفي الواقع، يُمكن التحقق بسهولة من أن حجم الفريق الذي تُنتجه الخوارزمية الاستدلالية الجشعة حتى في النُسخ التي تفشل في إيجاد الحلول المُثلى لها يكون قريبًا جدًا من حجم أفضل فريق ممكن.

إذا تمت إضافة ذلك إلى ميزة السرعة الهائلة، تجد أن الخوارزمية الاستدلالية خيار عمليّ أكثر للتطبيقات الواقعية، وستكتشف في الدرس التالي تقنيات تحسين أكثر ذكاءً، وستتعرّف على كيفية تطبيقها على مشكلات مُختلفة.

تمرينات

1 ما مزايا وعيوب استخدام كلٍّ من: خوارزمية القوة المفرطة والخوارزمية الاستدلالية الجشعة في حلّ مشكلات التحسين؟

2 حلِّلْ طريقة استخدام الخوارزميات الاستدلالية الجشعة لإيجاد الحلول المُثلى في مشكلات التحسين.

3 أنشئ خوارزمية حلّ جشعة لتحسين مشكلة تكوين أعضاء فريق، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم خوارزمية الحلّ الاستدلالية الجشعة لتكليف أعضاء الفريق بالمُهِمَّة:

```
def greedy_solver(problem):
    worker_skills=problem['worker_skills'] # worker skills for this problem
    required_skills=problem['required_skills'] # required skills for this problem

    uncovered_required_skills = required_skills.____________() # skills not covered
    best_team=[] # best solution
    uncovered_worker_skills={}
    for worker_id in worker_skills:

        uncovered_worker_skills[worker_id]=worker_skills[worker_id]._____
(uncovered_required_skills)
    while len(uncovered_required_skills) > 0:

        best_worker_id=__________ # the best worker to add next
        best_new_coverage=0 # number of uncovered required skills covered by the best worker
        for worker_id in uncovered_worker_skills: # for each worker
            my_uncovered_skills=uncovered_worker_skills[worker_id]
            # if this worker can cover more uncovered required skills than the best worker so far
            if  len(my_uncovered_skills)>best_new_coverage:
                best_worker_id=worker_id # makes this worker the best worker

                best_new_coverage=____________(my_uncovered_skills)

        if best_worker_id!=____________: # if a best worker was found

            best_team.___________(best_worker_id) # adds the worker to the solution
            #removes the best worker's skills from the skills to be covered
            uncovered_required_skills=uncovered_required_skills - uncovered_
worker_skills[best_worker_id]
            # for each worker
            for worker_id in uncovered_worker_skills:

                # remembers only the required uncovered skills that this worker has

                uncovered_worker_skills[worker_id]=uncovered_worker_

skills[worker_id].________(uncovered_required_skills)
        else: # no best worker has been found and some required skills are still uncovered

            return _______ # no solution could be found
    return best_team
```

4 اذكر ثلاث مشكلات تحسين مُختلفة من العالم الواقعي، وفي كل مشكلة:

- اضرب مثالًا على دالة موضوعية.
- اضرب مثالين على القيود إن وُجِدَتْ.

5 إذا قمتَ بزيادة عدد العمّال في خوارزمية القوة المُفرطة، كيف يؤثر ذلك على المشكلة من حيث عدد الحلول والزمن الحسابي؟

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

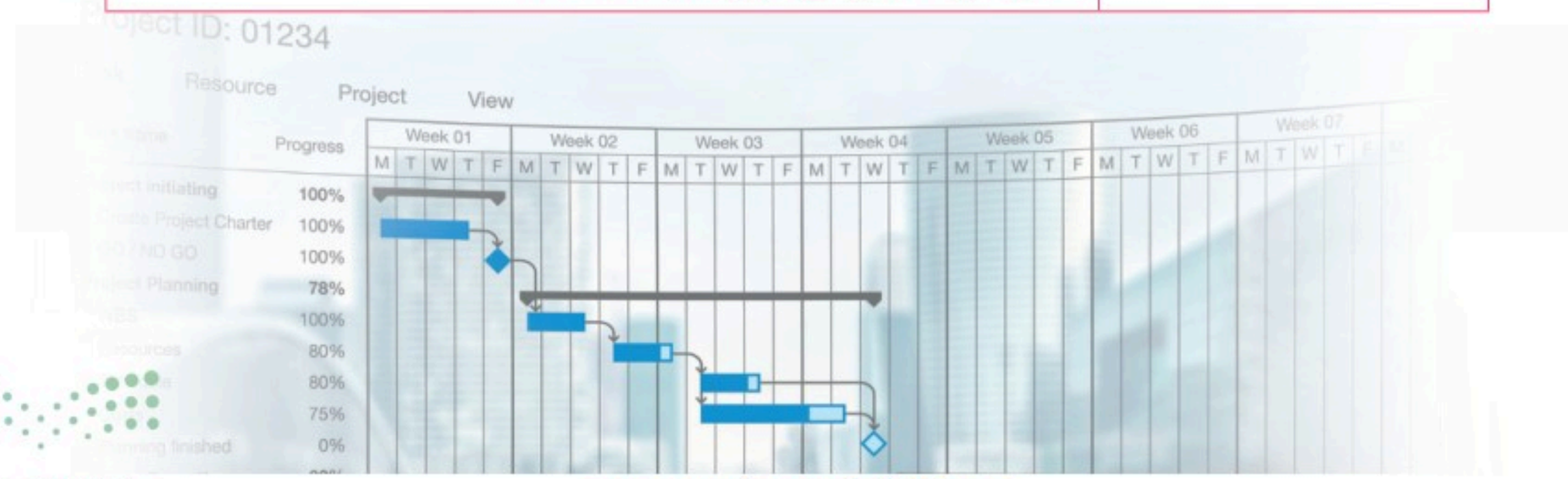
مشكلة جدولة الموارد

مشكلات الجدولة Scheduling Problems

مشكلات الجدولة شائعة في مجال التحسين؛ لأنها تتطلب تخصيص موارد محدودة لمهام متعددة بطريقة تُحسِّن بعض الدوال الموضوعية، وعادة ما تكون لمشكلات الجدولة قيود إضافية مثل: الحاجة إلى تنفيذ المهام بترتيب معيّن أو إنجازها في الموعد النهائي المحدَّد، وهذه المشكلات جوهرية في العديد من المجالات المختلفة بما فيها التصنيع والنقل والرعاية الصحية وإدارة المشاريع. ستتعمق في هذا الدرس في خوارزميات التحسين عن طريق إدخال تقنيات إضافية لحلّ جدولة المشكلات.

جدول 5.1: تطبيقات من مجالات مختلفة بحاجة إلى حلول الجدولة

جدولة المشاريع	تخصيص الموارد والمهام لأنشطة المشروع؛ لتقليل مدة المشروع وتكاليفه.
تخطيط الإنتاج	تحديد خطة الإنتاج المُثلى؛ لتلبية الطلب مع تقليل المخزون والتكاليف.
جدولة خطوط الطيران	جدولة إقلاع الطائرات وفترات عمل الطاقم؛ لتحسين جداول الرحلات مع تقليل التأخير والتكاليف.
جدولة مركز الاتصالات	تخصيص فترات عمل للموظفين؛ لضمان التغطية المناسبة لفترات العمل مع تقليل التكاليف والالتزام باتفاقيات مستوى الخدمة.
جدولة الإنتاج حسب الطلب	تخصيص الموارد في التصنيع؛ لتقليل زمن الإنتاج والتكاليف.
جدولة وسائل الإعلام	جدولة توقيت الإعلانات على التلفاز أو الإذاعة؛ لزيادة الوصول إلى الجمهور والإيرادات مع الالتزام بقيود الميزانية.
جدولة الممرضات	تخصيص فترات عمل للممرضات في المستشفيات؛ لضمان التغطية الكافية خلال فترات العمل مع تقليل تكاليف العمالة.



شكل 5.3: مُخطَّط قانت يبيّن جدول مشروع

في هذا الدرس ستُستخدم مشكلة **التباطُؤ الموزون للآلة الواحدة** (Single-Machine Weighted Tardiness - SMWT) كمثال عملي لتوضيح كيف يُمكن لخوارزميات التحسين أن تحلّ مشكلات الجدولة.

مشكلة التباطُؤ الموزون للآلة الواحدة
Single-Machine Weighted Tardiness (SMWT) Problem

لتوضيح هذه المشكلة، سنفترض أن مَصنعًا يرغبُ في جدولة مهام إنتاج عدة سلع على آلة واحدة، على النحو التالي:

- كل مُهِمَّة لها وقت معالجة محدَّد، وموعد محدَّد لابد أن تكتمل فيه.
- كل مُهِمَّة مرتبطة بوزن يمثل أهميتها.

إذا كان من المستحيل إنجاز كل المهام في الموعد النهائي، فسيكون عدم الالتزام بإنجاز المهام ذات الوزن الصغير في الموعد النهائي أقل تكلفة من عدم الالتزام بإنجاز المهام ذات الوزن الكبير في الموعد النهائي.

الهدف

الهدف (Goal) من جدولة المهام بطريقة محدَّدة هو تقليل المجموع الموزون للتأخير (التباطُؤ) لكل مُهِمَّة، وهكذا فإن مجموع التباطُؤ الموزون يكون بمثابة الدالة الموضوعية لخوارزميات التحسين المصمَّمة لحلّ هذه المشكلة.

حساب التأخير

يُحسبُ **التأخير** (Lateness) في أداء المُهِمَّة على أساس الفَرق بين زمن إنجازها والموعد المحدَّد لتسليمها، ثم تُستخدم أوزان المهام **كعوامل ضرب** (Multipliers) لإكمال المجموع الموزون النهائي. على سبيل المثال: افترض أن هناك جدولًا به ثلاث مهام هي: م1 وم2 وم3، وأوزان هذه المهام هي: 2 و1 و2 على الترتيب. وفقًا لهذا الجدول، ستُنجز المُهِمَّة رقم 1 في الموعد المحدَّد، وسيتأخر إنجاز المُهِمَّة رقم 2 ثلاث ساعات عن موعد تسليمها، أما المُهِمَّة رقم 3 فسيتأخر إنجازها ساعة واحدة عن موعد تسليمها، ويعني ذلك أن مجموع التباطُؤ الموزون يساوي $3\times1+1\times2=5$.

شكل 5.4: رسم توضيحي لتسلسل المهام

المُهِمَّة	الموعد المحدَّد لإنجازها	موعد تسليمها	التأخير	التباطُؤ الموزون
------	----------	----------	--------	----------
م1	14	11	0	0
م2	20	23	3	3
م3	17	18	1	2

شكل 5.5: حساب التباطُؤ الموزون

توجد صعوبة في حلّ مشكلة التباطُؤ الموزون للآلة الواحدة؛ لأن تعقُّدَها يتزايد تزايُدًا أُسِّيًّا مع عدد المهام، مما يجعل إيجاد أفضل حلّ ممكن لأحجام المُدخَلات الكبيرة مكلفًا للغاية وعادة ما يكون مستحيلًا.

تُستخدم خوارزميات التحسين للحصول على حلول شبه مثالية لمشكلة محدَّدة في مدة زمنية معقولة.

مشكلة جدولة الإنتاج حسب الطلب Job Shop Scheduling (JSS) Problem

مشكلة **جدولة الإنتاج حسب الطلب** (JSS) هي مشكلة اعتيادية أخرى في الجدولة حَظِيت بدراسات مُوسَّعة في مجال التحسين، وتتضمن جدولة مجموعة من المهام على عدة آلات، حيث يجب معالجة كل مُهِمَّة بترتيب ووقت معيّنان لكل آلة بالنسبة للمهام الأخرى.

الهدف

تقليل زمن الإنجاز الكليّ (فترة التصنيع) لجميع المهام.

متغيِّرات المشكلة

المتغيِّرات الأخرى من هذه المشكلة تفرض عدة قيود إضافية مثل:

- وجوب الالتزام بتاريخ إصدار كل مُهِمَّة؛ حيث إن لكل مُهِمَّة تاريخها الخاص ولا يمكن البدء بها قبل ذلك التاريخ، بالإضافة إلى مراعاة الموعد النهائي.
- وجوب جدولة بعض المهام قبل المهام الأخرى؛ بسبب ضوابط الأسبقية بينها.
- وجوب إخضاع كل آلة للصيانة الدورية وفقًا لضوابط جدول الصيانة، حيث لا يمكن للآلات تأدية المهام أثناء الصيانة، كما لا يمكن أن تتوقف المُهِمَّة بمجرد بدئها.

لا بد أن تمر كل آلة بفترة توقُّف عن الإنتاج بعد إكمال المُهِمَّة، وقد يكون طول هذه الفترة ثابتًا، وقد يتفاوت من آلة إلى أخرى، ومن الممكن أن يعتمد على الوقت الذي استغرقته الآلة في إكمال المُهِمَّة السابقة.

ما ورد أعلاه ليس سوى مجموعة فرعية من القيود المعقدة والمتعددة، ومن متغيِّرات المشكلة الموجودة في مشكلات الجدولة التي نواجها في واقع الحياة، حيث أن لكل متغيِّر خصائصه وتطبيقاته العملية الفريدة، وقد تكون خوارزميات التحسين المُختلفة أكثر ملاءمة لحلّ كل متغيِّر من متغيِّرات المشكلة.

استخدام البايثون والتحسين لحلّ مشكلة التباطُؤ الموزون للآلة الواحدة Using Python and Optimization to Solve the SMWT Problem

يُمكن استخدام المقطع البرمجي التالي لإنشاء نُسَخ عشوائية لمشكلة **التباطُؤ الموزون للآلة الواحدة** (SMWT):

```
import random

# creates an instance of the Single-Machine Weighted Tardiness problem.

def create_problem_instance(job_num, # number of jobs to create
                            duration_range, # job duration range
                            deadline_range, # deadline range
                            weight_range):# importance weight range

    # generates a random duration, deadline, and weight for each job
    durations = [random.randint(*duration_range) for i in range(job_num)]
    deadlines = [random.randint(*deadline_range) for i in range(job_num)]
    weights = [random.randint(*weight_range) for i in range(job_num)]

    # returns the problem instance as a dictionary
    return {'durations':durations,
            'deadlines':deadlines,
            'weights':weights}
```

تُستخدم الدالة random.randint(x,y) لتوليد عدد صحيح عشوائي بين x وy، وهناك طريقة مُختلفة لاستخدام هذه الدالة تتمثّل في توفير قائمة [x,y] أو مجموعة (x,y)، وفي هذه الحالة لا بد من كتابة الرمز * قبل القائمة، كما هو موضَّح في الدالة السابقة، على سبيل المثال:

```
for i in range(5):# prints 5 random integers between 1 and 10
    print(random.randint(*[1, 10]))
```

```
6
5
5
10
1
```

يَستخدم المقطع البرمجي التالي دالة ( )create_problem_instance لتوليد نسخة لمشكلة يتوفّر فيها ما يلي:

- تشتمل كلّ نسخة على عشرة مهام.
- يُمكن لكل مُهِمَّة أن تستمر ما بين 5 وحدات زمنية و20 وحدة زمنية، وسيتم افتراض أن الساعة هي الوحدة الزمنية المستخدَمة فيما تبقى من هذا الدرس.
- كل مُهِمَّة لها موعد نهائي يتراوح ما بين 5 ساعات و50 ساعة، وتبدأ ساعة الموعد النهائي من لحظة بدء المُهِمَّة الأولى في استخدام الآلة، على سبيل المثال: إذا كان الموعد النهائي لمُهِمَّة ما يساوي عشر ساعات، فهذا يعني أنه لا بد من إكمال المُهِمَّة في غضون عشر ساعات من بداية المُهِمَّة الأولى في الجدول.
- وزن كل مُهِمَّة هو عدد صحيح يتراوح بين 1 و3.

عدد المهام المراد إنشاؤها.

نطاق مدة المُهِمَّة.

```
create_problem_instance(10, [5, 20], [5, 50], [1, 3])
```

نطاق الموعد النهائي.

مدى أهمية الوزن.

```
{'durations': [18, 17, 17, 6, 9, 6, 20, 12, 9, 19],
 'deadlines': [39, 31, 6, 42, 48, 10, 39, 16, 34, 35],
 'weights': [2, 2, 3, 2, 1, 3, 2, 1, 3, 1]}
```

يُمكن استخدام الدالة التالية لتقييم جودة أي جدول أنتجته إحدى الخوارزميات لنسخة مشكلة محدَّدة، حيث تَقبل الدالة نسخة المشكلة وجدولًا لمهامها، ثم تمر على كل المهام بترتيب جدولتها نفسه حتى تَحسب أزمنة إنجازها ومجموع التباطُؤ الموزون لكامل الجدول، ويُحسب هذا التباطُؤ بحساب تباطؤ كل مُهِمَّة (مع مراعاة الموعد النهائي لها) وضربه في وزن المُهِمَّة وإضافة الناتج إلى المجموع:

```
# computes the total weighted tardiness of a given schedule for a given problem instance

def compute_schedule_tardiness(problem, schedule):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(schedule) # gets the number of jobs
    finish_times = [0] * job_num # stores the finish time for each job
    schedule_tardiness = 0 # initializes the weighted tardiness of the overall schedule to 0
    for pos in range(job_num): # goes over the jobs in scheduled order
```

```
        job_id=schedule[pos] # schedule[pos] is the id in the 'pos' position of the schedule

        if pos == 0: # if this is the job that was scheduled first (position 0)

            # the finish time of the job that starts first is equal to its run time
            finish_times[pos] = durations[job_id]

        else: # for all jobs except the one that was scheduled first

            # the finish time is equal to the finish time of the previous time plus the job's run time
            finish_times[pos] = finish_times[pos-1] + durations[job_id]

        # computes the weighted tardiness of this job and adds it to the schedule's overall tardiness
        schedule_tardiness += weights[job_id] * max(finish_times[pos] -
deadlines[job_id], 0)

    return schedule_tardiness,finish_times
```

ستُستخدم الدالة compute_schedule_tardiness() لتقييم الجداول، وستكون هذه الدالة بمثابة أداة مفيدة لكل الخوارزميات التي سيتم تقديمها في هذا الدرس لحلّ مشكلة **التباطُؤ الموزون للآلة الواحدة** (SMWT).

fx دالة التباديل Itertools.Permutations() Function

تستخدِم خوارزمية حلّ القوة المُفرطة الدالة itertools.permutations() لإنشاء كل الجداول المُمكنة (تجميعات المهام)، ثم تَحسب تباطؤ كل جدول ممكن وتستخرج أفضل جدول (الجدول ذو التباطؤ الكُليّ الأدنى).

تَقبل الدالة itertools.permutations() عنصرًا واحدًا متكررًا (مثل: قائمة) وتُنشئ كل تبديل ممكن لقيم المُدخَلات، ويوضِّح المثال البسيط التالي استخدام دالة permutations() ويُظهِر التبديلات لكل عناوين المهام المُعطاة:

```
job_ids = [0,1,2] # the ids of 3 jobs
for schedule in itertools.permutations(job_ids):
    print(schedule)
```

```
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

تُستخدم خوارزميات حلّ القوة المُفرطة بشكل أفضل لحلّ المشكلات الصغيرة، فالنسخة الخاصة بمشكلة التباطؤ الموزون للآلة الواحدة ذات عدد N من المهام، لديها عدد !N من الجداول الممكنة، فعندما يكون N = 5، سيكون الناتج 120 = !5 جدولًا، ولكن هذا العدد يتزايد بشكل كبير عندما يكون N = 10 إلى 3,628,800 = !10 ، وعندما يكون N = 11 إلى 39,916,800 = !11.

خوارزمية حلّ القوة المُفرطة Brute-Force Solver

لقد تعلّمت في الدرس السابق طريقة استخدام خوارزمية حلّ القوة المُفرطة في مشكلة تكوين فريق، وعلى الرغم من أن خوارزمية الحلّ هذه أظهرت بطئًا شديدًا في المشكلات الأكبر حجمًا، إلا أن قدرتها على إيجاد الحلّ الأمثل (أفضل حلّ ممكن) لنُسخ المشكلة ذات الحجم الصغير كانت مفيدة في تقييم جودة الحلول المُنتَجة بواسطة خوارزميات التحسين الأسرع التي لا تضمن إيجاد الحلّ الأمثل. وبالمثل: يُمكن استخدام خوارزمية حلّ القوة المُفرطة التالية لحلّ مشكلة **التباطُؤ الموزون للآلة الواحدة** (SMWT).

```python
import itertools

def brute_force_solver(problem):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations) # number of jobs

    # Generates all possible schedules
    all_schedules = itertools.permutations(range(job_num))

    # Initializes the best solution and its total weighted tardiness
    best_schedule = None  # initialized to None

    # 'inf' stands for 'infnity'. Python will evaluate all numbers as smaller than this value.
    best_tardiness = float('inf')

    # stores the finish time of each job in the best schedule
    best_finish_times = None  # initalized to None

    for schedule in all_schedules: # for every possible schedule

        #evalutes the schedule
        tardiness,finish_times=compute_schedule_tardiness(problem, schedule)

        if tardiness < best_tardiness: # this schedule is better than the best so far
            best_tardiness = tardiness
            best_schedule = schedule
            best_finish_times = finish_times

    # returns the results as a dictionary
    return  {'schedule':best_schedule,
             'tardiness':best_tardiness,
             'finish_times':best_finish_times}
```

خوارزمية الحلّ تعطي الجدول الأفضل، وزمن التباطؤ، وزمن إنجاز كل مُهِمَّة مُعطاة في هذا الجدول. على سبيل المثال، إذا كان الجدول يحوي ثلاث مهام، وكانت أوقات إنجاز جميع المهام تساوي [10، 14، 20]، فذلك يعني أن المُهِمَّة التي بدأت أولًا انتهت بعد 10 ساعات، والمُهِمَّة الثانية انتهت بعد ذلك بأربع ساعات، والمُهِمَّة الأخيرة انتهت بعد ست ساعات من اكتمال المُهِمَّة الثانية.

عدد المهام المراد إنشاؤها.

نطاق الموعد النهائي.

```python
sample_problem = create_problem_instance(5, [5, 20], [5, 30], [1, 3])
brute_force_solver(sample_problem)
```

نطاق مدة المُهِمَّة.

مدى أهمية الوزن.

```
{'schedule': (0, 2, 1, 3, 4),
 'tardiness': 164,
 'finish_times': [5, 11, 21, 36, 51]}
```

خوارزمية الحلّ الاستدلالية الجشعة Greedy Heuristic Solver

تستخدِم خوارزمية الحلّ الجشعة أسلوبًا استدلاليًا بسيطًا لفرز المهام واتخاذ قرار الترتيب الذي يجب جدولتها وفقًا له، ثم تُرتب المهام لحساب زمن إكمال كل مُهِمَّة ومجموع التباطُؤ الموزون لكامل الجدول، وفي هذا المثال الخاص تُظهر خوارزمية الحلّ الجشعة نوع المُخرَجات نفسه الذي أظهرته خوارزمية حلّ القوة المُفرطة.

تَقبل خوارزمية الحلّ الجشعة مُعامِلان هما: نسخة المشكلة المراد حلّها، ودالة الاستدلال التي ستُستخدم (معيار فرز المهام)، مما يسمح للمستخدِم بأن يُطبِّق أي دالة استدلال يختارها كدالة البايثون، ثم يمرِّره إلى خوارزمية الحلّ الجشعة باعتباره مُعامِلًا.

تُطبِّق الدالة التالية خوارزمية تحسين تستخدِم دالةً استدلاليةً جشعةً لحلّ المشكلة:

```
def greedy_solver(problem, heuristic):

    # gets the information for this problem
    durations, weights, deadlines = problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations)# gets the number of jobs

    # Creates a list of job indices sorted by their deadline in non-decreasing order
    schedule = sorted(range(job_num), key = lambda j: heuristic(j, problem))


    # evaluates the schedule
    tardiness, finish_times = compute_schedule_tardiness(problem, schedule)

    # returns the results as a dictionary
    return  {'schedule':schedule,
             'tardiness':tardiness,
             'finish_times':finish_times}
```

يُستخدم بناء الجملة lambda مع دالة البايثون sorted( ) عندما يتمثّل الهدف في فرز قائمة عناصر بناءً على قيمة يتم حسابها بطريقة منفصلة لكل عنصر.

يُستخدم في هذا المثال دالة استدلالية جشعة لتحديد المُهِمَّة التالية التي تحتاج إلى جدولة وهي المُهِمَّة التي لها أقرب موعد نهائي.

تُظهر الدالة التالية الموعد النهائي لمُهِمَّة محدَّدة في نسخة مشكلة مُعطاة:

```
# returns the deadline of a given job
def deadline_heuristic(job,problem):

    # accesses the deadlines for this problem and returns the deadline for the job
    return problem['deadlines'][job]
```

تمرير دالة deadline_heuristic كمُعَامِل إلى **خوارزمية الحلّ الجشعة** (greedy_solver) يعني أن الخوارزمية ستُجدول (تفرز) المهام وفق ترتيب تصاعدي حسب الموعد النهائي، مما يعني أن المهام التي لها أقرب موعد نهائي ستُجَدوَل أولًا.

```
greedy_sol = greedy_solver(sample_problem, deadline_heuristic)
greedy_sol
```

```
{'schedule': [3, 1, 4, 0, 2],
 'tardiness': 124,
 'finish_times': [15, 26, 32, 48, 57]}
```

تُطبِّق الدالة التالية استدلالًا بديلًا يأخذ في اعتباره أوزان المهام عند اتخاذ قرار ترتيبها في الجدول:

```
# returns the weighted deadline of a given job
def weighted_deadline_heuristic(job,problem):

    # accesses the deadlines for this problem and returns the deadline for the job
    return problem['deadlines'][job] / problem['weights'][job]
weighted_greedy_sol=greedy_solver(sample_problem, weighted_deadline_heuristic)
weighted_greedy_sol
```

```
{'schedule': [3, 2, 1, 4, 0],
 'tardiness': 89,
 'finish_times': [15, 24, 35, 41, 57]}
```

البحث المحلّي Local Search

على الرغم من أن خوارزمية الحلّ الجشعة أسرع بكثير من خوارزمية القوة المُفرطة، إلا أنها تميل إلى إنتاج حلول ذات جودة أقل بزمن تباطؤ أعلى، ويُعدُّ البحث المحلّي طريقة لتحسين حلّ تم حسابه بواسطة الخوارزمية الجشعة أو بأي طريقة أخرى.

البحث المحلّي (Local Search):
هو طريقة تحسين استدلالية تركِّز على اكتشاف حلول مجاورة لِحلّ معين بهدف تحسينه.

في البحث المحلّي، يُعدَّل الحلّ الذي تم التوصل إليه في البداية بشكلٍ متكرر من خلال فحص الحلول المجاورة التي وُجِدت عن طريق إجراء تعديلات بسيطة على الحلّ الحالي. بالنسبة للعديد من مشكلات التحسين، فهناك طريقة شائعة لتعديل الحلّ تتمثل في تبديل العناصر بشكل متكرر. على سبيل المثال، في مشكلة تكوين الفريق التي تم توضيحها في الدرس السابق، سيحاول أسلوب البحث المحلّي إنشاء فريق أفضل وذلك من خلال تبديل أعضاء الفريق بالعمّال الذين لا يُعدّون حاليًا جزءًا من الفريق.

أنشأت **خوارزمية الحلّ الاستدلالية الجشعة** (Greedy Heuristic Solver) حلًّا للمشكلة خطوة خطوة حتى حصلت في النهاية على حلّ كامل ونهائي، وعلى العكس من ذلك تبدأ طرائق البحث المحلّية بحلّ كامل قد يكون ذا جودة متوسطة أو سيئة، وتعمل بطريقة تكرارية لتحسين جودته. في كل خطوة يكون هناك تغيير بسيط على الحلّ الحالي، وتُقيَّم جودة الحلّ الناتج (يسمّى الحل المُجاور)، وإذا كان يتمتع بجودة أفضل، فإنهُ يستبدل الحلّ الحالي ويستمر في البحث، وإذا لم يكن كذلك، يتم تجاهل الحل المُجاور وتتكرر العملية لتوليد حل مجاور آخر، ثم ينتهي البحث عندما يتعذر العثور على حلّ مُجاور آخر يتمتع بجودة أفضل من الحلّ الحالي، ويتم تحديد أفضل حلّ تم العثور عليه.

دالة خوارزمية حلّ البحث المحلّي Local_search_solver( ) Function

تطبق الدالة التالية local_search_solver( ) خوارزمية حلّ البحث المحلّي القائم على المبادلة لِمشكلة **التباطؤ الموزون للآلة الواحدة** (SMWT)، حيث تَقبل هذه الدالة أربعة مُعامِلات وهي:

- نسخة المشكلة.
- خوارزمية استدلالية جشعة تستخدِمها دالة greedy_solver( ) لحساب حلّ أوّلي.
- دالة swap_selector المستخدَمة لانتقاء مُهِمَّتين ستتبادلان موقعيهما في الجدول. على سبيل المثال، إذا كان الحلّ الحالي للمشكلة المُكوَّنة من أربع مهام هو [1، 3، 2، 0]، وقرَّرت دالة swap_selector أن يحدث مبادلة بين المُهِمَّة الأولى والمُهِمَّة الأخيرة، سيكون الحلّ المرشَّح هو [0، 3، 2، 1].
- max_iterations عدد صحيح يُحدِّد عدد المبادلات التي يجب تجربتها قبل أن تتوصل الخوارزمية للحلّ الأفضل في حينه.

في كل تكرار، تنتقي الخوارزمية مُهِمَّتين للتبديل بينهما، ثم تُنشئ جدولًا جديدًا تتم فيه هذه المبادلة، وكل شيء في الجدول الجديد بخلاف ذلك سيكون مُطابقًا للجدول الأصلي. إذا كان للجدول الجديد تباطؤ موزون أقل من الجدول الأفضل الذي تم إيجاده حتى الآن، فإن الجدول الجديد يُصبح هو الأفضل بدلًا منه. خوارزمية الحلّ هذه لها نفس مُخرَجات خوارزمية الحلّ الجشعة وخوارزمية حلّ القوة المُفرطة.

سلوك خوارزميات التحسين القائمة على البحث المحلي يتأثر بشكل كبير بالاستراتيجية المستخدَمة بطريقة تكرارية لتعديل الحلّ.

```
def local_search_solver(problem, greedy_heuristic, swap_selector, max_
iterations):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations) # gets the number of jobs

    # uses the greedy solver to get a first schedule
    # this schedule will be then iteratively refined through local search
    greedy_sol = greedy_solver(problem, greedy_heuristic) # the best schedule so far

    best_schedule, best_tardiness, best_finish_times = greedy_sol['schedule'],
greedy_sol['tardiness'], greedy_sol['finish_times']

    # local search
    for i in range(max_iterations): # for each of the given iterations

        # chooses which two positions to swap
        pos1, pos2 = swap_selector(best_schedule)

        new_schedule = best_schedule.copy() # create a copy of the schedule

        # swaps jobs at positions pos1 and pos2
        new_schedule[pos1], new_schedule[pos2] = best_schedule[pos2],
                                                 best_schedule[pos1]
```

```
        # computes the new tardiness after the swap
        new_tardiness, new_finish_times = compute_schedule_tardiness(problem,
new_schedule)

        # if the new schedule is better than the best one so far
        if new_tardiness < best_tardiness:

            # the new_schedule becomes the best one
            best_schedule = new_schedule
            best_tardiness = new_tardiness
            best_finish_times = new_finish_times

    # returns the best solution
    return {'schedule':best_schedule,
            'tardiness':best_tardiness,
            'finish_times':best_finish_times}
```

جيران الحلّ في هذا المثال كلها حلول يتم الحصول عليها عن طريق انتقاء مُهمَّتين داخل الحلّ ومبادلة موقعيهما في الجدول.

تُطبِّق الدالة التالية مبادلة عشوائية بانتقاء مُهِمَّتين عشوائيتين في الجدول المُعطى الذي يستوجب تبديل مكانيهما:

```
def random_swap(schedule):

    job_num = len(schedule) # gets the number of scheduled jobs

    pos1 = random.randint(0, job_num - 1) # samples a random position

    pos2 = pos1
    while pos2 == pos1: # keeps sampling until it finds a position other than pos1
        pos2 = random.randint(0, job_num - 1) # samples another random position

    return pos1, pos2 # returns the two positions that should be swapped
```

تستخدِم الدالة التالية استراتيجية مُختلفة وذلك باختيارها الدائم لمُهِمَّتين عشوائيتين متجاورتين في الجدول لتبادلهما. على سبيل المثال، إذا كان الجدول الحالي لنسخة مشكلة مُكوَّنة من أربع مهام هو [2 ،1 ،3 ،0]، فإن المبادلات المُرشحة ستكون فقط 0<>3 و3<>1 و1<>2.

```
def adjacent_swap(schedule):

    job_num = len(schedule) # gets the number of scheduled jobs

    pos1 = random.randint(0, job_num - 2) # samples a random position (excluding the last
one)
    pos2 = pos1 + 1 # gets the position after the sampled one

    return pos1,pos2 # returns the two positions that should be swapped
```

يستخدِم المقطع البرمجي التالي استراتيجيتي المبادلة مع خوارزمية حلّ البحث المحلّي لحلّ المشكلة التي تم إنشاؤها في بداية هذا الدرس:

```
print(local_search_solver(sample_problem, weighted_deadline_heuristic, random_
swap, 1000))

print(local_search_solver(sample_problem, weighted_deadline_heuristic,
adjacent_swap, 1000))
```

```
{'schedule': [3, 4, 2, 1, 0], 'tardiness': 83, 'finish_times': [15, 21, 30,
41, 57]}
{'schedule': [3, 4, 2, 1, 0], 'tardiness': 83, 'finish_times': [15, 21, 30,
41, 57]}
```

تُظهر النتائج أفضل جدول [0، 1، 2، 4، 3] لهذا المثال، وإجمالي التباطُؤ 83، وأزمنة إكمال المهام (ستنتهي المُهِمَّة 3 في الوحدة 15 من الزمن، وتنتهي المُهِمَّة 4 في الوحدة 21 منه، وهكذا).

مقارنة خوارزميات الحلّ Comparing Solvers

يستخدِم المقطع البرمجي التالي الدالة create_problem_instance() لتوليد مجموعتي بيانات:

- مجموعة بيانات من 100 نسخة لمشكلة التباطُؤ الموزون للآلة الواحدة، وفي كل منها 7 مهام.
- مجموعة بيانات من 100 نسخة لمشكلة التباطُؤ الموزون للآلة الواحدة، وفي كل منها 30 مُهِمَّة.

سيتم استخدام مجموعة البيانات الأولى لمقارنة أداء جميع خوارزميات الحلّ الموضَّحة في هذا الدرس:

1. خوارزمية حلّ القوة المُفرطة.
2. خوارزمية الحلّ الجشعة المُتضمنة على استدلال خاص بالموعد النهائي.
3. خوارزمية الحلّ الجشعة المُتضمنة على استدلال خاص بالموعد النهائي الموزون.
4. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات عشوائية وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي لإيجاد الحلّ الأوّلي.
5. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات عشوائية وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي الموزون.
6. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات متجاورة وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي.
7. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات متجاورة وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي الموزون.

سيتم استخدام مجموعة البيانات الثانية لمقارنة جميع خوارزميات الحلّ باستثناء خوارزمية حلّ القوة المُفرطة البطيئة جدًا بالنسبة للمشكلات المشتملة على 30 مُهِمَّة.

```
#Dataset 1
problems_7 = []
for i in range(100):
    problems_7.append(create_problem_instance(7, [5, 20], [5, 50], [1, 3]))

#Dataset 2
problems_30 = []
for i in range(100):
    problems_30.append(create_problem_instance(30, [5,20], [5, 50], [1, 3]))
```

دالة المقارنة Compare( ) Function

تستخدِم الدالة التالية Compare( ) كل خوارزميات الحلّ؛ لحلّ كل المشكلات في مجموعة بيانات معيّنة، ثم تُظهر متوسطَ التباطُؤ الذي تحققه كل خوارزمية حلّ على كل المشكلات في مجموعة البيانات، وتَقبل الدالة كذلك المُعامِل المنطقي use_brute لتحديد إمكانية استخدام خوارزمية الحلّ بالقوة المُفرطة أم لا:

```
from collections import defaultdict
import numpy

def compare(problems,use_brute):
    # comparison on Dataset 1
    # maps each solver to a list of all tardiness values it achieves for the problems in the given dataset
    results = defaultdict(list)
    for problem in problems: # for each problem in this datset

        #uses each of the solvers on this problem
        if use_brute == True:
            results['brute-force'].append(brute_force_solver(problem)
['tardiness'])
        results['greedy-deadline'].append(greedy_solver(problem,deadline_
heuristic)['tardiness'])
        results['greedy-weighted_deadline'].append(greedy_
solver(problem,weighted_deadline_heuristic)['tardiness'])
        results['ls-random-wdeadline'].append(local_search_solver(problem,
weighted_deadline_heuristic, random_swap, 1000)['tardiness'])
        results['ls-random-deadline'].append(local_search_solver(problem,
deadline_heuristic, random_swap, 1000)['tardiness'])
        results['ls-adjacent-wdeadline'].append(local_search_solver(problem,
weighted_deadline_heuristic, adjacent_swap, 1000)['tardiness'])
        results['ls-adjacent-deadline'].append(local_search_solver(problem,
deadline_heuristic, adjacent_swap, 1000)['tardiness'])

    for solver in results: # for each solver
        # prints the solver's mean tardiness values
        print(solver,numpy.mean(results[solver]))
```

يُمكن الآن استخدام دالة compare( ) مع مجموعتي البيانات problems_7 و problems_30 كلتيهما:

```
compare(problems_7,True)
```

```
brute-force 211.49
greedy-deadline 308.14
greedy-weighted_deadline 255.61
ls-random-wdeadline 212.35
ls-random-deadline 212.43
ls-adjacent-wdeadline 220.62
ls-adjacent-deadline 224.36
```

```
compare(problems_30,False)
```

```
greedy-deadline 10126.18
greedy-weighted_deadline 8527.61
ls-random-wdeadline 6647.73
ls-random-deadline 6650.99
ls-adjacent-wdeadline 6666.47
ls-adjacent-deadline 6664.67
```

تمرينات

1 صِف استراتيجيتين مُختلفتين (مبادلة، انعكاس، تحويل، إلخ) لأسلوب البحث المحلي لِحلّ مشكلة التباطؤ الموزون للآلة الواحدة.

2 كم عدد الجداول المُمكنة (الحلول) لنسخة مشكلة التباطؤ الموزون للآلة الواحدة والتي تشتمل على تسع مهام؟

3 أنشئ خوارزمية حلّ بالقوة المُفرطة لمشكلة التباطؤ الموزون للآلة الواحدة، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم الدالةُ القوةَ المُفرطة لإيجاد تبديل الجدولة الأمثل.

```
def brute_force_solver(problem):
    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(_______________) # number of jobs
    # generates all possible schedules

    all_schedules = itertools._________________(range(job_num))
    # initializes the best solution and its total weighted tardiness

    best_schedule = ___________________ # initialized to None
    # 'inf' stands for 'infnity'. Python will evaluate all numbers as smaller than this value.

    best_tardiness = float('_________________')
    # stores the finish time of each job in the best schedule

    best_finish_times=_________________ # initalized to None

    for schedule in all_schedules: # for every possible schedule
        #evalute the schedule
        tardiness,finish_times=compute_schedule_tardiness(problem, schedule)
        if tardiness<best_tardiness: # this schedule is better than the best so far

            best_tardiness=___________________

            best_schedule=_____________

            best_finish_times=_______________

    # return the results as a dictionary
    return  {'schedule':best_schedule,
             'tardiness':best_tardiness,
             'finish_times':best_finish_times}
```

4 أنشئ خوارزمية حلّ البحث المحلّي لمشكلة التباطؤ الموزون للآلة الواحدة، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم الدالةُ البحث المحلّي لإيجاد تبديل الجدولة الأمثل.

```
def local_search_solver(problem, greedy_heuristic, swap_selector, max_
iterations):
    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(_______________)# gets the number of jobs
    # uses the greedy solver to get a first schedule.
    # this schedule will be then iteratively refined through local search

    greedy_sol = _______________(problem, greedy_heuristic) # remembers the best
schedule so far
    best_schedule, best_tardiness, best_finish_times=greedy_
sol['schedule'],greedy_sol['tardiness'],greedy_sol['finish_times']

    # local search

    for i in range(_____________): # for each of the given iterations
        # chooses which two positions to swap

        pos1,pos2=_______________(best_schedule)

        new_schedule = best_schedule._______________()# creates a copy of the
schedule
        # swaps jobs at positions pos1 and pos2
        new_schedule[pos1], new_schedule[pos2] = best_schedule[pos2], best_
schedule[pos1]
        # computes the new tardiness after the swap
        new_tardiness, new_finish_times = compute_schedule_tardiness(problem,
new_schedule)
        # if the new schedule is better than the best one so far
        if new_tardiness < best_tardiness:
            # the new_schedule becomes the best one

            best_schedule = _________________

            best_tardiness = ________________

            best_finish_times=_______________

    # returns the best solution
    return {'schedule':best_schedule,
            'tardiness':best_tardiness,
            'finish_times':best_finish_times}
```

5 صِف طريقة عمل البحث المحلّي.

6 اكتب ملاحظاتك عن نتائج خوارزميات الحلّ الجشعة مقارنة بخوارزميات حلّ البحث المحلّي في مشكلة تشتمل على ثلاثين مُهِمَّة. من وجهة نظرك، لماذا لم تُستخدم خوارزمية حلّ القوة المُفرطة في هذه المشكلة المكوّنة من ثلاثين مُهِمَّة؟

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

مشكلة تحسين المسار

البرمجة الرياضية في مشكلات التحسين

Mathematical Programming in Optimization Problems

في الدرسين السابقين تم توضيح كيفية استخدام الخوارزميات الاستدلالية لحلّ أنواع مُختلفة من مشكلات التحسين، وبالرغم من أن الاستدلالات بإمكانها أن تكون سريعة جدًا وتُنتج في العادة حلولًا جيدة، إلا أنها لا تضمن دائما إيجاد الحلّ الأمثل، وقد لا تكون مناسبة لكل أنواع المشكلات، وفي هذا الدرس ستُركِّز على أسلوب تحسين مُختلف وهو **البرمجة الرياضية** (Mathematical Programming).

البرمجة الرياضية (Mathematical Programming): هي تقنية تُستخدم لحلّ مشكلات التحسين عن طريق صياغتها على هيئة نماذج رياضية.

يُمكن للبرمجة الرياضية أن تحلّ العديد من مشكلات التحسين مثل: تخصيص الموارد، وتخطيط الإنتاج، والخدمات اللوجستية والجدولة، وتتميز هذه التقنية بأنها تُوفّر حلًّا مثاليًا مضمونًا ويُمكنها التعامل مع المشكلات المعقدة ذات القيود المتعددة.

يبدأ حلّ البرمجة الرياضية بصياغة مشكلة التحسين المُعطاة على شكل نموذج رياضي باستخدام المتغيِّرات، حيث تُمثِّل هذه المتغيِّرات القيم التي يجب تحسينها، ثم يتم استخدامها لتحديد الدالة الموضوعية والقيود، وهما يصفان المشكلة معًا ويُمكِّنان من استخدام خوارزميات البرمجة الرياضية.

تستخدِم البرمجة الرياضية **متغيِّرات القرار** (Decision Variables) التي تساعد مُتَّخِذ القرار في إيجاد الحل المناسب عن طريق ضبطها والتحكم فيها، كما يمكنها أن تستخدِم **متغيِّرات الحالة** (State Variables) التي لا يتحكم فيها مُتَّخِذ القرار وتفرضها البيئة الخارجية، وبالتالي لا يمكن ضبط متغيِّرات الحالة. تُوفِّر القوائم التالية أمثلة على متغيِّرات القرار ومتغيِّرات الحالة لبعض مشكلات التحسين الشائعة:

جدول 5.2: أمثلة على متغيِّرات القرار ومتغيِّرات الحالة

	متغيِّرات القرار	متغيِّرات الحالة
تخطيط الإنتاج	الكمية التي يجب إنتاجها من كل مُنتَج.	تَوفّر المواد الخام، وسعة آلات الإنتاج، وتَوفّر العمالة المطلوبة للإنتاج.
نقل الموارد	عدد السلع التي يجب نقلها من مكان لآخر.	المسافة بين الأماكن التي يجب زيارتها وسعة المركبات.
جدولة المهام	ترتيب كل مُهِمَّة والمدة الزمنية اللازمة لإجرائها.	تَوفّر العمّال والآلات، والمواعيد النهائية، ووزن أهمية كل مُهِمَّة.
توزيع الموظفين حسب المهام	تكليف العمّال وجدولتهم للقيام بمهام مُختلفة في أوقات مُختلفة.	مهارات كل عامل وتفضيلاته، وجاهزيّته، والمهارات المطلوبة منه لإنجاز كل مُهِمَّة.

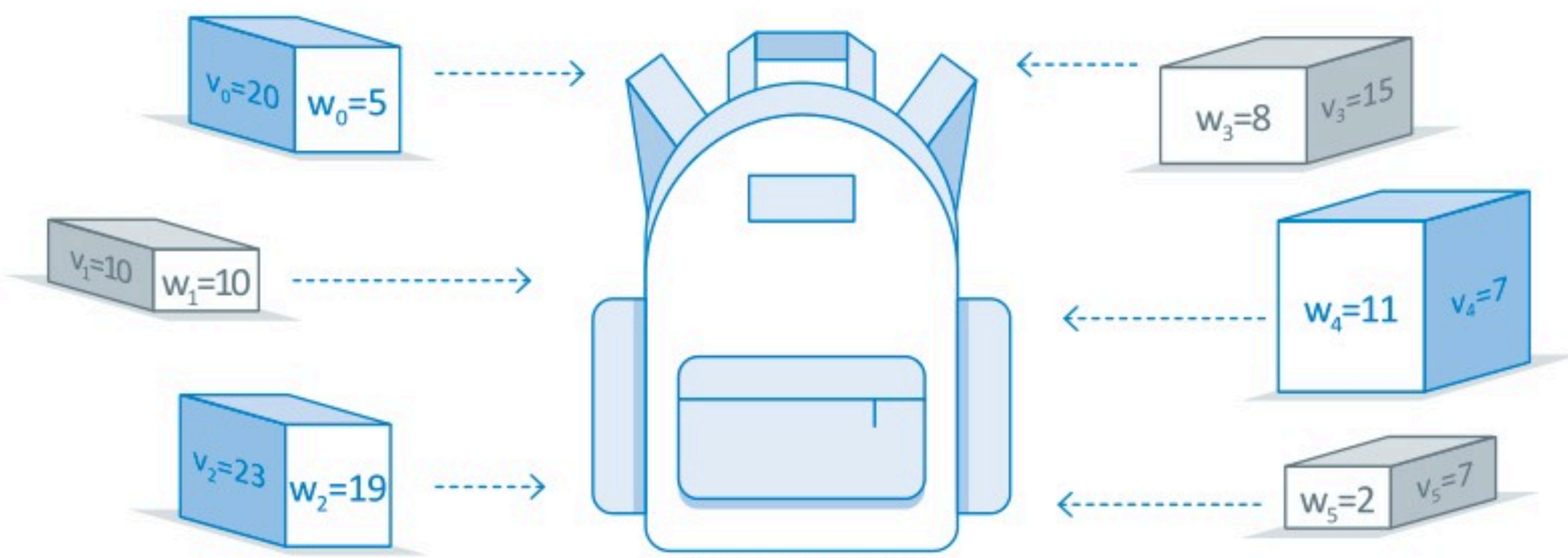
تتم صياغة الدالة الموضوعية **كتعبير رياضي** (Mathematical Expression) لتحسينها (بزيادتها أو تقليلها) بناءً على المتغيِّرات المناسبة، وتُمثِّل هذه الدالة الهدف من مشكلة التحسين مثل: زيادة الربح أو تقليل التكاليف، وتُحدَّد في العادة بناءً على متغيِّرات القرار، كما تُحدَّد أحيانًا بناءً على متغيِّرات الحالة، وبالمثل يُمكن صياغة القيود باستخدام المتغيِّرات والمتباينات الرياضية.

توجد عدة أنواع من البرمجة الرياضية، مثل: **البرمجة الخطية** (Linear Programming - LP)، و**البرمجة الرباعية** (Quadratic Programming - QP) و**برمجة الأعداد الصحيحة المختلطة** (Mixed Integer Programming - MIP). يركِّز هذا الدرس على برمجة الأعداد الصحيحة المختلطة المُستخدَمة في المشكلات التي تتقيّد فيها متغيِّرات القرار بالأعداد الصحيحة مثل: مشكلات الجدولة أو اختيار الطريق.

مشكلة حقيبة الظهر The Knapsack Problem

مشكلة حقيبة الظهر 1/0 هي مثال بسيط على استخدام برمجة الأعداد الصحيحة المختلطة لصياغة الدالة الموضوعية والقيود، وتُعرَّف المشكلة على النحو التالي: لديك حقيبة ظهر بسعة قصوى تبلغ C وحدة، ومجموعة من العناصر I، بحيث يكون لكل عنصر i متغيِّران من متغيِّرات الحالة هما وزن العنصر w_i وقيمته v_i، والمطلوب هو تعبئة الحقيبة بمجموعة العناصر ذات أقصى قيمة ممكنة في حدود سعة الحقيبة. يُستخدم متغيِّر القرار x_i لتتبع تجميعات العناصر التي ستُعبَّأ في حقيبة الظهر، حيث تكون $x_i = 1$ إذا تم اختيار العنصر i للإضافة للحقيبة، بينما تكون $x_i = 0$ خلاف ذلك، ويتمثّل الهدف في انتقاء مجموعة فرعية من العناصر من I بحيث تشمل:

- **القيد** (Constraint): مجموع أوزان العناصر المنتقاة بها لا يزيد عن السعة القصوى C.
- **الدالة الموضوعية** (Objective Function): مجموع قيم العناصر المنتقاة بها هي أقصى قيمة مُمكنة.



شكل 5.6: مشكلة حقيبة الظهر

يوضِّح الشكل 5.6 مثالًا على مسألة حقيبة ظهر مُكوَّنة من ستة عناصر بأوزان وقِيم محدَّدة، وحقيبة ظهر بسعة قصوى تساوي أربعين وحدة. يقوم المقطع البرمجي التالي بتثبيت مكتبة البايثون المفتوحة المصدر mip الخاصة ببرمجة الأعداد الصحيحة المختلطة لحلّ نسخة مشكلة حقيبة الظهر 1/0، ويستورد الوحدات الضرورية:

```
!pip install mip # install the mip library
```

```
# imports useful tools from the mip library
from mip import Model, xsum, maximize, BINARY
values = [20, 10, 23, 15, 7, 7] # values of available items
weights = [5, 10, 19, 8, 11, 2] # weights of available items
```

```
C = 40 # knapsack capacity

I = range(len(values)) # creates an index for each item: 0,1,2,3,...

solver = Model("knapsack") # creates a knapsack solver
solver.verbose = 0 # setting this to 1 will print more information on the progress of the solver

x = [] # represents the binary decision variables for each item.

# for each items creates and appends a binary decision variable
for i in I:
    x.append(solver.add_var(var_type = BINARY))

# creates the objective function
solver.objective = maximize(xsum(values[i] * x[i] for i in I))

# adds the capacity constraint to the solver
solver += xsum(weights[i] * x[i] for i in I) <= C

# solves the problem
solver.optimize()
```

```
<OptimizationStatus.OPTIMAL: 0>
```

يُنشئ المقطع البرمجي القائمة x لتخزين متغيِّرات القرار الثنائية للعناصر، وتُوفِّر المكتبة mip في البايثون ما يلي:

- أداة add_var(var_type=BINARY) لإنشاء المتغيِّرات الثنائية وإضافتها إلى خوارزمية الحلّ.
- أداة maximize( ) لمشكلات التحسين التي تحتاج لزيادة دالة موضوعية، أما مشكلات التحسين التي تتطلب تصغير الدالة الموضوعية، فتستخدِم الأداة minimize( ).
- أداة xsum( ) لإنشاء التعبيرات الرياضية التي تتضمن **المجاميع** (sums)، وفي المثال السابق تم استخدام هذه الأداة لحساب مجموع الوزن الإجمالي للعناصر في إنشاء قيد السّعة وحلّه.
- أداة optimize( ) لإيجاد حلّ يحسِّن الدالة الموضوعية في ظل الالتزام بالقيود، وتَستخدم الأداة برمجةَ الأعداد الصحيحة المختلطة للنظر بكفاءة في توليفات القيم المُختلفة لمتغيِّرات القرار ولإيجاد التوليفة التي تُحسِّن الهدف.
- المُعامِل += لإضافة قيود إضافية إلى خوارزمية الحلّ الموجودة.

في المقطع البرمجي أدناه تحتوي القائمة x على متغيِّر ثنائي واحد لكل عنصر، وبعد حساب الحلّ سيكون كل متغيِّر مساويًا للواحد إذا أُدرج العنصر في الحلّ، وسيُساوي صفرًا بخلاف ذلك. تَستخدم المكتبة mip بِناء الجملة x.[i]x لإظهار القيمة الثنائية للعنصر ذي الفهرس i، وتَحسب خوارزمية الحلّ متغيِّر القرار x، ثم تجد القيمة الإجمالية والوزن الإجمالي للعناصر المنتقاة عن طريق التكرار على متغيِّر القرار x، وتَجمع الأوزان والقيم لكل عنصر منتقى i، استنادًا إلى x[i]، وتَعرضها كما هو موضَّح في المقطع البرمجي التالي:

```
total_weight = 0 # stores the total weight of the items in the solution
total_value = 0 # stores the total value of the items in the solution
```

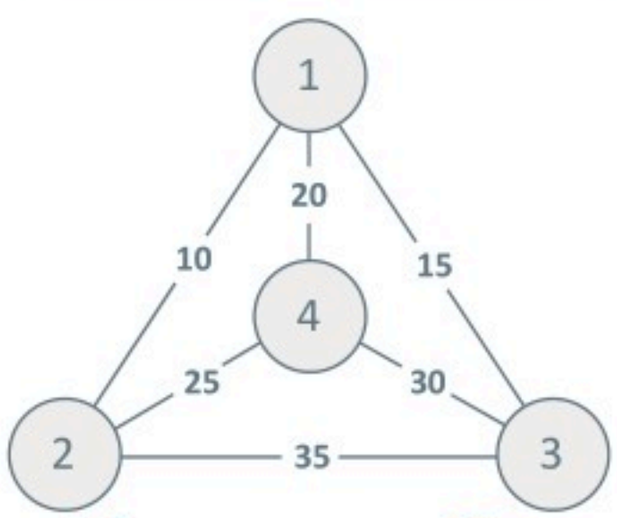
```
for i in I: # for each item
    if x[i].x == 1: # if the item was selected
        print('item', i, 'was selected')
        # updates the total weight and value of the solution
        total_weight += weights[i]
        total_value += values[i]

print('total weight', total_weight)
print('total value', total_value)
```

```
item 0 was selected
item 2 was selected
item 3 was selected
item 5 was selected
total weight 34
total value 65
```

مشكلة البائع المُتجوّل
Traveling Salesman Problem

مشكلة البائع المُتجوّل (Traveling Salesman Problem – TSP) من المشكلات الأخرى التي يُمكن حلّها ببرمجة الأعداد الصحيحة المختلطة، وهي مشكلة مألوفة تُعنى بتحديد أقصر مسار يمكن أن يسلكه بائع متجوّل لزيارة مدنٍ معيّنة مرة واحدة، دون أن يكرّر زيارة أيٍّ منها، ثمّ يعودَ للمدينة الأصلية، ويصوّرُ الشكل 5.7 نسخةً من هذه المشكلة.

الأمثلة الواردة في مُخطَّط مشكلة البائع المُتجوّل متصلة اتصالًا تامًّا؛ فهناك حافة تصل كل زوج من العُقد بالآخر.



شكل 5.7: نسخة على مشكلة البائع المُتجوّل

تُمثّل كل دائرة (عقدة) مدينة أو موقعًا يجب زيارته، وهناك حافة تربط بين موقعين إذا كان من الممكن السفر بينهما، ويُمثّل الرقم الموجود على الحافة التكلفة (المسافة) بين الموقعين. في هذا المثال، تم ترقيم المواقع وفقًا لترتيبها في الحلّ الأمثل للمشكلة، وتكون التكلفة الإجمالية للطريق 1 ← 2 ← 4 ← 3 ← 1 تساوي 80 = 15+30+25+10، وهو أقصر طريق ممكن لزيارة كل مدينة مرة واحدة فقط والعودة إلى نقطة البداية.

توجد تطبيقات عملية لمشكلة البائع المُتجوّل في الخدمات اللوجستية، والنقل، وإدارة الإمدادات والاتصالات، فهي تنتمي إلى عائلة أوسع من مشكلات تحديد الطريق التي تشمل أيضًا مشكلات شهيرة أخرى موضحة فيما يلي:

- تتضمن **مشكلة تحديد مسار المركبات** (Vehicle Routing Problem) إيجاد الطُرق المُثلى لأسطول من المركبات لتوصيل السلع أو الخدمات لمجموعة من العملاء في ظل تقليل المسافة الإجمالية المقطوعة إلى الحد الأدنى، وتشمل تطبيقاتها الخدمات اللوجستية وخدمات التوصيل وجمع النفايات.
- تتضمن **مشكلة الاستلام والتسليم** (Pickup and Delivery Problem) إيجاد الطُرق المُثلى للمركبات لكي تستلم (تُحمِّل أو تُرْكِب) وتسلِّم (تُوصِل) البضائع أو الأشخاص إلى مواقع مُختلفة، وتشمل تطبيقاتها خدمات سيارات الأجرة، والخدمات الطبية الطارئة، وخدمات النقل الجماعي.
- تتضمن **مشكلة جدولة مواعيد القطارات** (Train Timetabling Problem) إيجاد جداول زمنية مثالية للقطارات في شبكة سكك الحديد في ظل تقليل نسبة التأخير إلى الحد الأدنى وضمان الاستخدام الفعّال للموارد، وتشمل تطبيقاتها النقل بالسكك الحديدية والجدولة.

يُمكن استخدام المقطع البرمجي التالي لإنشاء نسخة من مشكلة البائع المُتجوّل، وتَقبل الدالة عدد المواقع المراد زيارتها، ونطاق المسافة يُمثِّل الفَرق بين المسافة الأقصر والمسافة الأطول بين موقعين، ثم تُظهر:

- مصفوفة المسافة التي تشمل المسافة المُسندة بين كل زوج ممكن من المواقع.
- مجموعة عناوين المواقع العددية (عنوان لكل موقع).
- الموقع الذي يكون بمثابة بداية الطريق ونهايته، ويُشار إليه باسم موقع startstop (الانطلاق والتوقف).

```
import random
import numpy
from itertools import combinations

def create_problem_instance(num_locations, distance_range):
    # initializes the distance matrix to be full of zeros
    dist_matrix = numpy.zeros((num_locations, num_locations))
    # creates location ids: 0,1,2,3,4,...
    location_ids = set(range(num_locations))
    # creates all possible location pairs
    location_pairs = combinations(location_ids, 2)
    for i,j in location_pairs: # for each pair
        distance = random.randint(*distance_range) # samples a distance within range
        # the distance from i to j is the same as the distance from j to i
        dist_matrix[j,i] = distance
        dist_matrix[i,j] = distance

    # returns the distance matrix, location ids and the startstop vertix
    return dist_matrix, location_ids, random.randint(0, num_locations - 1)
```

يستخدِم المقطع البرمجي التالي الدالة الواردة سابقًا لإنشاء نسخة من مشكلة البائع المُتجوّل، بحيث يتضمن 8 مواقع، ومسافات ثنائية تتراوح بين 5 و20:

```
dist_matrix, location_ids, startstop = create_problem_instance(8, (5, 20))
print(dist_matrix)
print(startstop)
```

```
[[ 0. 19. 17. 15. 18. 17.  7. 15.]
 [19.  0. 15. 18. 11.  6. 20.  5.]
 [17. 15.  0. 17. 15.  7.  5. 11.]
 [15. 18. 17.  0. 19.  7.  7. 16.]
 [18. 11. 15. 19.  0. 17. 20. 17.]
 [17.  6.  7.  7. 17.  0. 15. 14.]
 [ 7. 20.  5.  7. 20. 15.  0. 14.]
 [15.  5. 11. 16. 17. 14. 14.  0.]]
3
```

لاحظ أن الخط القُطري يُمثِّل المسافات من العُقد إلى نفسها (dist_ matrix[i،i])، وبالتالي فإن المسافات تساوي أصفارًا.

إنشاء خوارزمية حلّ القوة المُفرطة لمشكلة البائع المُتجوّل
Creating a Brute-Force Solver for the Traveling Salesman Problem

تستخدِم الدالة التالية خوارزمية حلّ القوة المُفرطة لتعداد جميع الطُرق المُمكنة (التباديل) وإظهار أقصر مسار، وتَقبل هذه الدالة مصفوفة المسافة وموقع الانطلاق والتوقف الذي تُظهره الدالة create_problem_instance( ). لاحظ أن الحل الممكن لنسخة **مشكلة البائع المُتجوّل** (TSP) هي تبديل مدن، يبدأ من مدينة startstop (الانطلاق والتوقف) ثم ينتهي إليها.

```
from itertools import permutations

def brute_force_solver(dist_matrix, location_ids, startstop):
    # excludes the starstop location
    location_ids = location_ids - {startstop}
    # generate all possible routes (location permutations)
    all_routes = permutations(location_ids)
    best_distance = float('inf') # initializes to the highest possible number
    best_route = None # best route so far, initialized to None

    for route in all_routes: # for each route
        distance = 0 # total distance in this route
        curr_loc = startstop # current location

        for next_loc in route:
            distance += dist_matrix[curr_loc,next_loc] # adds the distance of this step
            curr_loc = next_loc # goes to the next location
        distance += dist_matrix[curr_loc,startstop] # goes to the starstop location
        if distance < best_distance: # if this route has lower distance than the best route
            best_distance = distance
            best_route = route

    # adds the startstop location at the beginning and end of the best route and returns
    return [startstop] + list(best_route) + [startstop], best_distance
```

تستخدِم خوارزمية حلّ القوة المُفرطة أداة permutations( ) لإنشاء كل الطُرق المُمكنة. لاحظ أن موقع startstop (الانطلاق والتوقف) يُستبعد من التباديل؛ لأنه يجب أن يظهر دائمًا في بداية كلّ طريق ونهايته، فعلى سبيل المثال، إذا كانت لديك أربعة مواقع 0، و1، و2، و3، وكان الموقع 0 هو موقع startstop (الانطلاق والتوقف)، ستكون قائمة التباديل المُمكنة كما يلي:

```
for route in permutations({1,2,3}):
    print(route)
```

```
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)
```

تَحسب خوارزمية حلّ القوة المُفرطة المسافة الإجمالية لكل طريق، وتُظهر في النهاية الطريق ذا المسافة الأقصر. يُطبِّق المقطع البرمجي التالي خوارزمية الحلّ على نسخة مشكلة البائع المُتجوّل التي تم إنشاؤها سابقًا:

```
brute_force_solver(dist_matrix, location_ids, startstop)
```

```
([3, 5, 2, 7, 1, 4, 0, 6, 3], 73.0)
```

على غرار خوارزميات حلّ القوة المُفرطة التي تم توضيحها في الدروس السابقة، لا تُطبَّق هذه الخوارزمية إلا على نُسخ مشكلة البائع المُتجوّل الصغيرة؛ لأن عدد الطُرق المُمكنة يتزايد أضعافًا مضاعفة كلما زاد العدد N، ويساوي $(N-1)!$، وعلى سبيل المثال، عندما يكون $N = 15$، فإن عدد الطُرق المُمكنة يساوي $14! = 87,178,291,200$.

استخدام برمجة الأعداد الصحيحة المختلطة لحلّ مشكلة البائع المُتجوّل
Using MIP to Solve the Traveling Salesman Problem

لاستخدام **برمجة الأعداد الصحيحة المختلطة** (MIP) لحلّ **مشكلة البائع المُتجوّل** (TSP)، يجب إنشاء صيغة رياضية تُغطي كلًا من الدالة الموضوعية وقيود مشكلة البائع المُتجوّل.

تتطلب الصيغة متغيِّر قرار ثنائي x_{ij} لكل انتقال محتمل $j \leftarrow i$ من موقع i إلى موقع آخر j، واذا كانت المشكلة بها عدد N من المواقع، فإن عدد الانتقالات المُمكنة يساوي $N \times (N-1)$. إذا كانت x_{ij} تساوي 1، فإن الحلّ يتضمن الانتقال من الموقع i إلى الموقع j، وخلاف ذلك إذا كانت x_{ij} تساوي 0، فلن يُدرج هذا الانتقال في الحلّ.

يُمكن الوصول بسهولة إلى العناصر في مصفوفة numpy ثنائية الأبعاد عبر الصيغة البرمجية [i,j]، فعلى سبيل المثال:

```
arr = numpy.full((4,4), 0) # creates a 4x4 array full of zeros

print(arr)

arr[0, 0] = 1
arr[3, 3] = 1

print()
print(arr)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[[1 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 1]]
```

يستخدِم المقطع البرمجي الأداة product( ) من المكتبة itertools لحساب جميع انتقالات المواقع المحتملة، فعلى سبيل المثال:

```
ids = {0, 1, 2}
for i, j in list(product(ids, ids)):
    print(i, j)
```

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
```

يستخدِم المقطع البرمجي التالي مكتبة البايثون mip لإنشاء خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة، ثم يضيف متغيِّر قرار ثُنائي لكل انتقال ممكن في نسخة مشكلة البائع المُتجوّل التي تم إنشاؤها سابقًا:

```
from itertools import product # used to generate all possible transition
from mip import BINARY
from mip import Model,INTEGER

solver = Model() # creates a solver
solver.verbose = 0 # setting this to 1 will print info on the progress of the solver

# 'product' creates every transition from every location to every other location
transitions = list(product(location_ids, location_ids))

N = len(location_ids) # number of locations

# creates a square numpy array full of 'None' values
x = numpy.full((N, N), None)

# adds binary variables indicating if transition (i->j) is included in the route
for i, j in transitions:
    x[i, j] = solver.add_var(var_type = BINARY)
```

يستخدِم المقطع البرمجي السابق أداة numpy.full( ) لإنشاء مصفوفة numpy بحجم NxN لتخزين المتغيِّرات الثنائية x.

بعد إضافة متغيِّرات القرار x، يُمكن استخدام المقطع البرمجي التالي لصياغة وحساب الدالة الموضوعية لمشكلة البائع المُتجوّل، حيث تقوم الدالة بالتكرار على كل انتقال ممكن $i \rightarrow j$ وتُضرب مسافتها dist_matrix[i,j] مع متغيِّر قرارها x[i,j]، وإذا تم إدراج الانتقال في الحلّ سيؤخذ x[i,j] = 1 و dist_matrix[i,j] بعين الاعتبار، وبخلاف ذلك ستُضرب dist_matrix[i,j] في صفر ليتم تجاهُلُها:

```
# the minimize tool is used then the objective function has to be minimized
from mip import xsum, minimize

# objective function: minimizes the distance
solver.objective = minimize(xsum(dist_matrix[i,j]*x[i][j] for i,j in
transitions))
```

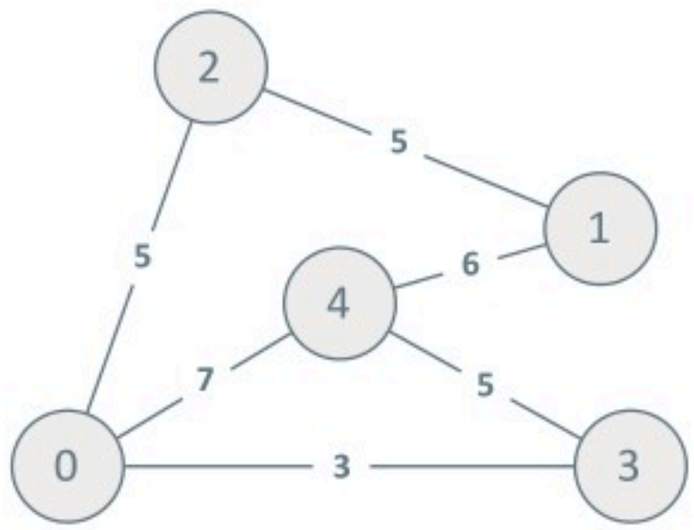
تهدف الخطوة التالية إلى التأكد بأن الخوارزمية تُظهر الحلول التي تضمن زيارة كل المواقع لمرَّة واحدة فقط، باستثناء موقع startstop (الانطلاق والتوقف) حسب ما تتطلبه مشكلة البائع المُتجوّل، وزيارة كل موقع مرة واحدة تعني أن الطريق الصحيح يُمكن أن:

- يصل إلى كل موقع مرة واحدة فقط.
- يغادر من كل موقع مرة واحدة فقط.

ويُمكن إضافة قيود الوصول والمغادرة هذه بسهولة كما يلي:

```
# for each location id
for i in location_ids:
    solver += xsum(x[i,j] for j in location_ids - {i}) == 1 # exactly 1 arrival
    solver += xsum(x[j,i] for j in location_ids - {i}) == 1 # exactly 1 departure
```

تشمل الصيغة الكاملة لمشكلة البائع المُتجوّل نوعًا إضافيًا آخرًا من القيود لضمان حساب الطُرق المتصلة، ففي نسخة مشكلة البائع المُتجوّل الواردة في الشكل 5.8 يُفترض أن الموقع 0 هو موقع الانطلاق والتوقف.



شكل 5.8: نسخة مشكلة البائع المُتجوّل

في هذا المثال، أقصر طريق ممكن هو 0 ← 3 ← 4 ← 1 ← 2، بمسافة سفر إجمالية قدرها 24، ولكن عند عدم وجود قيد اتصال سيكون هناك حلّ صحيح آخر يشمل طريقين غير متصلين هما: 0 ← 3 ← 4 ← 0 و 1 ← 2 ← 1، وهذا الحلّ المتمثل في وجود طريقين يمتثل لقيود الوصول والمغادرة التي تم تعريفها في المقطع البرمجي السابق؛ لأن كل موقع يدخل له ويخرج منه مرة واحدة فقط، ولكن هذا الحلّ غير مقبول لمشكلة البائع المُتجوّل.

يُمكن فرض حلّ يشمل طريقًا واحدًا متصلًا بإضافة متغيِّر القرار y_i لكل موقع i، وستحافظ هذه المتغيِّرات على ترتيب زيارة كل موقع في الحلّ.

```
# adds a decision variable for each location
y = [solver.add_var(var_type = INTEGER) for i in location_ids]
```

على سبيل المثال، إذا كان الحلّ هو: 0 ← 3 ← 4 ← 1 ← 2 ← 0، فستكون قيم y كما يلي: $y_1=2$، $y_4=1$، $y_3=0$، $y_2=3$، والموقع 0 هو موقع الانطلاق والتوقف، ولذلك لا تؤخذ قيمة y الخاصة به بعين الاعتبار.

يُمكن استخدام متغيِّرات القرار الجديدة هذه لضمان الاتصال من خلال إضافة قيد جديد لكل انتقال $i \leftarrow j$ لا يشمل موقع startstop (الانطلاق والتوقف).

```
# adds a connectivity constraint for every transition that does not include the startstop
for (i, j) in product(location_ids - {startstop}, location_ids - {startstop}):
    # ignores transitions from a location to itself
    if i != j:
        solver += y[j] - y[i] >= (N+1) * x[i, j] - N
```

إذا كانت $x_{ij}=1$ لانتقال $i \leftarrow j$ وتم إدراج هذا الانتقال في الحلّ، فإن المتباينة الواردة في الأعلى تصبح y[j] >= y[i] + 1، ومعنى ذلك أن المواقع التي ستُزارُ لاحقًا لا بد أن تكون قيمة y الخاصة بها أعلى، بالإضافة إلى قيود الوصول والمغادرة، وسيكون الطريق الذي لا يشمل موقع الانطلاق والتوقف صحيحًا فقط إذا:

- بدأ وانتهى بالموقع نفسه؛ لضمان أن يكون لكل موقع وصولٌ واحدٌ ومغادرة واحدة فقط.
- خُصصت قيم y أعلى لكل المواقع التي ستُزارُ لاحقًا؛ لأن y[j] يجب أن تكون أكبر من y[i] لكل الانتقالات التي تم إدراجها في الطريق، ويؤدي هذا أيضًا إلى تجنب إضافة الحافة نفسها من اتجاه مُختلف، على سبيل المثال: $i \leftarrow j$ و $j \leftarrow i$

ولكن إذا كان الموقع يمثل بداية الطريق ونهايته، فلا بدّ أن تكون قيمة y الخاصة به هي أكبر وأصغر من قيم كل المواقع الباقية في الطريق، ونظرًا لاستحالة هذا الأمر، فستُؤدي إضافة قيد الاتصال إلى استبعاد أية حلول بها طُرق لا تشمل موقع الانطلاق والتوقف.

على سبيل المثال، فكِّر في الطريق 1 ← 2 ← 1 الوارد في الحلّ المُكوَّن من طريقين لنسخة مشكلة البائع المُتجوّل الموضحة في الشكل السابق، حيث يتطلب قيد الاتصال أن تكون $y_2 \geq y_1 + 1$ وأن تكون $y_1 \geq y_2 + 1$، وهذا مستحيل، فلذلك سيتم استبعاد الحلّ.

في المقابل، يتطلب الحلّ الصحيح 0 ← 3 ← 4 ← 1 ← 2 ← 0 أن تكون $y_4 \geq y_3 + 1$ وأن تكون $y_1 \geq y_4 + 1$، وأن تكون $y_2 \geq y_1 + 1$، ويُمكن تحقيق ذلك بضبط قيم y كما يأتي: $y_3=0$ و $y_4=1$ و $y_1=2$ و $y_2=3$، ولا تنطبق قيود الاتصال على الانتقالات التي تشمل موقع startstop (الانطلاق والتوقف).

تَجمع الدالة التالية كل الأشياء معًا لإنشاء خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لمشكلة البائع المُتجوّل:

```
from itertools import product
from mip import BINARY,INTEGER
from mip import Model
from mip import xsum, minimize

def MIP_solver(dist_matrix, location_ids, startstop):
    solver = Model()# creates a solver
    solver.verbose = 0 # setting this to 1 will print info on the progress of the solver
    # creates every transition from every location to every other location
    transitions = list(product(location_ids,location_ids))
    N = len(location_ids) # number of locations
    # create an empty square matrix full of 'None' values
    x = numpy.full((N, N), None)
    # adds binary decision variables indicating if transition (i->j) is included in the route
    for i, j in transitions:
        x[i, j]=solver.add_var(var_type = BINARY)
    # objective function: minimizes the distance
    solver.objective = minimize(xsum(dist_matrix[i,j]*x[i][j] for i,j in transitions))
    # Arrive/Depart Constraints
    for i in location_ids:
        solver += xsum(x[i,j] for j in location_ids - {i}) == 1 # exactly 1 arrival
        solver += xsum(x[j,i] for j in location_ids - {i}) == 1 # exactly 1 departure
    # adds a binary decision variable for each location
    y = [solver.add_var(var_type=INTEGER) for i in location_ids]
    # adds connectivity constraints for transitions that do not include the startstop
    for (i, j) in product(location_ids - {startstop}, location_ids - {startstop}):
        if i != j: # ignores transitions from a location to itself
            solver += y[j] - y[i] >=(N+1)*x[i,j] - N
    solver.optimize() #solves the problem
    # prints the solution
    if solver.num_solutions: # if a solution was found
        best_route = [startstop] # stores the best route
        curr_loc = startstop # the currently visited location
        while True:
            for next_loc in location_ids:# for every possible next location
                if x[curr_loc,next_loc].x == 1: # if x value for the curr_loc->next_loc transition is 1
                    best_route.append(next_loc) # appends the next location to the route
                    curr_loc=next_loc # visits the next location
                    break
            if next_loc == startstop: # exits if route returns to the startstop
                break
    return best_route, solver.objective_value # returns the route and its total distance
```

يولِّد المقطع البرمجي التالي 100 نسخة من مشكلة البائع المُتجوّل تشمل 8 مواقع وتتراوح المسافات فيها بين 5 و20، كما أنه يستخدِم خوارزمية حلّ القوة المُفرطة، وخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لحلّ كل حالة، ويُظهِر النسبة المئوية للأسلوبين اللذين أظهرا طريقين لهما المسافة نفسها:

```
same_count = 0
for i in range(100):
    dist_matrix, location_ids, startstop=create_problem_instance(8, [5,20])
    route1, dist1 = brute_force_solver(dist_matrix, location_ids, startstop)
    route2, dist2 = MIP_solver(dist_matrix, location_ids, startstop)
    # counts how many times the two solvers produce the same total distance
    if dist1 == dist2:
        same_count += 1
print(same_count / 100)
```

```
1.0
```

تؤكد النتائج أن خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة تُظهِر الحلّ الأمثل بنسبة 100% لكل نُسخ المشكلة، ويوضِّح المقطع البرمجي التالي سرعة خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة من خلال استخدامها لحلّ 100 نسخة كبيرة تتضمن كلٌّ منها 20 موقعًا:

```
import time

start = time.time() # starts timer
for i in range(100):
    dist_matrix, location_ids, startstop = create_problem_instance(20, [5,20])
    route, dist = MIP_solver(dist_matrix, location_ids, startstop)

stop=time.time() # stops timer
print(stop - start) # prints the elapsed time in seconds
```

```
188.90074133872986
```

على الرغم من أن وقت التنفيذ الدقيق سيعتمد على قوة معالجة الجهاز الذي تستخدِمه لتنفيذ مفكرة جوبيتر، إلا أنه من المفترض أن يستغرق التنفيذ بضع دقائق لحساب الحلّ لجميع مجموعات البيانات المئة.

وهذا بدوره مذهل إذا تم الأخذ في الاعتبار أن عدد الطُرق المُمكنة لكل نسخة من النُسخ المئة هي: 121,645,100,000,000,000 = !19 طريقًا مُختلفًا، ومثل هذا العدد الكبير من الطُرق يفوق بكثير قدرات أسلوب القوة المُفرطة، ومع ذلك فإنه عن طريق البحث الفعّال في هذه المساحة الهائلة الخاصة بجميع الحلول المُمكنة يُمكن لخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة أن تجد الطريق الأمثل بسرعة.

وعلى الرغم من مزايا البرمجة الرياضية إلا أنها تملك قيودًا خاصة أيضًا، فهي تتطلب فهمًا قويًا للنمذجة الرياضية وقد لا تكون مناسبة للمشكلات المعقدة التي يصعب فيها التعبير عن الدالة الموضوعية والقيود بواسطة الصيغ الرياضية، وعلى الرغم من أن البرمجة الرياضية أسرع بكثير من أسلوب القوة المُفرطة إلا أنها قد تظل بطيئة جدًا بالنسبة لمجموعات البيانات الكبيرة، وفي مثل هذه الحالات يقدِّم الأسلوب الاستدلالي الموضَّح في الدرسين السابقين بديلًا أكثر سرعة.

تمرينات

1 اشرح طريقة استخدام البرمجة الرياضية لحلّ مشكلات التحسين المعقدة.

2 ما مزايا وعيوب أسلوب برمجة الأعداد الصحيحة المختلطة في حلّ مشكلات التحسين؟

3 قم بتحليل مشكلتين من مشكلات التحسين يُمكن حلهما باستخدام البرمجة الرياضية، ثم حدِّد متغيِّرات الحالة ومتغيِّرات القرار الخاصة بهما.

4 اذكر ثلاث مشكلات تحسين مُختلفة من عائلة مشكلات تحديد المسار.

5 أنشئ دالة خوارزمية حلّ القوة المُفرطة لمشكلة البائع المُتجوّل، من خلال إكمال المقطع البرمجي التالي بحيث تُظهر الدالة المسار الأفضل والمسافة الإجمالية المُثلى:

```
from itertools import permutations


def brute_force_solver(dist_matrix, location_ids, startstop):
    # excludes the startstop location
    location_ids = _____________ - {_______________}
    # generates all possible routes (location permutations)
    all_routes = ________________(_________________)
    best_distance = float('inf') # initializes to the highest possible number
    best_route = None  # best route so far, initialized to None

    for route in all_routes: # for each route
        distance = 0 # total distance in this route
        curr_loc = _____________ # current location

        for next_loc in route:
            distance += _________[curr_loc, next_loc] # adds the distance of this step
            curr_loc = ________________ # goes the next location
        distance += ___________________[curr_loc, _______________] # goes to
back to the startstop location
        if distance < best_distance: # if this route has lower distance than the best route
            best_distance = distance
            best_route = route

    # adds the startstop location at the beginning and end of the best route and returns
    return [startstop] + list(best_route) + [startstop], best_distance
```

6 أنشئ خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لمشكلة البائع المُتجوّل، من خلال إكمال المقطع البرمجي التالي، بحيث تنتقي متغيّرات القرار وقيود الاتصال انتقاءً صحيحًا:

```
def MIP_solver(dist_matrix, location_ids, startstop):

    solver = _____________() # creates a solver
    solver.verbose = 0  # setting this to 1 will print info on the progress of the solver
    # creates every transition from every location to every other location

    transitions = list(_________________(location_ids, location_ids))
    N = len(location_ids) # number of locations
    # creates an empty square matrix full of 'None' values
    x = numpy.full((N, N), None)
    # adds binary decision variables indicating if transition (i->j) is included in the route
    for i, j in transitions:

        x[i, j] = solver._______________(var_type=_________________)

    # objective function: minimizes the distance

    solver.objective = _________________(xsum(dist_matrix[i, j] * x[i][j] for
i, j in transitions))

    # Arrive/Depart Constraints
    for i in location_ids:

        solver += xsum(_________________ for j in location_ids - {i}) == 1

        solver += xsum(_________________ for j in location_ids - {i}) == 1

    # Adds a binary decision variable for each location

    y = [solver._____________________(var_type=_____________________) for i in
location_ids]

    # Adds connectivity constraints for transitions that do not include the startstop
    for (i, j) in product(location_ids - {startstop}, location_ids -
{startstop}):
        if i != j: # ignores transitions from a location to itself
            solver += y[j] - y[i] >= (N + 1) * x[i, j] - N


    solver._______________() # solves the problem
```

المشروع

افترض أنك تعمل في شركة توصيل، وطلب منك مديرك أن تجد المسار الأكثر كفاءة لتوصيل الطرود إلى مواقع متعددة في المدينة.

يتمثّل الهدف في إيجاد أقصر مسار ممكن لزيارة كل موقع مرة واحدة فقط ومن ثمَّ العودة إلى موقع البدء. هذه المشكلة مثال على **مشكلة البائع المُتجوّل** (TSP).

1 ستقوم بإنشاء أمثلة متعددة على مشكلة البائع المُتجوّل تشمل مواقع عددها من 3 إلى 12، وستتراوح المسافة في كل مثال من 5 وحدات إلى 20 وحدة.

2 أنشئ دالة رسم نقاط باستخدام مكتبة matplotlib ترسم أفضل مسار تُنتجه خوارزمية الحلّ، يمكنك استخدام هذه الدالة فقط مع النسخة التي تشمل 20 موقعًا.

3 أنشئ دالة رسم نقاط باستخدام مكتبة matplotlib ترسم نقاط أداء كل من خوارزمية حلّ القوة المُفرطة وخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة بالمقارنة بينهما.

4 اكتب تقريرًا موجزًا تناقش فيه النتائج التي توصلت إليها بخصوص كفاءة أداء خوارزميتي الحلّ، ومزايا وعيوب كل منهما.

ماذا تعلّمت

- تحديد أساليب التحسين الملائمة لحلّ المشكلات المعقدة.
- حلّ مشكلات تخصيص الموارد عن طريق تطبيق مقطع برمجي بلغة البايثون.
- حلّ مشكلات الجدولة عن طريق تطبيق مقطع برمجي بلغة البايثون.
- حلّ مشكلة حقيبة الظهر باستخدام خوارزميات التحسين المختلفة.
- حلّ مشكلة البائع المُتجوّل باستخدام خوارزميات التحسين المختلفة.

المصطلحات الرئيسة

البرمجة الرياضية	Mathematical Programming
برمجة الأعداد الصحيحة المختلطة	Mixed Integer Programming
مشكلة التحسين	Optimization Problem
البرمجة الرباعية	Quadratic Programing
مشكلة الجدولة	Scheduling Problem
تشكيل فريق	Team Formation
مشكلة البائع المُتجوّل	Traveling Salesman Problem

خوارزمية حلّ القوة المُفرطة	Brute-Force Solver
البرمجة القيدية	Constraint Programming
خوارزمية استدلالية جشعة	Greedy Heuristic Algorithm
خوارزمية حلّ جشعة	Greedy Solver
برمجة الأعداد الصحيحة	Integer Programming
خوارزمية حلّ مشكلة حقيبة الظهر	Knapsack Problem Solver

6. الذكاء الاصطناعي والمجتمع

سيتعرّف الطالب في هذه الوحدة على أخلاقيات الذكاء الاصطناعي وتأثيرها على تطوير أنظمته المتقدمة وتحديد توجهاتها، وسيُقيِّم مدى تأثير أنظمة الذكاء الاصطناعي واسعة النطاق على المجتمعات والبيئة، وكيفية تنظيم مثل هذه الأنظمة للاستخدام الأخلاقي المُستدام، وسيستخدِم بعد ذلك مُحاكي ويبوتس (Webots) لبرمجة طائرة مُسيَّرة على الحركة الذاتية واستكشاف منطقة ما من خلال تحليل الصور.

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُعرِّف أخلاقيات الذكاء الاصطناعي.
- يُفسِّر مدى تأثير التحيُّز والإنصاف على الاستخدام الأخلاقي لأنظمة الذكاء الاصطناعي.
- يُقيِّم كيفية حل مشكلة الشفافية وقابلية التفسير في الذكاء الاصطناعي.
- يُحلِّل كيفية تأثير أنظمة الذكاء الاصطناعي واسعة النطاق على المجتمع وكيفية وضع قوانين لتنظيمها.
- يُبرمج جهاز الطائرة المُسيَّرة على الحركة الذاتية.
- يُطوِّر نظام تحليل الصور لطائرة مُسيَّرة تُستخدم في استطلاع منطقة معينة.

الأدوات

- ويبوتس (Webots)
- مكتبة أوبن سي في (OpenCV Library)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

مقدمة في أخلاقيات الذكاء الاصطناعي

نظرة عامة على أخلاقيات الذكاء الاصطناعي
Overview of AI Ethics

مع استمرار تقدُّم الذكاء الاصطناعي تزايدت أهمية التفكير في الآثار الأخلاقية المترتبة على استخدام هذه التقنية، ومن المهم أن يفهم المواطن في عَالَمنا الحديث الدور الهام لأخلاقيات الذكاء الاصطناعي إذا أردنا تطوير أنظمة ذكاء اصطناعي مسؤولة واستخدامها. إن أحد الأسباب الرئيسة للتأكيد على أهمية أخلاقيات الذكاء الاصطناعي هو التأثير الكبير لأنظمة الذكاء الاصطناعي على حياة الانسان. على سبيل المثال، يُمكن استخدام خوارزميات الذكاء الاصطناعي لاتخاذ قرارات التوظيف والعلاج الطبي، وإذا كانت هذه الخوارزميات مُتحيِّزة أو تمييزية، فقد تُؤدي إلى نتائج غير عادلة تُضر بالأفراد والمجتمعات.

أخلاقيات الذكاء الاصطناعي (AI Ethics):

تشير أخلاقيات الذكاء الاصطناعي إلى المبادئ، والقيم، والمعايير الأخلاقية التي تُنظِّم تطوير أنظمة الذكاء الاصطناعي وانتشارها واستخدامها.

أمثلة من العَالَم الواقعي على المخاوف الأخلاقية في مجال الذكاء الاصطناعي
Real-World Examples of Ethical Concerns in AI

الخوارزميات التمييزية

هناك مواقف تدل على أن أنظمة الذكاء الاصطناعي تميل إلى التحيُّز والتمييز ضد فئات معيّنة من البشر. على سبيل المثال، وجدت دراسة أجراها **المعهد الوطني للمعايير والتقنية** (National Institute of Standards and Technology) أن نِسب الخطأ في تقنية التعرُّف على الوجه تكون أعلى عند التعرُّف على وجوه الأشخاص ذوي البشرة الداكنة؛ مما قد يُؤدي إلى تحديد هويات خاطئة واعتقالات خاطئة. ومن الأمثلة الأخرى على ذلك استخدام خوارزميات الذكاء الاصطناعي في نظام العدالة الجنائية، إذ أظهرت الدراسات أن هذه الخوارزميات يُمكن أن تكون مُتحيِّزة ضد الأقليات مما يُؤدي إلى عقوبات أقسى.

انتهاك الخصُوصية

يُمكن أن تكون أنظمة الذكاء الاصطناعي التي تجمع البيانات وتُحلِّلها مصدر تهديد للخصُوصية الشخصية. على سبيل المثال: جمعت شركة استشارات سياسية في عام 2018 م بيانات الملايين من مستخدِمي **فيسبوك** (Facebook) دون موافقتهم واستخدمتها للتأثير على الحملات السياسية، وأثار هذا الحادث المخاوِف بشأن استخدام الذكاء الاصطناعي وتحليلات البيانات في التلاعب بالرأي العام، وانتهاك حقوق خصوصية الأفراد.

الأسلحة ذاتية التحكم

تطوير الأسلحة ذاتية التحكم التي يُمكن أن تعمل دون تدخل بشري له مخاوف أخلاقية بشأن استخدام الذكاء الاصطناعي في الحروب، حيث يرى فريق من المنتقدين أن هذه الأسلحة يُمكن أن تتخذ قرارات مصيرية دون إشراف بشري ويُمكن برمجتها لاستهداف مجموعات معيّنة من الناس، مما قد ينتهك القانون الإنساني الدولي، ويُؤدي إلى وقوع إصابات في صفوف المدنيين.

التسريح من الوظائف

أثار الاستخدام المتزايد للذكاء الاصطناعي **والأتمتة** (Automation) في مختلف الصناعات المخاوِف بشأن تسريح البشر من وظائفهم وتأثيره على سُبل عيش العاملين، فعلى الرغم من أن الذكاء الاصطناعي يُمكنه أن يُؤدي إلى تحسين الكفاءة والإنتاجية، إلا أنه يُمكن أن يُؤدي أيضًا إلى فقدان البشر لوظائفهم وتزايد عدم المساواة في الدخل؛ مما قد يكون له عواقب اجتماعية واقتصادية سلبية.

التحيُّز والإنصاف في الذكاء الاصطناعي Bias and Fairness in AI

يُمكن أن يظهر **التحيُّز** (Bias) في أنظمة الذكاء الاصطناعي عندما تكون البيانات المستخدَمة لتدريب الخوارزميّة ناقصة التمثيل أو تحتوي على تحيُّزات أساسية، ويُمكن أن يظهر في أية بيانات تُمثِّلها مُخرَجات النظام، فعلى سبيل المثال لا الحصر: المُنتَجات والآراء والمجتمعات والاتجاهات كلها يمكن أن يظهر فيها التحيُّز.

يُعدُّ نظام التوظيف الآلي الذي يستخدِم الذكاء الاصطناعي لفحص المرشحين للوظائف من أبرز الأمثلة على الخوارزميّة المُتحيِّزة. افترض أن الخوارزميّة مُدرَّبة على بيانات مُتحيِّزة، مثل أنماط التوظيف التاريخية التي تُفضِّل مجموعات ديموغرافية معيّنة، ففي هذه الحالة قد يعمل الذكاء الاصطناعي على استمرار تلك التحيُّزات ويستبعد المرشّحين المؤهّلين بشكل غير عادل من بين المجموعات متجاهلًا الفئات غير الممثَّلة جيدًا في مجموعة البيانات. على سبيل المثال، افترض أن الخوارزميّة تُفضل المرشحين الذين التحقوا بجامعات النخبة، أو عملوا في شركات مرموقة، ففي هذه الحالة قد يلحق ذلك الضرر بالمرشحين الذين لم يحظوا بتلك الفُرص، أو الذين ينتمون إلى بيئات أقل حظًّا، ويُمكن أن يُؤدي ذلك إلى نقص التنوع في مكان العمل وإلى استمرارية عدم المساواة، ولذلك من المهم تطوير واستخدام خوارزميات توظيف للذكاء الاصطناعي تَستنِد على معايير عادلة وشفافة، وغير مُتحيِّزة.

يشير **الإنصاف** (Fairness) في الذكاء الاصطناعي إلى كيفية تقديم أنظمة الذكاء الاصطناعي لنتائج غير مُتحيِّزة وعلى معاملتها لجميع الأفراد والمجموعات مُعامَلة مُنصِفة، ولتحقيق الإنصاف في الذكاء الاصطناعي يتطلب ذلك تحديد التحيُّزات في البيانات والخوارزميات وعمليات اتخاذ القرار ومعالجتها. على سبيل المثال، تتمثّل إحدى طرائق تحقيق الإنصاف في الذكاء الاصطناعي في استخدام عملية تُسمى **إلغاء الانحياز** (Debiasing)، حيث يتم تحديد البيانات المُتحيِّزة وإزالتها أو تعديلها بما يضمن وصول الخوارزميّة إلى نتائج أكثر دقة دون تحيُّز.

تحيُّز الذكاء الاصطناعي (AI Bias):

في مجال الذكاء الاصطناعي، يدل التحيُّز على ميل خوارزميات التعلُّم الآلي إلى إنتاج نتائج تحابي بدائل، أو فئات معيّنة، أو تظلمها بأسلوب منهجي؛ مما يؤدي إلى القيام بتنبؤات خاطئة وإلى احتمالية التمييز ضد مُنتَجات معيّنة أو فئات بشرية محدَّدة.

جدول 6.1: العوامل التي تُحدِّد تحيُّز أنظمة الذكاء الاصطناعي

بيانات التدريب المُتحيِّزة	تتعلّم خوارزميات الذكاء الاصطناعي من البيانات التي تُدرَّب عليها؛ فإذا كانت البيانات مُتحيِّزة أو ناقصة التمثيل، فقد تصل الخوارزميّة إلى نتائج مُتحيِّزة. على سبيل المثال، إذا تم تدريب خوارزميّة التعرُّف على الصور على مجموعة بيانات تحتوي في الغالب على أفراد ذوي بشرة فاتحة، فربما تُواجه صعوبة في التعرُّف بدقة على الأفراد ذوي البشرة الداكنة.
الافتقار إلى التنوُّع في فِرق التطوير	إذا لم يكن فريق التطوير متنوعًا ولا يُمثِّل نطاقًا واسعًا من الفئات الثقافية والتقنية، فقد لا يتعرَّف على التحيُّزات الموجودة في البيانات أو الخوارزميّة، ويؤدي الفريق الذي يتكون من أفراد من منطقة جغرافية أو ثقافة معيّنة إلى عدم مراعاة المناطق أو الثقافات الأخرى التي قد تكون مُمثَّلة في البيانات المُستخدَمة لتدريب نموذج الذكاء الاصطناعي.
الافتقار إلى الرقابة والمسؤولية	يُمكن أن يُؤدي الافتقار إلى الرقابة والمسؤولية في تطوير أنظمة الذكاء الاصطناعي ونشرها إلى ظهور التحيُّز، فإذا لم تُطبِّق الشركات والحكومات آليات رقابة ومُساءلة مناسبة، فإنّ ذلك قد يؤدي إلى عدم تنفيذ اختبار التحيُّز في أنظمة الذكاء الاصطناعي وربما لا يكون هناك مجال لإنصاف الأفراد أو المجتمعات المتضررة من النتائج المُتحيِّزة.
الافتقار إلى الخبرة والمعرفة لدى فريق التطوير	قد لا تُحدِّد فِرق التطوير التي تفتقر إلى الخبرة مؤشرات التحيُّز في بيانات التدريب أو تُعالجها، كما أن الافتقار إلى المعرِفة في تصميم نماذج الذكاء الاصطناعي واختبارِها لتحقيق العدالة ربما يُؤدي إلى استمرارية التحيُّزات القائمة.

الحدّ من التحيُّز وتعزيز الإنصاف في أنظمة الذكاء الاصطناعي
Reducing Bias and Promoting Fairness in AI Systems

البيانات المتنوعة والمُمثَّلة

يُقصد بذلك استخدام البيانات التي تعكس تنوع المجموعة التي يتم تمثيلها، كما أنه من المهم مراجعة وتحديث البيانات المُستخدَمة لتدريب أنظمة الذكاء الاصطناعي بانتظام؛ للتأكد من أنها ما زالت ملائمة وغير مُتحيِّزة.

تقنيات إلغاء الانحياز

تتضمن أساليب إلغاء الانحياز تحديد وإزالة البيانات المُتحيِّزة من أنظمة الذكاء الاصطناعي؛ لتحسين معايير الدقة والإنصاف، فتشمل هذه التقنيات مثلًا: **زيادة العينات** (Oversampling) أو **تقليل العينات** (Undersampling) أو **زيادة البيانات** (Data Augmentation) لضمان تعرُّض نظام الذكاء الاصطناعي لنقاط بيانات مختلفة.

القابلية للتفسير والشفافية

إنّ جعل أنظمة الذكاء الاصطناعي أكثر شفافية وأكثر قابلية للتفسير يمكنه أن يساعد في تقليص مستوى التحيُّز من خلال السماح للمُستخدِمين بفهم كيفية اتخاذ النظام للقرارات، ويتضمن ذلك توضيح عملية اتخاذ القرار والسماح للمُستخدِمين باستكشاف مُخرَجات النظام واختبارها.

التصميم المعتمد على إشراك الإنسان

يُمكن أن يساهم إشراك العنصر البشري في حلقة تصميم أنظمة الذكاء الاصطناعي في التقليل من التحيُّز، وذلك بالسماح للبشر بالتدخل وتصحيح مُخرَجات النظام عند الضرورة، ويشمل ذلك تصميم أنظمة ذكاء اصطناعي بها مرحلة للتغذية الراجعة تُمكِّن البشر من مراجعة قرارات النظام والموافقة عليها.

المبادئ الأخلاقية

تعني دمج المبادئ الأخلاقية مثل: الإنصاف والشفافية والمساءلة، في تصميم وتنفيذ أنظمة الذكاء الاصطناعي، من أجل ضمان تطوير تلك الأنظمة واستخدامها بشكل أخلاقي ومسؤول، وذلك بوضع إرشادات أخلاقية واضحة لاستخدام أنظمة الذكاء الاصطناعي ومراجعة هذه الإرشادات بانتظام وتحديثها عند الضرورة.

المراقبة والتقييم بانتظام

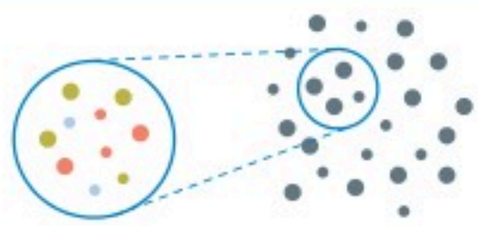
تُعدُّ المراقبة والتقييم بشكل دوري لأنظمة الذكاء الاصطناعي أمرًا ضروريًا لتحديد التحيُّز وتصحيحه، ويتضمن ذلك اختبار مُخرَجات النظام وإجراء عمليات تدقيق منتظمة؛ للتأكد من أن النظام يعمل بشكل عادل ودقيق.

تقييم تغذية المُستخدِم الراجعة

يُمكن أن تساعد التغذية الراجعة التي يقدمها المُستخدِم في تحديد التحيُّز في النظام؛ لأن المُستخدِمين غالبًا ما يكونون أكثر وعيًا بتجاربهم، ويُمكنهم تقديم رؤى عن التحيُّز المحتمل أفضل مما يُمكن أن تقدمه خوارزميات الذكاء الاصطناعي. على سبيل المثال، يُمكن أن يقدِّم المُستخدِمون تغذية راجعة عن رؤيتهم لأداء نظام الذكاء الاصطناعي أو تقديم اقتراحات مفيدة لتحسين النظام وجعله أقل تحيُّزًا.

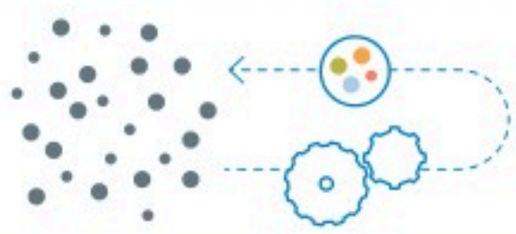
زيادة العينات (Oversampling):

تُشير زيادة العينة في تعلُّم الآلة إلى زيادة عدد عيّنات فئة ما داخل مجموعة بيانات لتحسين دقة النموذج، ويكون ذلك بواسطة المضاعفة العشوائية للعيّنات الموجودة في الفئة أو توليد عيّنات جديدة من الفئة نفسها.



تقليل العينات (Undersampling):

تقليل العينة هو عملية تقليل حجم مجموعة البيانات بحذف مجموعة فرعية من بيانات الفئة الأكبر للتركيز على العيّنات الأكثر أهمية. ويكون ذلك مفيدًا بشكل خاص إذا كانت مجموعة البيانات تفتقر إلى التوازن بين الفئات أو بين مجموعاتها المختلفة.



زيادة البيانات (Data Augmentation):

زيادة البيانات هي عملية توليد بيانات تدريب جديدة من البيانات الموجودة لتعزيز أداء نماذج تعلُّم الآلة، ومن الأمثلة على ذلك: **قلب الصور** (Image Flipping) وتدويرها وقصها وتغيير ألوانها وتحويلها **تحويلًا تآلفيًا** (Affine Transformation) والتشويش عليها.

مشكلة المسؤولية الأخلاقية في الذكاء الاصطناعي
The Problem of Moral Responsibility in AI

تُعدُّ مشكلة المسؤولية الأخلاقية عند استخدام أنظمة الذكاء الاصطناعي المتقدمة قضية مُعقَّدة ومتعددة الجوانب، وقد حظيت باهتمام كبير في السنوات الأخيرة.

تتمثّل إحدى التحديات الرئيسة لأنظمة الذكاء الاصطناعي المتقدمة في قدرتها على اتخاذ القرارات والقيام بإجراءات يُمكن أن يكون لها عواقب إيجابية أو سلبية كبيرة على الأفراد والمجتمع، ورغم ذلك، لا يكون الطرف الذي يجب تحميله المسؤولية الأخلاقية عن هذه النتائج محدّدًا دائمًا.

هناك رأي يقول: إن مطوِّري ومصمِّمي أنظمة الذكاء الاصطناعي يجب أن يتحملوا المسؤولية عن أي نتائج سلبية تَنتُج عن استخدامها، ويُؤكِّد هذا الرأي على أهمية ضمان تصميم أنظمة ذكاء اصطناعي تُراعي الاعتبارات الأخلاقية وتُحمِّل المطوِّرين المسؤولية عن أي ضرر قد تسببه اختراعاتهم.

ويرى آخرون أن المسؤولية عن نتائج الذكاء الاصطناعي هي مسؤولية مشتركة بين أصحاب المصلحة بما فيهم صُنّاع السياسات، والمنظمين ومُستخدِمي التقنية، ويسلط هذا الرأي الضوء على أهمية ضمان استخدام أنظمة الذكاء الاصطناعي بطرائق تتماشى مع المبادئ الأخلاقية، وتقييم المخاطر المرتبطة باستخدامها وإدارتها بعناية.

وهناك رأي ثالث يقول: إن أنظمة الذكاء الاصطناعي هي "ذاتٌ مسؤولة" لديها حسٌّ أخلاقي ومسؤولة عن أفعالها، وتقول هذه النظرية: إنّ أنظمة الذكاء الاصطناعي المُتقدِّمة يُمكن أن تتمتع بالفاعلية والاستقلالية؛ مما يجعلها أكثر من مجرد أدوات، كما تتطلب منها أن تكون مسؤولة عن أفعالها، إلا أن لهذه النظرية عدة مشكلات.

تستطيع أنظمة الذكاء الاصطناعي أن تُصْدِر أحكامًا وأن تتصرف من تلقاء نفسها، ولكنها ليست "ذاتًا مسؤولة" لديها حسٌّ أخلاقي وذلك للأسباب التالية:

أولًا: أن أنظمة الذكاء الاصطناعي تفتقر إلى الوعي والخبرات الذاتية؛ مما يُعدُّ سمة أساسية من سمات "الذات المسؤولة" التي لديها حسٌّ أخلاقي، وفي العادة تتضمن الفاعلية الأخلاقية القدرة على التفكير في المُثُل العليا للفرد وأفعاله.

ثانيًا: يقوم الأشخاص بتدريب أنظمة الذكاء الاصطناعي على اتباع قواعد وأهداف محدَّدة؛ مما يحدُّ من حكمها الأخلاقي، ويُمكن لأنظمة الذكاء الاصطناعي تكرار اتخاذ القرارات الأخلاقية، مع افتقارها للإرادة الحُرة والاستقلالية الشخصية.

وأخيرًا، فإن مُنشِئي أنظمة الذكاء الاصطناعي والقائمين على نشرها هم المسؤولون عن أفعالهم، ويُمكن لأنظمة الذكاء الاصطناعي أن تُساعد في اتخاذ القرارات الأخلاقية، على الرغم من أنها ليست "ذاتًا مسؤولة" لديها حسٌّ أخلاقي.

الشفافية وقابلية التفسير في الذكاء الاصطناعي ومشكلة الصندوق الأسود

Transparency and Explainability in AI and the Black-Box Problem

تكمن مشكلة الصندوق الأسود في الذكاء الاصطناعي في التحدي المُتمثِّل في فهم كيفية عمل **نظام قائم على الذكاء الاصطناعي** (AI-Based System) باتخاذ القرارات أو إنتاج المُخرَجات؛ مما قد يُصعِّب الوثوق بالنظام أو تفسيره أو تحسينه، وربما يؤثر الافتقار إلى الانفتاح وإلى قابلية التفسير على ثقة الناس في النموذج. تتزايد هذه التحديات بوجه خاص في مجال التشخيص الطبي، والأحكام التي تصدرها المركبات ذاتية القيادة. تُعدُّ التحيُّزات في نماذج تعلُّم الآلة إحدى المخاوف الأخرى المتعلقة بنماذج الصندوق الأسود، كما أن التحيُّزات الموجودة في البيانات التي يتم تدريب هذه النماذج عليها يُمكن أن تُؤدي إلى نتائج غير عادلة أو عنصرية. بالإضافة إلى ذلك، ربما يكون من الصعب تحديد المسؤولية عن القرارات التي يتخذها نموذج الصندوق الأسود؛ حيث يصعب تحميل أي شخص المسؤولية عن تلك القرارات لا سيما مع وجود الحاجة إلى الرقابة البشرية، كما هو الحال في أنظمة الأسلحة ذاتية التحكم. إن الافتقار إلى الشفافية في عملية اتخاذ القرار باستخدام الذكاء الاصطناعي يُصعِّب تحديد مشكلات النموذج وحلّها، كما أن عدم معرفة الطريقة التي يتخذ بها النموذج قراراته تجعل من الصعب إجراء التحسينات والتأكد من أنها تعمل بطريقة صحيحة، وهناك استراتيجيات عديدة لمعالجة مشكلة الصندوق الأسود في الذكاء الاصطناعي. تتمثّل إحدى تلك الاستراتيجيات في استخدام تقنيات ذكاء اصطناعي قابلة للتفسير لجعل نماذج تعلُّم الآلة أكثر شفافية وأكثر قابلية للتفسير، وقد تشمل هذه التقنيات: **مُفسرات اللغات الطبيعية** (Natural Language Explanation) أو تصوير البيانات للمساعدة في فهم عملية اتخاذ القرار، وهناك أسلوب آخر يتمثل في استخدام نماذج تعلُّم الآلة الأكثر قابلية للتفسير مثل: **أشجار القرار** (Decision Trees) أو **الانحدار الخطي** (Linear Regression)، وربما تكون هذه النماذج أقل تعقيدًا وأسهل في الفهم، ولكنها قد لا تكون قوية أو دقيقة مثل النماذج الأكثر تعقيدًا. تُعدُّ معالجة مشكلة الصندوق الأسود في الذكاء الاصطناعي أمرًا مهمًّا لبناء الثقة في نماذج تعلُّم الآلة وضمان استخدامها بأسلوب أخلاقي وعادل.

نظام الصندوق الأسود (Black-Box System):

هو نظام لا يكشف عن طرائق عمله الداخلية للبشر؛ إذ تتم التغذية بالمُدخَلات، ليتم إنتاج المُخرَجات دون معرِفة طريقة عملها، كما هو موضَّح في الشكل 6.1.

شكل 6.1: نظام الصندوق الأسود

طرائق تعزيز شفافية نماذج الذكاء الاصطناعي وقابليتها للتفسير

Methods for Enhancing the Transparency and Explainability of AI Models

النموذج المحايد المحلي القابل للتفسير والشرح

النموذج المحايد المحلي القابل للتفسير والشرح (Local Interpretable Model-Agnostic Explanations – LIME) تم استخدامه مسبقًا في مهام **معالجة اللغات الطبيعية** (NLP)، وتقوم هذه التقنية بتوليد تفسيرات محلية لتنبؤات مفردة يتم إجراؤها بواسطة نموذج، وتُنشئ هذه التفسيرات نموذجًا أبسط وقابلًا للتفسير يقارب نموذج الصندوق الأسود المُعقَّد حول تنبؤ محدَّد، ثم يُستخدم هذا النموذج البسيط لشرح كيف توصّل إلى قراره بشأن هذا التنبؤ المحدَّد. تتمثّل ميزة هذه التقنية في أنها تُوفر تفسيرات يُمكن للإنسان قراءتها، وبالتالي يُمكن لأصحاب المصلحة غير المتخصصين فهمها بسهولة؛ حتى فيما يتعلق بالنماذج المُعقَّدة مثل: **الشبكات العصبية العميقة** (Deep Neural Networks).

تفسيرات شابلي الإضافية

تفسيرات شابلي الإضافية (SHapley Additive exPlanations – SHAP) هي طريقة أخرى لتفسير مُخرَجات نماذج تعلُّم الآلة، وتعتمد على المفهوم الخاص بقيم شابلي من **نظرية الألعاب** (Game Theory) وتُخصِّص قيمة (أو وَزنًا) لكل خاصية

مساهمة في التنبؤ. يُمكن استخدام الطريقة مع أي نموذج، كما تقدم تفسيراتٍ في شكل درجات تبيّن أهميّة الخصائص، مما يُمكن أن يساعد في تحديد الخصائص الأكثر تأثيرًا في مُخرَجات النموذج.

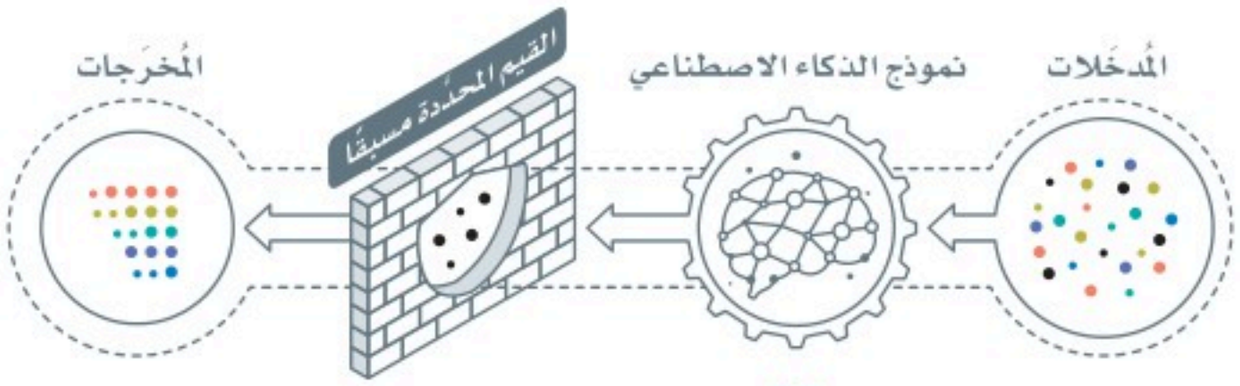
وهناك تقنية أخرى لتحسين قابلية تفسير الذكاء الاصطناعي مثل: أشجار القرار وقواعد القرار، وهي نماذج قابلة للتفسير يُمكن تصويرها بسهولة، حيث تقوم أشجار القرار بتقسيم **فضاء الخصائص** (Feature Space) بناءً على الخاصية الأكثر دلالة، وتقدِّم قواعد واضحة لاتخاذ القرارات، وتُعدُّ أشجار القرار مفيدة بشكل خاص عندما تتخذ البيانات شكل الجداول ويكون هناك عدد محدود من الخصائص. ولكن هذه النماذج محدودة أيضًا؛ لأن قابلية تفسير شجرة القرار التي تم إنشاؤها تتناسب تناسبًا عكسيًّا مع حجم الشجرة. على سبيل المثال، من الصعب فهم الأشجار التي تتكون من آلاف العقد ومئات المستويات. وأخيرًا، هناك أسلوب آخر يستخدِم تقنيات مثل: **وكلاء الذكاء الاصطناعي** (Artificial Intelligence Agents) أو **تحليل الحساسية** (Sensitivity Analysis) للمساعدة في فهم كيفية تأثير تغيير المُدخَلات أو الافتراضات على مُخرَجات النموذج، ويُمكن أن يكون هذا الأسلوب مفيدًا بشكل خاص في تحديد مصادر الغموض في النموذج وفي فهم حدوده.

الاستدلال القائم على القِيم في أنظمة الذكاء الاصطناعي
Value-Based Reasoning in AI Systems

يتمثّل الهدف من ذلك في إنشاء أنظمة ذكاء اصطناعي أكثر اتساقًا مع القيم والأخلاقيات البشرية؛ بحيث تتعامل هذه الأنظمة بطرائق مفيدة ومنصفة ومسؤولة. تتضمن الخطوة الأولى في الاستدلال القائم على القِيم، فهمَ وتمثيل القِيم الأخلاقية داخل أنظمة الذكاء الاصطناعي، حيث يجب أن تكون هذه الأنظمة قادرة على تفسير وتوطين القيم أو المبادئ التوجيهية الأخلاقية التي يُقدمها منشؤها البشريون أو أصحاب المصلحة، وقد تتضمن هذه العملية التعلُّم من الأمثلة أو التغذية الراجعة البشرية أو القواعد الواضحة، وعندما تفهم أنظمة الذكاء الاصطناعي هذه القيم بوضوح، يُمكنها أن تقوم بمواءمة أفعالها بطريقة أفضل مع المبادئ الأخلاقية المنشودة.

الاستدلال القائم على القِيم (Value-Based Reasoning):

الاستدلال القائم على القِيم في أنظمة الذكاء الاصطناعي يشير إلى العملية التي يستخدِمها وكلاء الذكاء الاصطناعي لاتخاذ قرارات أو استخلاص نتائج بناءً على مجموعة محدَّدة مسبقًا من القيم أو المبادئ أو الاعتبارات الأخلاقية.



شكل 6.2: تمثيل للاستدلال القائم على القِيم

يُركز الجانب الثاني من جوانب الاستدلال القائم على القِيم على تقييم القرارات أو الأفعال بناءً على **القِيم التي وُطنَت** (Internalized Values)، ويجب أن تقوم أنظمة الذكاء الاصطناعي بتقييم النتائج المحتملة للقرارات أو الإجراءات المختلفة بالنظر في عواقب كل خيار ومخاطره وفوائده، كما يجب أن تأخذ عملية التقييم هذه في الاعتبار القيم الأساسية التي تم تصميم نظام الذكاء الاصطناعي لدعمها، مما يضمن أن يتخذ النظام خيارات مستنيرة ومتوافقة مع القيم.

وأخيرًا، يتطلب الاستدلال القائم على القِيم من أنظمة الذكاء الاصطناعي اتخاذ قرارات تتماشى مع القِيم الراسخة، فبعد تقييم الخيارات المختلفة ونتائجها المحتملة، يجب على نظام الذكاء الاصطناعي أن ينتقي القرار أو الإجراء الذي يُمثِّل المبادئ والأهداف الأخلاقية التي صُمِّم لاتباعها، فمن خلال اتخاذ قرارات متوافقة مع القِيم، يمكن **لوكلاء الذكاء الاصطناعي** (AI Agents) التصرف بطرائق تتفق مع المبادئ التوجيهية الأخلاقية التي وضعها مُنشؤها؛ مما يعزِّز السلوك المسؤول والمفيد. على سبيل المثال: تُستخدم أنظمة الذكاء الاصطناعي في الرعاية الصحية للمساعدة في اتخاذ

قرارات التشخيص والعلاج، حيث يجب أن تكون هذه الأنظمة قادرة على التفكير في الآثار الأخلاقية المترتبة على العلاجات المختلفة مثل: الآثار الجانبية المحتملة أو التأثير على جودة الحياة، ومن ثَمّ تتخذ قرارات تُعطي الأولوية لسلامة المريض، ومن الأمثلة الأخرى: أنظمة الذكاء الاصطناعي المُستخدَمة في التمويل للمساعدة في اتخاذ قرارات الاستثمار، حيث يجب أن تكون هذه الأنظمة قادرة على أن تُفكر في الآثار الأخلاقية المترتبة على الاستثمارات المختلفة، كالتأثير على البيئة أو على الرعاية الاجتماعية، وبالتالي تتخذ القرارات التي تتماشى مع قِيم المستثمر.

يجب أن ندرك أن المسؤولية لا تقع بأكملها على عاتق نظام الذكاء الاصطناعي، بل إنها مسؤولية مشتركة بين الذكاء الاصطناعي والخبراء البشريين، فنظام الذكاء الاصطناعي يساعد في اتخاذ القرار بأن يُلخِّص الحالة ويقدِّم الخيارات أو العروض للمُستخدِم الخبير الذي يتخذ القرار النهائي؛ مما يؤكد أن الخبير البشري هو المتحكم والمسؤول عن النتيجة النهائية، في ظل الاستفادة من الأفكار والتحليلات التي يُوفرها نظام الذكاء الاصطناعي.

الذكاء الاصطناعي وتأثيره على البيئة AI and Environmental Impact

إن تأثير الذكاء الاصطناعي على البيئة وعلى علاقتنا بها مُعقَّد ومتعدد الأوجه.

شكل 6.3: تحليل الذكاء الاصطناعي لكميات ضخمة من البيانات

فوائده المحتملة

يُمكن للذكاء الاصطناعي أن يساعد في فهم التحديات البيئية والتعامل معها بشكل أفضل مثل: تغير المناخ، والتلوث، وفقدان التنوع البيولوجي، ويُمكنه أن يساعد في تحليل كميات هائلة من البيانات والتنبؤ بتأثير الأنشطة البشرية المختلفة على البيئة، ويُمكنه كذلك أن يساعد في تصميم أنظمة أكثر كفاءة واستدامة مثل: أنظمة شبكات الطاقة، والزراعة، والنقل، والمباني.

أخطاره أو أضراره المُحتملة

هناك مخاوف من تأثير الذكاء الاصطناعي نفسه على البيئة؛ إذ يتطلب تطوير أنظمة الذكاء الاصطناعي واستخدامها قدرًا كبيرًا من الطاقة والموارد؛ مما قد يُسهِّم في انبعاث غازات تُفاقِم من مشكلة الاحتباس الحراري وغيرها من الآثار البيئية. على سبيل المثال، قد يتطلب تدريب نموذج واحد للذكاء الاصطناعي قدرًا من الطاقة يعادل ما تستهلكه العديد من السيارات طوال حياتها. بالإضافة إلى ذلك، يمكن أن يساهم إنتاج المُكوِّنات الإلكترونية المُستخدَمة في تصنيع أنظمة الذكاء الاصطناعي في تلوث البيئة مثل: استخدام المواد الكيميائية السامة وتوليد النفايات الإلكترونية.

شكل 6.4: تتطلب أنظمة الذكاء الاصطناعي كميات هائلة من الطاقة والموارد

علاوة على ذلك، يُمكن أن يغير الذكاء الاصطناعي علاقتنا بالبيئة بطرائق ليست إيجابية دائمًا، فقد يُؤدي استخدام الذكاء الاصطناعي في الزراعة إلى ممارسات زراعية مكثَّفة ومركِّزة على الصناعة؛ مما يؤثر سلبًا على صحة التربة والتنوع البيولوجي. بالمثل، ربما يُؤدي استخدام الذكاء الاصطناعي في النقل إلى زيادة الاعتماد على السيارات وأساليب النقل الأخرى؛ مما يُسهِّم في تلوث الهواء وتدمير البيئات الطبيعية التي تسكنها الكائنات الحية.

الخاتمة

بوجه عام، يعتمد تأثير الذكاء الاصطناعي على البيئة وعلاقتنا بها على كيفية تطوير أنظمة الذكاء الاصطناعي واستخدامها، ومن المهم النظر في التأثيرات البيئية المحتملة للذكاء الاصطناعي وتطوير أنظمته واستخدامها بطرائق تُعطي الأولوية للاستدامة والكفاءة وسلامة كوكب الأرض.

الأُطر التنظيمية ومعايير الصناعة

Regulatory Frameworks and Industry Standards

تلعب الأُطر التنظيمية ومعايير الصناعة دورًا مهمًّا في تعزيز تطبيقات الذكاء الاصطناعي الأخلاقية، فبإمكان التنظيمات المُساعِدة أن تضمن تَحمُّل المنظمات التي تقوم بتطوير واستخدام أنظمة الذكاء الاصطناعي المسؤولية عن أفعالها عن طريق تحديد توقُّعات وعواقب واضحة لعدم الامتثال، وبإمكان التنظيمات والمعايير أن تُحفز المنظمات على إعطاء الأولوية للاعتبارات الأخلاقية عند تطوير واستخدام أنظمة الذكاء الاصطناعي.

الشفافية

يُمكن أن تعزِّز التنظيمات والمعايير الشفافية في أنظمة الذكاء الاصطناعي بمطالبة المؤسسات بالكشف عن كيفية عمل أنظمتها وعن البيانات التي تستخدِمها، ويُمكن أن يساعد ذلك في بِناء الثقة مع أصحاب المصلحة وتقليل المخاوف من التحيُّزات المحتملة أو التمييز المحتمل في أنظمة الذكاء الاصطناعي.

تقييم المخاطر

يُمكن تقليل مخاطر العواقب غير المقصودة أو النتائج السلبية الناتجة عن استخدام الذكاء الاصطناعي بوضع التنظيمات والمعايير المناسبة، وذلك بمطالبة المنظمات بإجراء تقييمات للمخاطر، وهذا يعني تحديد المخاطر والأخطار المحتملة وتنفيذ ضمانات مناسبة، مما يُمكِّن التنظيمات والمعايير من المساعدة في تقليل الأضرار المحتملة على الأفراد والمجتمع.

تطوير ونشر أُطر عمل واضحة للذكاء الاصطناعي

يُمكن أن تشجِّع التنظيمات والمعايير الابتكار بتوفير إطار عمل واضح لتطوير أنظمة الذكاء الاصطناعي واستخدامها؛ إذ أن استخدام التنظيمات والمعايير لتأسيس فرص متكافئة وتقديم التوجيه بخصوص الاعتبارات الأخلاقية يُمكن أن يساعد المنظمات على تطوير أنظمة الذكاء الاصطناعي ونشرها بطرائق تتفق مع القيم الأخلاقية والاجتماعية. تلعب الأُطر التنظيمية ومعايير الصناعة دورًا مهمًّا في تعزيز تطبيقات الذكاء الاصطناعي الأخلاقية، وذلك بتوفير إرشادات وحوافز واضحة للمؤسسات حتى تُعطي الأولوية للاعتبارات الأخلاقية والتنظيمات والمعايير؛ مما يضمن تطوير أنظمة الذكاء الاصطناعي واستخدامها بطرائق تتماشى مع القيم الاجتماعية والأخلاقية.

التنمية المستدامة للذكاء الاصطناعي في المملكة العربية السعودية

Sustainable AI Development in the Kingdom of Saudi Arabia

من المتوقَّع أن تصبح تقنيات الذكاء الاصطناعي وأنظمته أحد العوامل الرئيسة التي تُؤدي إلى إحداث خلل في القطاعات المالية في العديد من البلدان، وقد تؤثر بشكل كبير على سوق العمل، ومن المتوقَّع في السنوات القادمة أن يصبح حوالي 70 % من الأعمال الروتينية التي يقوم بها العمال مؤتمتة بالكامل. كما أنه من المتوقَّع أن تخلق صناعة الذكاء الاصطناعي سبعة وتسعين مليون وظيفة جديدة وتضيف ستة عشر تريليون دولار أمريكي إلى الناتج المحلي الإجمالي العَالَمي.

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

لقد طوَّرت **الهيئة السعودية للبيانات والذكاء الاصطناعي** (Saudi Data and Artificial Intelligence Authority - SDAIA) أهدافًا استراتيجية للمملكة لاستخدام تقنيات الذكاء الاصطناعي المُستدامة في تنمية المملكة، وستكون المملكة العربية السعودية مركزًا عالميًا للبيانات والذكاء الاصطناعي، كما أن المملكة استضافت أول قمة عالمية لهُ، حيث يُمكن للقادة والمبتكرين مناقشة مستقبل الذكاء الاصطناعي وتشكيله لصالح المجتمع. أما الهدف الآخر فيتمثل في تحويل القوى العاملة في المملكة من خلال تطوير البيانات المحلية ودعم المواهب في الذكاء الاصطناعي. وبما أن الذكاء الاصطناعي يقوم بتحويل أسواق العمل عالميًا، فإن معظم القطاعات تحتاج إلى تكييف البيانات والذكاء الاصطناعي ودمجها في التعليم والتدريب المهني والمعرِفة العامة، وبذلك يُمكن أن تكتسب المملكة العربية السعودية ميزة تنافسية من حيث التوظيف والإنتاجية والابتكار.

أما الهدف النهائي فيتمثّل في جذب الشركات والمستثمرين عن طريق أُطر عمل وحوافز تنظيمية مرنة ومستقرة، حيث ستركز الأنظمة على تطوير سياسات ومعايير للذكاء الاصطناعي، بما فيها استخدامه بشكل أخلاقي. وسيعمل إطار العمل على تعزيز التطوير الأخلاقي لأبحاث وحلول الذكاء الاصطناعي ودعمه في ظل توفير إرشادات ومعايير لحماية البيانات والخصوصية؛ مما سيُوفر الاستقرار والتوجيه لأصحاب المصلحة العاملين في المملكة.

مثال

تُخطط المملكة العربية السعودية لاستخدام أنظمة وتقنيات الذكاء الاصطناعي كأساس لمشروعي المدينتين العملاقتين نيوم (NEOM) وذا لاين (THE LINE). مشروع نيوم هو مدينة مستقبلية سيتم تشغيلها بالطاقة النظيفة، وبها أنظمة نقل متطورة، وتقدِّم خدمات ذات تقنية عالية، وستكون منصة للتقنيات المتطورة، بما في ذلك الذكاء الاصطناعي، وستستخدِم حلول المُدن الذكية؛ لتحسين استهلاك الطاقة وإدارة حركة المرور والخدمات المتقدمة الأخرى. وسيتم استخدام أنظمة الذكاء الاصطناعي فيها؛ لتحسين جودة الحياة للسكان ولتعزيز الاستدامة.

وبالمثل، ستكون مدينة ذا لاين مدينة خطية خالية من الكربون مبنية بتقنيات الذكاء الاصطناعي، وستستخدِم أنظمة الذكاء الاصطناعي لأتمتة بنيتها التحتية وأنظمة النقل فيها؛ مما يجعل حياة المقيمين فيها تتسم بالسلاسة والكفاءة، وستكون الطاقة التي ستُشغِّل المدينة طاقة نظيفة، كما أن الأولوية ستكون للمعيشة المستدامة، وسيتم استخدام الأنظمة التي تعمل بالذكاء الاصطناعي؛ لمراقبة استخدام الطاقة وتحسينها وانسيابية حركة المرور والخدمات المتقدمة الأخرى.

وبوجه عام، ستلعب أنظمة الذكاء الاصطناعي وتقنياته دورًا حاسمًا في تطوير مشروعي هاتين المدينتين العملاقتين، وتمكينهما من أن تصبحا مدينتين مستدامتين من مُدن المستقبل تتسمان بالكفاءة والابتكار.

الإرشادات العالمية لأخلاقيات الذكاء الاصطناعي International AI Ethics Guidelines

كما هو موضَّح في الجدول التالي، طوَّرت منظمة **اليونسكو** (UNESCO) وثيقة إرشادية توضِّح بالتفصيل القِيم والمبادئ التي يجب الالتزام بها عند تطوير أنظمة وتقنيات الذكاء الاصطناعي الجديدة.

جدول 6.2: قِيم ومبادئ أخلاقيات الذكاء الاصطناعي

القِيم	المبادئ
• احترام كرامة الإنسان وحمايتها وتعزيزها، وحفظ حريته وحقوقه الأساسية. • ازدهار البيئة والنظام البيئي. • ضمان التنوع والشمولية. • العيش في انسجام وسلام.	• التناسب وعدم الإضرار. • السلامة والأمن. • الإنصاف وعدم التمييز. • الاستدامة. • الخصوصية. • الرقابة البشرية والعزيمة. • الشفافية وقابلية التفسير. • المسؤولية والمساءلة. • الوعي والتثقيف. • الحوكمة والتعاون القائمان على تعدُّد أصحاب المصلحة.

تمرينات

1

حَدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. تهتم أخلاقيات الذكاء الاصطناعي بتطوير أنظمة الذكاء الاصطناعي فقط.	○	○
2. من المحتمل أن يُؤدي الذكاء الاصطناعي والأتمتة إلى تسريح البشر من الوظائف.	○	○
3. يُمكن أن يُؤدي الافتقار إلى التنوع في فِرق تطوير الذكاء الاصطناعي إلى عدم رؤية التحيُّزات أو عدم معالجتها.	○	○
4. يُمكن أن يساعد دمج المبادئ الأخلاقية في أنظمة الذكاء الاصطناعي في ضمان تطويرها واستخدامها بطريقة مسؤولة.	○	○
5. يتطلب التصميم المعتمد على إشراك الإنسان أن تعمل أنظمة الذكاء الاصطناعي دون أي تدخل بشري.	○	○
6. تدل مشكلة الصندوق الأسود في الذكاء الاصطناعي على صعوبة فهم كيفية وصول خوارزميات الذكاء الاصطناعي إلى قراراتها أو تنبؤاتها.	○	○
7. يُمكن تصميم نماذج الذكاء الاصطناعي لتكييف قراراتها أو نتائجها وفقًا للقِيم الأخلاقية الراسخة.	○	○
8. استخدام الذكاء الاصطناعي على نطاق واسع له آثار إيجابية فقط على البيئة.	○	○

2 صِف كيف يؤدي الذكاء الاصطناعي والأتمتة إلى تسريح البشر من وظائفهم.

3 اشرح كيف يمكن أن تساهم بيانات التدريب المُتحيِّزة في تحقيق نتائج ذكاء اصطناعي مُتحيِّزة.

4 عرِّف مشكلة الصندوق الأسود في أنظمة الذكاء الاصطناعي.

5 قارن بين الآثار الإيجابية والسلبية لأنظمة الذكاء الاصطناعي على البيئة.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

التطبيقات الروبوتية 1

إحداث ثورة في العَالَم باستخدام الروبوتية

Revolutionizing the World with Robotics

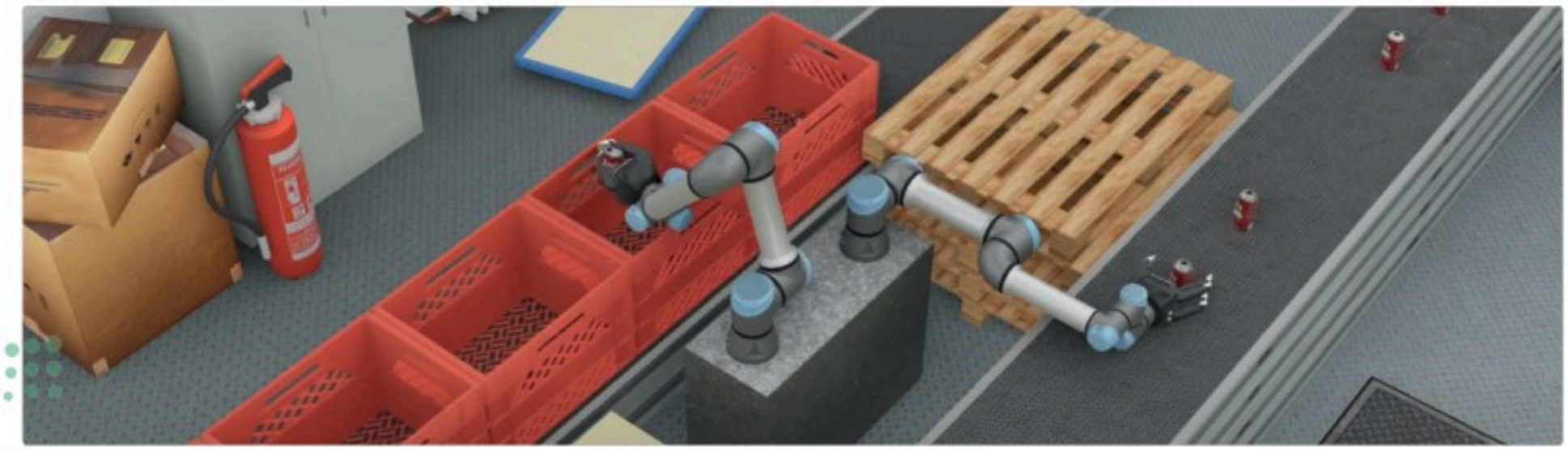
الروبوتية هي مجال سريع النمو أحدثَ ثورة في طريقة عمل الناس وفي عيشهم وتفاعلهم مع بيئتهم وتطبيقاتها، وتشمل مجموعة واسعة من المجالات: بداية من التصنيع وحتى استكشاف الفضاء، ومن الإجراءات الطبية إلى تنظيف المنزل، ومن الترفيه إلى المهام العسكرية. وتتمثّل الميزة الرئيسة للروبوتية في قدرتها على أداء المهام المتكررة بدرجة عالية من الدقة والإتقان، حيث يُمكن أن تعمل الروبوتات بلا تعب ودون أخطاء؛ مما يجعلها مثالية للقيام بالمهام الخطرة أو التي يصعب على البشر القيام بها. على سبيل المثال، في العمليات المصنعية تُستخدم الروبوتات لأداء بعض المهام مثل: اللحام والطلاء وتجميع المُنتَجات، وفي المجال الطبي تُستخدم الروبوتات لإجراء العمليات الجراحية بدقة أكبر، وفي استكشاف الفضاء تُستخدم الروبوتات لاستكشاف ودراسة الكواكب البعيدة.

الروبوتية (Robotics):
تهتم الروبوتية بدراسة الروبوتات، وهي آلات يمكنها أداء مجموعة متنوعة من المهام بطريقة مستقلة أو شبه مستقلة أو تحت تصرُّف البشر.

الروبوتية والمُحاكِيات Robotics and Simulators

هناك تحديان مهمان في مجال الروبوتية هما: التكلفة والوقت اللازمان لبناء أجهزة الروبوت الماديّة واختبارها، وهنا يأتي دور **المُحاكِيات** (Simulators) التي تُستخدم على نطاق واسع في أبحاث الروبوتية وتعليمها وصناعتها؛ لأنها توفر طريقة فعّالة من حيث التكلفة، كما أنها آمنة لاختبار الروبوتات وتجربتها، حيث تتيح المُحاكِيات للمطوِّرين إنشاء بيئات افتراضية تُحاكي سيناريوهات العَالَم الحقيقي؛ مما يسمح لهم باختبار قدرات الروبوتات وأدائها في مجموعة متنوعة من المواقف، ويُمكنها محاكاة مختلف الظروف الجوية والتضاريس والعقبات التي قد تواجهها الروبوتات في العَالَم الحقيقي. كما يُمكن للمُحاكِيات أن تُحاكي التفاعلات بين الروبوتات المتعددة وبين الروبوتات والبشر؛ مما يسمح للمطوِّرين بدراسة وتحسين الطرائق التي تتفاعل بها الروبوتات مع بيئتها.

المُحاكي (Simulator):
برنامج يسمح للمطوِّرين باختبار تصميماتهم وخوارزمياتهم وتحسينها في عالم افتراضي قبل بِناء الروبوتات الماديّة.



شكل 6.5: محاكاة للأذرع الصناعية

وهناك ميزة أخرى للمُحاكِيات تتمثّل في أنها تسمح للمطوِّرين بتعديل تصاميم وخوارزميات الروبوتات المختلفة، واختبارها بسهولة دون الحاجة إلى مُكوِّنات ماديّة حاسوبية باهظة الثمن؛ حيث تسمح بالتكرار والتجريب بطريقة أسرع، مما يُؤدي إلى دورات تطوير أكثر سرعة وتصميمات أكثر كفاءة.

وبوجه عام، تُعدُّ الروبوتية مجالًا سريع النمو يتضمن مجموعة واسعة من التطبيقات والمُحاكِيات التي تلعب دورًا مهمًّا في تطوير الروبوتات عن طريق السماح للمطوِّرين باختبار تصاميم الروبوتات وخوارزمياتها، وتحسينها بطريقة آمنة وغير مُكلفة، ومع استمرار تقدُّم التقنية، فمن المتوقَّع أن تنمو تطبيقات الروبوتية واستخدام المُحاكِيات، مما يمهّد الطريق لعَالَم أكثر أتمتةً وترابطًا.

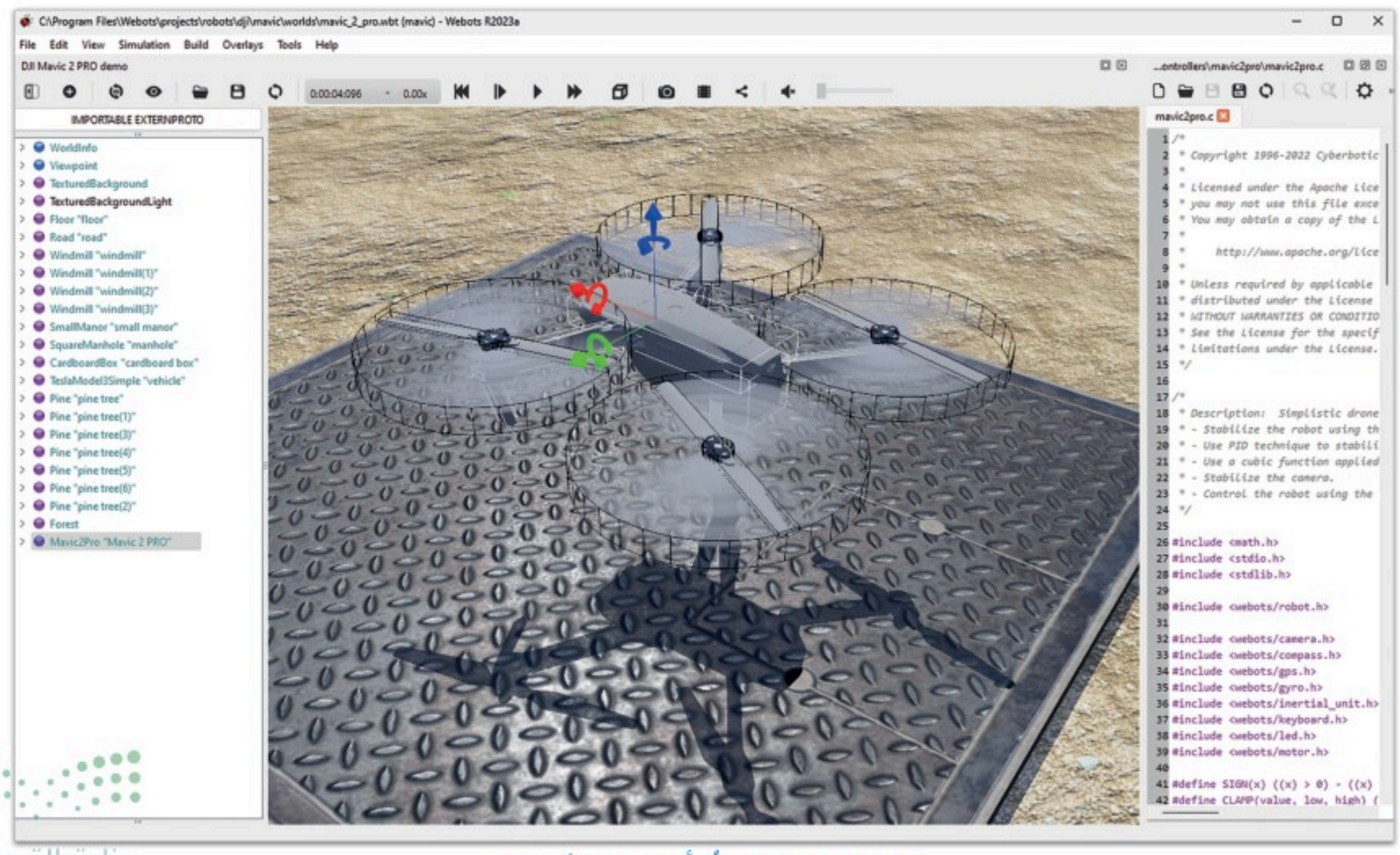
ويبوتس Webots

ويبوتس أداة برمجية قوية يُمكن استخدامها في محاكاة الروبوتات وبيئاتها، وهي منصة ممتازة تستحق إدخالها في عَالَم الروبوتات والذكاء الاصطناعي، حيث يستطيع الطلبة تصميم الأنظمة والخوارزميات الروبوتية ومحاكاتها واختبارها باستخدام هذه الأداة، دون الحاجة إلى معدات حاسوبية باهظة الثمن.

يُعدُّ استخدام أداة ويبوتس في الذكاء الاصطناعي مفيدًا بشكل خاص؛ لأنها تتيح للطلبة تجربة خوارزميات تعلُّم الآلة واختبار أدائها في بيئة تعتمد على المُحاكاة، فمن خلال إنشاء روبوتات وبيئات افتراضية يستطيع الطلبة أن يستكشفوا إمكانيات وقيود الذكاء الاصطناعي، وأن يتعلَّموا كيفية برمجة الأنظمة الذكية التي يُمكِنها اتخاذ القرارات بناءً على بيانات الزمن الواقعي.

يُمكنك تنزيل أداة ويبوتس من الرابط التالي:

https://github.com/cyberbotics/webots/releases/download/R2023a/webots-R2023a_setup.exe



شكل 6.6: مشروع طائرة مُسيَّرة باستخدام أداة ويبوتس

مراقبة المنطقة Area Surveillance

في هذا الدرس والدرس التالي ستستخدِم أداة ويبوتس لعمل مُحاكاة لطائرة مُسيَّرة تُحلق فوق أحد المنازل، ثم ستقوم بترقيتها لتكتشف الحدود البشرية كمُراقِبة، حيث تتكون المُحاكاة من طائرة مُسيَّرة تُقلع من وضع السكون على الأرض وتبدأ في الدوران حول المنزل. وفي الدرس التالي، ستُضيف ميزة رؤية الحاسب للطائرة المُسيَّرة باستخدام الكاميرا الخاصة بها باستخدام مكتبة **أوبن سي في** (OpenCV)، وهذا سيُمكنك من تحليل الصور التي التقطتها الكاميرا.

يتم التحكم في الطائرة المُسيَّرة بواسطة نصٍّ برمجي بلغة البايثون وهو مسؤول عن التحكم في جميع الأجهزة المُسيَّرة بما فيها مُحركات المراوح والكاميرا و**نظام تحديد المواقع العالمي** (Global Positioning System – GPS) وما إلى ذلك، كما أنه يحتوي على مقطع برمجي لمزامنة جميع المُحركات لتحريك الطائرة المُسيَّرة إلى **نقاط الطريق** (Waypoints) المتنوعة وجعلها مستقرة في الهواء.

نقطة الطريق (Waypoint):

نقطة الطريق هي موقع جغرافي محدَّد في فضاء ثلاثي الأبعاد تتم برمجة الطائرة المُسيَّرة لتطير إليها أو تمر من خلالها. وتُستخدم نقاط الطريق لإنشاء مسارات طيران معرَّفة مسبقًا لتتبعها الطائرات المُسيَّرة، ويمكن ضبطها باستخدام إحداثيات نظام تحديد المواقع العالمي أو أنظمة أخرى قائمة على المواقع.

البدء مع ويبوتس Starting with Webots

ستتعرَّف في هذا الدرس على أداة ويبوتس وبيئتها، حيث تتكون محاكاة ويبوتس من عنصرين:

- التعريف بروبوت واحد أو أكثر وبيئاتها في ملف **عَالَم ويبوتس** (Webots World).
- برنامج مُتحكِّم واحد أو أكثر للروبوتات المذكورة.

عَالَم ويبوتس (Webots World) هو وصف ثلاثي الأبعاد لخصائص الروبوت، حيث يتم تعريف كل كائن بما في ذلك موقعه، واتجاهه، وهندسته، ومظهره مثل: لونه أو سطوعه، وخصائصه المادية، ونوعه وما إلى ذلك، كما يُمكن أن تحتوي الكائنات على كائنات أخرى في الأنظمة الهرمية التي تُشكل العوالم. على سبيل المثال، قد يحتوي الروبوت على عجلتين، ومُستشعر مسافة، ومفصل يحتوي على كاميرا، ونحوها. يحدِّد **ملف العَالم** (World File) فقط اسم المُتحكِّم اللازم لكل روبوت، ولا يحتوي على المقطع البرمجي **للمُتحكِّم** (Controller) في الروبوتات، وتُحفظ العَوالِم في ملفات بتنسيق "wbt."، ويحتوي كل مشروع ويبوتس على مجلد فرعي بعنوان worlds (العَوالِم) تُخزَّن فيه الملفات بتنسيق "wbt.".

مُتحكِّم ويبوتس (Webots Controller) هو برنامج حاسب يتحكم في روبوت محدَّد في ملف العَالَم، ويُمكن استخدام أي لغة من لغات البرمجة التي يدعمها ويبوتس لتطوير المُتحكِّم مثل: لغة **سي بلس بلس** (C ++) ولغة **جافا** (Java)، ولكنك ستستخدِم في هذا المشروع لغة البايثون. يُطلِق ويبوتس كل برنامج من برامج المُتحكِّم المُعطاة كعملية منفصلة عندما تبدأ المُحاكاة، ويقوم بربط عمليات المُتحكِّم بالروبوتات التي تمت محاكاتها، وعلى الرغم من أن العديد من الروبوتات يُمكنها مشاركة المقطع البرمجي نفسه لبرنامج المُتحكِّم، إلا أن كل روبوت سيشغِّل العملية الخاصة به. يُخزَّن مَصدر كل برنامج مُتحكِّم وملفاته الثنائية معًا في **مجلد المُتحكِّم** (Controller Directory)، حيث يحتوي كل مشروع ويبوتس على مجلد مُتحكِّم داخل المجلد الفرعي الذي يتخذ اسم controllers (المُتحكِّمات).

بيئة الويبوتس The Webots Environment

عندما تفتح البرنامج، ستلاحظ عدة حقول ونوافذ، حيث تشمل المُكوِّنات الرئيسة لواجهة ويبوتس ما يلي:

شريط القائمة (Menu Bar): يقع في الجزء العلوي من الواجهة، ويُوفر شريط القوائم إمكانية الوصول إلى أوامر وخيارات متنوعة للعمل على المُحاكاة مثل: إنشاء نموذج روبوت أو استيراده، وتهيئة بيئة المُحاكاة، وتشغيل عمليات المُحاكاة.

شريط الأدوات (Toolbar): هو مجموعة من الأزرار الموجودة أسفل شريط القائمة ويُوفر الوصول السريع إلى الوظائف المُستخدَمة بشكل متكرر مثل: إضافة كائنات إلى المشهد، وبدء المُحاكاة وإيقافها، وتغيير عرض الكاميرا.

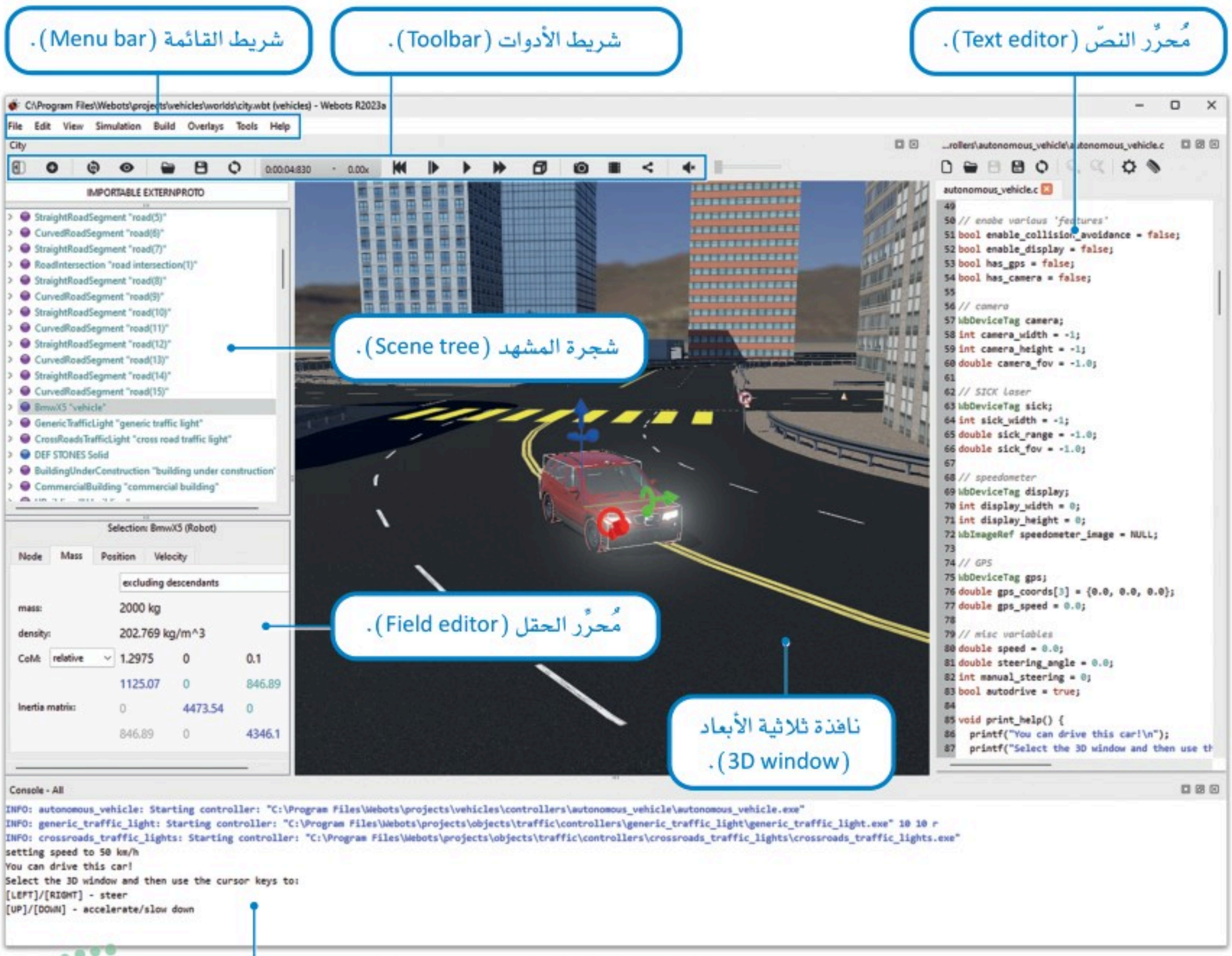
شجرة المشهد (Scene Tree): هي تمثيل هرمي للكائنات في بيئة المُحاكاة، حيث تتيح للمُستخدِمين التنقل في المشهد والتعامل معه مثل: إضافة أو حذف الكائنات، وتغيير خصائص الكائن، وتجميع الكائنات وإدارتها بشكل أسهل.

مُحرِّر الحقل (Field Editor): هو واجهة رسومات لتحرير خصائص الكائنات في بيئة المُحاكاة، حيث يُمكن للمُستخدِمين استخدامه لضبط مُعامِلات الكائن مثل: موضعه، واتجاهه، وحجمه، ومادته، وخصائصه الفيزيائية.

نافذة ثلاثية الأبعاد (3D Window): هي نافذة العرض الرئيس لبيئة المُحاكاة، وتعرض الكائنات وتفاعلاتها في فضاء ثلاثي الأبعاد، حيث يُمكن للمُستخدِمين التنقل في النافذة الثلاثية الأبعاد باستخدام عناصر تحكم الكاميرا المختلفة مثل: التحريك، والتكبير أو التصغير، والتدوير.

مُحرِّر النصّ (Text Editor): هو أداة لتحرير مصدر المقطع البرمجي أو الملفات النصّية الأخرى المُستخدَمة في المُحاكاة، ويقدِّم **تمييزًا لبناء الجمل** (Syntax Highlighting) وخصائص مفيدة أخرى لكتابة المقاطع البرمجية **وتصحيحها** (Debugging)، مثل: **الإكمال التلقائي** (Auto-Completion) **وإبراز الأخطاء** (Error Highlighting).

وحدة التحكم (Console): هي نافذة تعرض مُخرَجات قائمة على النصّ من المُحاكاة، بما في ذلك رسائل الخطأ ومعلومات التصحيح، وهي مفيدة في استكشاف الأخطاء التي تحدث أثناء المُحاكاة وإصلاحها.



شكل 6.7: نافذة ويبوتس

أولًا: عليك أن تقوم بتثبيت المكتبات اللازمة التي ستستخدِمها في مشروعك. يمكنك تثبيت مكتبة **أوبن سي في** (OpenCV) عن طريق **باي تشارم** (PyCharm).

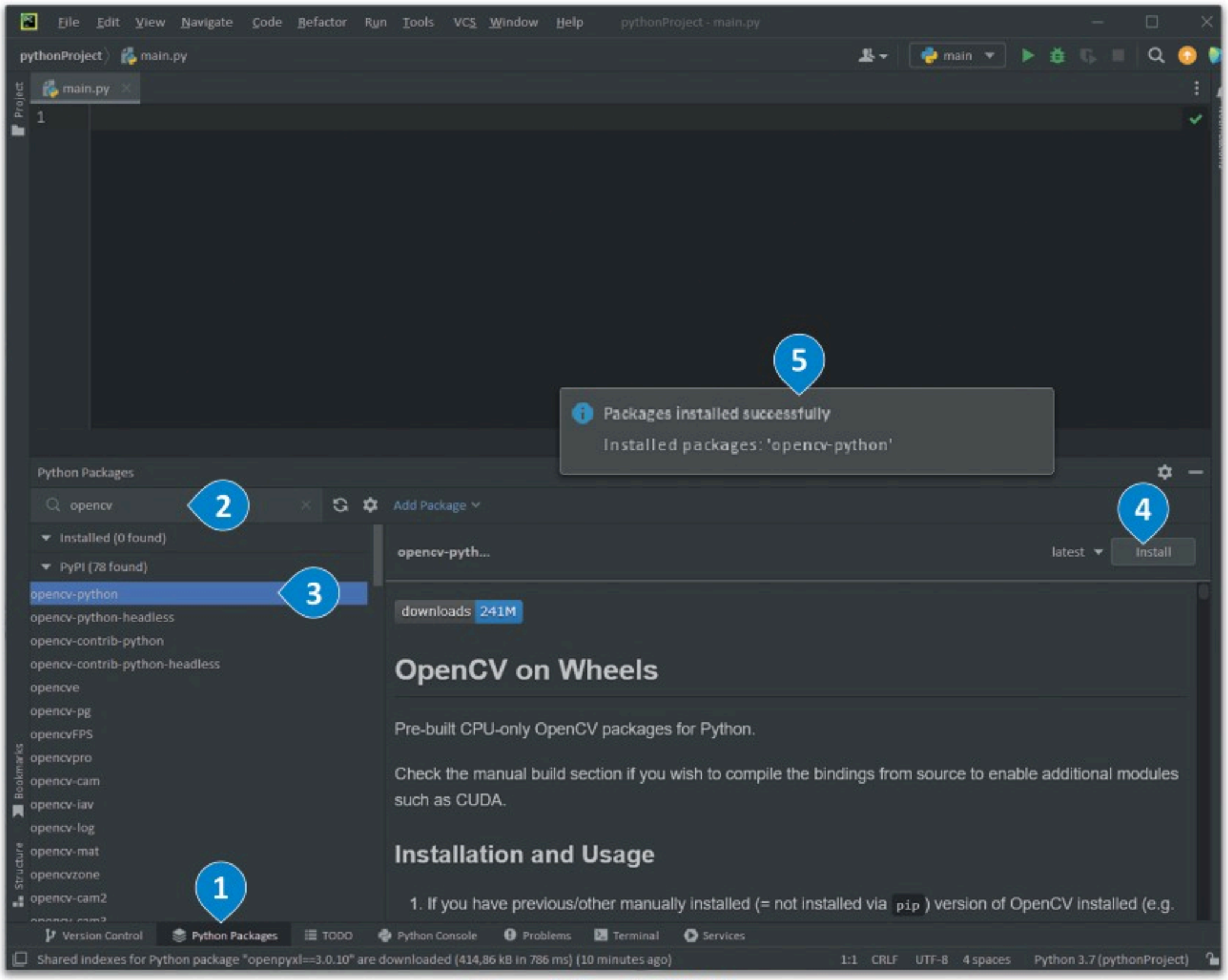
لتنصيب مكتبة أوبن سي في (OpenCV):

> في نافذة PyCharm (باي تشارم)، اضغط على Packages (حِزم). 1

> اكتب "opencv" (أوبن سي في) في شريط البحث. 2

> اختر opencv-python (أوبن سي في- بايثون)، 3 ثم اضغط على install (تثبيت). 4

> ستظهر لك رسالة تخبرك باكتمال التنصيب. 5



شكل 6.8: تثبيت مكتبة أوبن سي في

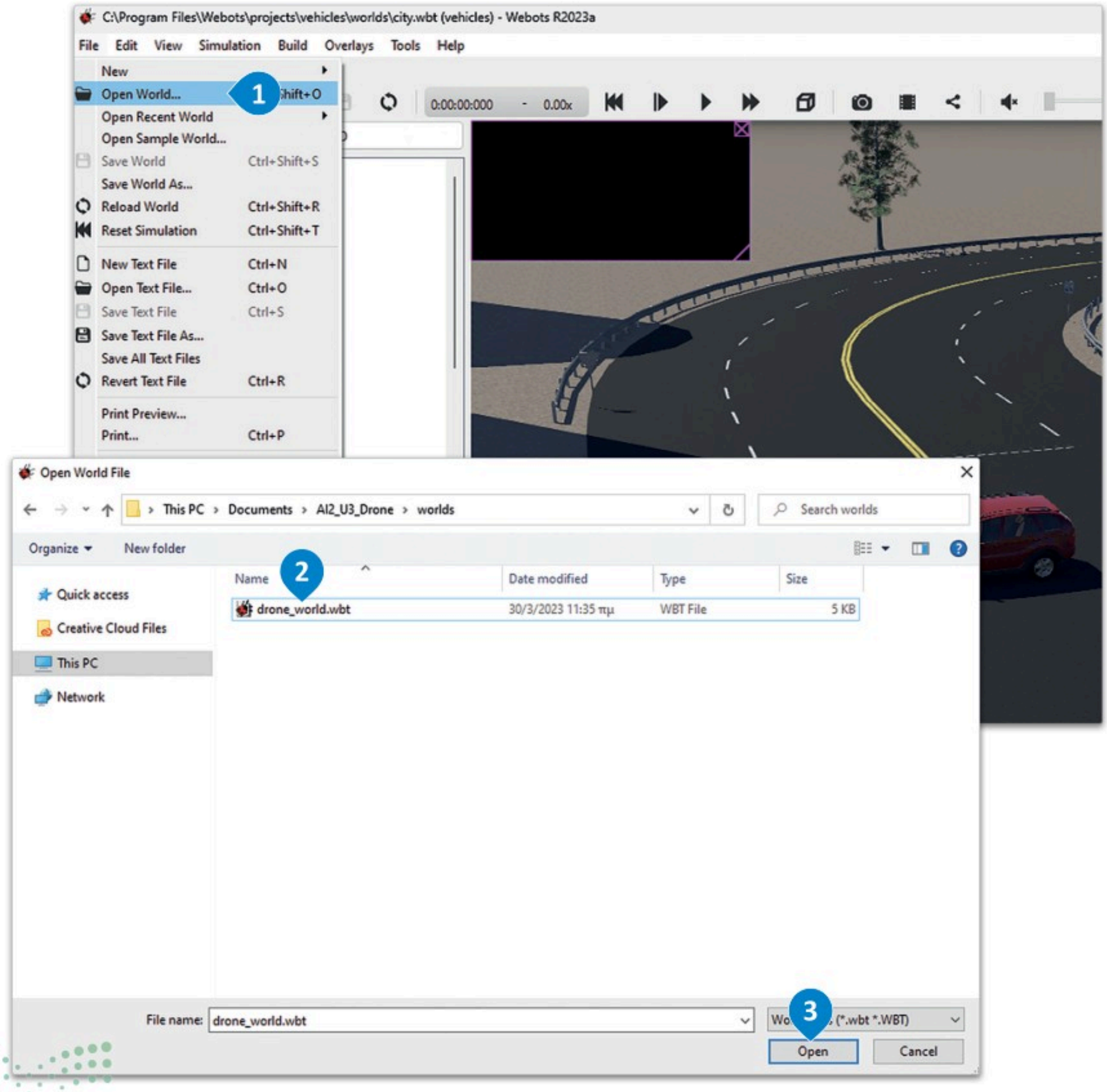
بالمثل، يمكنك تثبيت مكتبة بيلو (Pillow) من خلال البحث عن كلمة "pillow".

دعونا نُلقي نظرة على المشروع. أولًا: عليك أن تبحث عن ملف عَالَم ويبوتس وتقوم بتحميله.

لفتح عَالَم ويبوتس:

> من Menu bar (شريط القائمة)، اضغط على File (ملف)، ثم على Open World (افتح عَالَم). 1

> ابحث عن ملف drone_world.wbt (الطائرة المُسيَّرة_ العَالَم) في مجلد worlds (العَوالِم)، 2 ثم اضغط على Open (فتح). 3



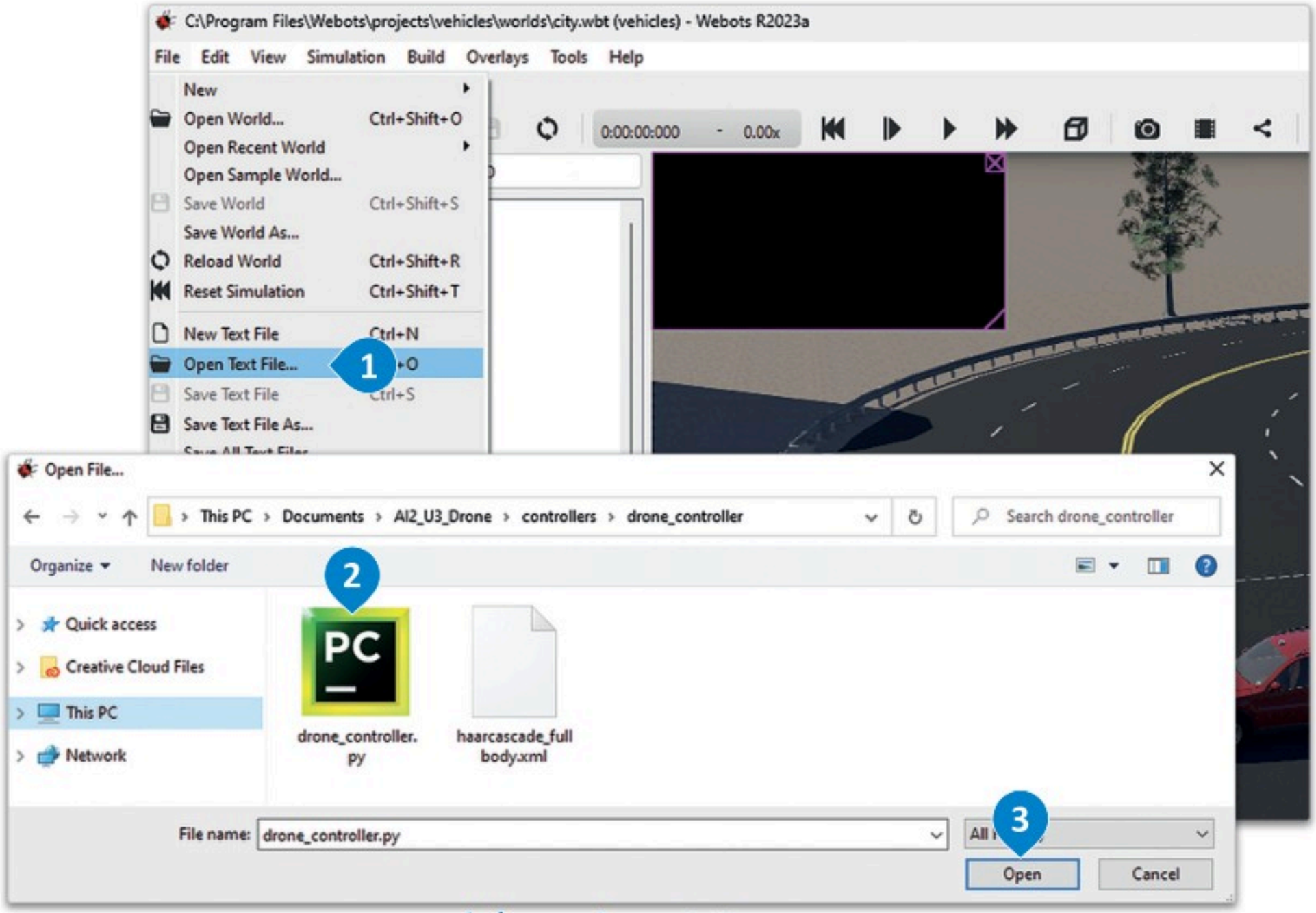
شكل 6.9: فتح عَالَم ويبوتس

بعدها افتح ملف النصّ البرمجي بلغة البايثون الذي سيُستخدم في التحكم في الطائرة المُسيَّرة.

لفتح النصّ البرمجي للمُتحكِّم:

> اضغط على File (ملف)، ثم Open Text File (افتح ملف نصّي) من شريط القائمة. 1

> ابحث عن ملف drone_controller.py (مُتحكِّم_الطائرة المُسيَّرة) في مجلد controllers (المُتحكِّمات) ثم مجلد drone_controller (مُتحكِّم_الطائرة المُسيَّرة)، 2 ثم اضغط على Open (فتح). 3



شكل 6.10: فتح النصّ البرمجي لمُتحكِّم ويبوتس

موضع الكائن ودورانه Object Position and Rotation

تُستخدم الإحداثيات ثلاثية الأبعاد X وY وZ لتمثيل موضع كائن في الفضاء، حيث يُمثِّل X المحور الأفقي، وY المحور الرأسي، وZ محور العمق، وتُشبه إحداثيات العَالَم الحقيقي لخطّ العرض وخطّ الطول والارتفاعات المُستخدَمة لوصف المواقع على الأرض.

الانحدار (Pitch) و**الالتفاف** (Roll) و**الانعراج** (Yaw) توجيهات دورانية يُمكن استخدامُها لوصف حركة كائن ما بالنسبة للإطار المرجعي كما يظهر في الشكل 6.11، **فالانحدار** (Pitch) هو دوران الكائن حول محوره X؛ مما يجعله يميل لأعلى أو لأسفل بالنسبة للمستوى الأفقي، أما **الالتفاف** (Roll) فهو دوران الكائن حول محوره Y؛ مما يجعل الجسم يميل جانبًا أو من جانب إلى آخر، و**الانعراج** (Yaw) هو دوران الكائن حول محوره Z؛ مما يجعل الجسم يلتف إلى اليسار أو اليمين بالنسبة للإطار المرجعي.

يُمكن استخدام هذه القيم الست معًا (X، Y، Z، الانحدار، الالتفاف، الانعراج) لوصف موضع كائن في الفضاء ثلاثي الأبعاد واتّجاهه، حيث تُستخدم بشكل شائع في الروبوتات، وأنظمة الملاحة، والتطبيقات الأخرى التي تتطلب تحديد المواقع والتحكم بدقة.

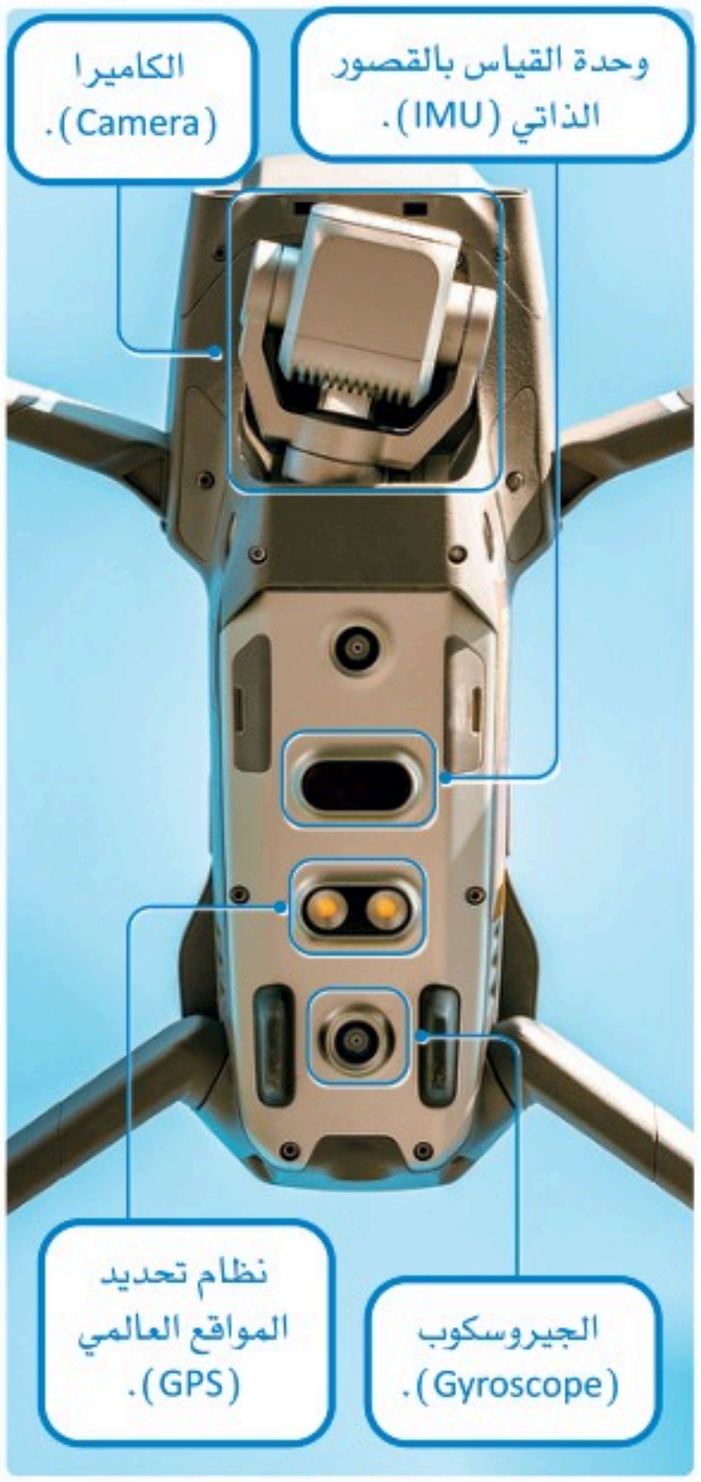
أجهزة الطائرة المُسيَّرة Drone Devices

تم تجهيز **الطائرة المُسيَّرة** (Drone) بعدة **مُستشعرات** (Sensors) تتيح لها أن تجمع المُدخَلات من بيئتها، ويوفّر المُحاكي الدّالتين getDevice( ) و enable( ) للتفاعل مع المُستشعرات و**المُشغِّلات** (Actuators) المختلفة لروبوت المُحاكاة.

تُستخدم دالة getDevice( ) للحصول على قراءات جهاز مثل: المُستشعر أو المُشغِّل من نموذج روبوت ويبوتس، وتأخذ مُعامِلًا نصّيًّا وتحدِّد اسم الجهاز المراد الوصول إليه.

تُستخدم الدالة enable( ) لتنشيط جهاز، بحيث يُمكنه البدء في تقديم البيانات أو تنفيذ إجراء محدَّد.

شكل 6.11: محاور الدوران



شكل 6.12: طائرة مُسيَّرة بمُستشعرات وكاميرا

يُمكن **لوحدة القياس بالقصور الذاتي** (Inertial Measurement Unit – IMU) قياس التسارع الخطيّ للطائرة المُسيَّرة وسرعتها الزاويّة، وقياس القوى مثل الجاذبية، بالإضافة إلى قوى الدوران المؤثرة على الطائرة المُسيَّرة، كما يُمكنها أن توفر معلومات عن وضع الطائرة المُسيَّرة (الانحدار، والالتفاف، والانعراج)، وهو أمر بالغ الأهمية لتحقيق الاستقرار والتحكم.

نظام تحديد المواقع العَالَمي (Global Positioning System – GPS) هو نظام ملاحة يعتمد على القمر الصناعي ويُوفر للطائرة المُسيَّرة معلومات دقيقة عن المواقع، ويمكِّن نظام تحديد المواقع العَالَمي الطائرة المُسيَّرة من معرفة موقعها الحالي وارتفاعها وسرعتها بالنسبة إلى الأرض، وهذه المعلومات مهمة؛ للتنقل والتحكم في الطائرة المُسيَّرة.

المُستشعرات (Sensors) هي أجهزة تكشف الكميات الفيزيائية أو الأحوال البيئية وتقيسها، وتحوّلها إلى إشارة كهربائية للمراقبة أو التحكم.

المُشغِّلات (Actuators) هي أجهزة تحوّل الإشارات الكهربائية إلى حركة ميكانيكية لأداء عمل معيّن أو مُهِمَّة معيّنة.

بينما تقيس السرعة الخطية المسافة التي يقطعها الجسم خلال الثانية، فإنَّ السرعة الزاوية تقيس سرعة دوران الجسم حول نقطة مركزية أومحور، حيث تقيس مقدار التغيّر في الزاوية المركزية لجسم خلال وحدة الزمن، وعادةً ما تُقاس بالراديان في الثانية (rad/s) أو الدرجات في الثانية (°/s).

الجيروسكوب (Gyroscope) هو مُستشعر يقيس السرعة الزاويَّة، أو معدل الدوران حول محور معيّن، ويُعدُّ الجيروسكوب مفيدًا بشكل خاص في اكتشاف التغيرات الصغيرة في اتجاه الطائرة المُسيَّرة وتصحيحها، وهو أمر مهم للحفاظ على الاستقرار والتحكم أثناء الطيران.

كاميرا الطائرة المُسيَّرة (Drone's Camera) تُستخدم لالتقاط الصور أثناء الطيران، ويُمكن تثبيتها على الطائرة المُسيَّرة، بحيث تتمكّن من التقاط صور من جهات وزوايا مختلفة عن طريق ضبط زاوية **انحدار الكاميرا** (Camera Pitch) باستخدام الدالة setPosition( ). وفي هذا المشروع، ضُبط الموضع على 0.7، أي حوالي 45 درجة بالنظر إلى الأسفل.

أجهزة **المروحيات الأربعة** (Four Propeller) في الطائرة المُسيَّرة هي مُشغِّلات تتحكم في سرعة دوران **المروحية الرباعية** (Quadcopter) واتجاهها، وهي طائرات مُسيَّرة مُجهزة بأربعة **دوَّارات** (Rotors)، اثنان منهما يدوران في اتجاه عقارب الساعة والاثنان الآخران يدوران عكس اتجاهها، حيث يولِّد دوَران هذه الدوَّارات **قوة رفع** (Lift) ويسمح للطائرة المُسيَّرة بالإقلاع والمناورة في الهواء. وكما هو الحال مع باقي الأجهزة، تُسترد المحرِّكات وتوضع في موضعها، وتُستخدم الدالة setVelocity( ) كذلك لضبط السرعة الأولية للأجهزة المروحية.

شكل 6.13: طائرة مُسيَّرة بأربع مروحيات

التحرُّك نحو الهدف Moving to a Target

للانتقال من موقع إلى آخر، تستخدِم الطائرة المُسيَّرة دالة move_to_target( ) التي تحتوي على **منطق التحكم** (Control Logic)، حيث تأخذ قائمة الإحداثيات كمُعامِل، في شكل أزواج [X، Y]؛ لاستخدامها كنقاط طريق.

في البداية، تتحقق الدالة ممّا إذا تمّت **تهيئة** (Initialized) **الموضع المستهدَف** (Target Position) أم لا، وفي تلك الحالة تضبطه على نقطة الطريق الأولى، ثم تتحقق مما إذا كانت الطائرة المُسيَّرة قد وصلت إلى الموضع المستهدف بالدقة المُحدَّدة في المُتغيِّر target_precision. وإذا كان الأمر كذلك، تنتقل الدالة إلى نقطة الطريق المستهدفة التالية.

ويجب حساب الزاوية بين الموضع الحالي للطائرة المُسيَّرة وموضعها المستهدف؛ لمعرفة مدى قوة الدوران التي يجب أن تكون عليه في الخطوة التالية، حيث تمت معايرة هذه القيمة وضبطها على النطاق [π, π−].

وبعد ذلك، تقوم الدالة بحساب اضطرابات الانعراج والانحدار المطلوبة لتوجيه الطائرة المُسيَّرة نحو نقطة الطريق المستهدفة وضبط زاوية انحدار الطائرة المُسيَّرة على التوالي.

حسابات المحرِّكات Motor Calculations

أخيرًا، يجب حساب السرعة التي تضبط بها **المحرِّكات** (Motors)، وذلك بقراءة القيم المبدئية للمُستشعرات، أي قراءة: قيم الالتفاف والانحدار، والانعراج من وحدة القياس بالقصور الذاتي، ويتم الحصول على قيم مواضع X وY وZ من نظام تحديد المواقع العَالَمي، بينما يتم الحصول على قيم تسارع الالتفاف والانحدار من الجيروسكوب.

ويتم استخدام **الثوابت** (Constants) المختلفة التي تم تعريفها في المقطع البرمجي مسبقًا لإجراء الحسابات والتعديلات بالتزامن مع مُدخَلات المُستشعرات، وفي النهاية يتم ضبط **الدفع** (Thrust) الصحيح.

معلومة

يمكن للمروحية أن تتحرك في أي اتجاه وأن تُحافظ على طيرانها مُستقرًا من خلال التحكّم في سرعة المروحيات الأربع واتّجاهها، فعلى سبيل المثال، عند زيادة سرعة الدوَّارين الموجودين على جانب واحد وتقليل سرعة الدوَّارين الآخرين، فإن الطائرة المُسيَّرة باستطاعتها الميلان والتحرك في اتجاه معيَّن.

```python
from controller import Robot
import numpy as np    # used for mathematic operations
import os    # used for folder creation
import cv2    # used for image manipulation and human detection
from PIL import Image    # used for image object creation
from datetime import datetime    # used for date and time

# auxiliary function used for calculations
def clamp(value, value_min, value_max):
    return min(max(value, value_min), value_max)

class Mavic (Robot):

    # constants of the drone used for flight
    # thrust for the drone to lift
    K_VERTICAL_THRUST = 68.5
    # vertical offset the drone uses as targets for stabilization
    K_VERTICAL_OFFSET = 0.6
    K_VERTICAL_P = 3.0          # P constant of the vertical PID
    K_ROLL_P = 50.0             # P constant of the roll PID
    K_PITCH_P = 30.0            # P constant of the pitch PID

    MAX_YAW_DISTURBANCE = 0.4
    MAX_PITCH_DISTURBANCE = -1
    # precision between the target position and the drone position in meters
    target_precision = 0.5

    def __init__(self):
        # initializes the drone and sets the time interval between updates of the simulation
        Robot.__init__(self)
        self.time_step = int(self.getBasicTimeStep())

        # gets and enables devices
        self.camera = self.getDevice("camera")
        self.camera.enable(self.time_step)

        self.imu = self.getDevice("inertial unit")
        self.imu.enable(self.time_step)

        self.gps = self.getDevice("gps")
        self.gps.enable(self.time_step)

        self.gyro = self.getDevice("gyro")
        self.gyro.enable(self.time_step)

        self.camera_pitch_motor = self.getDevice("camera pitch")
        self.camera_pitch_motor.setPosition(0.7)

        self.front_left_motor = self.getDevice("front left propeller")
        self.front_right_motor = self.getDevice("front right propeller")
        self.rear_left_motor = self.getDevice("rear left propeller")
        self.rear_right_motor = self.getDevice("rear right propeller")
        motors = [self.front_left_motor, self.front_right_motor,
                  self.rear_left_motor, self.rear_right_motor]
        for motor in motors: # mass initialization of the four motors
            motor.setPosition(float('inf'))
            motor.setVelocity(1)
```

تحتوي مكتبة برنامج المُتحكِّم على فئة Robot (روبوت) التي ستُستخدم طرائقها للتحكم في الطائرة المُسيَّرة.

استيراد المكتبات المطلوبة للحسابات والمعالجة.

تُستخدم الثوابت (Constants) الموجودة بشكل تجريبي لحساب الطيران والاستقرار.

```
        self.current_pose = 6 * [0]  # X, Y, Z, yaw, pitch, roll
        self.target_position = [0, 0, 0]
        self.target_index = 0
        self.target_altitude = 0

    def move_to_target(self, waypoints):

        # Moves the drone to the given coordinates
        # Parameters:
        #    waypoints (list): list of X,Y coordinates
        # Returns:
        #    yaw_disturbance (float): yaw disturbance (negative value to go on the right)
        #    pitch_disturbance (float): pitch disturbance (negative value to go forward)

        if self.target_position[0:2] == [0, 0]:  # initialization
            self.target_position[0:2] = waypoints[0]

        # if the drone is at the position with a precision of target_precision
        if all([abs(x1 - x2) < self.target_precision for (x1, x2)
                    in zip(self.target_position, self.current_pose[0:2])]):

            self.target_index += 1
            if self.target_index > len(waypoints) - 1:
                self.target_index = 0
            self.target_position[0:2] = waypoints[self.target_index]

        # computes the angle between the current position of the drone and its target position
        # and normalizes the resulting angle to be within the range of [-pi, pi]
        self.target_position[2] = np.arctan2(
            self.target_position[1] - self.current_pose[1],
            self.target_position[0] - self.current_pose[0])
        angle_left = self.target_position[2] - self.current_pose[5]
        angle_left = (angle_left + 2 * np.pi) % (2 * np.pi)
        if (angle_left > np.pi):
            angle_left -= 2 * np.pi

        # turns the drone to the left or to the right according to the value
        # and the sign of angle_left and adjusts pitch_disturbance
        yaw_disturbance = self.MAX_YAW_DISTURBANCE * angle_left / (2 * np.pi)
        pitch_disturbance = clamp(
            np.log10(abs(angle_left)), self.MAX_PITCH_DISTURBANCE, 0.1)

        return yaw_disturbance, pitch_disturbance

    def run(self):

        # time intevals used for adjustments in order to reach the target altitude
        t1 = self.getTime()

        roll_disturbance = 0
        pitch_disturbance = 0
        yaw_disturbance = 0
```

تهيئة موضع المُسيَّرة (x، y، z) ودورانه (الالتفاف، الانحدار، الانعراج).

```
        # specifies the patrol coordinates
        waypoints = [[-30, 20], [-60, 30], [-75, 0], [-40, -10]]
        # target altitude of the drone in meters
        self.target_altitude = 8

        while self.step(self.time_step) != -1:

            # reads sensors
            roll, pitch, yaw = self.imu.getRollPitchYaw()
            x_pos, y_pos, altitude = self.gps.getValues()
            roll_acceleration, pitch_acceleration, _ = self.gyro.getValues()
            self.current_pose = [x_pos, y_pos, altitude, roll, pitch, yaw]

            if altitude > self.target_altitude - 1:
                # as soon as it reaches the target altitude,
                # computes the disturbances to go to the given waypoints
                if self.getTime() - t1 > 0.1:
                    yaw_disturbance, pitch_disturbance = self.move_to_target(
                        waypoints)
                    t1 = self.getTime()

            # calculates the desired input values for roll, pitch, yaw,
            # and altitude using various constants and disturbance values
            roll_input = self.K_ROLL_P * clamp(roll, -1, 1) +
                         roll_acceleration + roll_disturbance
            pitch_input = self.K_PITCH_P * clamp(pitch, -1, 1) +
                          pitch_acceleration + pitch_disturbance
            yaw_input = yaw_disturbance
            clamped_difference_altitude = clamp(self.target_altitude -
                              altitude + self.K_VERTICAL_OFFSET, -1, 1)
            vertical_input = self.K_VERTICAL_P *
                             pow(clamped_difference_altitude, 3.0)

            # calculates the motors' input values based on the
            # desired roll, pitch, yaw, and altitude values
            front_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                   - yaw_input + pitch_input - roll_input
            front_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                    + yaw_input + pitch_input + roll_input
            rear_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                  + yaw_input - pitch_input - roll_input
            rear_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                   - yaw_input - pitch_input + roll_input

            # sets the velocity of each motor based on the motors' input values calculated above
            self.front_left_motor.setVelocity(front_left_motor_input)
            self.front_right_motor.setVelocity(-front_right_motor_input)
            self.rear_left_motor.setVelocity(-rear_left_motor_input)
            self.rear_right_motor.setVelocity(rear_right_motor_input)


    robot = Mavic()
    robot.run()
```

waypoints (نقاط الطريق) الخاصة بالمسار الذي ستطير فيه الطائرة المُسيَّرة.

حان الوقت الآن لإدراج النصّ البرمجي في الطائرة المُسيَّرة وتشغيل المُحاكاة.

لإدراج برنامج المُتحكّم وتشغيل المُحاكاة:

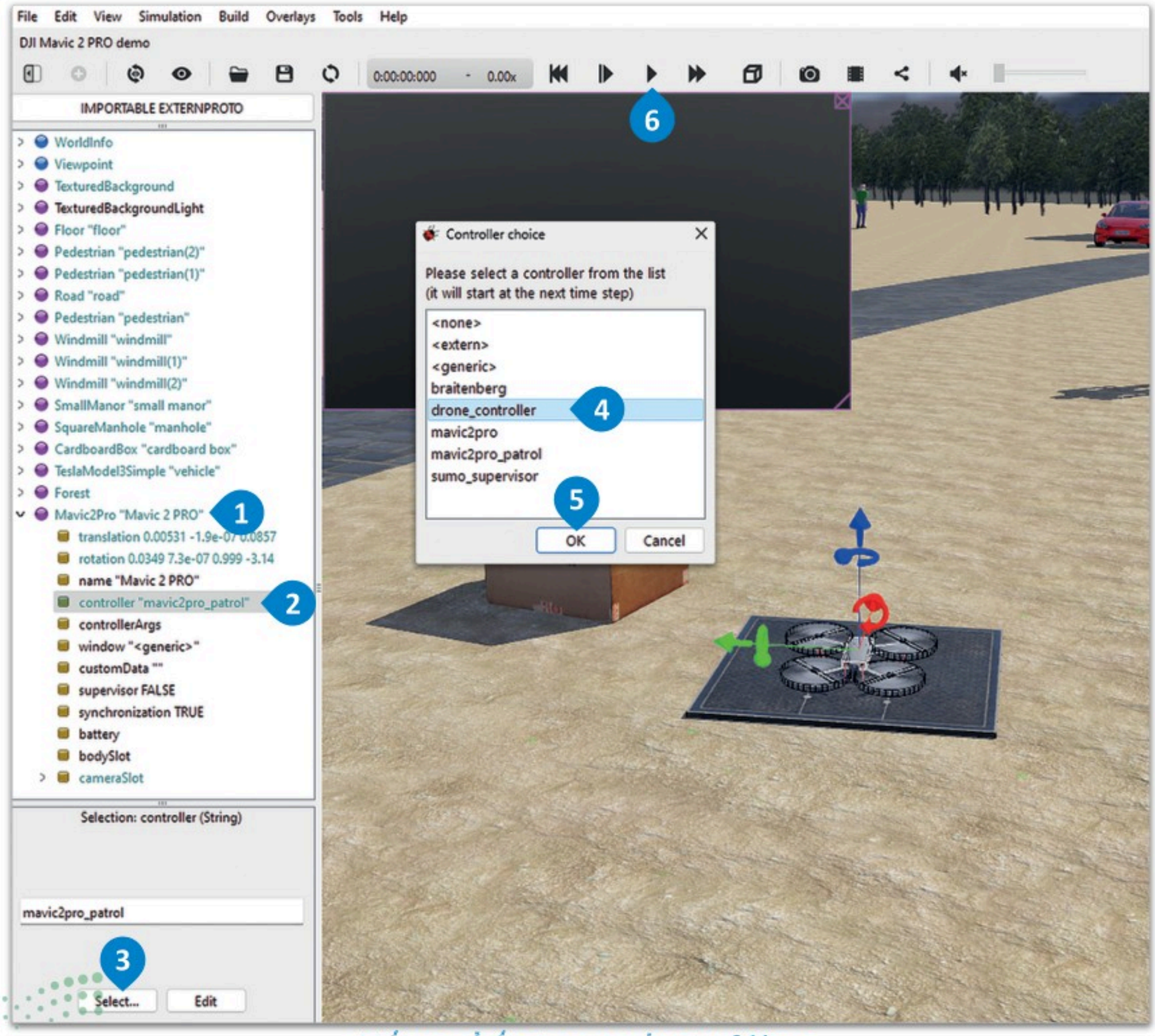
> من Scene tree (شجرة المشهد)، اضغط على "Mavic2Pro "Mavic 2 Pro"، 1 ثم اضغط على "controller "mavic2pro_patrol". 2

> من Field editor (مُحرِّر الحقل)، اضغط على ... Select (اختيار). 3

> حدِّد drone_controller (مُتحكِّم_الطائرة المُسيَّرة)، 4 ثم اضغط على OK (موافق). 5

> من Toolbar (شريط الأدوات)، اضغط على Run the simulation in real-time (شغّل المحاكاة بشكل فوري). 6

عند إجراء تغييرات على النصوص البرمجية، لا تنسَ أن تضغط على Ctrl + S.



شكل 6.14: إدراج النصّ البرمجي لبرنامج المُتحكّم وتشغيل المُحاكاة

عندما تبدأ المُحاكاة، ستعمل محركات الطائرة المُسيَّرة وستُقلع، ثم ستتبع الطريق المحدَّدة مسبقًا حول المنزل، وتمر عبر نقاط الطريق.

شكل 6.15: إقلاع الطائرة المُسيَّرة

تمرينات

1 حلِّل الدالة ( )move_to_target واشرح كيفية قيام الطائرة المُسيَّرة بحساب موضعها التالي في قائمة نقاط الطريق. كيف يمكن تحسين مسار الطائرة المُسيَّرة لتقليل زمن الطيران بين نقاط الطريق؟

2 قيِّم عيوب خوارزميّة التحكُّم الحالية في الطائرة المُسيَّرة عند مواجهة عوامل خارجية مثل: الرياح أو العوائق أو عدم دقة نظام تحديد المواقع العالمي، ثم اقترح وناقش التحسينات التي يمكن القيام بها في خوارزميّة التحكم لجعل الطائرة المُسيَّرة أكثر صمودًا في وجه هذه التحديات.

3 استكشف الآثار الأخلاقية للطائرات المُسيَّرة الهوائية في التطبيقات الواقعية مثل: المراقبة وتوصيل الطرود وعمليات البحث والإنقاذ، ثم اكتب عن المخاوف المحتملة الخاصة بالخصوصية، وقضايا السلامة، واحتمالات إساءة استخدام هذه التقنية.

4 أضف خاصية تُسجِّل موضع الطائرة المُسيَّرة وارتفاعها واتجاهها على فترات منتظمة أثناء الطيران، ثم اكتب كل الأنماط التي قد تجدها في بيانات السجل.

5 جرِّب استخدام قيم مختلفة لثوابت PID في برنامج المُتحكِّم (K_VERTICAL_P، K_ROLL_P، K_PITCH_P). ولاحظ كيفية تأثير هذا التغيرات على استقرار الطائرة المُسيَّرة واستجابتها، ثم ناقش الموازنات بين الاستقرار والاستجابة.

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

التطبيقات الروبوتية 2

الروبوتية ورؤية الحاسب والذكاء الاصطناعي
Robotics, Computer Vision and AI

رؤية الحاسب (Computer Vision) و**الروبوتية** (Robotics) مجالان متطوران من مجالات التقنية يعملان معًا على متابعة التغيير السريع لطريقة حياة الناس وعملهم، وعندما يُدمجان فإنهما يفتحان مجموعة واسعة من الإمكانيات **للأتمتة** (Automation) والتصنيع وتطوير التطبيقات الأخرى.

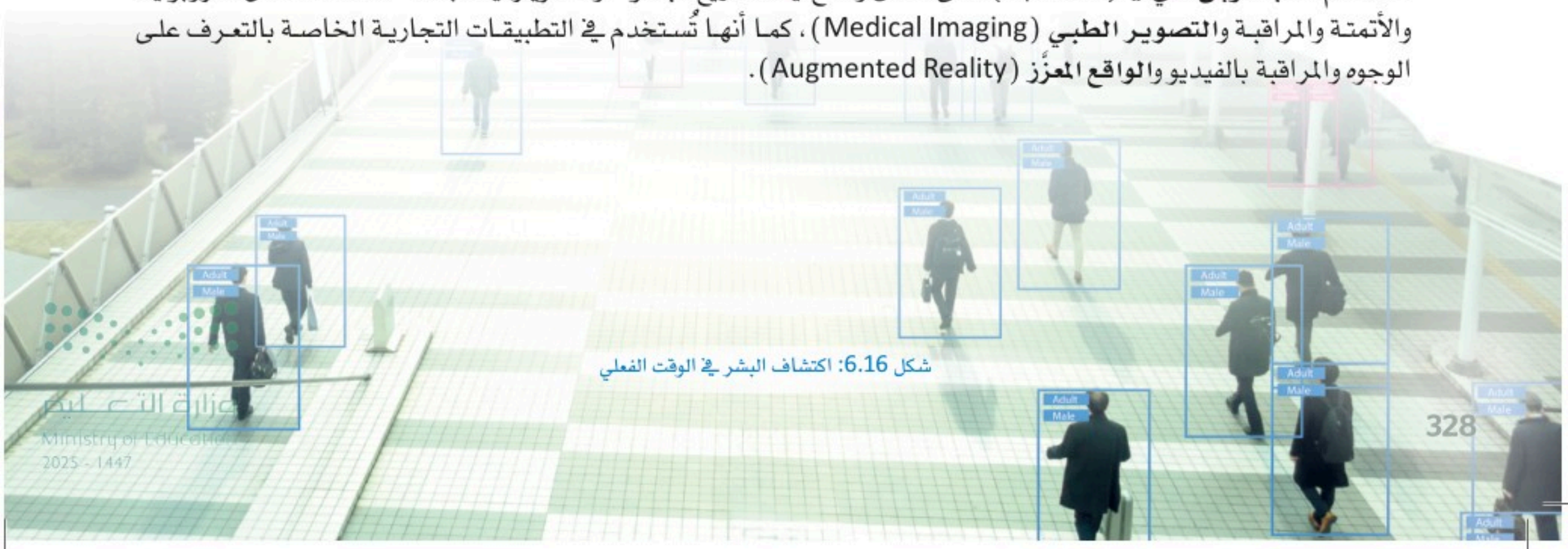
يُعدُّ الذكاء الاصطناعي مُكوِّنًا رئيسًا من مُكوِّنات رؤية الحاسب والروبوتية على حدّ سواء؛ مما يُمكِّن الآلات من التعلُّم والتكيُّف مع بيئتها بمرور الوقت، حيث تستطيع الروبوتات باستخدام خوارزميات الذكاء الاصطناعي أن تُحلِّل وتُفسِّر كميات هائلة من البيانات المرئية؛ مما يسمح لها باتخاذ قرارات والقيام بإجراءات في الوقت الفعلي. كما يُمكِّن الذكاء الاصطناعي الروبوتات من تحسين أدائها ودقتها بمرور الوقت، إذ أنها تتعلّم من تجاربها وتُعدِّل سلوكها وفقًا لذلك، وهذا يعني أن الروبوتات المزودة برؤية الحاسب وقدرات الذكاء الاصطناعي يُمكنها أداء مهام شديدة التعقيد بشكل أكثر دقّة وكفاءة.

في هذا الدرس ستعمل على ترقية المشروع الأوليّ للطائرة المُسيَّرة الذي تم توضيحه في الدرس السابق، وذلك باستخدام رؤية الحاسب لاكتشاف وتحديد الشّخوص البشرية القريبة من المنزل، حيث يُمكن النظر إليهم على أنهم أعداء في سيناريو العَالَم الواقعي، وتَستخدِم الطائرة المُسيَّرة الكاميرا المزود بها؛ لتكون بمثابة نظام مراقبة، كما يُمكن تطبيق هذا المثال وتنفيذه بسهولة على العديد من المباني الأخرى والبنية التحتية والممتلكات الخاصة والشركات مثل: المصانع ومحطات توليد الطاقة.

سيتم استخدام مكتبة **أوبن سي في** (OpenCV) من لغة البايثون لاكتشاف الشّخوص البشرية، وهي مكتبة رؤية حاسوبية مفتوحة المصدر توفر مجموعة من خوارزميات رؤية الحاسب ومعالجة الصور بالإضافة إلى مجموعة من أدوات البرمجة؛ لتطوير التطبيقات في هذه المجالات.

يُمكن استخدام مكتبة **أوبن سي في** (OpenCV) في الروبوتية للقيام بمهام مثل: اكتشاف الكائنات وتتبُّعها، وإعادة البناء ثلاثي الأبعاد، والملاحة، وتشمل ميزاتها كذلك اكتشاف الكائنات والتعرف عليها، واكتشاف الوجوه والتعرف عليها، ومعالجة الصور ومقاطع الفيديو، و**معايرة الكاميرا** (Camera Calibration)، وتعلُّم الآلة، وغيرها.

تُستخدم مكتبة **أوبن سي في** (OpenCV) على نطاق واسع في مشاريع البحوث والتطوير في مجالات متعددة تشمل: الروبوتية والأتمتة والمراقبة و**التصوير الطبي** (Medical Imaging)، كما أنها تُستخدم في التطبيقات التجارية الخاصة بالتعرف على الوجوه والمراقبة بالفيديو و**الواقع المعزَّز** (Augmented Reality).



شكل 6.16: اكتشاف البشر في الوقت الفعلي

لنستعرض التغييرات التي ستُجريها لإضافة وظائف رؤية الحاسب للطائرة المُسيَّرة.

إضافة المؤقِّت Adding a Timer

يُمكن أن يكون التقاط صورة ومعالجتها وحفظها مكلفًا من الناحية الحاسوبية إذا حُسب لكل إطار من إطارات المُحاكاة، ولذلك ستضيف مؤقِّتًا زمنيًا لاستخدامه؛ لتنفيذ هذه الإجراءات كل خمسِ ثوانٍ فقط.

```
# time intervals used for adjustments in order to reach the target altitude
t1 = self.getTime()
# time intervals between each detection for human figures
t2 = self.getTime()
```

إنشاء مجلد Creating a Folder

سيتم حفظ الصور المُلتَقطة التي يتم فيها اكتشاف الشّخوص البشرية في مجلد، حيث يُعدّ جزءًا من أرشيف المراقبة الأمنية الذي سيساعد على فحص الصور في المستقبل.

أولًا: عليك أن تستخدِم الدالة getcwd( ) لتسترد مسار دليل العمل الحالي لبرنامج المُتحكِّم (وهو المجلد الذي يتضمّن برنامج المُتحكِّم) حتى يتعرف البرنامج على المكان الذي يضع فيه المجلد الجديد باسم: detected (تم الاكتشاف)، بحيث تُستخدم الدالة path.join( ) لربط اسم المسار بسلسلة اسم المجلد النصّية، وتتمثّل الخطوة الأخيرة في التحقق مما إذا كان المجلد موجودًا بالفعل أم لا، وفي تلك الحالة يتم إنشاء مجلد جديد.

```
# gets the current working directory
cwd = os.getcwd()
# sets the name of the folder where the images
# with detected humans will be stored
folder_name = "detected"
# joins the current working directory and the new folder name
folder_path = os.path.join(cwd, folder_name)

if not os.path.exists(folder_path):
# creates the folder if it doesn't exist already
    os.makedirs(folder_path)
    print(f"Folder \"detected\" created!")
else:
    print(f"Folder \"detected\" already exists!")
```

معالجة الصورة Image Processing

في هذا التوقيت يمكنك الآن استرداد (قراءة) الصورة من الجهاز لمعالجتها قبل محاولة الكشف. لاحظ أن كل ما يتعلق بمعالجة الصورة وصولًا إلى حفظها يحدث كل خمسِ ثوانٍ فقط، كما هو مبيَّن في الشرط "self.getTime( ) – t2> 5.0".

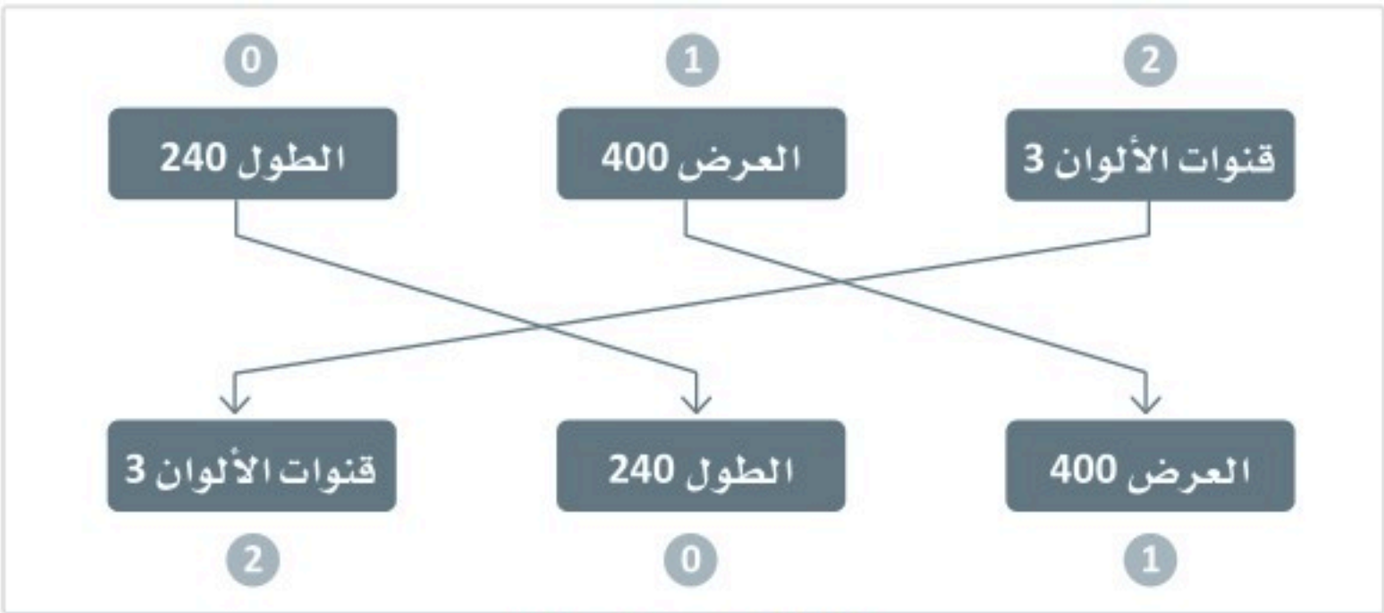
```
# initiates the image processing and detection routine every 5 seconds
if self.getTime() - t2 > 5.0:

    # retrieves image array from camera
    cameraImg = self.camera.getImageArray()
```

بعد التحقق من استرداد الصورة بنجاح، تنتقل الخوارزميّة إلى تعديل بعض خصائصها، بحيث تكون الصورة ثلاثية الأبعاد، ولها أبعاد طول وعرض وقنوات ألوان، حيث تلتقط كاميرا الطائرة المُسيَّرة صورًا بارتفاع 240 بكسل وعرض 400 بكسل، كما أنها تستخدِم 3 قنوات ألوان لحفظ معلومات الصورة وهي: الأحمر والأخضر والأزرق.

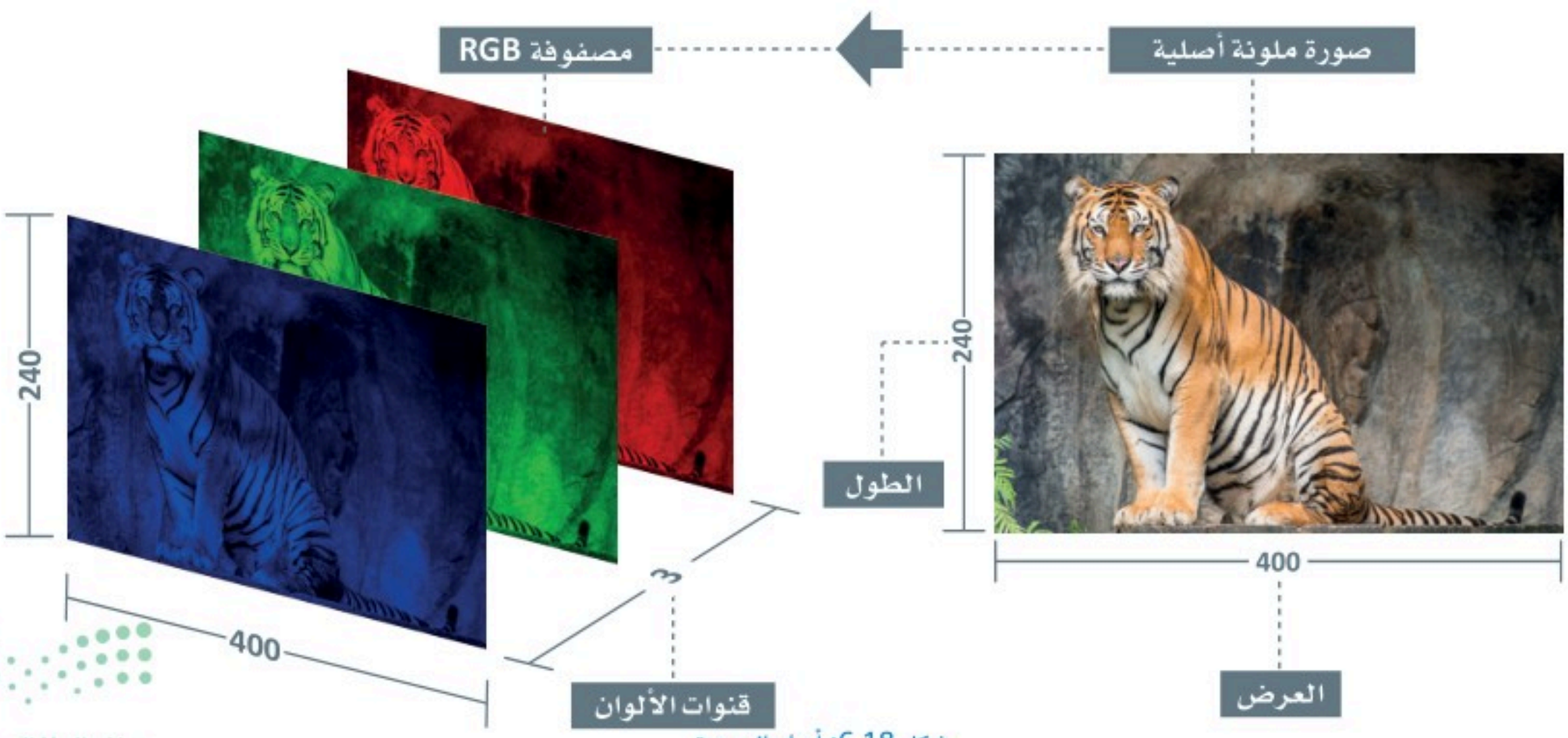
يجب معالجة الصورة أولًا حتى يتم استخدامها في الكشف، ولكي يتم تطبيق الدوال بشكل صحيح في وقت لاحق، لا بُدّ أن تحقق الصورة تركيبًا معيّنًا. في هذا المثال، يجب أن يتغير تسلسل الأبعاد من (الطول، والعرض، وقنوات الألوان) إلى (قنوات الألوان، والطول، والعرض) باستخدام الدالة ()transpose، حيث تُقدَّم **صورة الكاميرا** (CameraImg)، والتسلسل الجديد (1، 0، 2) كمُعامِلات لهذه الدالة، بافتراض أن الترتيب الأصلي كان (2، 1، 0).

كما يجب تعديل أحجام الأبعاد بعد تغيير التسلسل، حيث تُستخدم الدالة ()reshape بالطريقة نفسها، ولكن أحجام الأبعاد المعنيّة كالمُعامِل الثاني منها تكون (400، 240، 3).

```
# reshapes image array to (channels, height, width) format
cameraImg = np.transpose(cameraImg, (2, 0, 1))
cameraImg = np.reshape(cameraImg, (3, 240, 400))
```



شكل 6.17: تغيير تسلسل الأبعاد



شكل 6.18: أبعاد الصورة

بعد ذلك، يجب تغيير الصورة إلى التدرج الرمادي حيث أن خوارزميّة الاكتشاف تستلزم ذلك، مع وجوب تخزينها أولًا في كائن صورة ووجوب الجمع بين قنوات ألوانها الثلاثة، وهنا يجب دمج قنوات الألوان وتخزينها باستخدام الدالة ( )merge في تسلسل عكسي: أي أن يكون تسلسل الألوان (أزرق، أخضر، أحمر) بدلًا من (أحمر، أخضر، أزرق)، وأن يكون تسلسلها الرقمي (0، 1، 2) بدلًا من (2، 1، 0) على الترتيب.

```
# creates RGB image from merged channels
img = Image.new('RGB', (400, 240))
img = cv2.merge((cameraImg[2], cameraImg[1], cameraImg[0]))
```

وأخيرًا، يتم تحويل الصورة إلى التدرج الرمادي باستخدام الدالة ( )cvtColor التي تستخدِم مُعامِل COLOR_BGR2GRAY لتغيير الألوان من الأزرق والأخضر والأحمر إلى التدرج الرمادي.

```
# converts image to grayscale
gray = cv2.cvtColor(np.uint8(img), cv2.COLOR_BGR2GRAY)
```

اكتشاف صور الحدود البشرية Human Silhouette Detection

لكي تكتشف الصورة، عليك أن تستخدم **مصنِّف هار كاسكيد** (Haar Cascade Classifier)، وهو خوارزميّة لاكتشاف الكائنات تعتمد على تعلُّم الآلة، وتُستخدم لتحديد الكائنات في الصور أو مقاطع الفيديو. ولاستخدام هذا المُصنِّف تحتاج أن تُدرِّب نموذج تعلُّم الآلة على مجموعة من الصور التي تحتوي على الكائن الذي تريد البحث عنه، وعلى صور أخرى لا تحتوي على هذا الكائن، حيث تقوم الخوارزميّة بالبحث عن أنماط معيّنة في الصور لتحديد مكان الكائن. وفي العادة تُستخدم هذه الخوارزميّة للعثور على أشياء محدَّدة مثل: الوجوه، أو أشخاص يسيرون في مقطع فيديو. ومع ذلك قد لا تعمل هذه الخوارزميّة بشكل جيد في بعض المواقف التي يكون فيها الكائن محجوبًا جزئيًّا أو كليًّا أو معرضًا لإضاءة منخفضة. تم تدريب المصنِّف في مشروعك تدريبًا خاصًّا على اكتشاف البشر، وعليك أن تستخدِم ملف haarcascade_fullbody.xml الذي ستُزود به، وهو نموذج تعلُّم آلة مُدرَّب مسبقًا ويشكِّل جزءًا من مكتبة **أوبن سي في** (OpenCV)، ويُقدَّم كمُعامِل لكائن ( )CascadeClassifier، ثم تَستخدم الدالة ( )detectMultiScale للقيام بعملية الاكتشاف.

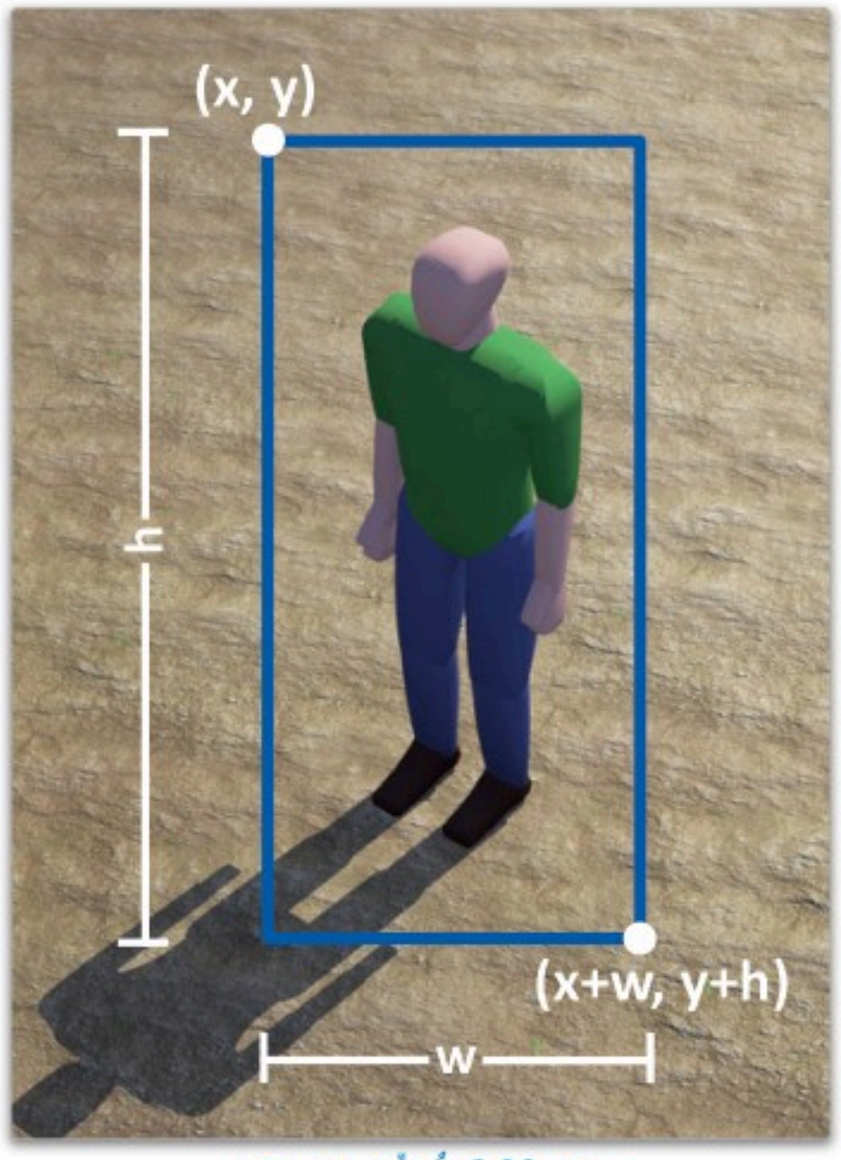
```
# loads and applies the Haar cascade classifier to detect humans in image
human_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')
humans = human_cascade.detectMultiScale(gray)
```

شكل 6.19: مثال على إكتشاف صور الحدود البشرية

تقرير الطائرة المُسيَّرة وحفظ الصور المُكتشَفة
Drone Report and Saving of the Detected Images

الإضافة النهائية لبرنامج المُتحكِّم الخاص بك هو نظام تقرير بسيط تُقدِّمه الطائرة المُسيَّرة عن طريق طباعة رسالة على **وحدة التحكم** (Console) عند اكتشاف شكل بشري، وحفظ الصورة في المجلد الذي أنشأتَه من قبل.

يقوم المُتغيِّر humans (البشر) بحمل المستطيلات الإطارية التي يُكتشف البشر بداخلها في حال عُثر عليهم. تُعرَّف المستطيلات بواسطة أربعة مُتغيِّرات: وهي الزوج x و y اللذان يمثِّلان الإحداثيين اللذين في الصورة وذلك في الزاوية العُليا من الجهة اليُسرى للمستطيل، وكذلك الزوج w و h، الذي يمثِّل عرض المستطيل وارتفاعه. في جميع الاكتشافات الموجودة في الصورة تُحدِّد الدالة rectangle( ) البشر بمستطيل أزرق، حيث تنظر الدالة إلى مُتغيِّرات الصورة على أنها تتمثَّل في الزاوية اليسرى العُلوية (x، y) والزاوية اليمنى السُفلية (x+w، y+h) من المستطيل، ولون المستطيل وعرضه، وفي الصورة الموضَّحة تلاحظ أن لون المستطيل أزرق (B=255، G=0، R=0) وعرضه 2.



شكل 6.20: مُتغيِّرات المستطيل

سيقوم نظام التقرير باسترجاع التاريخ والوقت الحاليين باستخدام الدالة datetime.now( ) وطباعتها على وحدة التحكم، بالإضافة إلى إحداثيات الطائرة المُسيَّرة في وقت التقرير، ويتم تعديل تنسيق التاريخ والوقت بطريقة بسيطة عن طريق إدراج الشرطات العُلوية (-) والشرطات السُفلية (_) لاستخدامها كجزء من اسم الملف المحفوظ، ثم يتم حفظها في المجلد باستخدام الدالة imwrite( )، وعند اكتمال كل شيء تقوم الدالة getTime( ) بإعادة ضبط المؤقِّت.

```
        # loop, through detected human images, annotates them with a bounding box
        # and prints a timestamp and an info message on the console
        for (x, y, w, h) in humans:

            # the image, the top left corner, the bottom right corner, color and width of the rectangle
            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            current_time = datetime.now()
            print(current_time)
            print("Found a person in coordinates [{:.2f}, {:.2f}]"
                          .format(x_pos, y_pos))

            # saves annotated image to file with timestamp
            current_time = current_time.strftime("%Y-%m-%d_%H-%M-%S")
            filename = f"detected/IMAGE_{current_time}.png"
            cv2.imwrite(filename, img)

        t2 = self.getTime()
```

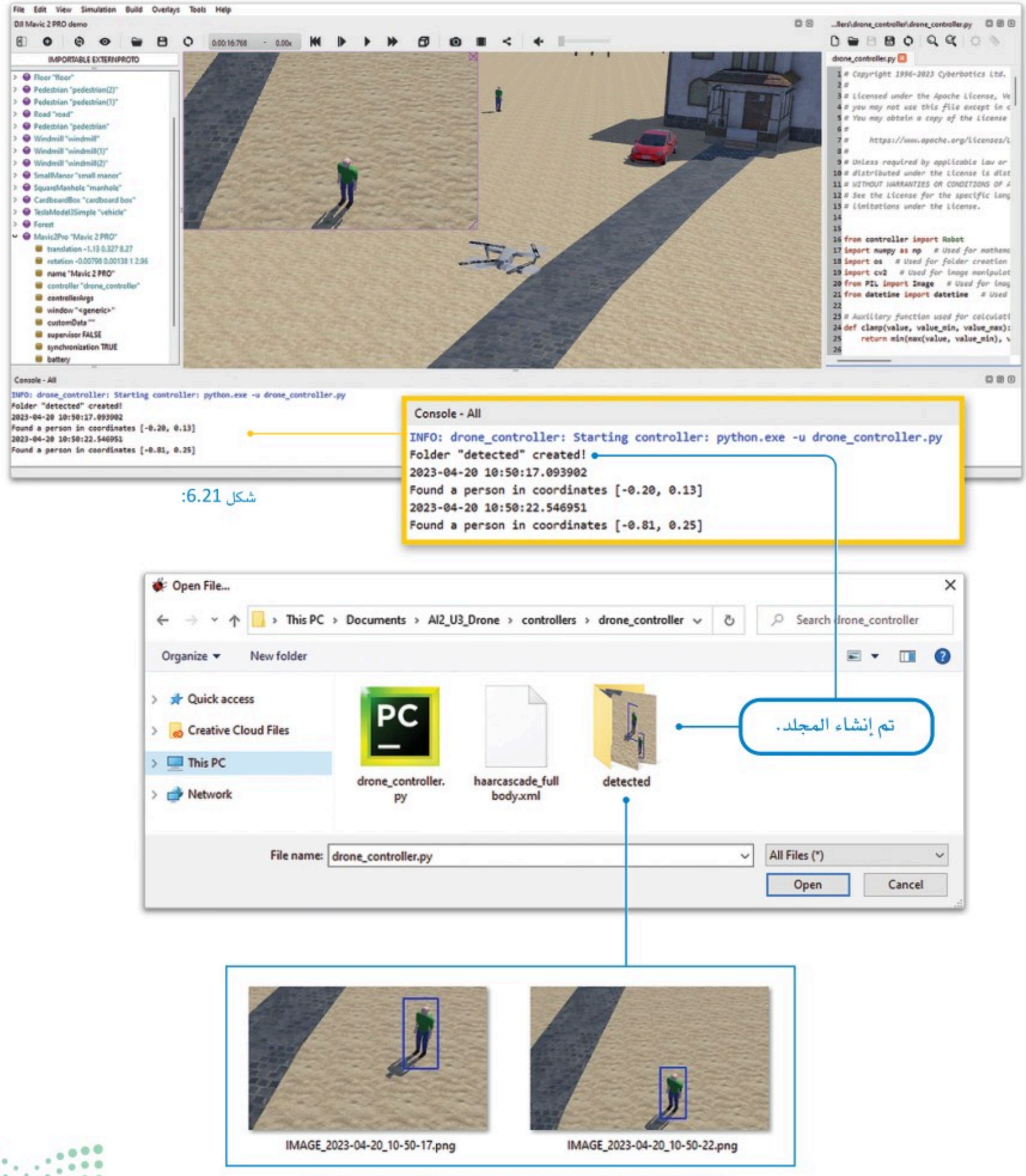
في السلسلة النصّية، يتم استخدام الترميز {:.2f} كاختصار لعدد حقيقي (floating number) ذي خانتين عشريتين، وهنا يتم استخدام الاختصارين للمُتغيِّرين x_pos وy_pos.

بعد إضافة كل هذه الوظائف يجب أن تظهر الدالة ( )run الخاصة ببرنامج المُتحكِّم كما يلي:

```
def run(self):

    # time intervals used for adjustments in order to reach the target altitude
    t1 = self.getTime()
    # time intervals between each detection for human figures
    t2 = self.getTime()

    roll_disturbance = 0
    pitch_disturbance = 0
    yaw_disturbance = 0

    # specifies the patrol coordinates
    waypoints = [[-30, 20], [-60, 30], [-75, 0], [-40, -10]]
    # target altitude of the drone in meters
    self.target_altitude = 8

    # gets the current working directory
    cwd = os.getcwd()
    # sets the name of the folder where the images
    # with detected humans will be stored
    folder_name = "detected"
    # joins the current working directory and the new folder name
    folder_path = os.path.join(cwd, folder_name)

    if not os.path.exists(folder_path):
    # creates the folder if it doesn't exist already
        os.makedirs(folder_path)
        print(f"Folder \"detected\" created!")
    else:
        print(f"Folder \"detected\" already exists!")

    while self.step(self.time_step) != -1:

        # reads sensors
        roll, pitch, yaw = self.imu.getRollPitchYaw()
        x_pos, y_pos, altitude = self.gps.getValues()
        roll_acceleration, pitch_acceleration, _ = self.gyro.getValues()
        self.current_pose = [x_pos, y_pos, altitude, roll, pitch, yaw]

        if altitude > self.target_altitude - 1:
            # as soon as it reaches the target altitude,
            # computes the disturbances to go to the given waypoints
            if self.getTime() - t1 > 0.1:
                yaw_disturbance, pitch_disturbance = self.move_to_target(
                   waypoints)
                t1 = self.getTime()

        # initiates the image processing and detection routine every 5 seconds
        if self.getTime() - t2 > 5.0:

            # retrieves image array from camera
            cameraImg = self.camera.getImageArray()

            # checks if image is successfully retrieved
            if cameraImg:
```

```
            # reshapes image array to (channels, height, width) format
            cameraImg = np.transpose(cameraImg, (2, 0, 1))
            cameraImg = np.reshape(cameraImg, (3, 240, 400))

            # creates RGB image from merged channels
            img = Image.new('RGB', (400, 240))
            img = cv2.merge((cameraImg[2], cameraImg[1], cameraImg[0]))

            # converts image to grayscale
            gray = cv2.cvtColor(np.uint8(img), cv2.COLOR_BGR2GRAY)

            # loads and applies the Haar cascade classifier to detect humans in image
            human_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')
            humans = human_cascade.detectMultiScale(gray)

            # loop, through detected human images, annotates them with a bounding box
            # and prints a timestamp and an info message on the console
            for (x, y, w, h) in humans:

                cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
                current_time = datetime.now()
                print(current_time)
                print("Found a person in coordinates [{:.2f}, {:.2f}]"
                    .format(x_pos, y_pos))

                # saves annotated image to file with timestamp
                current_time = current_time.strftime("%Y-%m-%d_%H-%M-%S")
                filename = f"detected/IMAGE_{current_time}.png"
                cv2.imwrite(filename, img)

            t2 = self.getTime()

        # calculates the desired input values for roll, pitch, yaw,
        # and altitude using various constants and disturbance values
        roll_input = self.K_ROLL_P * clamp(roll, -1, 1)
                                    + roll_acceleration + roll_disturbance
        pitch_input = self.K_PITCH_P * clamp(pitch, -1, 1)
                                    + pitch_acceleration + pitch_disturbance
        yaw_input = yaw_disturbance
        clamped_difference_altitude = clamp(self.target_altitude
                                    - altitude + self.K_VERTICAL_OFFSET, -1, 1)
        vertical_input = self.K_VERTICAL_P * pow(clamped_difference_altitude, 3.0)

        # calculates the motors' input values based on the desired roll, pitch, yaw, and altitude values
        front_left_motor_input = self.K_VERTICAL_THRUST
                    + vertical_input - yaw_input + pitch_input - roll_input
        front_right_motor_input = self.K_VERTICAL_THRUST
                    + vertical_input + yaw_input + pitch_input + roll_input
        rear_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
                    + yaw_input - pitch_input - roll_input
        rear_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
                    - yaw_input - pitch_input + roll_input

        # sets the velocity of each motor based on the motors' input values calculated above
        self.front_left_motor.setVelocity(front_left_motor_input)
        self.front_right_motor.setVelocity(-front_right_motor_input)
        self.rear_left_motor.setVelocity(-rear_left_motor_input)
        self.rear_right_motor.setVelocity(rear_right_motor_input)
```

الآن شغِّل المُحاكاة لترى الطائرة المُسيَّرة وهي تُقلع وتُحلِّق حول المنزل. لاحظْ مُخرَجات وحدة التحكم الجديدة والصور التي تم إنشاؤها في المجلد.



شكل 6.21:

شكل 6.22: إنشاء المجلد والصور المحفوظة التي تحتوي على الاكتشافات

تمرينات

1 عدّل برنامج المُتحكِّم الخاص بك بحيث لا يتحقق من وجود المجلد بالفعل في المسار. هل يتسبب ذلك في أية تعقيدات في تنفيذ المُحاكاة؟

2 عدّل برنامج المُتحكِّم بحيث يقوم بالاكتشاف كل 10 ثوانٍ. هل تلاحظ أي فَرق في تكرار ما تطبعه وحدة التحكم وفي الصور المحفوظة؟

3 ماذا سيحدث لمُخرَجات الصورة إذا قمت بدمج أبعاد الألوان حسب التسلسل المعتاد بدلًا من التسلسل المعكوس؟ دوّن ملاحظاتك وفقًا لذلك.

4 أجرِ تجارب على المُعامِلين الرابع والخامس في الدالة rectangle(). دوّن ملاحظاتك وفقًا لذلك.

5 عدّل برنامج المُتحكّم الخاص بك بحيث يطبع قيم الالتفاف والانحدار والانعراج للطائرة المُسيَّرة عند اكتشاف أي شخص.

المشروع

في الوقت الحاضر، هناك العديد من مشاريع تكامل الذكاء الاصطناعي كبيرة الحجم التي يتم تطويرها لمختلف الصناعات والقطاعات المختلفة في البلدان، ويُعدُّ القطاع الصحي من أهم القطاعات التي تتبنى تقنيات الذكاء الاصطناعي، وهذا يعني أن تطوير المشاريع في هذا القطاع لا بُدّ أن يأخذ أخلاقيات الذكاء الاصطناعي بعين الاعتبار.

1 أجرِ بحثًا عن أنظمة الرعاية الصحية التي تعمل بالذكاء الاصطناعي وعن آثارها الأخلاقية، وحدِّد المنافع والمخاطر المحتملة لتطبيق نظام تقنية معلومات يعمل بالذكاء الاصطناعي في مؤسسة صحية.

2 حلِّل المخاوف الأخلاقية التي تنشأ عند استخدام الذكاء الاصطناعي في اتخاذ قرارات تؤثر على صحة المريض، وضَعْ مجموعة من المبادئ الأخلاقية لاستخدام الذكاء الاصطناعي في الرعاية الصحية تعطي الأولوية لسلامة المريض وصحته.

3 أنشئ عرضًا تقديميًا يحدِّد المبادئ الأخلاقية المقترحة والأسباب التي تدعو إلى الالتزام بها، واعرض المبادئ على زملائك في الفصل، ثم ناقش معهم مزايا وتحديات المبادئ المقترحة.

ماذا تعلّمت

- معرفة لمحة عامة عن أخلاقيات الذكاء الاصطناعي.
- فحص كيف يُمكن للتحيُّز والافتقار إلى الإنصاف أن يُؤديا إلى إساءة استخدام أنظمة الذكاء الاصطناعي.
- تحديد طرائق التخفيف من مشكلة الشفافية لقابلية التفسير في الذكاء الاصطناعي.
- تقييم كيفية توجيه التنظيمات والمعايير الحكومية للاستخدام الأخلاقي والمستدام لأنظمة الذكاء الاصطناعي.
- برمجة الطائرة المُسيَّرة للتنقل في بيئة ما دون تدخل بشري.
- تعديل نظام الطائرة المُسيَّرة لتشمل قدرات المراقبة من خلال تحليل الصور.

المصطلحات الرئيسة

وحدة قياس بالقصور الذاتي	Inertial Measurement Unit - IMU
محرِّك	Motor
مكتبة أوبن سي في	OpenCV Library
الانحدار	Pitch
مروحية	Propeller
الروبوتية	Robotics
الالتفاف	Roll
مُحاكِي	Simulator
الاستدلال القائم على القِيم	Value-Based Reasoning
الانعراج	Yaw

أخلاقيات الذكاء الاصطناعي	AI Ethics
مراقبة المنطقة	Area Surveillance
التحيُّز	Bias
مشكلة الصندوق الأسود	Black-Box Problem
إلغاء الانحياز	Debiasing
نظام تحديد المواقع العَالَمي	Global Positioning System - GPS
الجيروسكوب	Gyroscope
اكتشاف البشر	Human Detection