الفصل الأول

مقدمة

المقدمة

بعد أن تحول العالم إلى قرية صغيرة مختزلة الحدود الجغرافية بفضل التكنولوجيا التي ماتزال تفاجئنا كل يوم بتطور وأفكار غير مسبوقة وهذا التطور السريع بنيانه وهدفه الأساسي شيء واحد ألا وهو خدمة الإنسان وتوفير طاقاته. و لكن و على الرغم من السباق التكنولوجي التطوري في حلبة العالم الافتراضي إلا أن الاحتفاظ بالسجلات و الوثائق عموماً يتم ورقياً مع الاحتفاظ بنسخ الالكترونية منها, و يتم إدخال هذه النسخ الالكترونية يدوياً في الحاسب , إلا أن هذه الآلية تؤدي تماماً إلى هدر الكثير من الوقت و الجهد الثمينين في عصر السرعة و هنا كان لابد من التقدم خطوة أخرى في هذا المضمار لتلافي الهدر المُحدث للوقت و الجهد على حد سواء, فكان للذكاء الصنعي بما تتضمنه من تقنيات دور كبير في إحداث ثورة بعالم التكنولوجيا ناهيك عن الاختزال الهائل للوقت و الجهد اللذان بطبيعة الحال يعتبران العاملان الأساسيان لتقييم أي عمل في كافة المجالات عموماً و على صعيد التكنولوجي على وجه الخصوص . لذا وبناءً على ما سبق يقودنا مشروعنا الذي سنتناوله إلى بعض التقنيات التي تتضمنها الذكاء الصنعي ألا وهي معالجة الصورة وإدراك الخط اليدوي اللتان تعتبران من أهم التقنيات الحديثة تلافياً للهدر.

تم ترتيب الفصول على النحو التالي في الفصل الأول مقدمة عن المشروع.

الفصل الثاني يتضمن لمحة عن الية التعرف على الخط اليدوي.

الفصل الثالث: يتحدث فيه عن الشبكات العصبية بشكل عام وعن الشبكات العصبية الالتفافية مع توضيح استخداماتها والفائدة منها .

الفصل الرابع: يتحدث عن المكتبات التي تم استخدامها في المشروع وتوضيح وظيفة كل منها على حدا.

الفصل الخامس: يتحدث عن النتائج التي تم التوصل اليها من خلال المشروع والتنفيذ الفعلي له مع توضيح الكود المستخدم في المشروع.

الفصل السادس والأخير: يتضمن الاستنتاجات والاقتراحات التي تم التوصل اليها من خلال المشروع.

الفصل الثاني

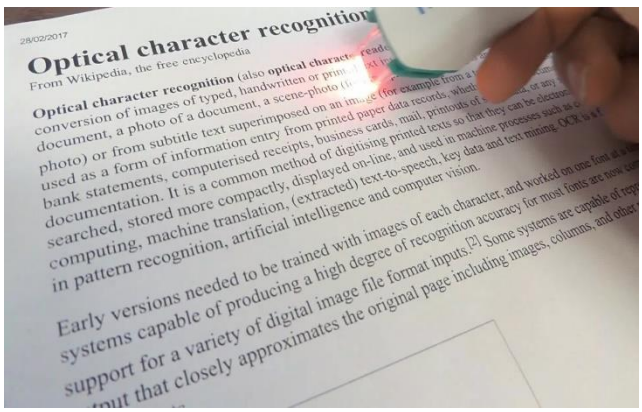
التعرف على الكتابة اليدوية

2-1 نبذة تاريخية عن التعرف على الكتابة اليدوية :

تم التحقق من التعرف التلقائي على الكتابة اليدوية منذ الخمسينات عندما أصبحت طلبات تكنولوجيا الكمبيوتر الرقمية الجديدة نسبيًا ذات أهمية. ومنذ ذلك الحين كان هناك جهد بحثي مطرد في المعالجة التلقائية والتعرف على الكتابة اليدوية. و قد أحرز التعرف التلقائي على الصيغ الرياضية والحروف المطبوعة والكتابة المتعرجة والتحقق من التوقيعات تقدم مشجع ، وكان هناك قدر محدود من البحوث في تطبيق المعالجة التلقائية للصور لتحديد الهوية الشخصية وتوثيق الوثائق من خلال تحديد هوية الكاتب أو الكشف عن الكتابة اليدوية المقنعة أو المزورة. بالإضافة إلى التعرف على النصوص والرموز الرومانية، كان هناك عمل مكثف في التعرف على الكتابة الصينية واليابانية وغيرها من النصوص المكتوبة بخط اليد.

2-2 ما هو التعرف على خط اليد؟

يستخدم التعرف على الكتابة اليدوية في أغلب الأحيان لوصف قدرة الكمبيوتر على ترجمة الكتابة البشرية إلى نص. وقد يحدث ذلك بإحدى طريقتين، إما عن طريق مسح النص المكتوب أو بالكتابة مباشرة على جهاز إدخال طرفي.



إن أول تقنيات التعرف على الكتابة اليدوية هو المعروفة باسم التعرف البصري على الحروف (OCR)، و هو الأكثر نجاحاً في هذا التيار. معظم مجموعات المسح الضوئي تقدم شكلا من أشكال التعرف الضوئي على الحروف ، مما يسمح للمستخدمين مسح في وثائق مكتوبة بخط اليد وترجمتها إلى وثائق نصية رقمية. ويستخدم بعض أمناء المحفوظات أيضاً عملية التعرف على الحروف كوسيلة لتحويل كميات هائلة من الوثائق التاريخية المكتوبة بخط اليد إلى أشكال رقمية يمكن البحث فيها ويسهل الوصول إليها.

المجموعة الثانية من تقنيات التعرف على الكتابة اليدوية، التي غالباً ما يشار إليها على أنها تعرف أونلاين شهدت انحساراً وتدفقاً في الشعبية. وذلك في التسعينات، حيث أصدرت Apple Computers جهازًا محمولًا يسمى نيوتن والذي استخدم أول واجهة للتعرف على الكتابة اليدوية متاحة على نطاق واسع. باستخدام قلم صغير، كان المستخدم قادرا على الكتابة مباشرة على شاشة نيوتن و (نظريا) التعرف على رسائلهم وتحويلها إلى نص. في الممارسة العملية ، كان البرنامج الذي استخدمه نيوتن في محاولة تعلم أنماط الكتابة اليدوية للمستخدم أقل من مثالي ، ونتيجة لذلك لم تكن شعبيته كبيرة أبدًا.

في وقت لاحق، حاولت شركة Palm في نظام جديد للتعرف على خط اليد، والتي أطلقوا عليها الكتابة على الجدران. بدلا من الاعتماد على الاستخدام البديهي للحروف الأبجدية الرومانية التقليدية، حدد نظام الكتابة على الجدران نظامه الخاص line-strokes والوقوف في كل حرف. وقد سمح ذلك بارتفاع معدل النجاح في تحديد الحروف وتعلم الاختلافات لدى المستخدم . وقد بدأت البحوث على برامج التعرف على الكتابة اليدوية ، مع تعميم الأجهزة الشخصية والهواتف الخلوية بمدخلات القلم.

يعد التعرف على الكتابة اليدوية مشكلة صعبة بسبب الاختلاف الكبير في أساليب الكتابة الفردية. النهج التقليدي لحل هذا المشكلة سيكون استخراج ميزات اللغة تعتمد على انحناء الحروف المختلفة ، والمسافات بين الحروف .

3-2 التعرف في الذكاء الصنعي :

ومن بين الأنماط السبعة للذكاء الاصطناعي التي تمثل الطرق التي يجري بها تنفيذ الذكاء الاصطناعي، الأكثر شيوعاً هو نمط التعرف. الفكرة الرئيسية لنمط التعرف في الذكاء الاصطناعي هي أننا نستخدم التعلم الآلي والتكنولوجيا المعرفية للمساعدة في تحديد وتصنيف البيانات غير المهيكلة إلى تصنيفات محددة. يمكن أن تكون هذه البيانات الغير المهيكلة صورًا أو فيديو أو نصًا أو حتى بيانات كمية. قوة هذا النمط هي أننا نمكن الآلات من القيام بالشيء الذي يبدو أن أدمغتنا تفعله بسهولة: تحقيق إدراك العالم الحقيقي من حولنا.

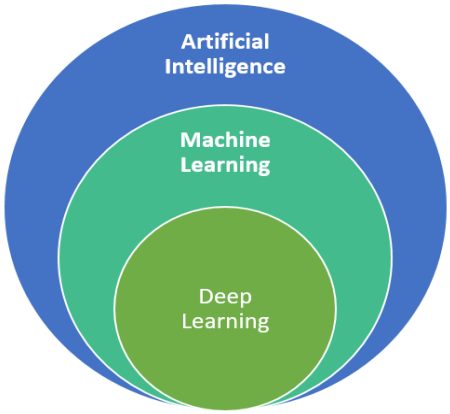
نمط التعرف الملحوظ من حيث أنه كان في المقام الأول هو التعرف على الصور التي جلبت الاهتمام المتزايد في نهج التعلم العميق للذكاء الاصطناعي، وساعدت على بدء موجة من الاستثمار في الذكاء الاصطناعي. ومع ذلك، فإن نمط التعرف على الصور أوسع من مجرد التعرف على الصور في الواقع، يمكننا استخدام التعلم الآلي للتعرف على الصور والصوت والكتابة اليدوية والعناصر والوجه والإيماءات وفهمها. والهدف من هذا النمط هو أن تتعرف الآلات على البيانات غير المُهَمَّة وتفهمها. هذا النمط من الذكاء الاصطناعي هو عنصر ضخم من الحلول الذكاء الاصطناعي بسبب تنوع تطبيقاته.

الفصل الثالث

الشبكات العصبية

3-1 مقدمة عن الشبكات العصبية :

الذكاء الاصطناعي هو علم مثل الرياضيات أو علم الأحياء. يدرس طرق بناء برامج وآلات ذكية يمكنها حل المشكلات بشكل إبداعي ، والتي لطالما اعتُبرت من اختصاص الإنسان. التعلم الآلي هو مجموعة فرعية من الذكاء الاصطناعي (AI) يوفر للأنظمة القدرة على التعلم والتحسين تلقائيًا من التجربة دون أن تتم برمجتها بشكل صريح. في (ML) ، توجد خوارزميات مختلفة (مثل الشبكات العصبية) تساعد في حل المشكلات. التعلم العميق ، أو التعلم العصبي العميق ، هو مجموعة فرعية من التعلم الآلي ، والذي يستخدم الشبكات العصبية لتحليل العوامل المختلفة بهيكل مشابه للنظام العصبي البشري.



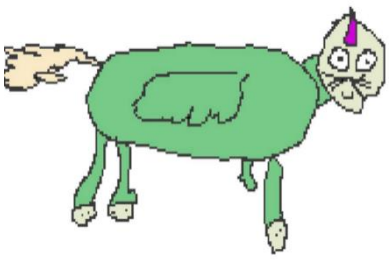
الشكل (3-1) يوضح ما يمثله التعلم العميق في الذكاء الصنعي

3-2 فهم الشبكات العصبية

مفهوم الخلايا العصبية :

أولاً نحتاج إلى فهم ما هي الشبكة العصبية من أجل القيام بذلك ، سنبدأ من مثال لمشكلة واقعية وحلها باستخدام منطق الشبكة العصبية.

المثال :

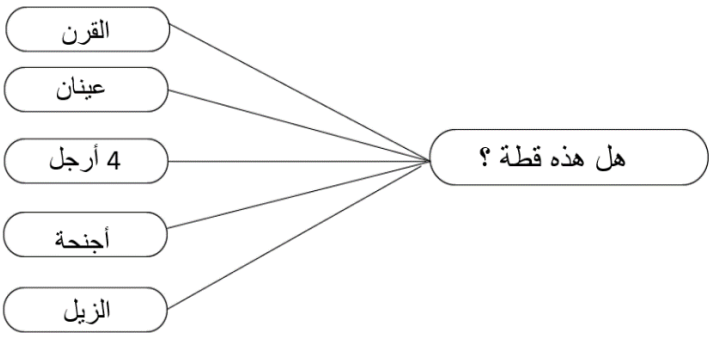


افترض أنك في غرفتك تبحث عن بعض مفاهيم البرمجة وأن ابن أخوك البالغ من العمر 5 سنوات يأتي إليك ويظهر لك لوحته , تنظر إليها وترى شيئًا كهذا:

السؤال الذي يبادر إلى ذهنك على الفور: ما هذا شيء؟

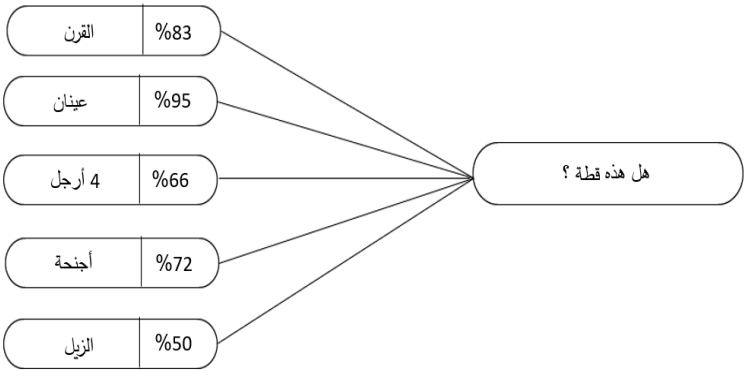
3-2-1 السمات :

لكنك ذكي! لذلك ما يمكنك فعله هو التوصل إلى فرضية "هذه قطة" لكن كيف تعرف؟ يمكنك تدوين كل سمات المخلوقات التي تعرفها. سيكون من المنطقي أن تؤثر كل هذه السمات على الفرضية ، أليس كذلك؟



الشكل (3-2)

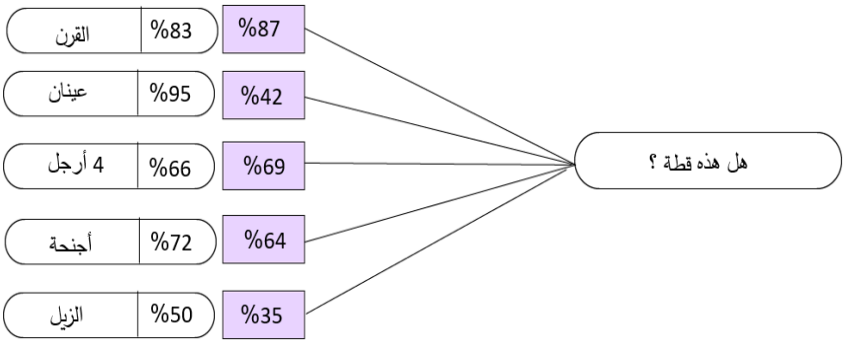
ولكن كيف يمكنك التأكد من أن كل هذه الميزات هي في الواقع ما تعتقده ؟ كيف يمكنك أن تعرف أن ما تراه كجناح هو حقًا جناح؟ حسنًا ، يمكنك دائمًا أن تسأل ، ولكن نظرًا لأن ذلك سيكون سهلاً للغاية ولن يكون متعلقًا بالتعلم الآلي ، ما يمكنك فعله هو تعيين الاحتمال للأشياء التي تراها. يمكنك أن تكون متأكدًا تمامًا من وجود عينين ، ولكن مع وجود 4 أرجل ، ليس كثيرًا ، لذا يجب أن يكون احتمال وجود 4 أرجل أقل. باستخدام الذيل ، لا يمكنك أن تكون متأكدًا تمامًا مما إذا كان المقصود حقًا أن يكون ذيلًا نظرًا لأنه لونه ومظهره غائم ، لنقدم قائمة كاملة من الميزات مع احتمالاتها.



الشكل (3-3)

3-2-2 الأوزان :

ثم مرة أخرى عليك أن تسأل نفسك ، هل كل هذه الميزات مهمة بنفس القدر؟ في الواقع ، لا يزال القط قطة إذا فقد إحدى عينيه. من ناحية أخرى ، فإن المخلوق الذي لديه أجنحة هو الأكثر تميزًا وليس قطة ، باستثناء ربما بعض الطفرات الجينية الناتجة عن العيش بجوار محطة للطاقة النووية التي تلوث المنطقة المحلية لمدة 20 عامًا. لذا فإن هدفنا التالي هو تحديد مدى أهمية كل من هذه الميزات في اتخاذ القرار النهائي وما إذا كانت تزيد أو تنقص الاحتمالية المطروحة.

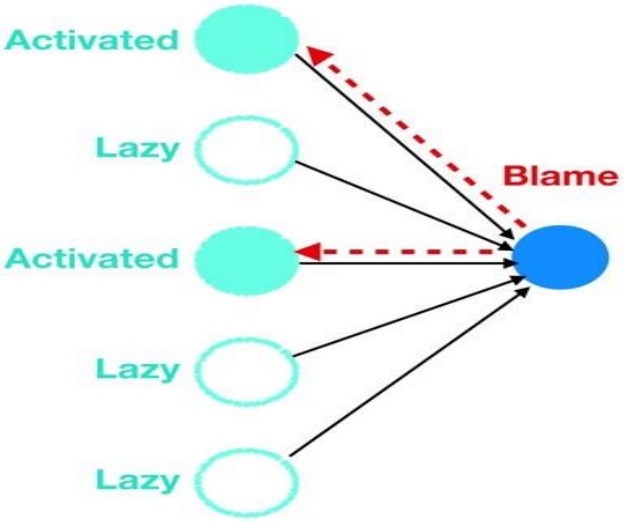


الشكل (3-4)

الآن بعد أن قررنا مدى أهمية الميزات وقمنا بتحجيمها بشكل صحيح ويمكننا ببساطة إضافتها معًا والحصول على بعض النتائج.

$$Y = 0.83 \times (-0.87) + 0.95 \times 0.42 + 0.66 \times 0.69 + 0.72 \times (-0.64) + 0.5 \times 0.35 = -0.1535$$

3-2-3 تابع التفعيل :

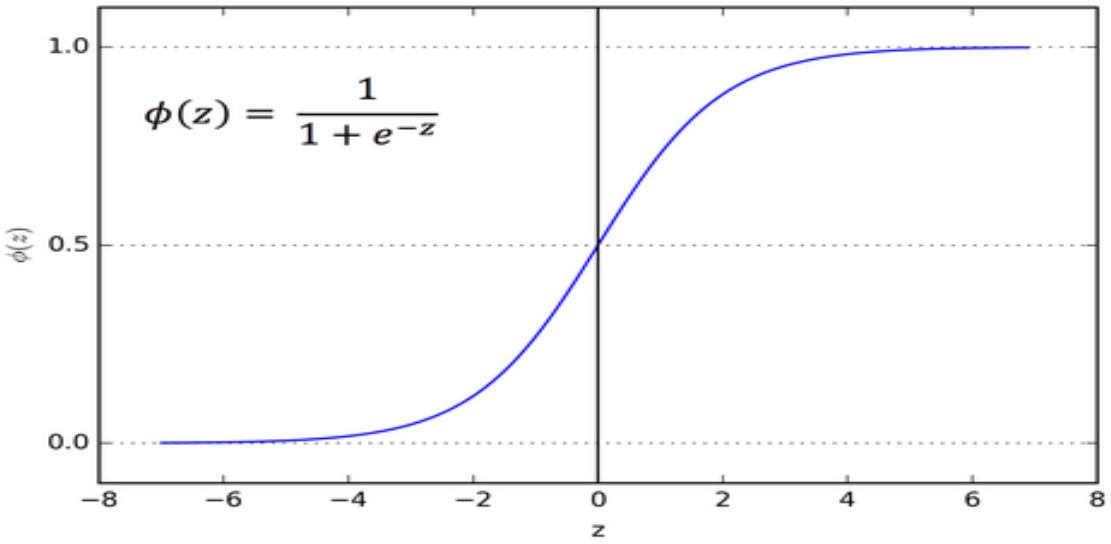


الشكل (3-5)

نلاحظ أننا نكتب 0.83 بدلاً من 83٪ لكن القيمة هي نفسها. نضيف أيضًا السمات الإيجابية ونطرح السمات السلبية مضروبة في وزنها. يمكننا أن نرى أن الناتج سلبي لذا يمكننا أن نستنتج أن هذه ربما ليست قطة ولكن هناك شيء أخير علينا القيام به. قد يكون هناك موقف يكون لدينا فيه الكثير من السمات وتكون درجة مخرجات كبيرة جدًا. نود الحصول على بيانات بتنسيق يقتصر على بعض

الفواصل الزمنية ، على سبيل المثال <0،1>. يمكننا التفكير في الأمر على أنه احتمال بين 0-100.

ما تابع التفعيل الذي يجب أن نستخدمه لجعل الخرج ملائماً ؟ حسنًا ، هناك الكثير للاختيار وليكن تابع هو Sigmoid.

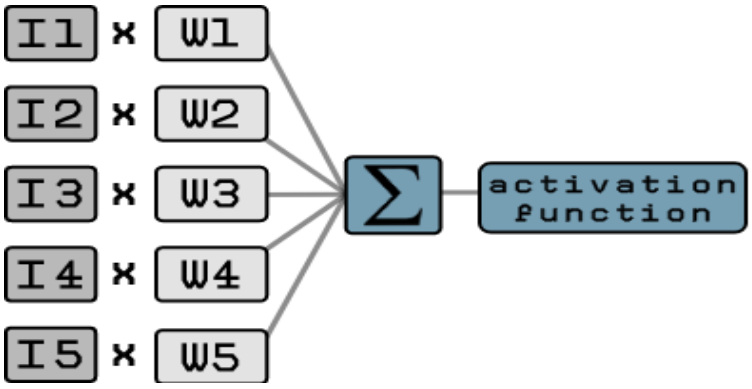


المخطط (3-1): مخطط يوضح شكل تابع الsigmoid بيانيا

حيث يقوم بإرجاع الناتج في النطاق <0،1> ، لذلك يمكننا التفكير في الناتج على أنه احتمال بين 0-100٪. الآن يمكننا كتابة صيغتنا بهذا التنسيق:

$$Y = \text{sigmoid}\ (0.83 \times (-0.87) + 0.95 \times 0.42 + 0.66 \times 0.69 + 0.72 \times (-0.64) + 0.5 \times 0.35)$$

3-2-4 العصبون :



الشكل (3-6)

إذن ماذا تفعل الخلايا العصبية؟

تأخذ المدخلات وتضربها بأوزانها ،ثم تلخصهم ،بعد ذلك يتم تطبيق وظيفة التنشيط على المجموع.

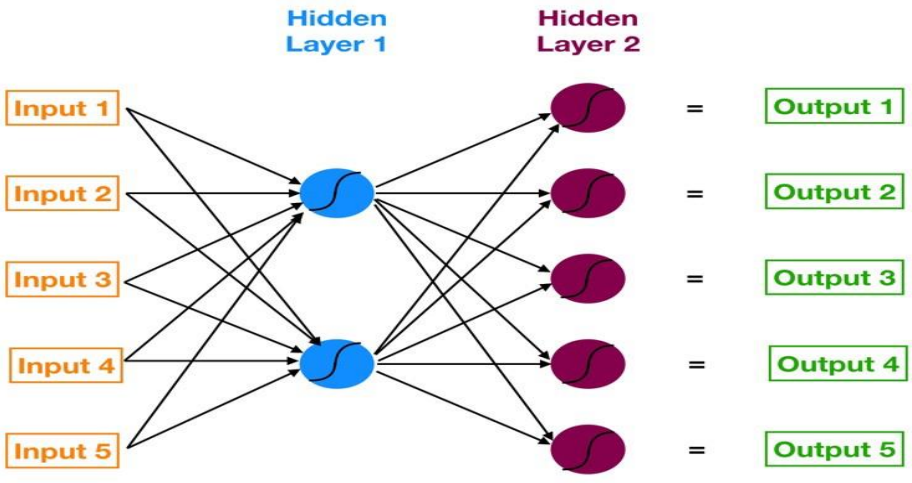
3-2-5 كيف تتعلم الخلايا العصبية؟

الهدف من الخلايا العصبية هو ضبط الأوزان بناءً على الكثير من أمثلة المدخلات والمخرجات. لنفترض أننا نعرض على الخلية العصبية آلاف الأمثلة لرسومات القطط ورسومات ليست قطط ولكل من هذه الأمثلة نعرض الميزات الموجودة ومدى قوة تأكدنا من وجودها هنا. بناءً على هذه الآلاف من الصور ، يقرر العصبون:

- ما هي السمات المهمة والإيجابية (على سبيل المثال ، كان لكل رسم قطة ذيل ، لذا يجب أن يكون الوزن موجبًا وكبيرًا) ،
- السمات غير المهمة (على سبيل المثال ، فقط عدد قليل من الرسومات لها عينان ، لذلك يجب أن يكون الوزن صغيرًا) ،
- ما هي السمات المهمة والسلبية (على سبيل المثال ، كل رسم يحتوي على قرن كان في الواقع رسمًا وحيد القرن وليس قطة ، لذا يجب أن يكون الوزن كبيرًا وسلبيًا).

3-2-6 الشبكات العصبية :

يمكن أن تكون الخلايا العصبية مدخلاً لبعض الخلايا العصبية الأخرى. و يمكن أيضًا أن تكون مدخلات الخلايا العصبية ناتجًا عن بعض الخلايا العصبية الأخرى.



الشكل (3-7): يبين بينة الشبكة العصبية

على سبيل المثال ، نريد أن نتنبأ بما إذا كانت هناك ميزات مثل القرون والأرجل وما إلى ذلك بناءً على وحدات البكسل في الصورة ، ثم بناءً على حقيقة أنها رسم لقط ، نريد توقع أنواع الحيوانات التي يحبها الطفل. لذلك يعد تابع التفعيل مفيد في ملائمة مخرجات جميع الخلايا العصبية ، بحيث يمكن أن تكون بمثابة مدخلات في الطبقة التالية من الشبكة.

لاحظ أنه عادةً ما يكون ناتج كل خلية عصبية غير متصل بخلايا عصبية واحدة من الطبقة التالية ولكن بالعديد من الخلايا العصبية إن لم يكن كلها. هذا بسبب حقيقة أن المعلومات التي تنبأت بها الخلايا العصبية يمكن أن تكون مفيدة للعديد من الخلايا العصبية الأخرى

3-2-7 الصندوق الأسود (الطبقات المخفية) :

في هذا المثال ، أوضحنا أن العصبون يقرر ما إذا كانت الصورة تحتوي على قطة بناءً على ميزات معينة. في الحياة الواقعية ، لا نعرف عادةً الميزات التي تم استخدامها للتنبؤ بالمخرجات النهائية. في مثالنا ، ذكرنا أن هذا ليس قطة بناءً على ميزات مثل القرن. ولكن في الحياة الواقعية ، يمكن للشبكة العصبية اختيار ميزات مثل البكسل في موضع معين ومعالجتها من خلال العديد والعديد من التصنيفات التي لا نعرف عنها شيئًا وتستند إلى توقع النتيجة النهائية. وهذا ما يسمى نهج الصندوق الأسود لأننا نرى فقط المدخلات والمخرجات وليس كل الحسابات التي تتم في المنتصف. بسبب ذلك تسمى الطبقات الموجودة بين طبقات الإدخال والإخراج الطبقات المخفية.

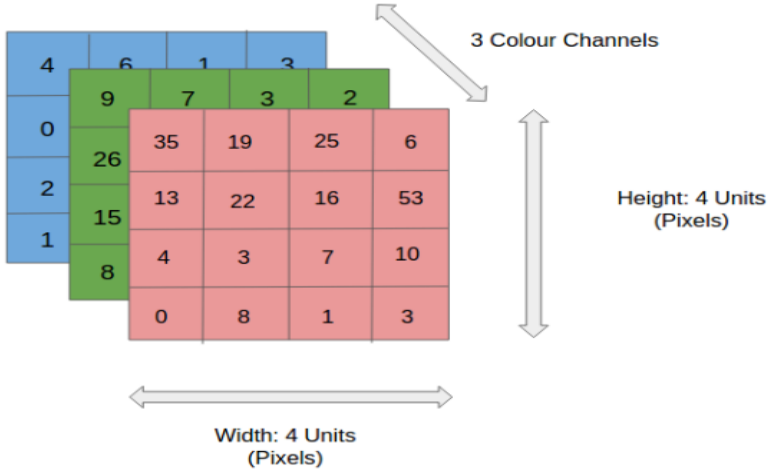
إذن ما هي الشبكة العصبية؟

" الشبكة العصبية هي مجموعة من الخلايا العصبية منظمة في طبقات. كل خلية عصبية هي عملية رياضية تأخذ مدخلاتها وتضربها بأوزانها ثم تمرر المجموع من خلال وظيفة التنشيط إلى الخلايا العصبية الأخرى. تتعلم الشبكة العصبية كيفية تصنيف المدخلات من خلال تعديل أوزانها بناءً على أمثلة سابقة ".

3-3 الشبكات العصبية الملتفة CNN:

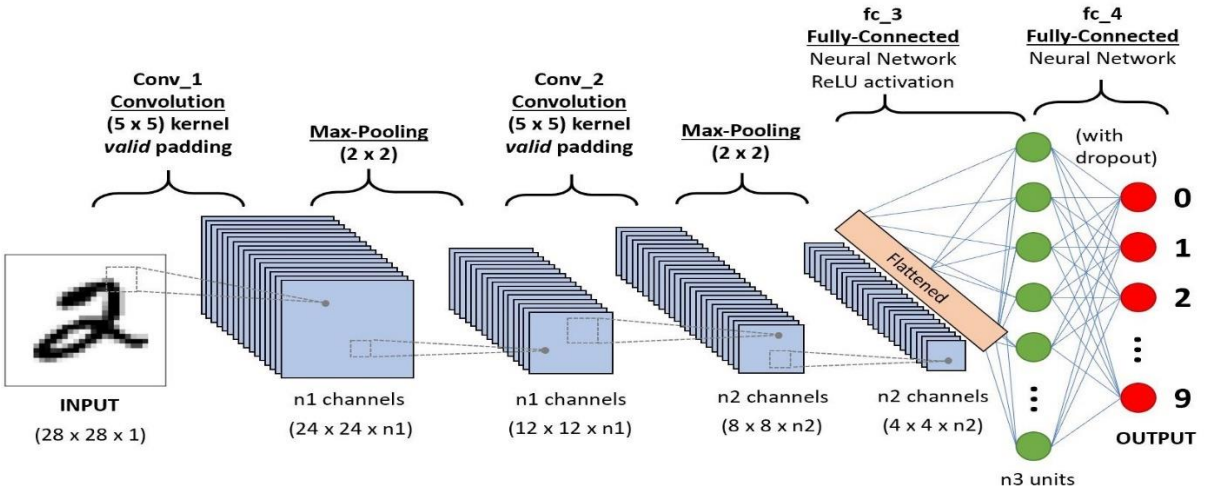
الشبكة العصبية التلافيفية (ConvNet / CNN) هي خوارزمية التعلم العميق التي يمكن أن تأخذ صورة إدخال ، وتعيين الأهمية (الأوزان القابلة للتعلم والتحيزات) للجوانب / الكائنات المختلفة في الصورة وتكون قادرة على تمييز أحدهما عن الآخر. المعالجة المسبقة المطلوبة في ConvNet أقل بكثير مقارنة بخوارزميات التصنيف الأخرى. بينما في الأساليب البدائية ، يتم تصميم المرشحات يدويًا ، مع التدريب الكافي ، تمتلك ConvNets القدرة على تعلم هذه المرشحات / الخصائص. حيث تأخذ تصنيفات صور الإدخال وتعالجها وتصنفها ضمن فئات معينة على سبيل المثال (الكلب ، القط ، النمر ، الأسد). ترى

أجهزة الكمبيوتر صورة الإدخال كمصفوفة من البكسل وتعتمد على دقة الصورة. بناءً على دقة الصورة، سترى الارتفاع × العرض × العمق. على سبيل المثال، مصفوفة صورة 4 × 4 × 3 (3 تشير إلى قيم RGB) و مصفوفة صورة 4 × 4 × 1(صورة ذات تدرج رمادي).



الشكل (3–8): دخل الشبكة العصبية CNN ثنائية الأبعاد

نماذج CNN للتعلم العميق للتدريب والاختبار، ستمر كل صورة إدخال عبر سلسلة من طبقات الالتفاف مع المرشحات (Kernals)، والتجميع، والطبقات المتصلة بالكامل (FC) وتطبيق وظيفة Softmax لتصنيف كائن بقيم احتمالية بين 0 و 1. الشكل أدناه هو تدفق كامل لـ CNN لمعالجة صورة إدخال وتصنيف الكائنات بناءً على القيم.

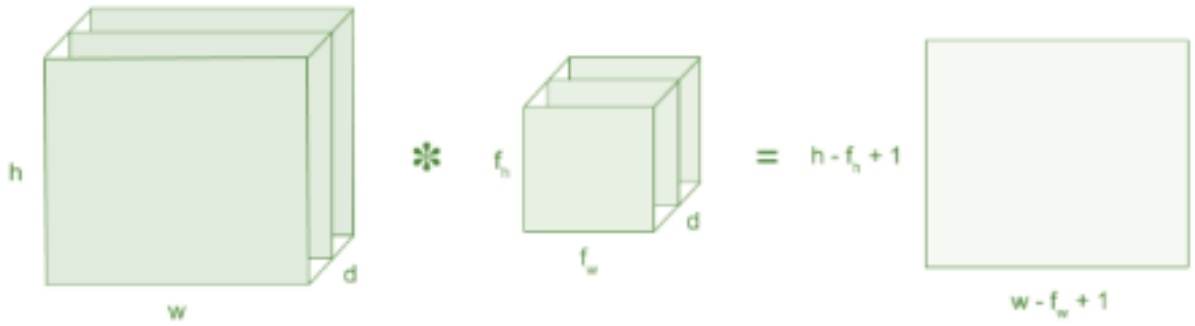


الشكل (3–9): بنية شبكة CNN

3-3-1 الطبقة الالتفافية (Convolution layer):

الالتفاف هو الطبقة الأولى لاستخراج المعالم من صورة المدخلة. يحافظ الالتفاف على العلاقة بين وحدات البكسل عن طريق تعلم ميزات الصورة باستخدام مربعات صغيرة من بيانات الإدخال. إنها عملية حسابية تأخذ مدخلين مثل مصفوفة الصورة ومرشح أو نواة.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

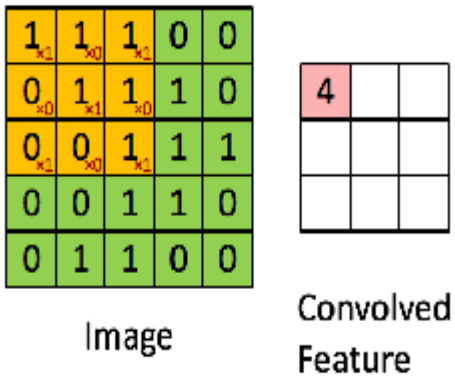


الشكل (3-10)

باعتبار 5 × 5 التي تكون قيم بكسل الصورة فيها 0 ، 1 ومصفوفة المرشح 3 × 3 كما هو موضح أدناه .

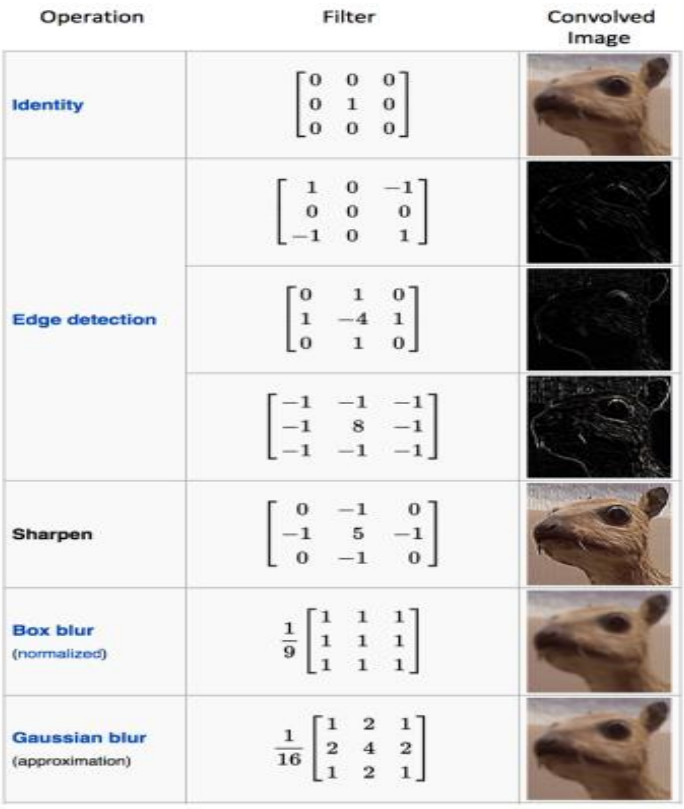
الشكل (3-11)

ثم تُضرب مصفوفة الالتفاف للصورة 5 × 5 مع مصفوفة مرشح 3 × 3 والتي تسمى" **Feature Map** "
كما هو موضح أدناه .



الشكل (3-12)

يمكن أن يؤدي التفاف الصورة باستخدام مرشحات مختلفة إلى عمليات مثل اكتشاف الحواف والتشويش والحدة من خلال تطبيق المرشحات. ويوضح المثال أدناه صورة التفاف مختلفة بعد تطبيق أنواع مختلفة من المرشحات (Kernels).



الشكل (3-13)

3-3-2 كيف يتم اختيار حجم النواة (Kernel):

أثناء عملية تعلم CNN هذه ، تجد أحجامًا مختلفة للنواة في أماكن مختلفة في الكود ، ثم يطرح هذا السؤال في ذهن المرء ما إذا كانت هناك طريقة محددة لاختيار مثل هذه الأبعاد أو الأحجام. إذن ، الجواب لا. في عالم التعلم العميق الحالي ، نستخدم الخيار الأكثر شيوعًا الذي يستخدمه كل ممارس للتعلم العميق ، وهو حجم نواة 3 × 3. الآن ، هناك سؤال آخر يخطر ببالك ، لماذا 3 × 3 فقط ، وليس 1 × 1 ، 2 × 2 ، 4 × 4. في الأساس ، نقسم أحجام النواة إلى أحجام أصغر وأكبر. تتكون أحجام النواة الأصغر من 1 × 1 و 2 × 2 و 3 × 3 و 4 × 4 ، بينما تتكون الأحجام الأكبر من 5 × 5 وما إلى ذلك ، لكننا نستخدم حتى 5 × 5 للالتفاف ثنائي الأبعاد. في عام 2012 ، عندما تم تقديم بنية AlexNet CNN ، استخدمت 11 × 11 ، 5 × 5 مثل أحجام النواة الأكبر التي استغرقت أسبوعين إلى ثلاثة أسابيع في التدريب. لذلك نظرًا لاستهلاك وقت أطول للغاية في التدريب والتكلفة ، لم نعد نستخدم مثل هذه الأحجام الكبيرة من النواة.

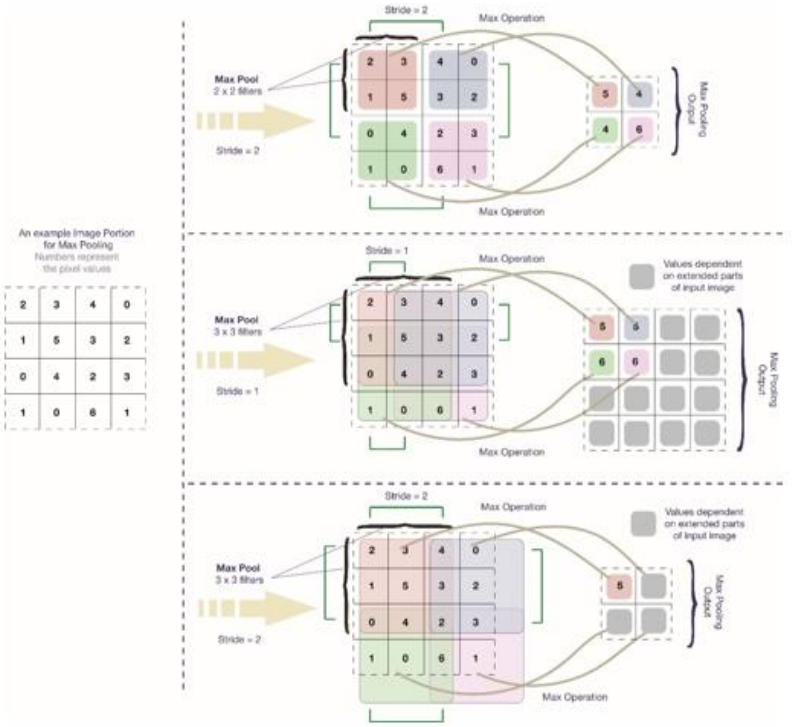
أحد أسباب تفضيل أحجام النواة الصغيرة على الشبكة المتصلة بالكامل هو أنها تقلل من التكاليف الحسابية وتقاسم الوزن الذي يؤدي في النهاية إلى أوزان أقل للانتشار العكسي. ثم جاءت الشبكات العصبية الملتفة VGG في عام 2015 والتي استبدلت طبقات الالتفاف الكبيرة هذه بطبقات التفاف 3 x 3 ولكن مع الكثير من المرشحات. ومنذ ذلك الحين ، أصبحت النواة بحجم 3 × 3 خيارًا شائعًا. ولكن مع ذلك ، لماذا لا تكون 1 x 1 أو 2 x 2 أو 4 x 4 كنواة أصغر حجمًا؟

يستخدم حجم النواة 1 x 1 فقط لتقليل الأبعاد الذي يهدف إلى تقليل عدد القنوات. إنه يلتقط تفاعل قنوات الإدخال في بكسل واحد فقط من خريطة المعالم. لذلك ، تم التخلص من 1 x 1 لأن الميزات المستخرجة ستكون محببة بدقة ومحلية أيضًا بدون أي معلومات من وحدات البكسل المجاورة.

لا يُفضل عموماً 2 x 2 و 4 x 4 لأن المرشحات ذات الحجم الفردي تقسم بشكل متماثل بكسلات الطبقة السابقة حول بكسل الإخراج. وإذا لم يكن هذا التناظر موجودًا ، فستحدث تشوهات عبر الطبقات والتي تحدث عند استخدام حبات ذات حجم متساوٍ ، أي 2 x 2 و 4 x 4. لذلك ، لهذا السبب لا نستخدم أحجام النواة 2 x 2 و 4 x 4.لذلك ، 3 x 3 هو الأمثل.

3-3-3 خطوات (Strides):

Stride هو عدد بكسلات الإزاحة على مصفوفة الإدخال. عندما تكون الخطوة 1، فإننا ننقل المرشحات بكسل واحد في المرة الواحدة. عندما تكون الخطوة 2، فإننا ننقل المرشحات 2 بكسل في كل مرة وهكذا. يوضح الشكل أدناه أن الالتواء سيعمل بخطوة 2 .



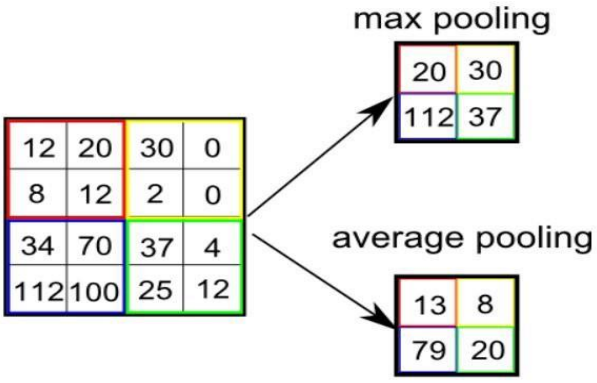
الشكل (3-14)

3-3-4 الحشوة (Padding):

أحيانًا لا يتلاءم المرشح تمامًا مع صورة الإدخال. لدينا خياران : قم بتثبيت الصورة بالأصفار (المساحة المتروكة الصفرية) بحيث تناسبها او قم بإسقاط(إهمال) جزء الصورة حيث لم يكن الفلتر مناسبًا. وهذا ما يسمى الحشو الصالح الذي يحتفظ فقط بجزء صالح من الصورة.

3-3-5 طبقة التجميع (Pooling):

على غرار الطبقة التلافيفية، تكون طبقة التجميع مسؤولة عن تقليل الحجم المكاني للميزة الملتفة. هذا لتقليل القوة الحسابية المطلوبة لمعالجة البيانات من خلال تقليل الأبعاد. علاوة على ذلك، فهو مفيد لاستخراج السمات السائدة التي تكون ثابتة في التناوب والموضع ، وبالتالي الحفاظ على عملية التدريب الفعال للنموذج.



الشكل (3-15)

هناك عدة انواع من التجميع: Max Pooling و Average Pooling و Sum pooling. يُرجع Max Pooling القيمة العظمى من جزء الصورة الذي يغطيه Kernel. من ناحية أخرى، يُرجع متوسط التجميع متوسط جميع القيم من جزء الصورة الذي تغطيه Kernel.

يعمل Max Pooling أيضًا كعامل مانع للضوضاء. إنه يتجاهل عمليات التنشيط الصاخبة تمامًا ويقوم أيضًا بإزالة الضوضاء جنبًا إلى جنب مع تقليل الأبعاد. من ناحية أخرى، يقوم متوسط التجميع ببساطة بتقليل الأبعاد كآلية لقمع الضوضاء. ومن ثم، يمكننا القول أن أداء Max Pooling أفضل بكثير من متوسط التجميع.

تشكل الطبقة التلافيفية وطبقة التجميع معًا الطبقة الأولى من الشبكة العصبية التلافيفية. اعتمادًا على التعقيدات في الصور، يمكن زيادة عدد هذه الطبقات لالتقاط تفاصيل ذات مستويات منخفضة بشكل أكبر، ولكن على حساب المزيد من القوة الحسابية. بعد إجراء العملية المذكورة أعلاه، نجحنا في تمكين النموذج لفهم الميزات. للمضي قدمًا، سنقوم بتسوية الناتج النهائي تغذيته إلى شبكة عصبية منتظمة لأغراض التصنيف

3-3-6 **كيف يتم أختيار نوع Pooling ؟**

يتم إجراء التجميع في الشبكات العصبية لتقليل التباين وتعقيد الحساب فيما يلي مقارنة بين ثلاث طرق تجميع أساسية مستخدمة على نطاق واسع.الأنواع الثلاثة لعمليات التجميع هي:

Max pooling: يتم تحديد الحد الأقصى لقيمة البكسل للدُفعة

Min pooling: يتم تحديد الحد الأدنى لقيمة البكسل للدُفعة.

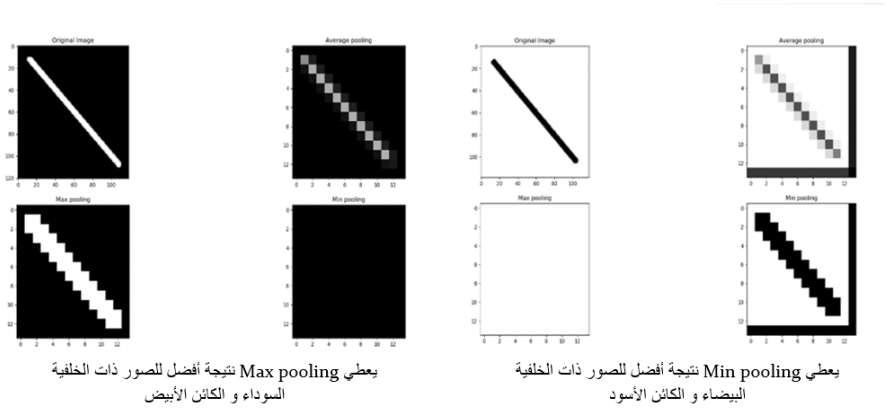
Average pooling: يتم تحديد متوسط قيمة جميع وحدات البكسل في الدُفعة.

الدُفعة هنا تعني مجموعة من وحدات البكسل ذات الحجم المساوي لحجم المرشح الذي يتم تحديده بناءً على حجم الصورة. في المثال التالي ، تم اختيار مرشح 9x9. يختلف ناتج طريقة التجميع باختلاف القيمة المتغيرة لحجم المرشح.

الشكل (3–16): يبين الفرق بين انواع الpooling

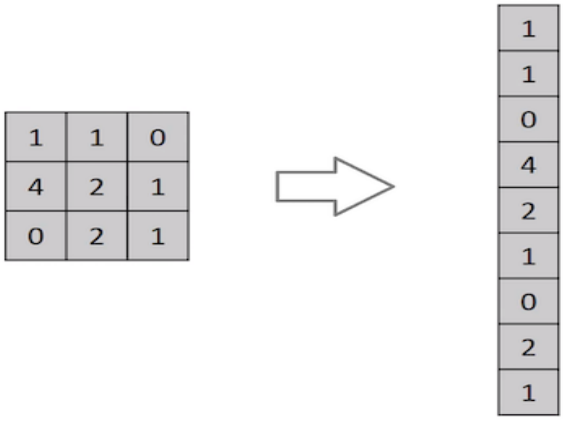
يتم توضيح العمليات من خلال الأرقام التالية لا يمكننا القول أن طريقة تجميع معينة أفضل من غيرها بشكل عام. يتم اختيار عملية التجميع بناءً على البيانات الموجودة. تعمل طريقة التجميع المتوسطة على تنعيم الصورة وبالتالي قد لا يتم تحديد الميزات الحادة عند استخدام طريقة التجميع هذه. يحدد Max pooling وحدات البكسل الأكثر سطوعًا من الصورة. يكون مفيدًا عندما تكون خلفية الصورة مظلمة ونحن مهتمون فقط بوحدات البكسل الأفتح من الصورة. على سبيل المثال: في مجموعة بيانات MNIST ، يتم تمثيل الأرقام باللون الأبيض والخلفية سوداء. لذلك ، يتم استخدام أقصى تجمع. وبالمثل ، يتم استخدام min pooling في الاتجاه المعاكس.

توضح الأشكال التالية تأثيرات التجميع على صورتين بمحتوى مختلف

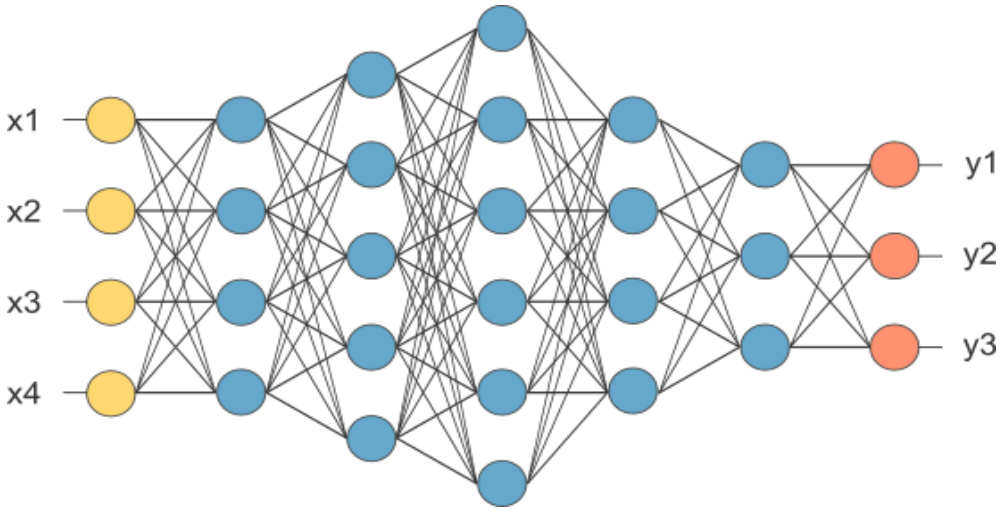


الشكل (3-17): يوضح الفرق بين الmin وال max

يتم تحويل مصفوفة خريطة السمات إلى متجه (x1، x2، x3 ، ...). ويتم تغذيته المسطح إلى شبكة عصبية ذات تغذية أمامية (backpropagation) على كل تكرار للتدريب. على مدى سلسلة من دورات Epochs ، يكون النموذج قادرًا على التمييز بين السمات المسيطرة وبعض الميزات منخفضة المستوى في الصور وتصنيفها باستخدام تقنية Softmax Classificationاو Binary Classification ..



الشكل 3-18



الشكل (3-19) يوضح ارتباط عصبونات الطبقة الأخيرة من الشبكة

يتم تحويل مصفوفة خريطة السمات إلى متجه (x1، x2، x3 ، ...). مع الطبقات المتصلة بالكامل، ثم يتم دمج هذه الميزات معًا لإنشاء نموذج. أخيرًا، لدينا توابع تفعيل مثل softmax أو sigmoid لتصنيف النواتج على أنها قط ، كلب ، سيارة ، شاحنة وما إلى ذلك.

3-3-7 طبقة Dropout :

يطبق Dropout على الإدخال. حيث تقوم بتعيين وحدات الإدخال بشكل عشوائي على 0 مع معدل تكرار في كل خطوة أثناء وقت التدريب ، مما يساعد على منع فرط التجهيز. المدخلات التي لم يتم ضبطها على 0 يتم تحجيمها بمقدار 1 / (1 − rate) بحيث لا يتغير المجموع على جميع المدخلات.

3-3-8 Flatten layer (طبقة التسطيح):

التسطيح هو تحويل البيانات إلى مصفوفة ذات بعد واحد لإدخالها إلى الطبقة التالية. نقوم بتسطيح إخراج الطبقات التلافيفية لإنشاء متجه ميزة طويلة واحدة. وهي متصلة بنموذج التصنيف النهائي ، والذي يسمى طبقة متصلة بالكامل(Full connected). بمعنى آخر ، نضع جميع بيانات البكسل في سطر واحد ونجري اتصالات بالطبقة النهائية.

الفصل الرابع

بيئة العمل

4-3 **Anaconda**: هو توزيع مجاني ومفتوح المصدر للغات البرمجة مثلPython و R للحوسبة العلمية (علوم البيانات وتطبيقات تعلم الآلة ومعالجة البيانات واسعة النطاق والتحليلات التنبؤية وما إلى ذلك) ، والتي تهدف إلى تبسيط إدارة الحزم و تعيينها. يتضمن التوزيع حزم علوم البيانات المناسبة لأنظمة Windows و Linux وmacOS تم تطويره وصيانته بواسطة Anaconda, Inc التي أسسها Peter Wang و Travis Oliphant في عام 2012. كمنتج Anaconda, Inc ، يُعرف أيضًا باسم Anaconda Distribution . ويتم إدارة إصدارات الحزمة في Anaconda بواسطة نظام conda لإدارة الحزم. تم إنشاء مدير الحزم هذا كحزمة منفصلة مفتوحة المصدر حيث انتهى بها الأمر لتكون مفيدة بمفردها ولأشياء أخرى غير بايثون. هناك أيضًا إصدار صغير من Anaconda يسمى Miniconda ، والذي يتضمن فقط conda و Python والحزم التي يعتمدون عليها وعددًا صغيرًا من الحزم الأخرى.

5-3 **Keras** : هي API للتعلم العميق مكتوبة بلغة Python، وتعمل على منصة التعلم الآلة TensorFlow. تم تطويره مع التركيز على تمكين التجريب السريع. والقدرة على الانتقال من الفكرة إلى النتيجة بأسرع ما يمكن. تم تطوير Keras في البداية كجزء من الجهود البحثية لمشروع ONEIROS (نظام تشغيل روبوت ذكي إلكتروني عصبي مفتوح).

6-3 **TensorFlow** : هي مكتبة مفتوحة المصدر للحوسبة الرقمية السريعة . وتم إنشاؤه وصيانته بواسطة Googleوتم إصداره بموجب ترخيص Apache 2.0 مفتوح المصدر على عكس المكتبات العددية الأخرى المخصصة للاستخدام في التعلم العميق مثلTheano، تم تصميم TensorFlow للاستخدام في كل من البحث والتطوير وأنظمة الإنتاج.

7-3 **Numpy** : هي مكتبة بايثون تستخدم للتعامل مع المصفوفات وهي تملك توابع العمل في نطاق الجبر الخطي وتحويلات فوريه والمصفوفات وهي مفتوحة المصدر.

8-3 **Tkinter**: هي مكتبة GUI القياسية لبايثون. عندما يقترن Tkinter مع البايثون يوفر طريقة سريعة وسهلة لإنشاء تطبيقات واجهة المستخدم الرسومية. يوفر Tkinter واجهة كائن موجهة قوية لمجموعة أدوات واجهة المستخدم الرسومية TK.

9-3 **win32gui:** وحدة نمطية التي توفر واجهة لواجهة المستخدم الرسومية WIN32 الأصلي.

10-3 **Jupyter** : تطبيق ويب مفتوح المصدر يسمح لنا بإنشاء ومشاركة المستندات التي تحتوي على تعليمات برمجية مباشرة ومعادلات وتصورات ونص سردي. تشمل الاستخدامات: تنظيف البيانات وتحويلها ، والمحاكاة العددية ، والنمذجة الإحصائية ، وتصور البيانات ، والتعلم الآلي.

الفصل الخامس

القسم العملي والنتائج

1-5 القسم العملي:

- حجم البيانات المستخدمة 16800 مقسمة إلى قسمين : القسم الأول للتدريب 13440 و القسم الثاني للاختبار 3360 . و لدينا لكل حرف 440 صورة مختلفة في نماذج التدريب و 120 صورة في نماذج الاختبار. حيث أبعاد كل صورة 32 * 32 مخزنة على القرص المحلي. و يتم جلب البيانات إلى المشروع عن طريق التابع () ImageDataGenerator الذي يقوم بإنشاء دفعات من بيانات صورة مع زيادة البيانات في الوقت الحقيقي. تحتوي على ثلاث طرق flow() و flow_from_directory() و flow_from_dataframe () لقراءة الصور من مصفوفة كبيرة ومجلدات تحتوي على صور. يجب تعيين الدليل على المسار حيث توجد "n" فئة من المجلدات .

```
datagen.flow_from_directory(
'Pictures\letters\Train',
target_size=(32, 32),
batch_size =batch_size,
class_mode='categorical')
```

- **target_size :** هو حجم الصور المدخلة ، سيتم تغيير حجم كل صورة إلى هذا الحجم.
- **batch_size** : عدد الصور التي سيتم الحصول عليها من المولد لكل دفعة.
- **class_mode :** عيّن "Binary" إذا كان لديك فئتان فقط للتنبؤ بهما ، وإذا كان أكثر من فئتان يتم ضبطه على "categorical" ، أما في حال إذا كنا نقوم بتطوير نظام Autoencoder ، فمن المحتمل أن يكون كل من الإدخال والإخراج هو نفس الصورة ، في هذه الحالة ، تم تعيينه على الدخل.

```
model = Sequential()

model.add(Conv2D(128, kernel_size=3,activation='relu',input_shape=input_shape))
model.add(Conv2D(64,kernel_size=3, activation='relu'))
model.add(Conv2D(32,kernel_size=3, activation='relu'))
model.add(Conv2D(16,kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))
```

- نوع النموذج الذي سنستخدمه متسلسل . التسلسلي هو أسهل طريقة لبناء نموذج في Keras. يسمح لك ببناء نموذج طبقة بطبقة. نستخدم وظيفة "add()" لإضافة طبقات إلى نموذجنا.
- أول ثلاثة طبقات لدينا هي طبقات Conv2D .هذه هي طبقات الالتفاف التي سوف تتعامل مع صور الإدخال لدينا ، والتي يُنظر إليها على أنها مصفوفات ثنائية الأبعاد. 64 في الطبقة الأولى و 32 في الطبقة الثانية و 16 في الطبقة الثالثة هي عدد العقد في كل طبقة.يمكن تعديل هذا الرقم ليكون أعلى أو أقل ، حسب على حجم مجموعة البيانات.حجم النواة هو حجم مصفوفة المرشح للالتفاف. لذلك حجم نواة 3 يعني أنه سيكون لدينا مصفوفة مرشح 3x3. التفعيل هو تابع التفعيل للطبقة. تابع التفعيل أستخدمنا لأول ثلاث طبقات ReLU . تأخذ الطبقة الأولى أيضًا شكل إدخال. هذا هو شكل كل صورة مدخلة (32،32،3) مع 3 يدل على الأبعاد اللونية RGB. بين طبقات Conv2D والطبقة Dense ، توجد طبقة "Flatten". يعمل Flatten كحلقة وصل بين الالتواء والطبقات Dense. "Dense" هو نوع الطبقة الذي تستخدم كطبقة المخرجات, وهي الطبقة (Full connected) المستخدمة في كثير من الشبكات العصبية. سيكون لدينا 28 عقدة في طبقة الإخراج ، واحدة لكل نتيجة محتملة (أ – ي). تابع التفعيل "softmax" يجعل الناتج يصل إلى 1 وبالتالي فإن خرج النموذج يمكن أن يفسر على أنه احتمالات. حيث يتم التنبؤ بناءً على الخيار الذي له أعلى احتمالية.

```
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta() ,metrics=['accuracy'])
```

- نحتاج إلى تجميع نموذجنا. يستغرق تجميع النموذج ثلاثة المعلمات: Optmizer ، Loss و metrics. Optmizer يتحكم في معدل التعلم. سنستخدم "Adadelta" يأخذ معدل تعلم كبرامتر, قد يؤدي معدل التعلم الأصغر إلى أوزان أكثر دقة, ولكن الوقت المستغرق لحساب الأوزان سيكون

طويـل. سـوف نسـتخدم "categorical_crossentropy" مـن اجـل Function Loss. وهـذا هـو الاختيار الأكثر شيوعة للتصنيف. و لتسهيل تفسير الأمور ، سنستخدم مقياس "accuracy".

```
hist = model.fit( train,  verbose=1,  epochs=epochs,
validation_data= test)
```

تـدريب ، سـوف نسـتخدم تـابع "fit()" فـي نمـوذج بالمعـايير التاليـة: بيانـات التـدريب (train) ، بيانات التحقق (Test) وعدد الدورات , ال verbose فقط لإظهار سير التدريب.

```
print("The model has successfully trained")

model.save('Pictures\letters\latter.h5')
print("Saving the model as mnist.h5")
score = model.evaluate(test, verbose=0)
model.summary()
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

- مجموعـة التعليمـات هـذه لحفـظ النمـوذج المـدرب (model.save) وتقييمـه (model.evaluate) واظهار ملخص عن الشبكة (model.summary) .

التابع predict() يمرر صور جديدة للشبكة المدربة لإختبار الشبكة فيما اذا تدربت أم لا.

```
def predict(img):
   img = img.resize((64,48))
   #convert rgb to grayscal
   img = img.convert('L')
   img = np.array(img)
   #reshaping to support our model input and normalizing
   img = img.reshape(1,32 , 32 ,3)
   res = model.predict(img)[0]
   return np.argmax(res)
```

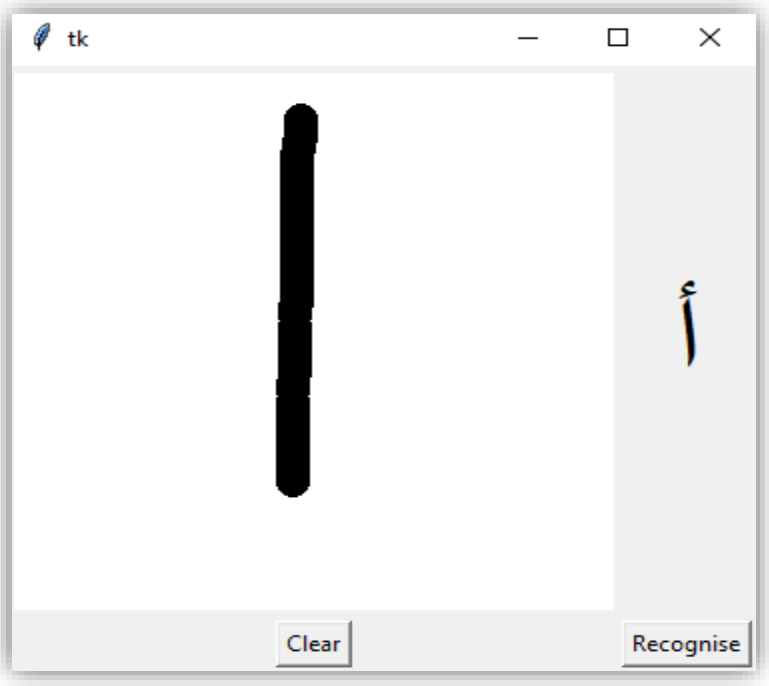
بإستخدام مكتبة Tkinter في مكتبة Python القياسية قمنا بإنشاء فئة التطبيقات المسؤولة عن إنشاء واجهة المستخدم الرسومية لتطبيقنا. حيث أنشأنا Canvas يمكننا الرسم عليه عن طريق التقاط حدث الماوس وباستخدام زر ، نقوم بتشغيل تابع ال predict() وعرض النتائج.

```
class App(tk.Tk):

    def __init__(self):
        tk.Tk.__init__(self)
        self.x = self.y = 0
        # Creating elements
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white",
        cursor="cross")
        self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise", command =
        self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)
        # Grid structure
        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)
        #self.canvas.bind("<Motion>", self.start_pos)
        self.canvas.bind("<B1-Motion>", self.draw_lines)
    def clear_all(self):
        self.canvas.delete("all")
    def classify_handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas
        rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas
        im = ImageGrab.grab(rect)
        digit = predict_digit(im)
        self.label.configure(text= str(alpha[digit]))
    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')
app = App()
mainloop()
```

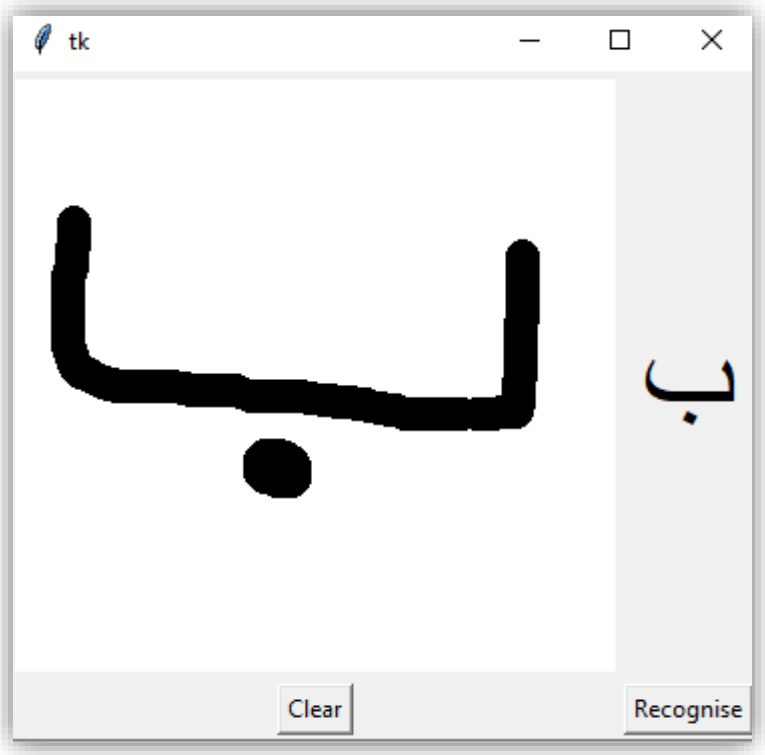
بعد أن تدربت الشبكة الخرج يكون على النحو التالي عند استخدام ال canvas

مثلا لندخل الحرف الألف تكون النتيجة كتالي:



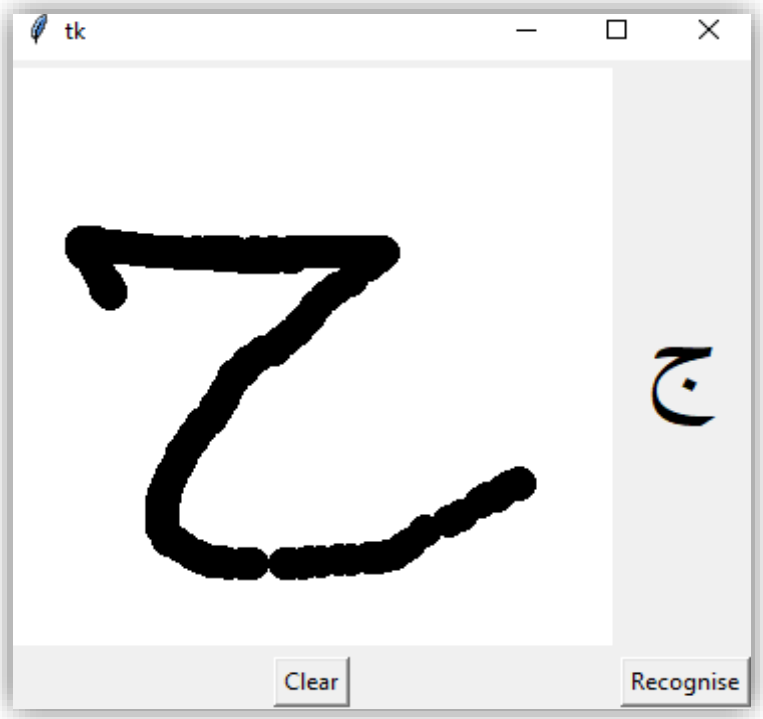
ادخال حرف اخر مثالا الغين :

ادخال الحرف باء :



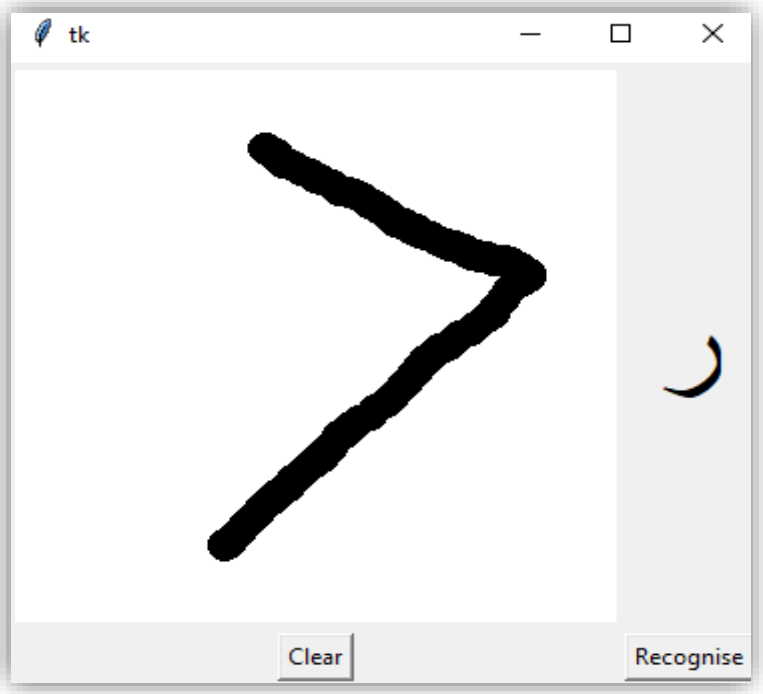
ادخال حرف الميم:

ادخال حرف الحاء:



الخرج قريب للحرف المدخل.

ندخل حرف الدال :



2-5 النتائج التي تم التوصل اليها:

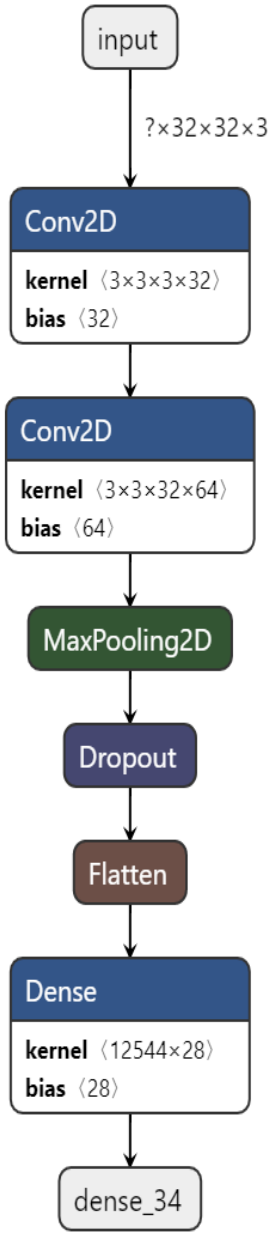
المرحلة الأولى: في بداية التدريب تم تعيير الشبكة وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	**Learning 6** Rate
20	**2**	**Adadelta**	**0.001**

الجدول (5-1)

مراحل التدريب :

```
Epoch 1/20
13440/13440 [==============================] - 203s 15ms/step - l
oss: 44.5353 - accuracy: 0.0359 - val_loss: 29.1762 –
 val_accuracy: 0.0298
Epoch 2/20
13440/13440 [==============================] - 201s 15ms/step - l
oss: 28.8087 - accuracy: 0.0453 - val_loss: 18.4644 –
val_accuracy: 0.03048.8097 - accuracy: 0.04
Epoch 3/20
13440/13440 [==============================] - 194s 14ms/step - l
oss: 21.3365 - accuracy: 0.0596 - val_loss: 31.0814 –
val_accuracy: 0.0292
Epoch 4/20
13440/13440 [==============================] - 209s 16ms/step - l
oss: 16.6048 - accuracy: 0.0688 - val_loss: 28.7631 –
val_accuracy: 0.0351
Epoch 5/20
13440/13440 [==============================] - 202s 15ms/step - l
oss: 13.3176 - accuracy: 0.0797 - val_loss: 13.1933 –
 val_accuracy: 0.0411
Epoch 6/20
13440/13440 [==============================] - 200s 15ms/step - l
oss: 10.8318 - accuracy: 0.0885 - val_loss: 0.0520 –
val_accuracy: 0.0435
Epoch 7/20
13440/13440 [==============================] - 190s 14ms/step - l
oss: 8.8686 - accuracy: 0.1004 - val_loss: 9.4752 –
val_accuracy: 0.0482
Epoch 8/20
13440/13440 [==============================] - 184s 14ms/step - l
oss: 7.2861 - accuracy: 0.1104 - val_loss: 3.9934 –
val_accuracy: 0.0446
Epoch 9/20
13440/13440 [==============================] - 187s 14ms/step - l
oss: 6.0679 - accuracy: 0.1154 - val_loss: 6.3797 –
val_accuracy: 0.0446
Epoch 10/2
```



مخطط بنية الشبكة

شكل (3-1)

13440/13440 [==============================] - 185s 14ms/step - loss:5.1201 - accuracy: 0.1261 - val_loss: 7.0771 - val_accuracy: 0.0506

Epoch 11/20

13440/13440 [==============================] - 226s 17ms/step - loss: 4.4273 - accuracy: 0.1357 - val_loss: 4.2086 - val_accuracy: 0.0494

Epoch 12/20

13440/13440 [==============================] - 181s 13ms/step - loss: 3.9438 - accuracy: 0.1410 - val_loss: 4.1507 - val_accuracy: 0.0500

Epoch 13/20

13440/13440 [==============================] - 186s 14ms/step - loss: 3.5715 - accuracy: 0.1508 - val_loss: 5.2314 - val_accuracy: 0.0524

Epoch 14/20

13440/13440 [==============================] - 188s 14ms/step - loss: 3.3110 - accuracy: 0.1647 - val_loss: 5.9062 - val_accuracy: 0.0512

Epoch 15/20

13440/13440 [==============================] - 193s 14ms/step - loss: 3.1115 - accuracy: 0.1863 - val_loss: 5.2185 - val_accuracy: 0.0554

Epoch 16/20

13440/13440 [==============================] - 188s 14ms/step - loss: 2.9685 - accuracy: 0.2005 - val_loss: 3.1892 - val_accuracy: 0.0565

Epoch 17/20

13440/13440 [==============================] - 189s 14ms/step - loss: 2.8637 - accuracy: 0.2208 - val_loss: 2.5184 - val_accuracy: 0.0577

Epoch 18/20

13440/13440 [==============================] - 195s 14ms/step - loss: 2.7610 - accuracy: 0.2369 - val_loss: 4.7781 - val_accuracy: 0.0577

Epoch 19/20

13440/13440 [==============================] - 186s 14ms/step - loss: 2.6759 - accuracy: 0.2561 - val_loss: 4.3546 - val_accuracy: 0.0577

Epoch 20/20

13440/13440 [==============================] - 185s 14ms/step - loss: 2.6018 - accuracy: 0.2702 - val_loss: 6.1469 - val_accuracy: 0.0577

كانت النتائج على الشكل التالي:

Test loss: 6.146929740905762

Test accuracy: 0.057738095521926

قيمة الخسارة كانت كبيرة نوعا ما ولم تتدرب الشبكة مع هذه التعييرات.

المرحلة التانية: تعيرات الشبكة كانت وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	Learning rate
50	**3**	**Adadelta**	**0.001**

الجدول (5-2)

ملخص التدريب :

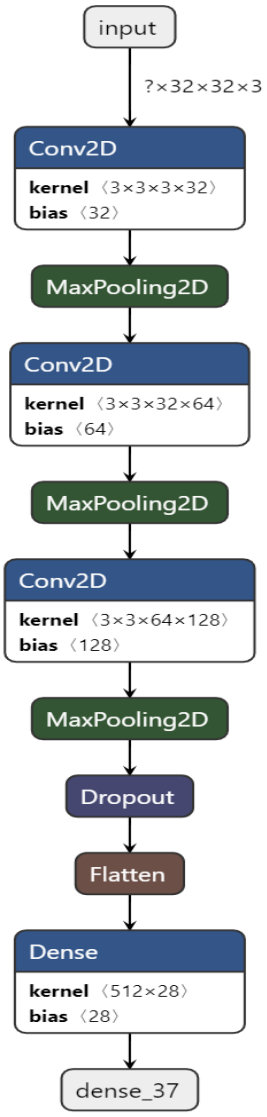
Layer (type)	Output Shape	Param #
conv2d_51 (Conv2D)	(None, 30, 30, 32)	**896**
max_pooling2d_35 (MaxPooling)	(None,15, 15, 32)	**0**
conv2d_52 (Conv2D)	(None,13, 13, 64)	**18496**
max_pooling2d_36 (MaxPooling)	(None, 6, 6, 64)	**0**
conv2d_53 (Conv2D)	(None, 4, 4, 128)	**73856**
max_pooling2d_37 (MaxPooling)	(None, 2, 2, 128)	**0**
dropout_37 (Dropout)	(None, 2, 2, 128)	**0**
flatten_23 (Flatten)	(None, 512)	**0**
dense_37 (Dense)	(None, 28)	**14364**

الجدول (5-3)

تم الحصول على النتائج التالية:

Total params: 107,612
Trainable params: 107,612
Non-trainable params: 0

Test loss: 6.437424659729004
Test accuracy: 0.06309524178504944



مخطط بنية الشبكة

شكل (2-3)

نلاحظ ان النتائج كانت مثل المرحلة الأولى وقيمة الخسارة هي نفسها.

المرحلة الثالثة: تم تعديل تعييرات الشبكة وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	Learning rate
100	**2**	**Adadelta**	**0.001**

الجدول (5-4)

ملخص التدريب :

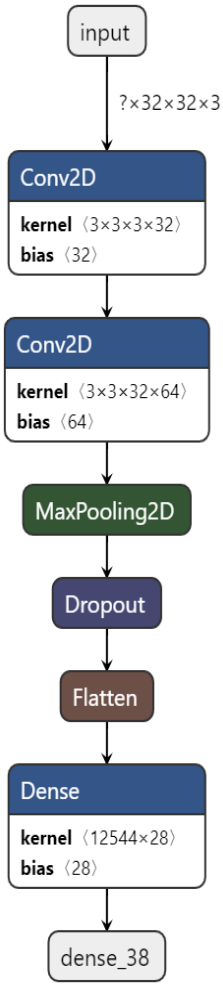
Layer (type)	Output Shape	Param #
conv2d_54 (Conv2D)	(None, 30, 30, 32)	**896**
conv2d_55 (Conv2D)	(None,28, 283, 64)	**18496**
max_pooling2d_38 (MaxPooling)	(None, 14, 14, 64)	**0**
dropout_37 (Dropout)	(None, 2, 2, 128)	**0**
flatten_23 (Flatten)	(None, 512)	**0**
dense_37 (Dense)	(None, 28)	**14364**

الجدول (5-5)

تم الحصول على النتائج التالية:

Total params: 370,652
Trainable params: 370,652
Non-trainable params: 0

Test loss: 6.746800422668457
Test accuracy: 0.04642857238650322



مخطط بنية الشبكة

شكل (3-3)

نلاحظ هنا ايضا تم الحصول على نتائج مقاربة للمرحلة السابقة وقيمة خسارة كبيرة

المرحلة الرابعة: تعييرات الشبكة كانت وفق الجدول اتالي:

Epochs	2_layer Conv2D	Optimizer	Learning Rate
30	**2**	**Adadelta**	**0.1**

الجدول (5-6)

ملخص التدريب :

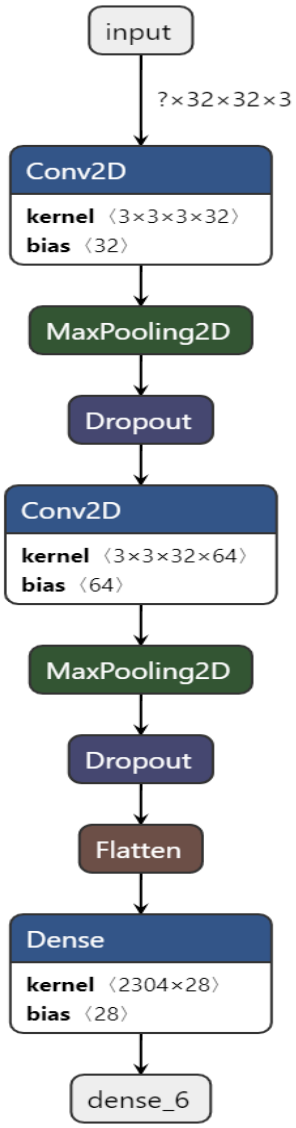
Layer (type)	**Output Shape**	**Param #**
conv2d_54 (Conv2D)	(None, 30, 30, 32)	**896**
max_pooling2d_11 (MaxPooling)	(None, 15, 15, 32)	**0**
dropout_9 (Dropout)	(None, 15, 15, 32)	**0**
conv2d_15 (Conv2D)	(None,13, 13, 64)	**18496**
max_pooling2d_12 (MaxPooling)	(None, 6, 6, 64)	**0**
flatten_6 (Flatten)	(None, 2304)	**0**
dense_6 (Dense)	(None, 28)	**64540**

الجدول (5-7)

تم الحصول على النتائج التالية:

Total params: 83,932
Trainable params: 83,932
Non-trainable params: 0

Test loss: 1.4442405700683594
Test accuracy: 0.8109890222549438



مخطط بنية الشبكة

شكل (3-4)

تم الحصول على قيمة خسارة اقل ولكن الشبكة لم تتعرف على الصور بشكل صحيح.

المرحلة الخامسة: التعييرات كانت وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	Learning Rate
15	**2**	**Adadelta**	**1**

الجدول (5-8)

ملخص التدريب :

Layer (type)	**Output Shape**	**Param #**
conv2d_43 (Conv2D)	(None, 28, 28, 32)	**2432**
conv2d_44 (Conv2D)	(None, 24, 24, 64)	**51264**
max_pooling2d_16 (MaxPooling)	(None, 12, 12, 64)	**0**
dropout_23 (Dropout)	(None,12, 12, 64)	**0**
flatten_15 (Flatten)	(None9216)	**0**
dense_23 (Dense)	(None, 28)	**258076**

الجدول (5-9)

تم الحصول على النتائج التالية:

Total params: 311,772
Trainable params: 311,772
Non-trainable params: 0
_____

Test loss: 0.9072326421737671
Test accuracy: 0.8062499761581421

نلاحظ قيمة الخسارة اقل من السابق وتعرفت الشبكة على حرفين وهما (أ , و).

ولكن مازالت الشبكة غير التمييز بشكل صحيح.

المرحلة السادسة: تعييرات الشبكة كانت وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	Learning Rate
15	**2**	**Adadelta**	**1**

الجدول (5-10)

ملخص التدريب :

Layer (type)	**Output Shape**	**Param #**
conv2d_48 (Conv2D)	(None, 28, 28, 32)	**2432**
conv2d_49 (Conv2D)	(None, 26, 26, 64)	**18496**
conv2d_50 (Conv2D)	(None, 26, 26, 128)	**8320**
max_pooling2d_18 (MaxPooling)	(None,13, 13, 128)	**0**
dropout_25 (Dropout)	(None, 13,13, 64)	**0**
flatten_17 (Flatten)	(None, 21632)	**0**
dense_25 (Dense)	(None, 28)	**605724**

الجدول (5-11)

تم الحصول على النتائج التالية:

Total params: 634,972
Trainable params: 634,972
Non-trainable params: 0
_____

Test loss: 1.3077809810638428
Test accuracy: 0.7461309432983398

نسبة الخسارة اعلى من المرحلة السابقة ولكن تعرف ايضا على الحرفين (أ , و).

المرحلة السابعة: تعييرات الشبكة وفق الجدول التالي:

Epochs	2_layer Conv2D	Optimizer	Learning Rate
10	**2**	**Adadelta**	**1**

الجدول (5-12)

ملخص التدريب :

Layer (type)	**Output Shape**	**Param #**
conv2d_71 (Conv2D)	(None, 30, 30, 32)	**896**
conv2d_72 (Conv2D)	(None, 26, 26, 64)	**51264**
max_pooling2d_28 (MaxPooling)	(None, 13, 13, 64)	**0**
dropout_35 (Dropout)	(None, 13,13, 64)	**0**
flatten_27 (Flatten)	(None, 10816)	**0**
dense_35 (Dense)	(None, 28)	**302876**

الجدول (5-13)

وكانت النتائج كالتالي:

Total params: 355,036
Trainable params: 355,036
Non-trainable params: 0

Test loss: 0.48462599515914917
Test accuracy: 0.8383928537368774

نلاحظ ان قيمة الخسارة أقل ولكن الشبكة لم تتعرف ابدا هنا.

يمكن الأستعانة بالتابع plt لإظهار رسوم بيانية توضح علاقة بين عدد الدورات و الدقة و كذلك العلاقة بين الدقة و مقدار الخسارة كما هو موضح في مخططات المبينة في الملحق الصفحة (48) .

الفصل السادس

الاستنتاجات والاقتراحات

الاستنتاجات والاقتراحات

يعد التعرف التلقائي على خط اليد مكونًا مهمًا للعديد من التطبيقات في مختلف المجالات. إنها مشكلة صعبة استحوذت على الكثير من الاهتمام في العقود الثلاثة الماضية. ركزت الأبحاث على التعرف على الكتابة اليدوية للغات اللاتينية. تم إجراء دراسات أقل للغة العربية. تحتوي مجموعة البيانات الخاصة بنا على 16800 حرفًا كل حرف كتب ب 600 شكل. بالإضافة إلى ذلك، نقترح نموذج التعرف التلقائي على خط اليد بناءً على الشبكات العصبية التلافيفية (CNN). نقوم بتدريب نموذجنا للتعرف على هذه الأحرف. تظهر النتائج أن أداء نموذجنا لم يكن يعمل بشكل جيد، حيث حقق دقة تبلغ 30% على مجموعة البيانات الموجودة لدينا.في المستقبل، سيكون من المفيد دراسة مختلف نماذج التعلم الآلي الناجحة في مجموعة البيانات هذه، يمكن استخدام نموذجنا للتعرف على خط اليد باللغة العربية في أي تطبيق يتطلب هذه الوظيفة. يمكن استخدامه كجزء من خط الأنابيب الذي يربط بين الأحرف العربية، للتعرف على الكلمات الكاملة. يمكن استخدام النموذج في بيئة تعلم النقل للتعرف على الأحرف في اللغات الأخرى. نخطط لدمج شبكتنا العصبية في تطبيقات أخرى.

المراجع

1) By Dataflair Team , AUGUST 6, 2020
https://data-flair.training/blogs/python-deep-learning-project-handwritten-digit-recognition/

2) By Mohamed Loey , 2017
https://github.com/mloey/Arabic-Handwritten-Characters-Dataset

3) By Prabhu in Data Science, Mar 4, 2018
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

4) https://cs231n.github.io/convolutional-networks/

5) by Adrian Rosebrock on October 14, 2019
https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/

6) Keras Documentation
https://keras.io/getting_started/

7) Tensorflow Documentation
https://www.tensorflow.org/guide/keras/sequential_model

8) Making Digital Ink Editable via Deep Generative Modeling
By Emre Aksan Fabrizio Pece Otmar Hilliges

الشكر والتقدير

" كن عالما.. فإن لم تستطع فكن متعلما، فإن لم تستطع فأحب العلماء، فإن لم تستطع فلا تبغضهم "

بعد رحلة بحث وجهد واجتهاد تكللت بإنجاز هذا المشروع، نحمد الله عز وجل على نعمه التي منَ بها علينا فهو العلي القدير. كما لا يسعنا إلا أن نخص بأسمى عبارات الشكر والتقدير للدكتور "محمد الخطيب" لما قدمه لنا من نصح ومعرفة طيلة انجاز هذا البحث. ونتوجه بكلمة شكر إلى الذين كانوا عوناً لنا في مشروعنا هذا ونوراً يضيء الظلمة التي كانت تقف أحيانا في طريقنا إلى من زرعوا التفاؤل في دربنا وقدموا لنا المساعدات والتسهيلات والمعلومات، فلهم منا كل الشكر. أما الشكر الذي من النوع الخاص فنحن نتوجه به أيضا إلى كل من لم يقف إلى جانبنا، ومن وقف في طريقنا وعرقل مسيرة بحثنا فلولا وجودهم لما أحسسنا بمتعة العمل وحلاوة البحث، ولما وصلنا إلى ما وصلنا إليه فلهم منا كل الشكر ...

الملحق : الكود الكامل للمشروع:

```
import keras

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
batch_size = 128
num_classes = 28
epochs = 30
input_shape = (32, 32,3)
datagen = ImageDataGenerator()
train = datagen.flow_from_directory(
        'Pictures\letters\Train',
        target_size=(32, 32),
        batch_size =batch_size,
        class_mode='categorical')
test = datagen.flow_from_directory(
        'Pictures\letters\Test',
        target_size=(32, 32),
        batch_size =batch_size,
        class_mode='categorical')
model = Sequential()
model.add(Conv2D(128, kernel_size=3,activation='relu',
input_shape=input_shape))
model.add(Conv2D(64,kernel_size=3, activation='relu'))
model.add(Conv2D(32,kernel_size=3, activation='relu'))
model.add(Conv2D(16,kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])
hist = model.fit( train,
       # steps_per_epoch=129,
        verbose=1,
        epochs=epochs,
        validation_data= test)
model.save('Pictures\letters\mnist.h5')
print("Saving the model as mnist.h5")
score = model.evaluate(test, verbose=0)
model.summary()
```

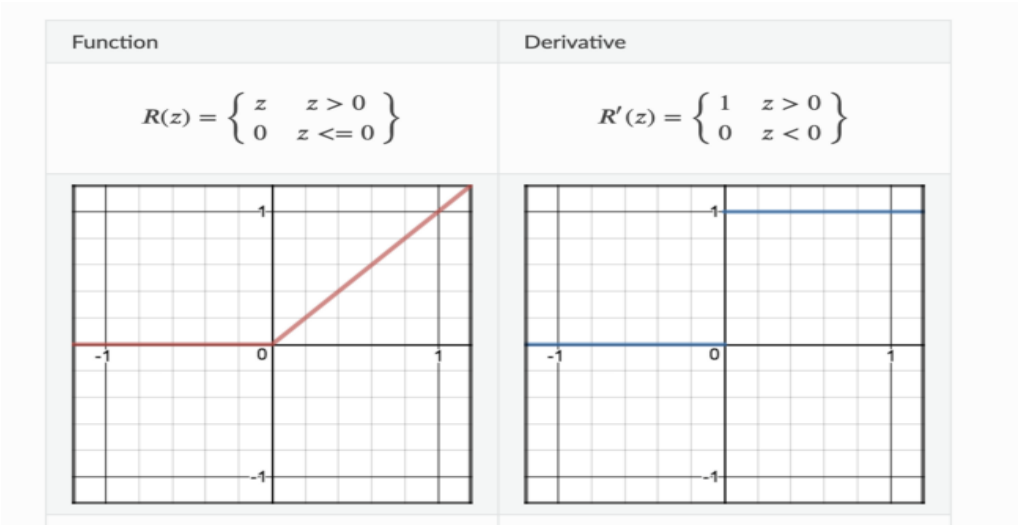
```
def predict_digit(img):
    img = img.resize((64,48))
    #convert rgb to grayscal
    img = img.convert('L')
    img = np.array(img)
    #reshaping to support our model input and normalizing
    img = img.reshape(1,32 , 32 ,3)
    res = model.predict(img)[0]
    return np.argmax(res)
class App(tk.Tk):

    def __init__(self):
        tk.Tk.__init__(self)
        self.x = self.y = 0
        # Creating elements
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross"         )
        self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise", command =
        self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)
        # Grid structure
        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)
        #self.canvas.bind("<Motion>", self.start_pos)
        self.canvas.bind("<B1-Motion>", self.draw_lines)
    def clear_all(self):
        self.canvas.delete("all")
    def classify_handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas
        rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas
        im = ImageGrab.grab(rect)
        digit = predict_digit(im)
        self.label.configure(text= str(alpha[digit]))
    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')
app = App()
mainloop()
```

ReLU (Rectified Linear Unit):

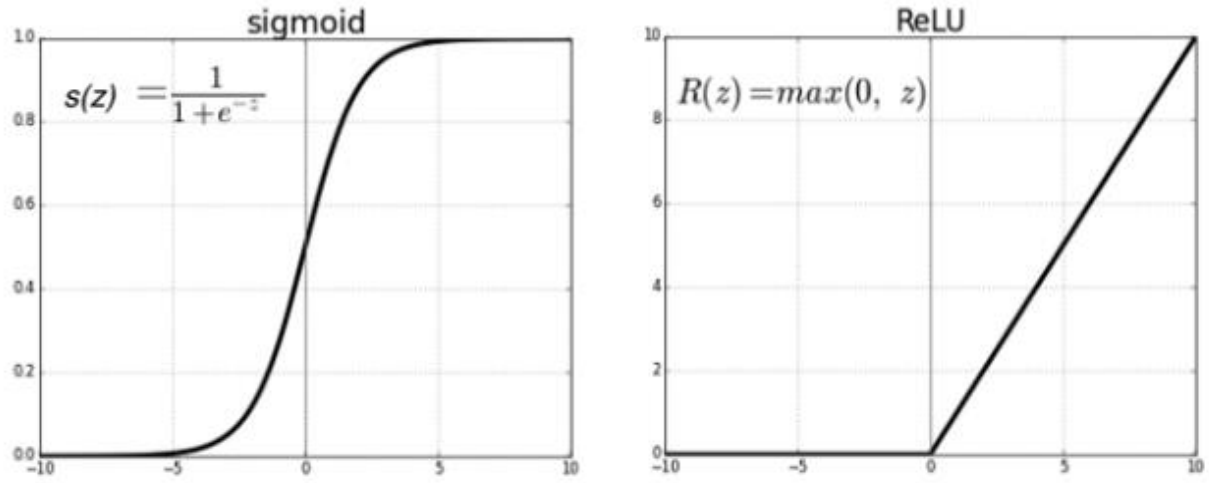
- هو تابع التفعيل الأكثر شيوعًا والتي تُستخدم في الطبقة المخفية من NN. صيغتها بسيطة بشكل مخادع:

$\max(0, z)$. على الرغم من اسمها ومظهرها ، إلا أنها ليست خطية وتوفر نفس مزايا Sigmoid ولكن مع أداء أفضل.



وتتمثل الميزة الرئيسية في أنه يتجنب ويصحح مشكلة التدرج اللوني المتلاشي وأقل تكلفة من الناحية الحسابية من , sigmoid tanh. لكن له بعض التراجع. في بعض الأحيان يمكن أن تكون بعض التدرجات هشة أثناء التدريب ويمكن أن تموت. وهذا يؤدي إلى موت الخلايا العصبية ، وبعبارة أخرى ، بالنسبة لعمليات التنشيط في المنطقة (x <0) من ReLu ، سيكون التدرج صفرًا بسبب عدم تعديل الأوزان أثناء الهبوط. هذا يعني أن الخلايا العصبية التي تدخل في هذه الحالة ستتوقف عن الاستجابة للتغيرات في الخطأ / الإدخال (ببساطة لأن التدرج هو 0 ، لا شيء يتغير). لذلك يجب أن نكون حذرين للغاية في اختيار وظيفة التنشيط ، ويجب أن تكون وظيفة التنشيط حسب متطلبات العمل.

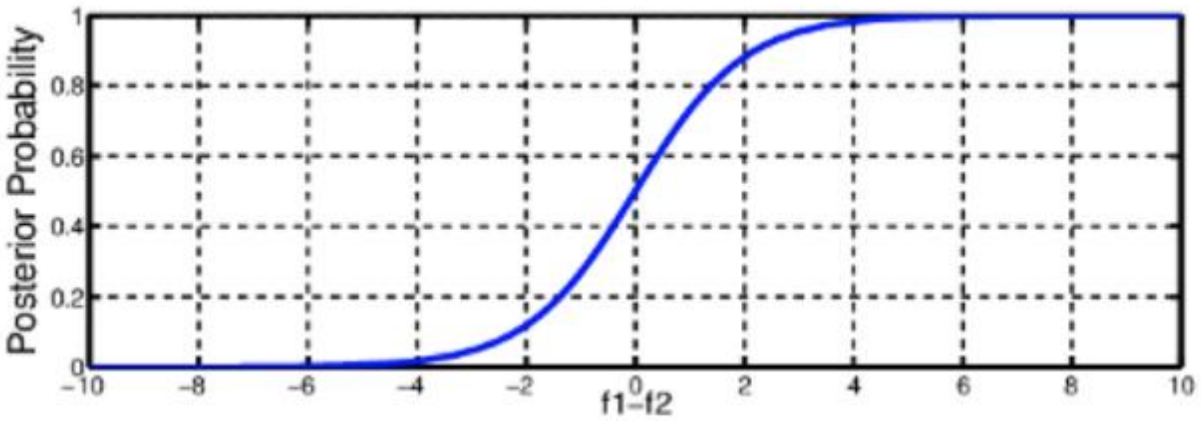
عندما نقارن مع تابع التعفيل Sigmoid ، يبدو الأمر:



Sigmoid Vs ReLU

Softmax:

بشكل عام ، نستخدم التابع في آخر طبقة من الشبكة العصبية التي تحسب توزيع الاحتمالات للحدث على أحداث مختلفة. الميزة الرئيسية للتابع هي القدرة على التعامل مع فئات متعددة.



عندما نقارن بين وظائف التنشيط الSigmoid و softmax ، فإنها تنتج نتائج مختلفة.

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid input values: −0.5, 1.2, −0.1, 2.4

Sigmoid output values: 0.37, 0.77, 0.48, 0.91

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

SoftMax input values: −0.5, 1.2, −0.1, 2.4

SoftMaxoutput values: 0.04, 0.21, 0.05, 0.70

المراحل الرئيسية في تاريخ دراسة وتطبيق الشبكات العصبية الاصطناعية:

1943 – قام كل من دبليو مكولوتش و دبليو بيتس بإضفاء الطابع الرسمي على مفهوم الشبكة العصبية في مقال أساسي عن الحساب المنطقي للأفكار والنشاط العصبي.

1948 – ينشرن. وينر مع زملائه عملاً عن علم التحكم الآلي. الفكرة الرئيسية هي تقديم العمليات البيولوجية المعقدة بنماذج رياضية.

1949 – يقدم د. هب أول خوارزمية تعلم. في عام 1958 ، اخترع F. Rosenblatt مدركًا أحادي الطبقة وأظهر قدرته على حل مشاكل التصنيف. اكتسب Perceptron شعبية – فهو يستخدم للتعرف على الأنماط والتنبؤ بالطقس وما إلى ذلك. في عام 1960 ، قام Widrow مع تلميذه Hoff ، بناءً على قواعد دلتا (صيغ Widrow) ، بتطوير Adalin ، والذي بدأ على الفور في استخدامه للتنبؤ ومشاكل التحكم التكيفي. (Adaline adder) Adaline هو الآن ميزة قياسية للعديد من أنظمة معالجة الإشارات. في عام 1963 في معهد مشاكل نقل المعلومات التابع لأكاديمية العلوم في اتحاد الجمهوريات الاشتراكية السوفياتية. أجرى A. P. Petrov دراسة تفصيلية للمهام "الصعبة" لبروسبترون. في عام 1969 ، نشر إم. مينسكي إثباتًا رسميًا لقيود الإدراك الحسي وأظهر أن المدرك لا يستطيع حل بعض المشكلات المرتبطة بثبات التمثيلات. الاهتمام بالشبكات العصبية ينخفض بشكل حاد. في عام 1972 ، اقترح كل من T. Kohonen و J. Anderson بشكل مستقل نوعًا جديدًا من الشبكات العصبية القادرة على العمل كذاكرة. في عام 1973 ، اقترح B.V. Khakimov نموذجًا غير خطي مع المشابك المستندة إلى الخيوط وقدمها لحل المشكلات في الطب والجيولوجيا والبيئة.

1974 – بول ج. ابتكر جالوشكين في نفس الوقت خوارزمية انتشار الخطأ لتعليم الإدراك متعدد الطبقات1975 – تمثل فوكوشيما Cognitron – وهي شبكة ذاتية التنظيم مصممة للتعرف على الأنماط الثابتة ، ولكن هذا يتحقق فقط من خلال حفظ جميع حالات الصورة تقريبًا.

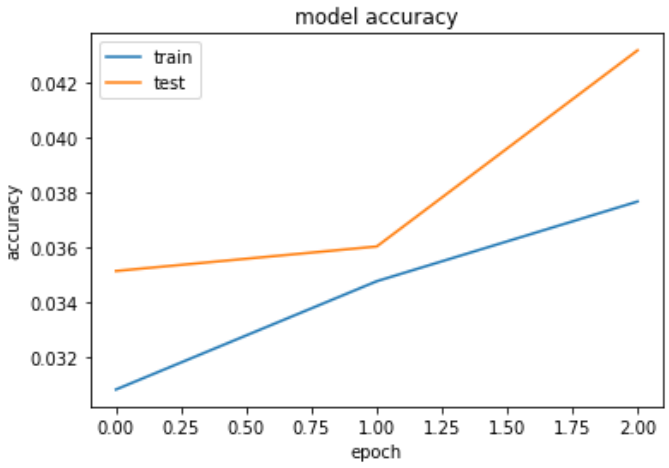
1982 – بعد فترة من النسيان ، يتزايد الاهتمام بالشبكات العصبية مرة أخرى. أظهر جيه هوبفيلد أن الشبكة العصبية ذات التغذية المرتدة هي نظام يقلل الطاقة (ما يسمى بشبكة هوبفيلد). يقدم Kohonen نماذج لشبكة التعلم غير الخاضعة للإشراف (شبكة Kohonen العصبية) ، ويحل مشاكل التجميع ، وتصور البيانات (خريطة Kohonen ذاتية التنظيم) وغيرها من مشاكل التحليل الأولي للبيانات.1986 –

ديفيد آي روميلهارت وجي إي هينتون ورونالد جيه وليامز وفي نفس الوقت ج. S. I. Bartsev و V. A. Okhonin (مجموعة كراسنويارسك) أعاد اكتشاف وتطوير طريقة backpropagation. بدأ انفجار الاهتمام بالشبكات العصبية المدربة. 2007 أنشأ جيفري هينتون من جامعة تورنتو خوارزميات للتعلم العميق للشبكات العصبية متعددة الطبقات. يعود هذا النجاح إلى أن Hinton استخدم آلة Boltzmann المحدودة (RBM – آلة Boltzmann المقيدة) لتدريب الطبقات السفلية من الشبكة.

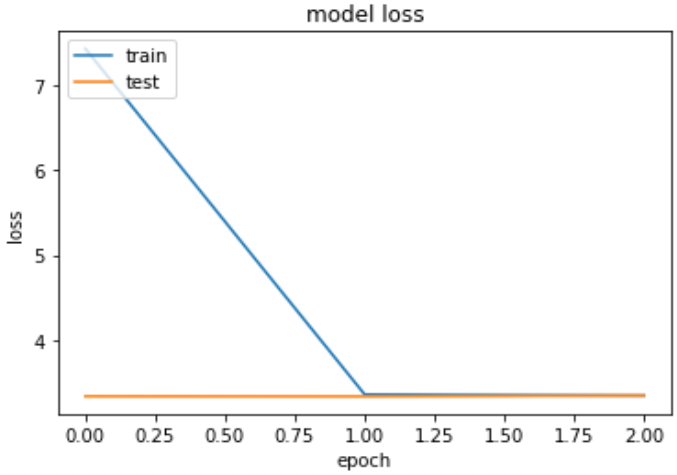
مخططات التوضحية للعلاقات بين عدد الدورات و الدقة أثناء تدريب الشبكة العصبية :

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

الرسم البياني :



مخططات التوضحية للعلاقات بين عدد الدورات و الخسارة أثناء تدريب الشبكة العصبية :


model loss
train
test
loss
epoch
7
6
5
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00