التعلم العميق Deep Learning

- **الدرس الأول : التعلم العميق و الشبكات العصبية**
  - الأسبوع الأول : مقدمة للتعلم العميق
  - الأسبوع الثاني : أساسيات الشبكات العصبية
  - الأسبوع الثالث : الشبكات العصبية المجوفة
  - الاسبوع الرابع : الشبكات العصبية العميقة
- **الدرس الثاني : تطوير الشبكات العميقة : المعاملات العليا**
  - الأسبوع الأول : السمات العملية للتعلم العميق
  - الأسبوع الثاني : الحصول علي القيم المثالية
  - الأسبوع الثالث : ضبط قيم الشبكات العميقة
- **الدرس الثالث : هيكلية مشاريع الـ ML**
  - الأسبوع الأول : استراتيجيات الـ ML - 1
  - الأسبوع الثاني : استراتيجيات الـ ML - 2

- **الدرس الرابع : الشبكات العصبية الملتفة CNN**
  - الأسبوع الأول : أساسيات الشبكات العصبية الملتفة
  - الأسبوع الثاني : حالات عملية من الشبكات العصبية الملتفة
  - الأسبوع الثالث : التعرف علي الأشياء
  - الاسبوع الرابع : التعرف علي الوجه
- **الدرس الخامس : الشبكات العصبية المتكررة RNN**
  - الأسبوع الأول : مفهوم الشبكات العصبية المتكررة
  - الأسبوع الثاني : المعالجة اللغوية الطبيعية NLP
  - الأسبوع الثالث : نماذج التتابع

درس 4: الشبكات العصبية الملتفة CNN

الأسبوع الثالث : التعرف علي الأشياء

==========

الهدف من هذا الأسبوع :

- التعرف علي مفهوم "موضع الجسم" , "اتجاه الجسم" , "البحث عن الخطوط التعريفية"
- فهم طريقة حذف القيم الدنيا non-max suppression
- فهم طريقة التقاطع عبر الاتحاد
- التعرف علي كيفية عنونة قاعدة بيانات للتعرف علي الاشياء
- التعرف علي مصطلحات التعرف علي الأشياء

-

بداية مجال التعرف علي الأشياء object detection هو من أكثر مجالات الـ DL تطور في السنين الاخيرة , حيث تضاعفت البحوث و الخوارزميات الخاصة به في الفترة الأخيرة أضعافا كثيرة .

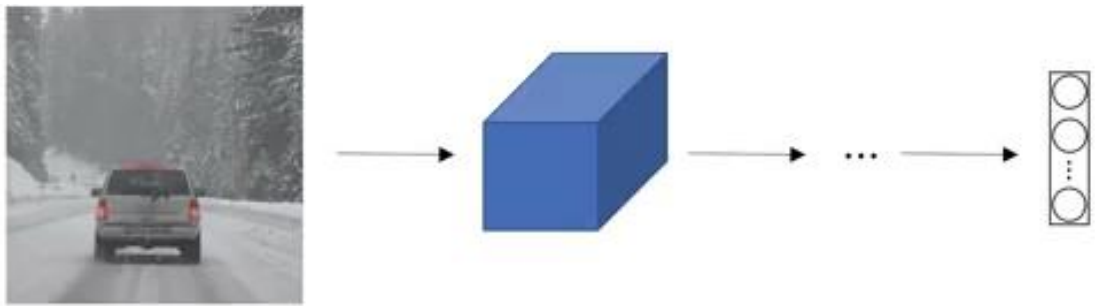
و للتعامل مع التعرف علي الأشياء , لابد أولا من التعرف علي ما يسمي موضع الأشياء Object Localization

درسنا في السابق , كيف يقوم الخوارزم بتصنيف الصور التي تدخل اليه اذا ما كانت سيارة ام غيرها .

لكن الجديد , أن يقوم الخوارزم أولا بوضع إطار (غالبا بلون أحمر) , علي الجزء المطلوب , قبل تصنيفه اذا ما كان سيارة أم غيرها , وهو ما يسمي تحديد الموضع مع التصنيف classification with localization

وعبر تطوير هذا الأمر , يظهر ما يسمي التعرف علي الأشياء object detection وهي امكانية الخوارزم علي التعرف علي جميع الاشياء التي تظهر امامه في الصورة , سواء سيارات او مشاة او دراجات او غيرها

بفرض ان لدينا شبكة ملتفة CNN تتناول الصورة الملتقطة من كاميرا السيارة لتصنيفها بين أربع اختيارات . .



إما : مشاة , او سيارة , او موتوسيكل , او شئ أخر (جز من الخلفية)

ما الجديد ؟

أن هذا التكنيك ليس فقط لتصنيف الصور , ولكن كذلك لتحديد موضعها , فهي كما ذكرنا تكنيك : classification with localization , لذا فعلينا أن نقوم بتحديد موضع كل صورة . .

ويتم هذا عبر تحديد أعلي يسار الصورة علي أساس أنه المركز , ويكون 0,0 , علي أن يكون اسفل يسار الصورة نهاية التحديد و يكون بـ 1,1

ثم ان يقوم الخوارزم بإيجاد اربع قيم أخري و هي :

bx , by , bh , bw

و المقصود بها قيم موقع مركز المربع الأحمر , وكذلك ارتفاعه و عرضه . .

ففي الصورة , يمكن أن نقول أن bx = 0.5 بينما by = 0.7 أما bh = 0.3 و bw = 0.4

و بالتالي مصفوفة y النهائية للخوارزم يمكن أن تكون كالتالي :

Pc
bx
by
bh
bw
c1
c2
c3

حيث قيمة Pc هي تحدد احتمالية probability تواجد جسم او هو خلفية , فلو كان الجسم هو مشاة او سيارة او موتوسيكل , ستكون قيمتها 1 اما لو كان شئ اخر او جزء من الخلفية فستكون صفر

قيم bx , by , bh , bw هي القيم التي تحدد مكان و ابعاد مربع التعيين الأحمر

قيم c1,c2,c3 هي التي تحدد اذا ما كان الشئ المحدد مشاة, ام سيارة او موتوسيكل , فسيكون الرقم الخاص بها 1 و الباقي اصفار , وبالطبع هذه السيهات ستزيد اذا ما كان مطلوب التصنيف بين 10 اختيارات مثلا .

فمع هذا المثال :

ستكون المصفوفة :

1
0.5
0.7
0.3
0.4
0
1
0

أما في حالة عدم وجود أي شئ من المذكور في الصورة , فتكون المصفوفة كالتالي :

0
?
?
?
?
?
?
?

علامات الاستفهام تعني أننا لا نهتم بقيمها , ولن نحسبها , فهي لن تمثل لنا اهمية طالما اختفي الشئ المطلوب

أما عن معادلة الخطأ التي سيقوم الخوارزم بتقليلها , ففكرتها تقوم علي مجموع مربعات الفوارق بين كل قيمة من قيم المصفوفة الثمانية (هنا القيم ثمانية لان الاختيارات ثلاث , و عادة تكون عدد قيم المصفوفة هي 5 + اختيارات الخوارزم)

ففي حالة كانت Pc = 1 ستكون :

$$L(\hat{y}-y) = (\hat{y}1-y1)^2 + (\hat{y}2-y2)^2 \ldots \text{till y 8}$$

أي اننا نقوم بطرح كل القيم الثمانية للمصفوفة الناتجة , من القيم الحقيقية التي سيتم ادخالها في الـ inputs

أما في حالة كانت Pc = 0 فستكون :

$$L(\hat{y}-y) = (\hat{y}1-y1)^2$$

فقط , وذلك أن الجسم أساسا غير موجود , فلو قام الخوارزم بتوقع ان الجسم غير موجود $\hat{y}1=0$
فيختفي الخطأ , اما لو توقع انه موجود , فيكون الخطأ 1 , دون النظر لباقي العوامل الغير موجودة اساسا

-

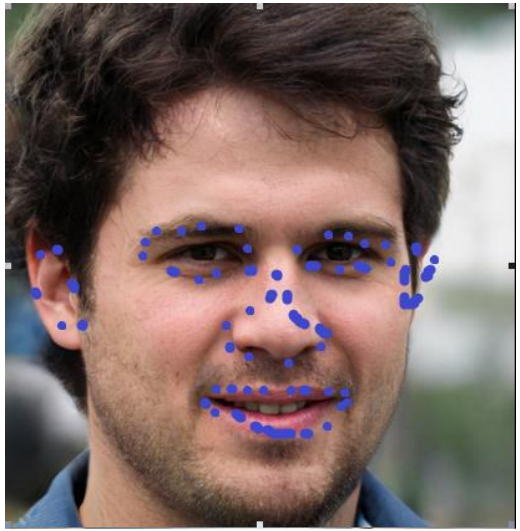
نتعرف الآن علي ما يسمي : الخطوط الفاصلة landmark . .

وهي الخطوط التي يقوم الخوارزم بتحديدها لمعرفة المكان التقريبي للجزء المهم , ليبدأ بمعالجته .

فإذا تناولنا صورة مثل هذه :

فلو اردنا بناء خوارزم , يقوم بتحديد نقاط معينة في الوجه , مثل العيون , الانف , الفم , او نريد ان نقوم بتحديد نقاط حول الفم مثلا , لتحديد هل هو مبتسم ام غير مبتسم ام ماذا . .

فلو قلنا انه مطلوب عدد 64 نقطة مثل هذا :



فنقوم وقتها بتعيين 129 رقم , أول رقم هو face اي تري هل هذا وجه ام لا فإذا كان قيمته 1 يكون وجه , اذا كان 0 يكون ليس وجه .

و باقي الارقام الـ 128 , هي عبارة عن قيم x , y لكل قيمة من النقاط الـ 64 , بحيث تكون :

L1x
L1y
L2x
L2y
.

وهكذا إلي باقي النقاط . . .

و عبر تدريب الخوارزم بالاف الصور و تعيين ارقام له , سيمكنه من تحديد هذه النقاط , وهي ما نسميها الخطوط الفاصلة landmark لأن هذه النقاط و الخطوط مهمة في تحديد اجزاء لاحقة

و اهميتها تتجاوز التعرف علي الوجه , بل هي في استخدامات الجرافيك (لعمل جروح او تشوهات علي الوجه), والفلترز المستخدمة في الصورة (فيسبوك و سنابشات) و غيرها

و كذلك الأمر في الحركات الجسدية , للاعب او شخص يمشي او يجري , يمكن تحديد نقاط معينة في جسده لمعرفة شكل حركته :

وبالطبع لابد من توحيد مسميات النقاط L فلو كانت L1 هي خاصة بطرف الانف , فلابد ان تكون كذلك في كل الصور , لتجنب اي اختلاط في الارقام

-

الان نتجه الي التعرف علي الأشياء object detection وهي المبنية علي كلا من موضع الأشياء Object Localization , و الخطوط الفاصلة landmark .

و سيتم استخدام تكنيك النوافد المتحركة sliding windows .

بفرض أننا نريد بناء خوارزم يقوم بفحص الصورة , والتعرف علي السيارات , فعلينا أولا البحص عن قاعدة بيانات dataset تشمل صور سيارات , ويكون قيمة y لها 1 و صور اخري و قيمة yصفر مثل هذا :

Training set:

x	y
	1
	1
	1
	0
	0

و ينصع ان يتم ضبط الصور crop تماما , بحيث تحتوي علي السيارة فقط , دون احتوائها علي اي عناصر اخري , لتسهيل عمل الخوارزم .

فيتم بناء شبكة ملتفة , بحيث تأخذ صور السيارات و وتتوقع قيمة y هل هي صفر او واحد

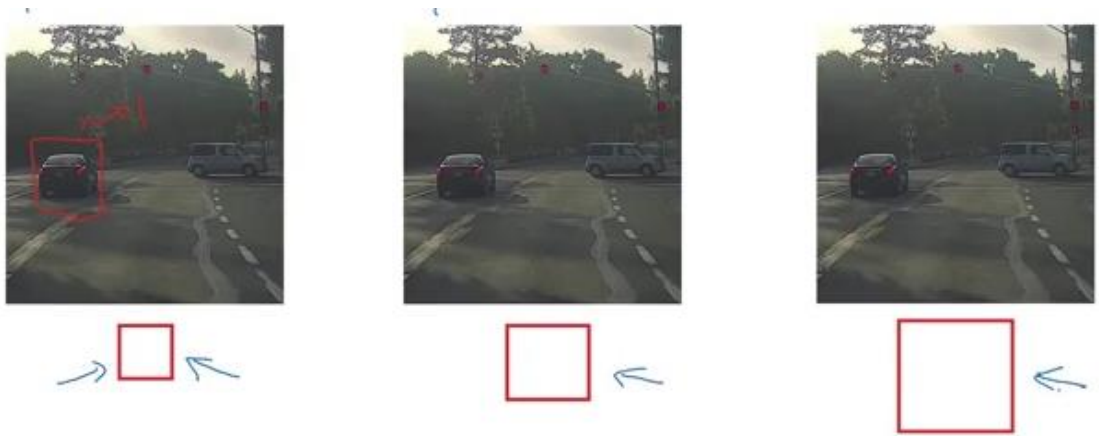
وفكرة النوافذ المتحركة , تقوم علي انشاء مربع بمقاس مناسب , وان يقوم هو بالتحرك عبر الصورة من اليسار إلي اليمين , ثم ينزل أسفل و يعيد التجربة , بداية من أعلي يسار الصورة حتي يصل إلي أسفل يمين الصورة :

ويجب علي المربع ان يكون بمقاس مناسب , لا هو كبير فيكون به اشياء اكثر من السيارة , ولا هو صغير فلا يستطيع تغطية السيارة بالكامل , يجب ان يكون بمقاس مناسب , حسب الصور التي سيتعامل معها .

اما الخطوة فهي تكون غالبا بمقدار بسيط للغاية ,حتي لا يقوم بتفويت اي جزء من الصورة .

وحينما تقل قيمة الخطوة stride تزيد الدقة , لكن الوقت و التكلفة تزداد , واذا قمنا بتقليل قيمة الخطة , سيقل الوقت لكن ستقل الكفاءة , لان من ان يعجز الخوارزم عن ايجاد عدد من الصور

و غالبا ما تتكرر العملية عدة مرات , كل مرة بحجم مختلف للنافذة , حتي يتم التأكد من تغطية الصورة بشكل كامل .



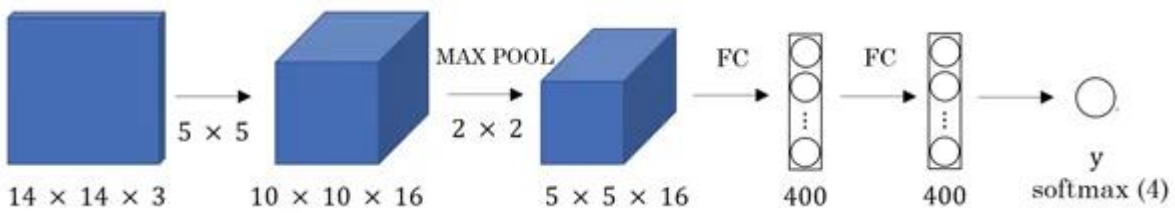
والعيب الأساسي في هذا التكنيك , هو التكلفة العالية + الوقت المطلوب لها , حيث ان الصورة الواحدة قد تحتاج لالاف النوافذ , وكل نافذة فيهم يتم عمل فحص كامل لها في الشبكة الملتفة , وهذا الوقت الكبير المستغرق لا يتناسب بالطبع مع السيارات ذاتية القيادة , التي تحتاج لقرارات لحظية فورية .

و لهذا فهذه الفكرة بشكلها الحالي غير مفيدة , فتم عمل تطوير لها عبر الـ CNN

-

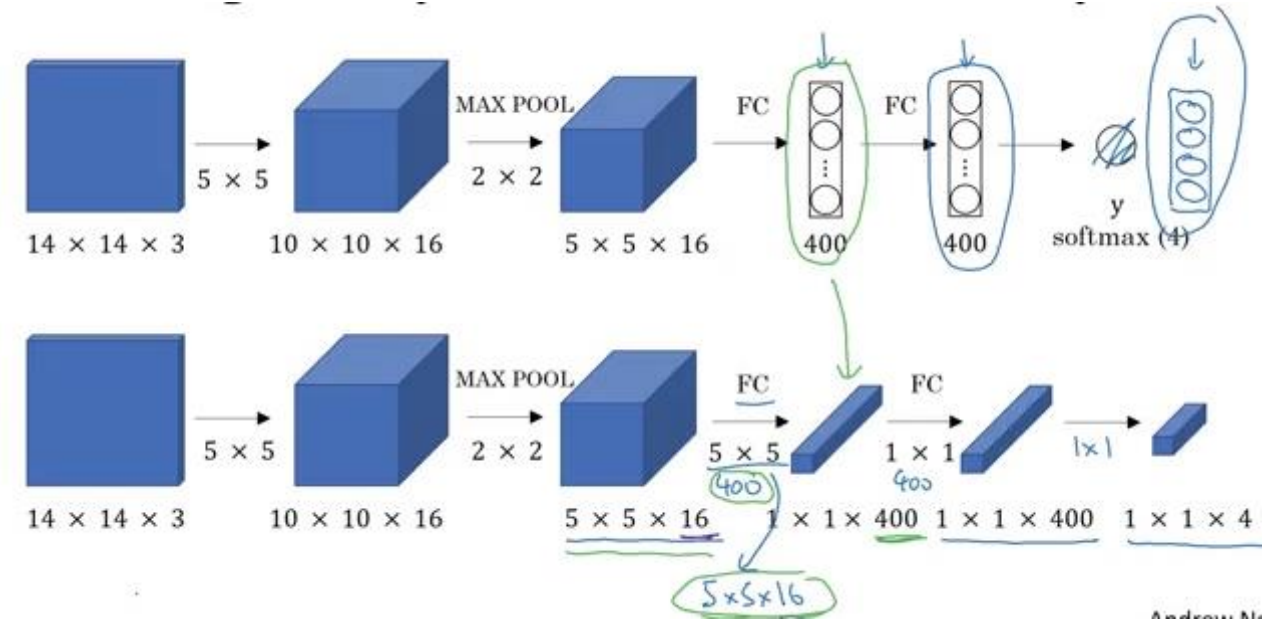
لفهم كيفية تطوير فكرة النوافذ المتحركة عبر الـ CNN علينا أولا ان نتعرف علي كيفية تحويل الـ Fully connected layer الي CNN

لو كان لدينا صورة صغيرة 14*14 , فيمكن ان تمر بهذه المراحل للوصول الي سوفتماكس 4 اختيارات (مشاة , سيارة , موتوسيكل , آخر) :



و لبدء تحويل الـ FC إلي CNN , فسنترك المراحل الأولي كما هي , وسنمسك طبقة الـ FC الأولي , وسنستبدلها بفلتر 5*5 , و عدده 400 , ويجب أن يكون بنفس عمق المكعب السابق له وهو 16

و هذا سينتج مكعب طويل بابعاد 400*1*1 , لان الفلتر 5*5 و المكعب السابق له بنفس 5*5 , وهو ما يعني ان الابعاد يجب ان تكون 1*1 , والعمق بنفس عدد الفلتر 400



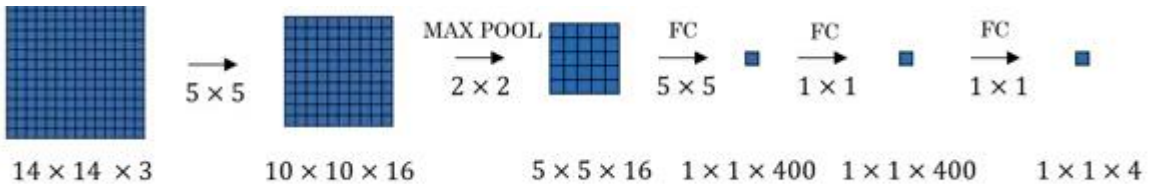
لاحظ أن هذا المكعب 400*1*1 هو مشابه الي حد ما مع الـ FC المناظرة لها و التي هي كذلك 400 صف في عمود واحد

كذلك يتم تكرار الأمر عبر عمل فلتر 1*1 , بعمق 400 , وعدده 400 كذلك , فينتج مكعب جديد 400*1*1 .

بعدها يتم عمل فلتر 1*1 و عدده اربع , فينتج 4*1*1 . وهي المناظرة للسوفتماكس .

ما هو التطوير الذي حدث في تكنيك النوافذ المتحركة ؟

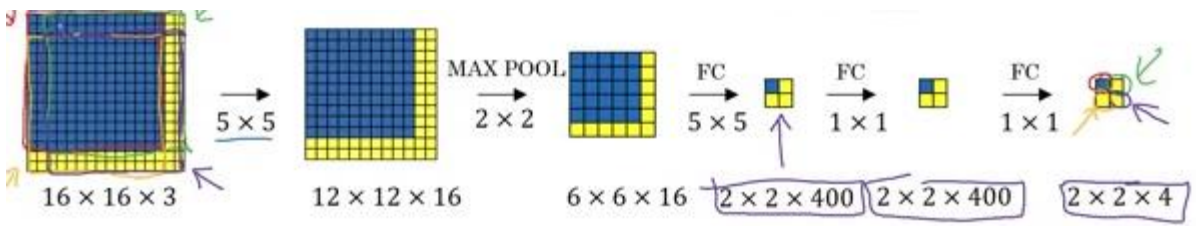
اذا فرضنا ان الصورة ذات الابعاد 14*14 و هي 3 الون , يتم معالجتها هكذا عبر الـ CNN :



واذا كان لدينا الصورة التي سنقوم باختبارها , هي 16 في 16 .

فالطريقة التقليدية , ان نقوم بعمل نافذة متحركة اربع مرات اعلي اليسار , اعلي اليمين , اسفل اليسار و اسفل اليمين .

لكن يمكن معالجة الأمر أسرع , عبر معالجة الـ 16 رقم مرة واحدة , عبر استخدام شبكة ملتفة 14*14 , و سيكون الناتج هو 4*4 (و لها عمق 4 لانه عدد الاختيارات)

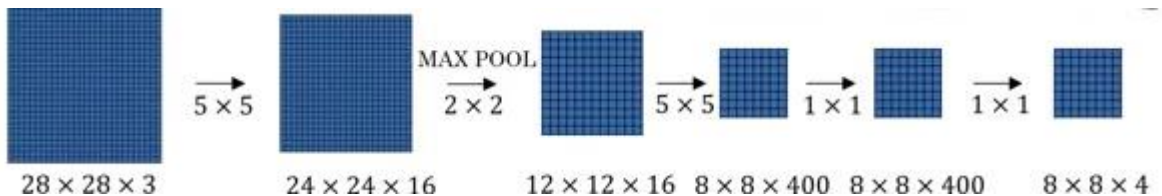


الان اذا اردنا ان نعرف نتيجة النافذة اعلي اليسار , فهي الرقم اعلي اليسار في المصفوفة اليمني , واعلي اليمين هو اعلي اليمين و هكذا

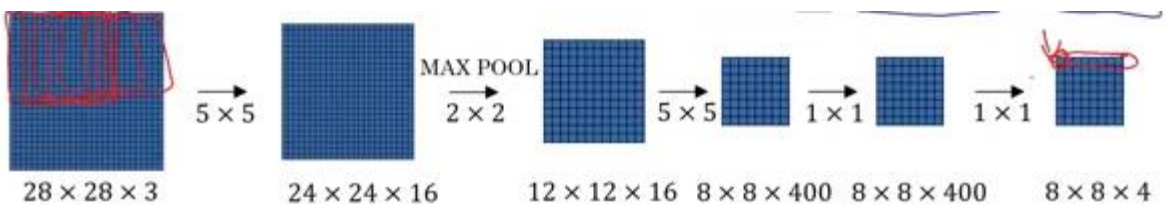
فكل سهل من المصفوفة اليمني (الناتج) يشير الي نتيجة فحص النافذة المناظرة لها في الصورة الاصلية .

فهكذا قمنا باختصار وقت كبير , بدلا من عمل النوافذ عدة مرات .

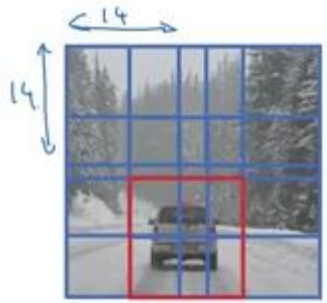
واذا تكلمنا عن مثال اكبر , وليكون صورة 28*28, ويكون الفلتر 14*14 , فيكون الناتج 8*8 (بعمق 4 اختيارات)

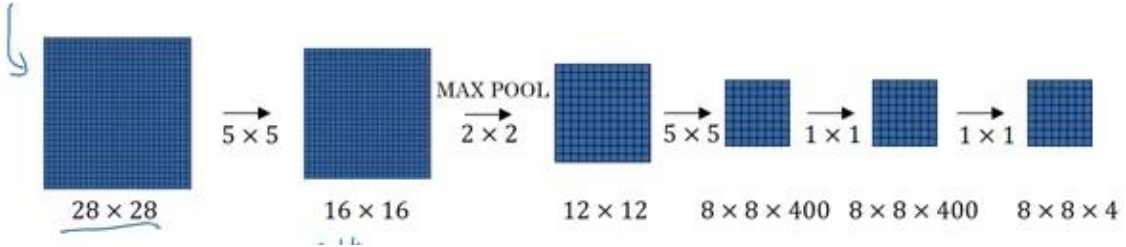


وهكذا , فتكون اول نافذة اعلي اليسار , هي اول قيمة اعلي يسار المصفوفة الناتجة و هكذا س



وبالتالي حينما تأتي صورة كبيرة لنا , فيمكن عمل فلتر مناسب لها , بحيث يتم فحصها مرة واحدة , وتكون النقطة المناظرة للصورة هي بالقيمة المطلوبة

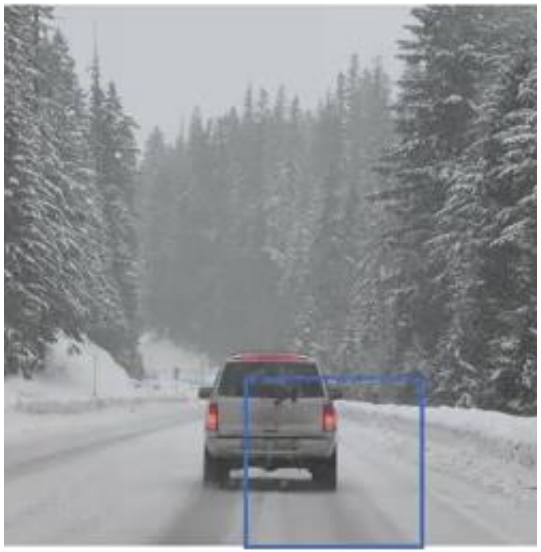


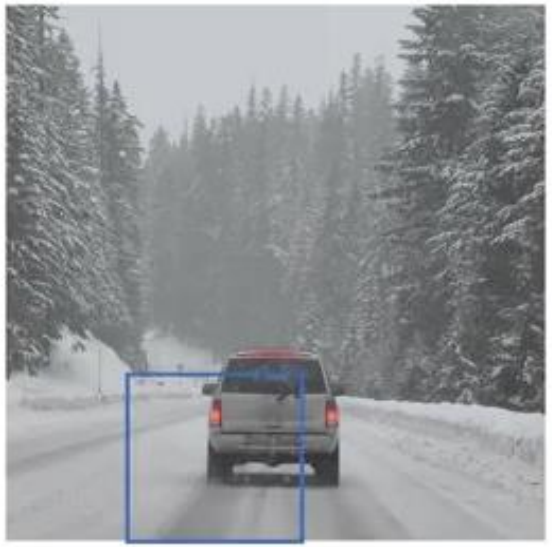


-

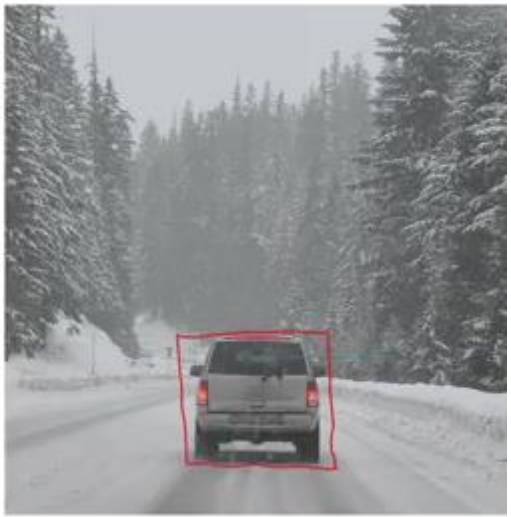
تكنيك النوافذ المتحركة , علي الرغم من كفائته إلا أن به مشكلة , وهو عدم دقته في توقع مربع التحديد bounding box prediction , وهو ما سنتكلم عنه الان . .

ولفهم معناه , علينا ان نتذكر ان النوافذ المتحركة , قد تعجز عن تغطية و تحديد مكان الجزء المطلوب (السيارة في هذا المثال) , فقد تكون قبلها قليلا او بعدها قليلا مثل هذا :





كما ان السيارة نفسها لن يتناسب معها مربع , لكن يجب أن يكون مستطيل :

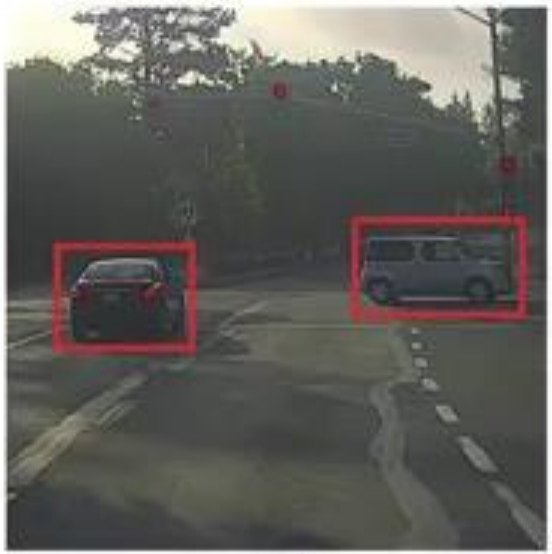


كيف نجعل الخوارزم يتوقع المقاس المطلوب لتغطية السيارة , والمكان المناسب لها ؟

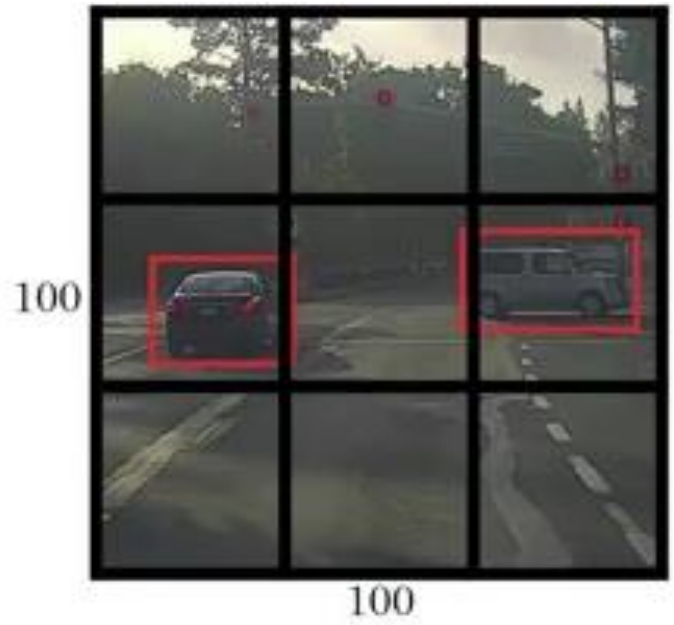
أحد الطرق المستخدمة لعلاج هذه المشكلة تسمي YOLO و هي اختصار you only look once يعني انت تنظر مرة واحدة .

و فكرتها تقوم علي تقسيم الصورة الموجودة لعدد من الاقسام المربعة (مثلا 3*3) , وان يقوم كل مربع علي حدة بفحص الصورة لتحديد هل فيه الجزء المطلوب ام لا .

فلو كان لدينا صورة مثل هذه , ونريد معرفة اذا كان فيها سيارات ام لا :



فيمكن تقسيمها لتسع مربعات متساوية , لفحصها كلها بشكل سريع هكذا :



بحيث يكون ناتج فحص كل مربع المصفوفة ذات الثمانية أرقام التي تحدد هل توجد سيارة ام لا , ومكان و مقاس مربع التحديد :

Pc
bx
by
bh
bw
c1
c2
c3

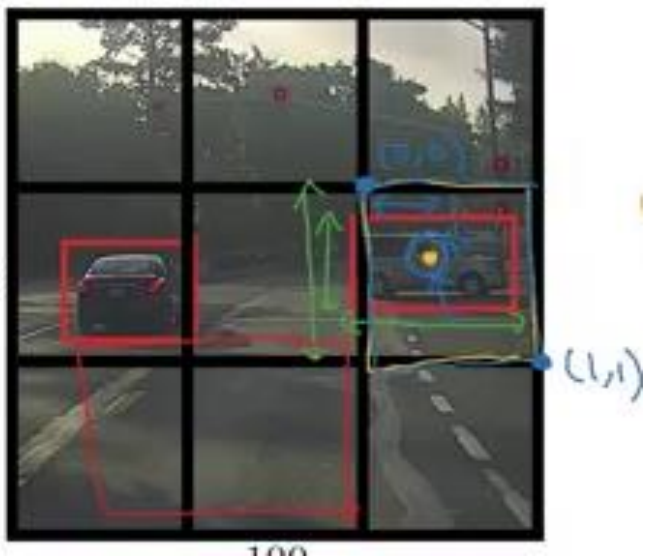
فيبدو واضحا أن المربعات (1,2,3,5,7,8,9) ستكون قيم Pc تساوي 0 , وباقي الارقام ليس لها قيمة , بينما المربعين الرابع و السادس ستكون Pc تساوي 1 , وباقي القيم تحدد التفاصيل المطلوبة .
وكأن المصفوفات التسعة الناتجة , وكل مصفوفة فيهم 8 ارقام , ستكون كأنها مكعب 3*3 بعمق 8

و يكون واضحا ان ناتج فحص المربع أعلي يسار الصورة , سيكون قيمته اعلي يسار المكعب و هكذا .

و غالبا ما تكون المربعات التي يتم تقسيمها اكثر من 9 (3*3) , وذلك للحرص علي الدقة المطلوبة , وتجبنا لتفويت سيارة قد تضيع اذا ما كانت المربعات كبيرة .

و من المشاكل المتوقعة ان يكون في نفس المربع أكثر من سيارة , وهذه المشكلة يتم حلها بجعل المربعات اصغر و أكثر .

و إذا تناولنا المربع السادس مثلا :



فسنجد أن أعلي يسار المربع هو المركز 0,0 , وأسفل يمين المربع هو النهاية 1,1 , و قيم مركز مربع التحديد ممكن ان تكون : bx = 0.4 , by = 0.3 .

بينما ابعاد مربع التحديد : bh =0.9 , bw =0.5

و لاحظ أن قيم bx , by يجب ان تكون بين الصفر و الواحد , فمركز مربع التحديد لابد ان يكون داخل المربع المختار , بينما قيم bh , bw ممكن ان تزيد عن 1 في حالة كانت السيارة متواجدة في اكثر من مربع معا .

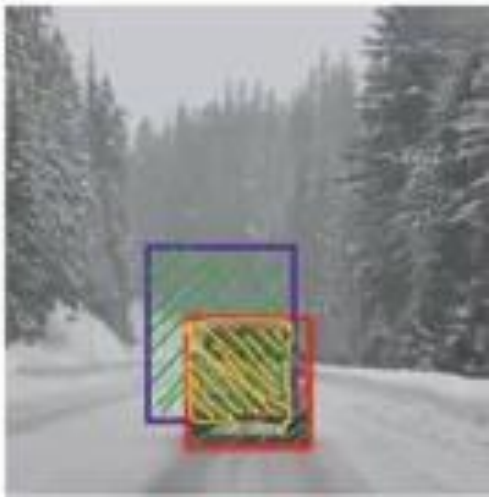
-

نتناول الآن طريقة لتقييم الخوارزم , وتحديد مدي كفائته , وتسمي : مقارنة التقاطع والإتحاد Intersection over union أو بالاختصار IOU

و فكرتها تقوم علي المقارنة بين مربع التحديد الذي قام الخوارزم بعمله , ومربع التحديد السليم الذي يحيط بالسيارة بشكل مناسب (لا اكبر منها ولا اصغر)

فإذا كان المربع الأحمر هو السليم , والازرق هو الذي قام الخوارزم بعمله :

فيتم حساب IOU عبر قسمة مساحة التقاطع بينهم (الصفراء) علي المساحة الكلية للمربعين (الخضراء)



وإذا كان المربعان متطابقان بدقة , فستكون قيمة IOU هي 1 , وهي القيمة المثلي , وإذا كان المربعان بعيدان عن بعضهما أساسا , فستكون IOU هي صفر , وهي أسوء قيمة .

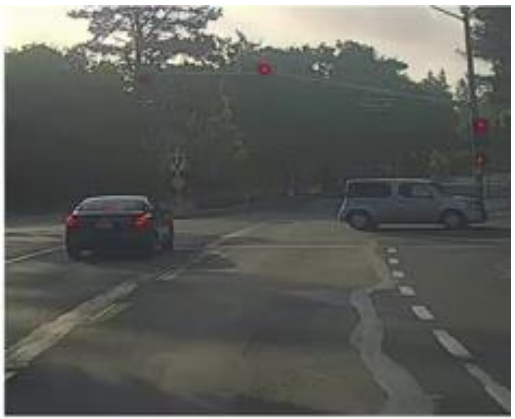
فكلما زادت قيمة IOU واقتربت من 1 كلما كان هذا دليل علي دقة الخوارزم , وكلما كانت صغيرة , كلما كان هذا يعني عدم كفائته

و غالبا ما يتم تحديد رقم 0.5 كرقم مناسب , اذا ما كانت قيمة IOU تساويه أو أكبر منه , وأحيانا لمزيد من الدقة يقولون أن IOU لابد ان تساوي او تزيد عن 0.6 واحيانا 0.7

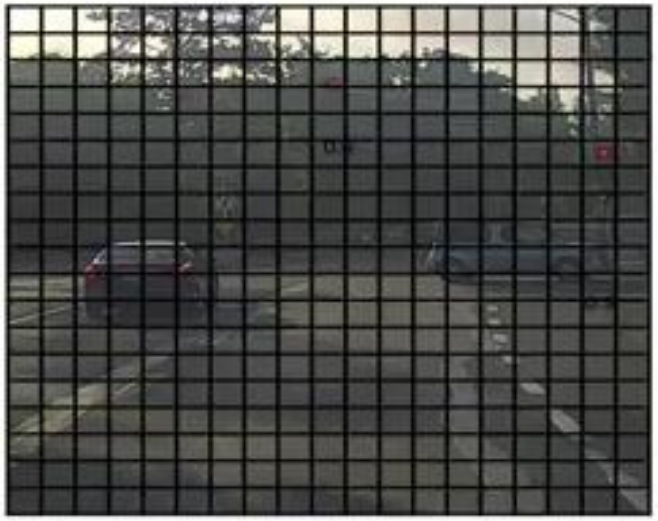
-

نتعرف الآن علي تكنيك : حذف القيم الدنيا non-max suppression , والذي يقوم بالتأكد من أن الخوارزم سيتعرف علي الجزء المطلوب مرة واحدة , وليس أكثر من مرة . .

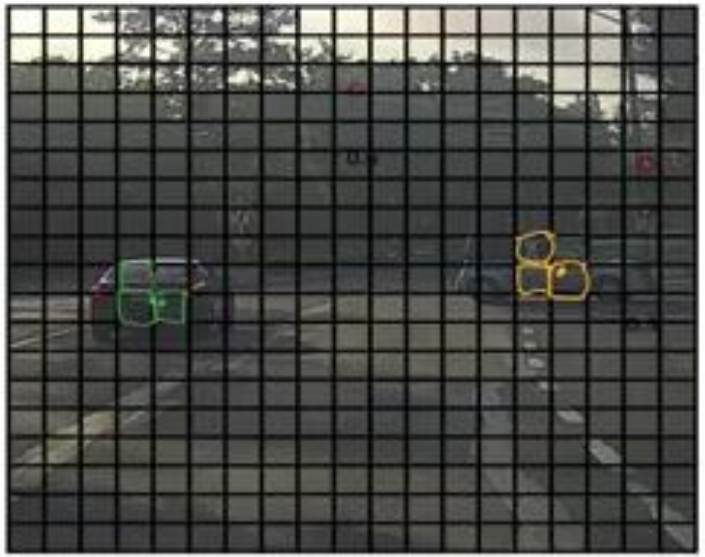
تظهر المشكلة في حالة , كان لدينا صورة مثل هذه :



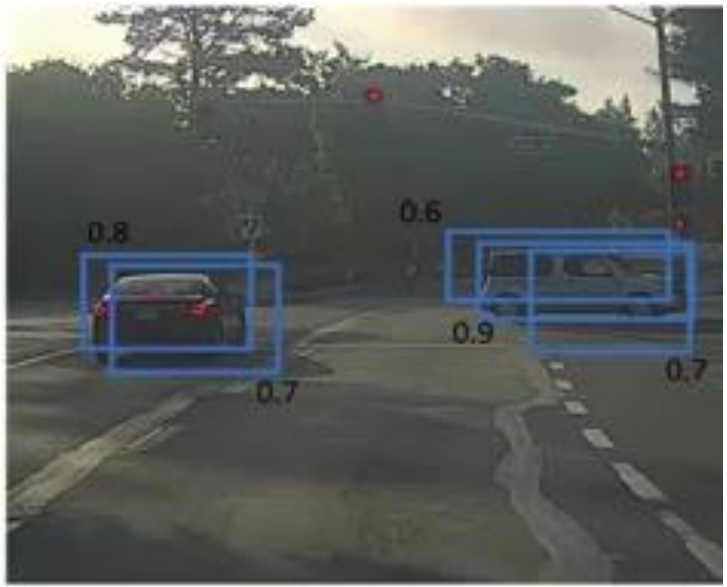
فللبحث خلالها يمكن عمل شبكة 19*19 للحبث عن سيارات , مثل هذه :



فالمشكلة ان كل المربعات التي تقع عليها السيارة , ستري اجزاء منها , وبالتالي فمركز المربع الأحمر الذي سيقوم بتحديدها قد يتكرر اكثر من مرة في اكثر من مربع متجاور .

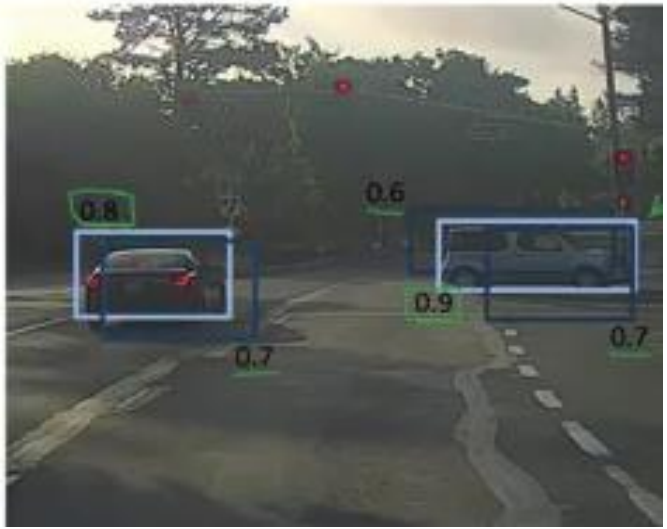


وبالتالي من الممكن ان يكون لكل سيارة مربع تحديد خاص بها , له طول و و عرض , مثل هذا :



وقتها سيقوم الخوارزم , بتحديد كفاءة كل مربع فيهم في تغطية الصورة بشكل جيد , وتحديد الكفاءة يتم عبر تحد الاختبارات ,مثل اختبار IOU السالف ذكره , و بالتالي يقوم بكتابة رقم الكفاءة علي كل مربع تحديد . .

و منها يقوم بحذف القيم الدنيا non-max suppression أي الابقاء فقط علي المربع الأعلي كفاءة , وحذف الباقين كهذا :



و تكون خطوات هذا الأمر كالتالي :

- أولا يتم حساب الاحتمالية Pc الخاصة بكل النقاط حول السيارة
- يتم استبعاد جميع المربعات ذات الإحتمالية القليلة (مثلا أقل من 0.6) , والإبقاء فقط علي ذوات الإحتمالية المناسبة
- بفرض أننا وجدنا مثلا 10 مربعات باحتمالية جيدة حول السيارة , وقتها نمسك كل مربع علي حدة , ونقوم بحساب IOU مع باقي المربعات التسعة , وذلك لتحديد مدي تكابق كل مربع مع باقي المربعات , واذا وجدنا ان هناك قيمة IOU كبيرة بين مربعين يتم استبعاد أحدهما
- و السبب في ذلك , انه حينما تكون قيمة IOU كبيرة بين مربعين , فهذا يعني انها شبه متطابقين علي السيارة , وهذا معناه ان واحد فيهم يكفي و الثاني لا نريده

- حينما تقل المربعات , يتم تحديد ايهما اعلي في الـ Pc و اختياره
- يتم تكرار كل هذه الخطوات من البداية للمشاة , فإذا كان الخوارزم الخاص بنا يقوم بتحديد سيارات + مشاة + دراجات , فيجب ان يقوم بإعادة هذا الأمر من البداية لكل عنصر علي حدة , لان كل عنصر فيهم منفصل عن الباقين

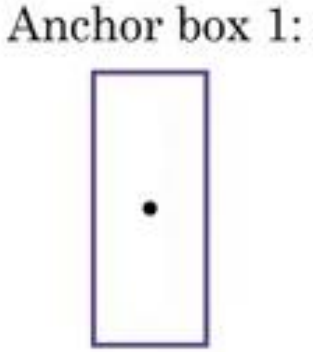
-

ماذا عن خوارزم واحد يقوم بتحديد اكثر من عنصر في الصورة ؟ ؟

هذا ما يسمي طريقة المربع المحدد Anchor boxes

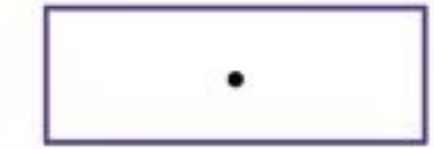
و الفكرة تقوم علي تحديد عدد من المربعات او الإطارات . والتي ابعادها تتناسب مع ابعاد الاجزاء التي نريد رؤيتها في الصورة .

فإذا كنا في الصورة نريد من الخوارزم اكتشاف مشاة + سيارات , فسنجد ان المشاة لهم مربع تحديد مثل هذا



بينما السيارة لها مربع تحديد مثل هذا

Anchor box 2:



وبالتالي يتم تحديد كلا من المربعين في الخوارزم , وهنا تكون المصفوفة السابقة هذه :

Pc
bx
by
bh
bw
c1
c2
c3

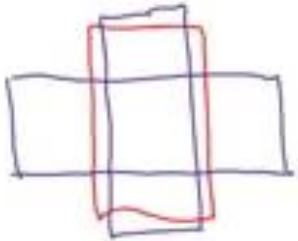
لن تكون مناسبة , حيث اننا سنحتاج الي 16 خانة بدلا من ثمانية لوضع 8 ارقام لكل مربع تحديد :

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

حيث الجزء الأول (1-8) من المصفوفة للمربع الأول , و الجزء الثاني (9-16) للمربع الثاني .

و إذا أردنا خوارزم يكتشف مثلا خمس اشياء في نفس الوقت , ستكون المصفوفة 40 عنصر , ثمانية لكل مربع منهم .

و حينما يبدأ الخوارزم في البحث عن الأشياء , سيقوم بمقارنة المربع الذي قام هو بتحديده , مع المربعات الموجودة لديه , لمعرفة ايهما يناسبه أكثر , وذلك بتكنيك IOU



والذي سيري منه ان الجسم الذي رآه (لون بنفسجي) , يتناسب مع المربع الأول و ليس الثاني , لان قيمة IOU معه اكبر .

وهو ما معناه أنه سيكون قادر علي معرفة , اي مربع من المربعات السابق معرفته بها , يتناسب مع الجسم الذي رآه .

وبالتالي اذا كان لدي صورة فيها مشاة و سيارة , وكان لدي المربعين بهذا التصميم :

Anchor box 1: Anchor box 2:

فتخرج المصفوفة تحتوي علي المعلومات عن كلا الجزئين معا هكذا (الأحمر للماشي و الأزرق للسيارة):

1
0.5
0.3
0.2
0.1
1
0
0
1
0.6
0.5
0.8
0.7
0
1
0

وفي حالة كانت الصورة بها ماشي فقط , وليس سيارة , فيكون الجزء الأول من المصفوفة به التفاصيل , والجزء الثاني فارغ طالما كان اول رقم به هو الصفر

1
0.5
0.3
0.2
0.1
1
0
0
0
?
?
?
?
?
?
?

والعكس اذا كانت الصورة بها سيارة دون ماشي :

0
?
?
?
?
?
?
?
1
0.6
0.5
0.8
0.7
0
1
0

وبالتالي مصفوفة الخارج في حالة كانت الصورة مقسمة لتسع مربعات (3*3) ستكون :

3 x 3 x 16 (2*8)

ولو كنا نريد فحص اكتر من جزء في نفس الوقت , فسيكون الرقم مضروب في 8

و هناك مصدرين للمشاكل , أولا ان يكون هناك جسم ثالث بابعاد مختلفة انت لم تقم بتحديد مربع anchor خاص به , فهذا يمكن ان يؤثر علي حساب الخوارزم

ثانيا ان يكون هناك جسم ثالث لكن مشابه لابعاد احد مربعات الـ anchor فممكن ان يقوم بلغبطة الخوارزم .

أخيرا , كيف يمكن تحديد عدد المربعات المطلوبة ؟ ؟

هناك ثلاث توجهات :

الأول ان يتم يدويا , يعني ان نقوم بتحديد المربعات من البداية بنسب محددة , وهذا في حالة اننا نعلم بالتأكيد ما هي الاجسام التي سنراها في الصورة

الثاني ان نقوم بتحديد مثلا 6 مربعات بنسب مختلفة , لكي تتناسب مع اغلب الاجسام التي قد تظهر

الثالث ان نستخدم آلية للـ Unsupervised ML مثل الـ K-means لنقوم بإعطائه عدد كبير من الصور , ويقوم هو بتقسيمها لمجموعات متشابهة , لمعرفة ما هي الابعاد المناسبة .

-

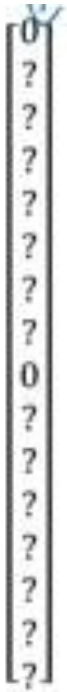
و لعمل ملخص و تجميع لما سبق . .

فلو كان لدينا صورة محددة , ونريد اكتشاف لو فيها سيارة او مشاة مثلا , وبالتالي يكون لدينا اثنين من الـ anchor box :

فنقوم ببناء مصفوفة 16 رقم :

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

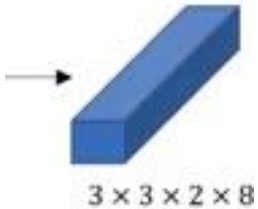
و حينما يقوم بفحص الأقسام التسعة في الصورة , سيجد أن الأقسام (1,2,3,4,5,6,7,9) كلها ليس فيها اي مشاة او سيارات , فستكون المصفوفة لهم هكذا :



أما حينما يصل للمربع الثامن , فسيجد فيه سيارة , وتكون المصفوفة :

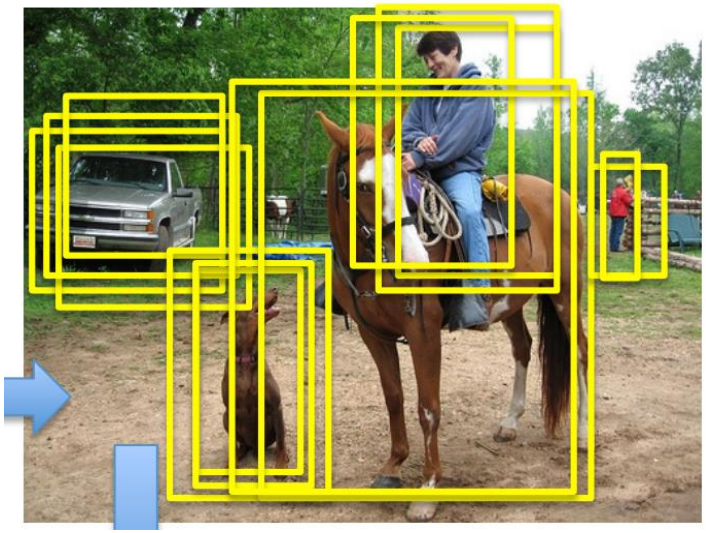
$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

و بالتالي سيكون المخرج النهائي هو مصفوفة 16*3*3 , وبالطبع سيختلف الرقم 16 اذا ما كان اكثر من جزء يتم رؤيته .

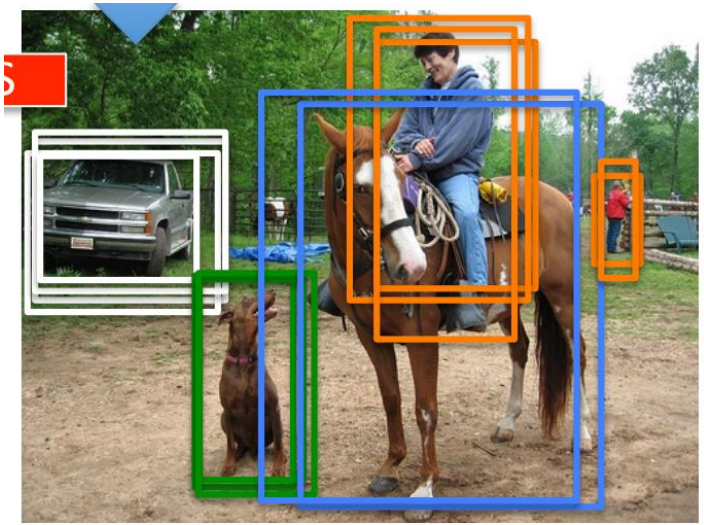


الخطو ة الأخيرة تطبيق الـ non-max suppression , فإذا كان لدينا صورة :

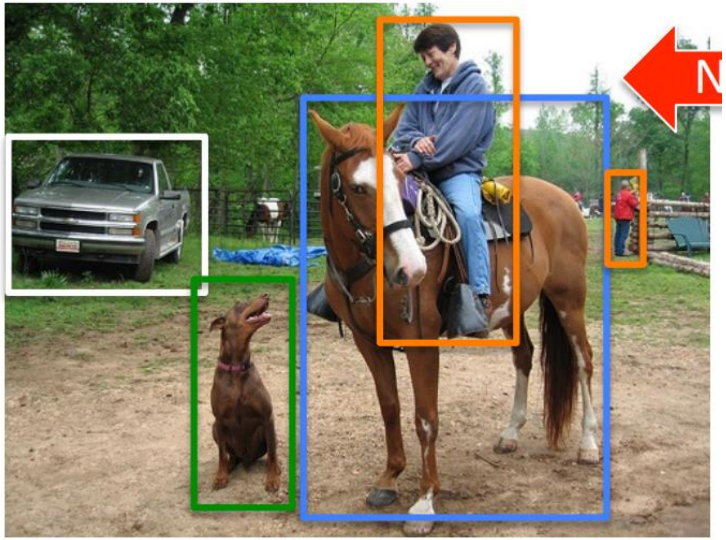
فيبدأ الخوارزم بتوقع عدد كبير من مربعات التحديد لها :



ثم يقوم باستبعاد كل المربعات ذات الاحتمالية Pc القليلة :



ثم عبر تطبيق suppression يقوم باستبعاد المربعات المتشابهة , الإبقاء علي أكبرها كفاءة

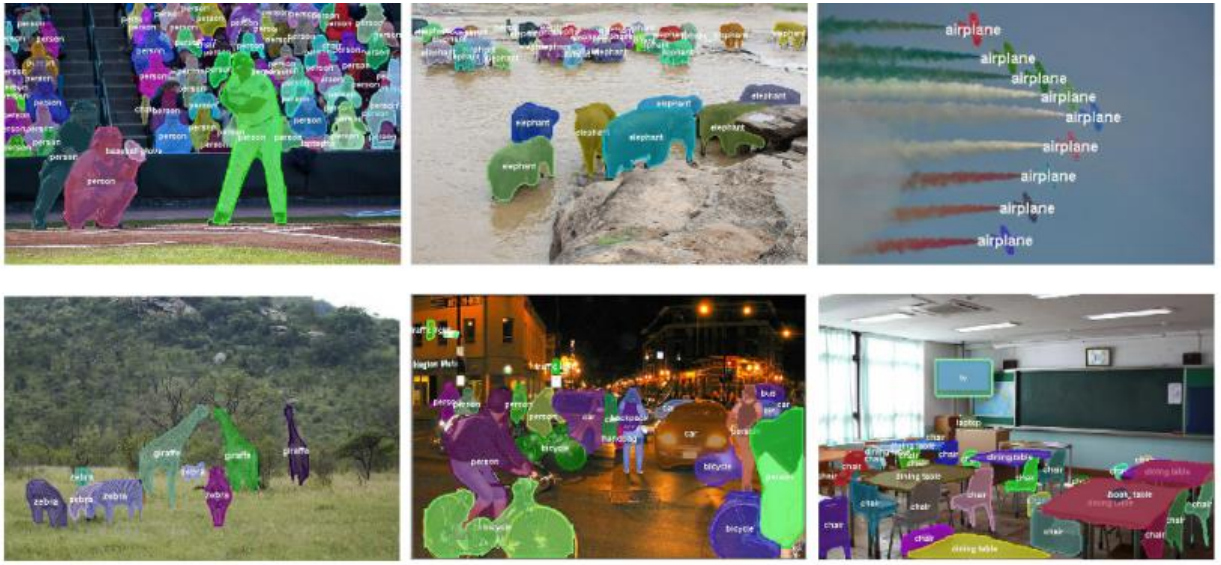


-

اخيرا , هناك تكنيك نادرا ما يستخدم , لكنه قد يكون هام , وهو يسمي : الشبكات الملتفة المقسمة للمناطق Region Conv Neural Network و تختصر R-CNN

و بالطبع ليس لها علاقة بالـ RNN و التي سنتكلم عنها لاحقا .

و هي تتعلق بفكرة , ان يقوم الخوارزم بتقسيم الصورة الي عدد من المناطق المتشابهة في الألوان , ويقوم بفحص الالوان المختلفة نوعا , بحثا عن الشئ المطلوب :



فهنا نري ان المساحات قد تم تلوينها تبعا لتشابهها , وبالتالي البحث عن الشئ المطلوب يكون اسها نوعا

لكن مشكلتها الأساسية البطئ الشديد

-

نهاية الاسبوع الثالث