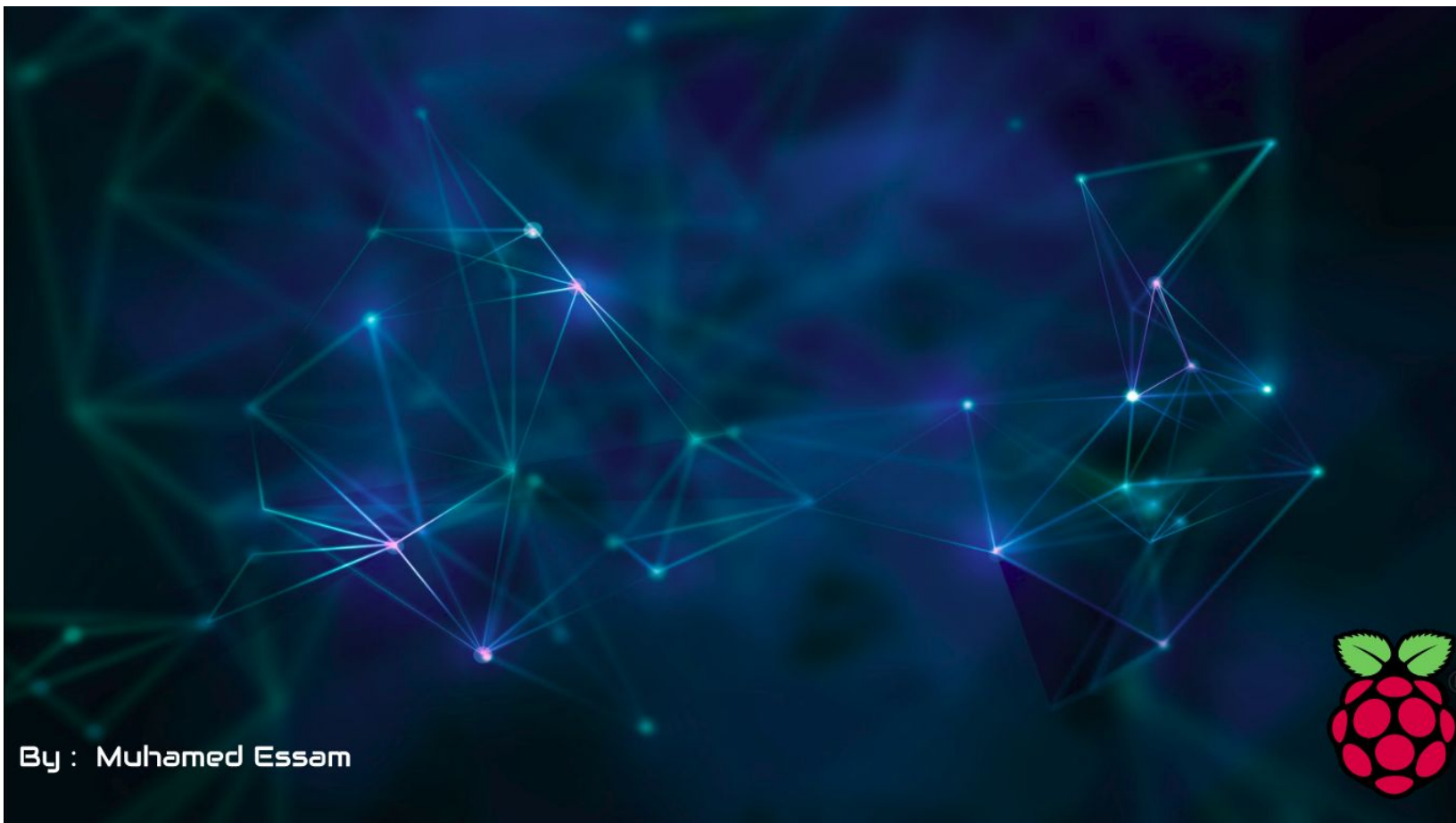


Muhammed Essam

September 1, 2018

Road to Deep Learning



By : Muhammed Essam

Chapter 1

Basics

- Intro to Neural Networks
- Biological Neural Networks
- ANN
- MLP
- Activation functions
- Optimization
- Gradient Descent
- First Model with Keras

Intro to Neural Networks

لو انت **Developer** فأكيد زهقت من bugs في Code بتاعك من ان في مشاكل كتيرة أوي بتقابلك عشان تطلع حاجة تناسب Client وممكن تدخل نفسك في لسته من conditions لمجرد بس انك خايف من أي Action هوا ياخده وتبقي انت مش متوقعه .. وده ببدا يوصلنا لسؤال مهم .. هو مفيش طريق غير ده ؟ مينفعش أوصل لأسلوب أخلي الكود بتاعي يفكر ؟ وياخد قرار ؟؟

- عمرك سألت نفسك , انت ازاي بتفهم أي معلومة ؟ أو إزاي دماغك بتشتغل ؟

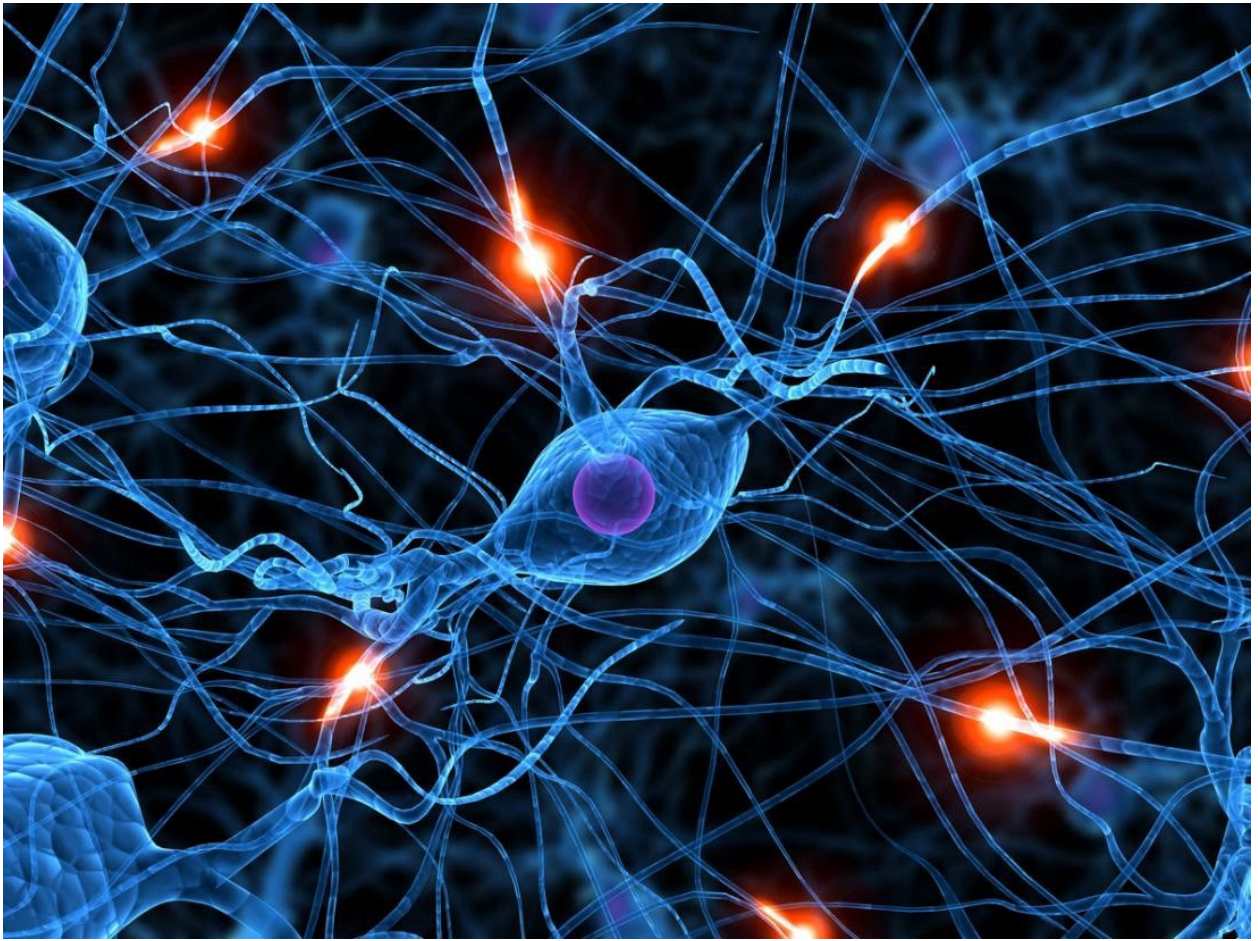
- تفتكر لو خيّرتك انك تبدل دماغك بكومبيوتر هتقبل ؟

- طيب تفتكر مين أفضل Computer ولا human brain ؟

- تفتكر ازاي المعلومات في دماغك مش بتخش في بعضها؟

Biological Neural Networks

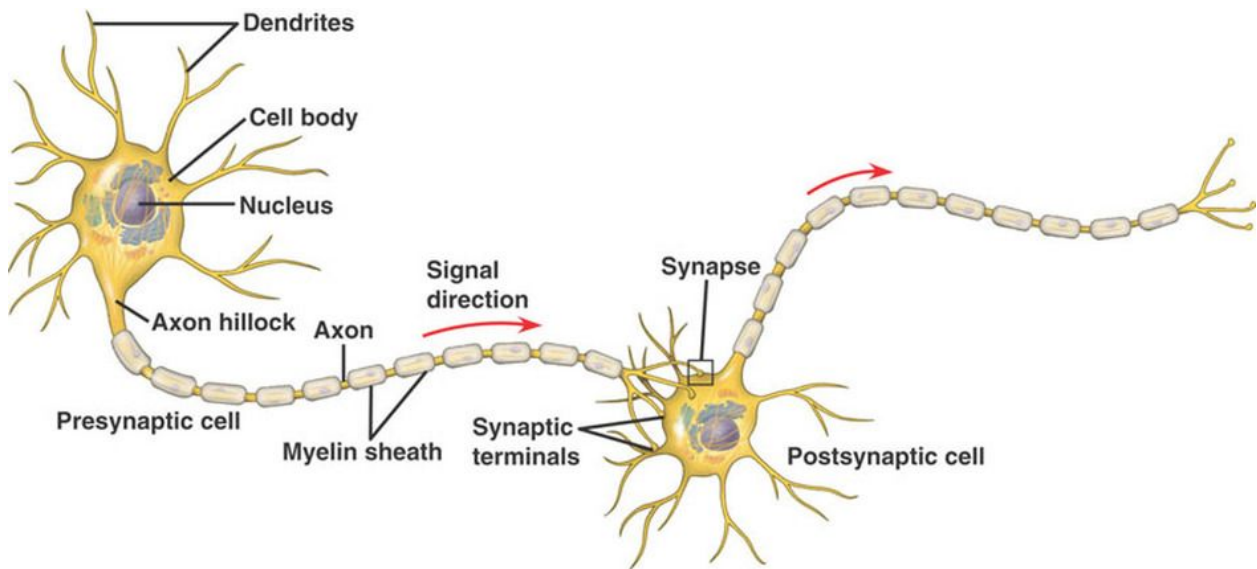
لان احنا محتاجين نعمل نظام مشابه لطريقة تفكير الانسان فاحنا كان لازم ندرس ازاى المخ البشري بيتعامل مع المعلومات .



تعالى نحاول نفهم الدنيا بطريقة بسيطة فى الحقيقة المخ بيتكون من خلايا عصبية بنسميها **Neuron** وهى فى الحقيقة بتستقبل المعلومة من الحواس بتاعتنا زى العين أو الأذن ...

الخلايا العصبية دي بتبقي مترتبة و متوصلة ببعض باشكال مختلفة وبتكون شبكة بنسميها الشبكة العصبية.

في الشكل اللي تحت ده هتلاقي فيه رسمة Neuron من جوا وهي متوصلة ب Neuron تانية .



أهم النقط في الشكل ده بالنسبة لنا هي ان قدامك راس الخلية وهو اللون الأصفر الكبير ده وبنسميه Soma وليه ديل أو طرف وهو اسمه Axon ... محتاجين ناخد بالنا من ايه بقي ؟ ان احنا بيبقي عندنا وليكن information في neuron وعايزة تبدأ تنقلها ف axon بينتهي ب Axon terminals أو بنسميها Synapses وبتستقبلها أطراف ال neuron التانية عن طريق dendrites وتوصلها لجوة Soma عند Nucleus نفسها وهكذا تبدأ هي كمان توصل اللي وراها بنفس الطريقة ..

طيب ايه النقطة المهمة بالنسبة لنا ؟ ان في الحقيقة كل neuron متوصلة بالاف من neurons فانت لو فكرت هتقول ان في فوضي ممكن تحصل والمعلومات تخش في بعض ... بس في الحقيقة يحصل نقطة مبهرة جدا وهي ان كل neuron يتحكم في جيرانها اللي متوصلة بيهم ياما تعملهم **firing** يعني تشغلهم أو لا ! فبالتالي neurons اللي شايلة المعلومة المطلوبة بس هي اللي بتوصل..

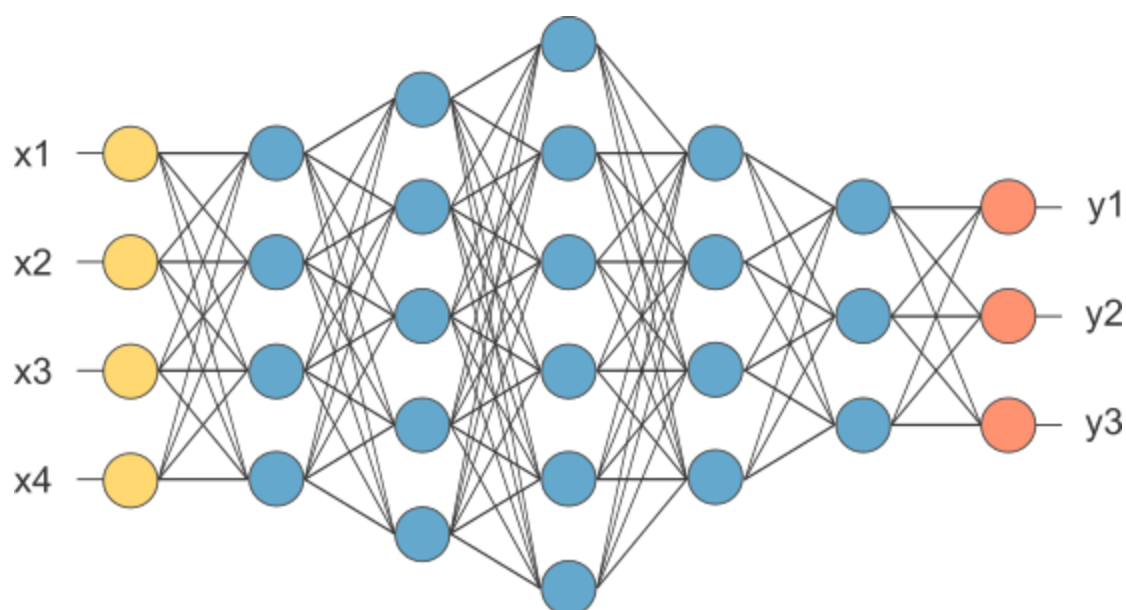
نوضح كلامنا مرة ثانية كل اللي قولناه :-

- المخ بيتكون من شوية **Neurons** متوصلة ببعض بتنقل المعلومات .
- كل **Neuron** بتبقي متوصلة بالاف من neurons وتعمل activation / firing لل neurons اللي بتأثر معاها في المعلومة
- المعلومات دي بتتمثل في الاخر ب **image** أو **memory** معينة أو... الخ

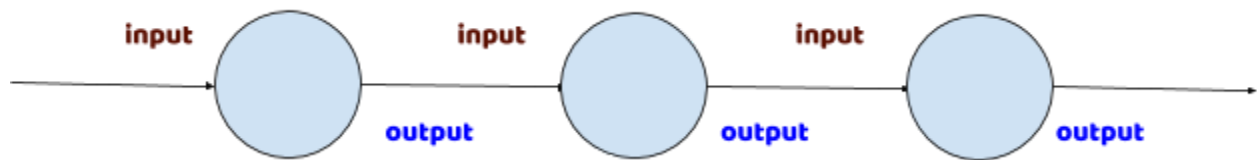
المفروض دلوقتي ان انت بدأت تربط انا احنا هحاول نقلد الفكرة دي بشكل اصطناعي **Artificial** ونشوف هنقدر نوصل منها لايه !

Welcome to Artificial Neural Networks

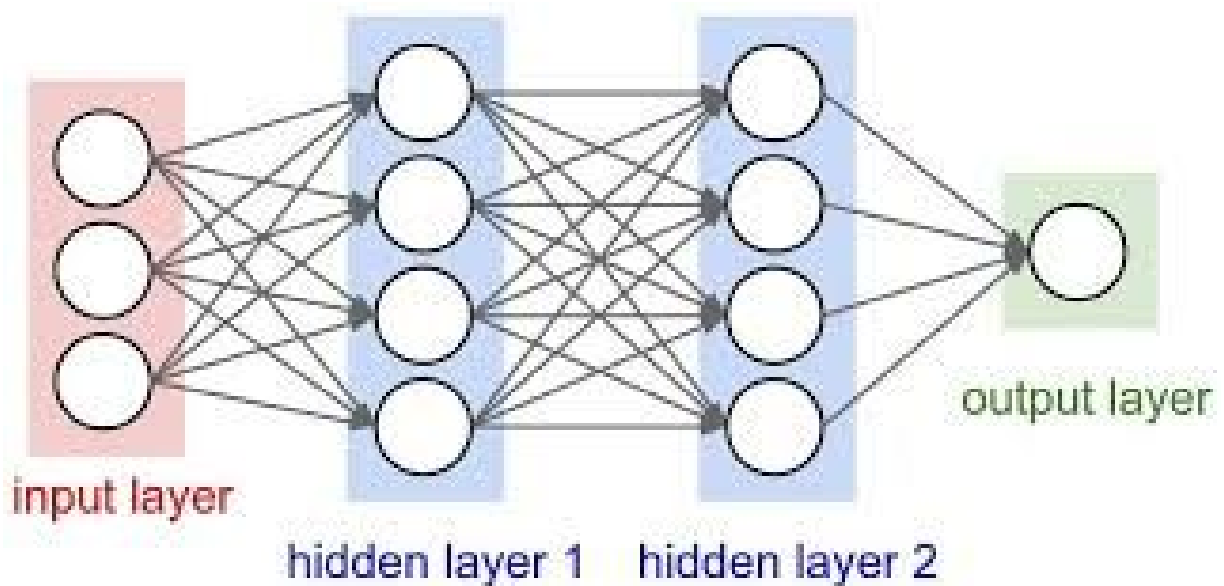
من هنا هنبداً ندخل مجالنا وهو Computer Science فاحنا دلوقتي في خطوة ان احنا نقلد فكرة عمل neurons بتاعت المخ تمام؟



الكرات الزرقا والحمرا دي بالنسبة لنا هي **Neurons** أو ممكن تسميها **Nodes** طيب احنا برده هنا عايزين نوصل Neurons ببعض زي ما اتكلمنا في human brain عن Dendrites , Synapses هنا احنا هنوصل Neurons ببعض بحيث ان ((كل Neuron ليها input وهو output اللي قبلها وليها output وهو input اللي بعدها))! * بص تحت علي الرسمة و اقراها مرة ثانية*



تمام زي ماقلنا في human brain ان neuron متوصلة بعدد كبير جدا من neurons فاحنا برده هنا هنعوصل كل neuron باكثر من neuron .. بالمنظرده



دې في الحقيقة شكل Basic لل ANN = Artificial Neural Network ملاحظ ايه ؟

- في 3 حاجات رئيسية وهي , Input layer , Output layer , hidden layer
- كل layer متكونة من عدد من Neurons

- كل Neuron متوصلة ب كل Neurons اللي وراها واللي قبلها ... وده اللي بنسميه **Fully Connected Neural Network** .
- في neuron واحدة في output layer يعني Single output طبعا ممكن يكون عندك Multiple outputs علي حسب Task

في سؤال مهم ؟ ازاى neural network بتتعلم ؟
طب هرد عليك بسؤال أهم وهو ازاى انت نفسك إتعلمت ؟؟

. في الحقيقة انت في فترة كبيرة من حياتك مشيت فيها بمبدأ Supervised Learning يعني بتشوف حاجة قدامك حد بيعملها



وانت بتقلده لحد ما تقدر تعملها

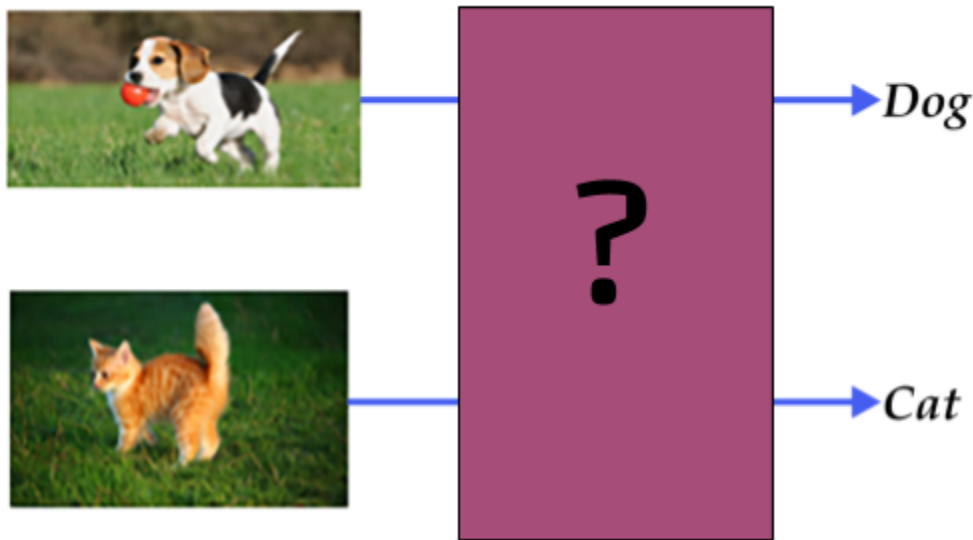


في أمثلة كثيرة أوي لكدة لو فاكّر مثلاً انت ازاي اتعلمت تميز ما بين الحيوانات كنت مثلاً بتشوف صور ليها و يقعدوا قلوبك دي قطعة دي قطعة مثلاً ده كلب وبعد كده يبدأوا يسألوك ده ايه ؟ فانت تقول قطعة فيقولك صح .. فانت عرفت منين ؟ او اتعلمت منين ؟ من تجارب سابقة ليك!

طيب وصلت لايه من الكلام ده ؟

لان احنا بنحاول نقلد طريقة تعليمنا احنا كبشر فاحنا هنبدأ نستخدم أسلوب مشابه في Neural Networks !

فاحنا هنبداً نوخر شوية Data ولو تخيلنا ان احنا معنا Neural Network model ده وأكنه Black Box يعني حاجة انا مش عارف شغالة ازاي ... انا بس بوفرلها Data عبارة عن شوية Inputs وال Outputs بتاعتها بالمنظر ده

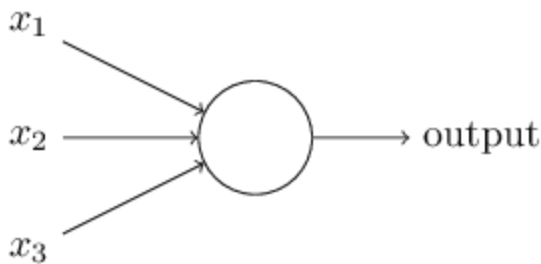


فانت دلوقتي معاك Model ببداً يعملك Classification يعني بتدخله صورة ويقولك هل ده قطعة ولا كلب ! ال Model بالنسبة لنا Black box يعني انت بتتعامل معاه دلوقتي انك بتديله inputs , outputs وهو هيبدأ عن طريق Data دي يقدر يتعلم ولو سألته علي حاجة بعد كده هل هي قطعة أو كلب يقدر يقولك توقعه أو تصنيفه ..

يبقي نقدر نقول ان Neural Network شرط انها تتعلم انك توفر لها Labeled Data يعني data ليها output سليم بنسميه Label .

بس في الحقيقة عشان Neural Network تتعلم بتحتاج تمشي علي مرحلتين أول حاجة هي Forward Propagation وتاني حاجة هي Backward Propagation .. بس قبل مانوضح معناهم ايه وبيتموا ازاى تعالوا نبدأ ب أساسيات Neural Network !

زي ما ال basic structure بتاع اي جسم عندك هو ال ذرة عندنا هنا في Neural Network ابسط وحدة هي **Perceptron**



ده شكل Perceptron بيبقي
ليه أكثر من input اللي هما
 x_1 , x_2 , x_3

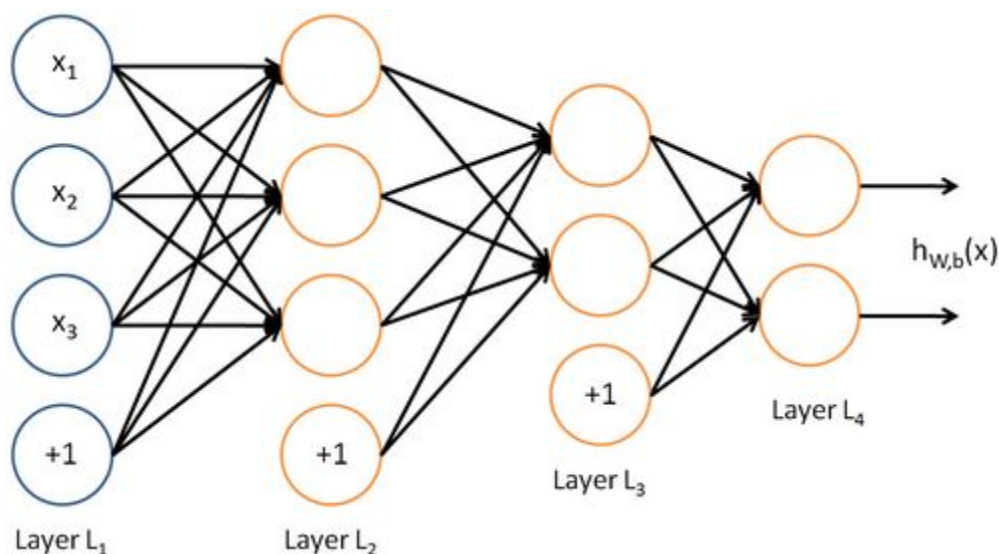
فال Perceptron بتقوم بعملية جمع ليهم وتطلعك Output
بس في Trick ان دلوقتي ده لو System عندك تفتكر هل
كل Inputs بتأثر في Output بنفس القيمة ؟ أكيد لا

كل واحد ليه أهميته .. مثال يعني لو بنميز ما بين راجل وست
هل لون البشرة بيأثر زي مثلا طول الشعر ؟ أكيد لا كل حاجة

ليها عندك أهمية بنسبها احنا هنا **Weight** وعشان كده بنبدأ نضرب كل Input اللي هي x عندنا في أهميته أو **Weight** يعني اللي هنسميه W

طيب يبقى كده وصلنا ان $y = w_1 * x_1 + w_2 * x_2 + w_3 * x_3$

فاضل خطوة بسيطة جدا وهي ان احنا بنعمل Adding لل **Bias** وهو بالنسبة لنا **Free term** يعني هو مش بيبقى ليه **variable input** ولا حتي **Connected** بال **Previous Layer**



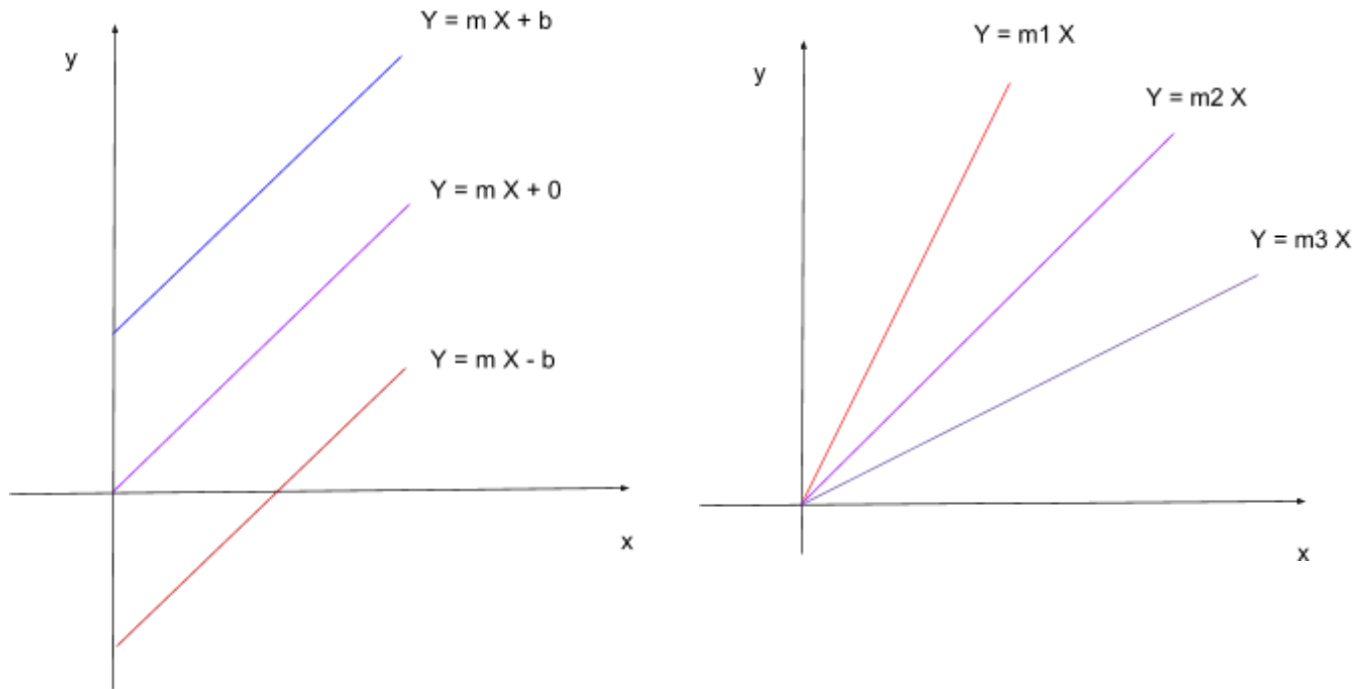
فهو بتديك قدرة علي انك تتحكم في Output من غير أي اعتماد علي previous input وده حاجة مميزة جدا ..

بس لو حابب تفتكر انت شوفت **Bias** دي فين قبل كده فانت لو خدت مثال لأبسط شكل **Neural Network** هيبقى عبارة

عن **input 1** و **output 1** فهيبقى $y = w_1 x$ لما تمثّلها graphically هتلاقىها عبارة عن خط مستقيم **Linear function**

انك تتحكم في w_1 هتديك فرصة انك تتحكم في ميل الخط

يعني في الزاوية يعتبر طيب لو عايز اعمل Shift اصلا لل
Graph كله ؟ كنا ساعتها بنضيف Bias term عشان المعادلة
تبقى $Y = mx + b$ اللي هي هنا $y = w_1 x + b$

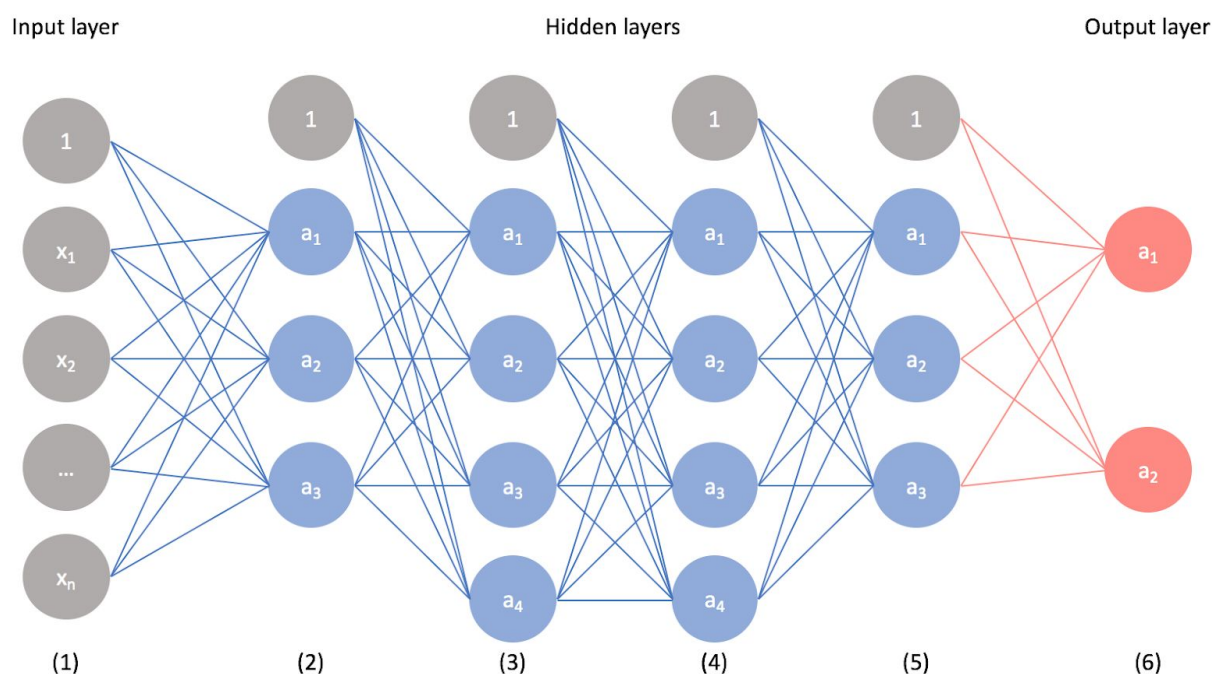


يبقي كده وصلنا ان احنا يعتبر في كل شغلنا بنضيف Node
زيادة وهي مش متوصلة بال previous layer وبيبقى ليها
weight بنسميه bias وبيبقى ليها input و قيمته 1

يبقي المعادلة النهائية اللي بتمثل **Perceptron**

$$Y = w_1 x_1 + w_2 x_2 + w_3 x_3 + 1 * b \text{ (for Perceptron)}$$

تفكر احنا بيبقي عندنا perceptron واحدة ؟ لا غالبا بيبقي عندك اكثر من perceptron في Layer الواحدة وكلهم بيقوا متوصلين ب inputs بتاعتك وبنأخذ output بتاع كل واحدة ونغذي بيها ال Layer اللي بعدها واكنه input بتاعهم وهكذا



Input layer (1) , Hidden Layers (2:5) , Output Layer(6)

الشكل اللي قدامك ده بيمثل **MLP** اللي هي Multi Layer Perceptron.... دلوقتي المفروض بيدأ يجيك سؤال ؟ هو ان في عدد كبير جدا من weights و biases موجود ... طيب هو مين اللي بيحسب قيمها ؟ لان اكيد كل ما هتغير فيهم كل ما هيتغير Outputs

في الحقيقة مش انت اللي بتحسب قيمهم .. انت بس بتعمل Random initialization ليهم يعني بتفرض أرقام عشوائية

ليهم كلهم و عن طريق Gradient Descent اللي موجود في Back Propagation بيبدأ يتعدّلوا للرقم الأقرب للصح ..

لحد دلوقتي احنا مشرحناش ايه هي Back ولا حتي Forward propagation بس بما اننا بنعمل explode لأشهر أساسيات Neural Networks فاحنا لازم نعرف دلوقتي يعني ايه gradient Descent و ايه أنواعه وده اللي احنا هنعرفه بشكل أوسع في Optimization .

Optimization

في علم Data Analysis بشكل عام دائما يبقي عندنا Error وال Error ده بنمثله ب equation وبنسميها Cost function

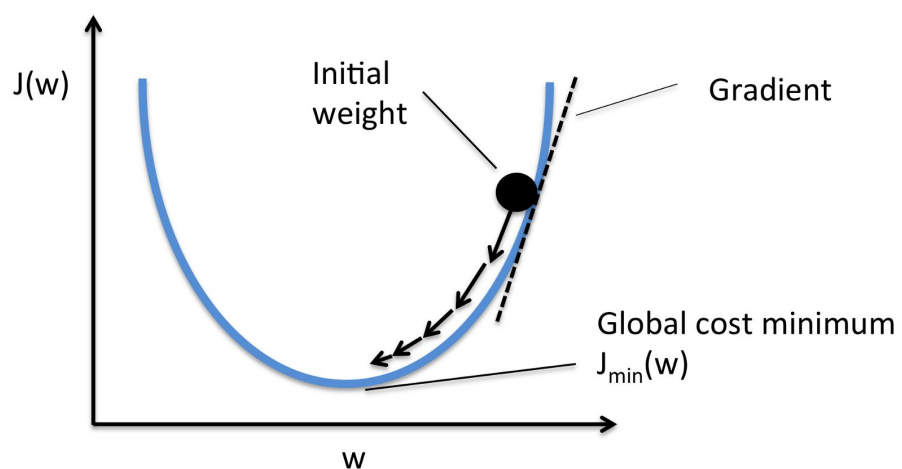
فكلمة Optimization المقصود منها انك تعمل reduction لل Cost Function دي او ساعات بنسميها Losses بس تعالوا نفهم ازاى بنقل Error ده لو قلنا ان انت دلوقتي مثلا معاك Output معين Predicted وليكن قيمته 60 والمفروض الناتج الصح الحقيقي Actual يعني هو 100

يبقي بالنسبة لك Error لو خدنا احد أشكاله وهو انك تطرح بس ال 2 من بعض فكداه

$$\text{Error} = \text{Predicted} - \text{Actual}$$

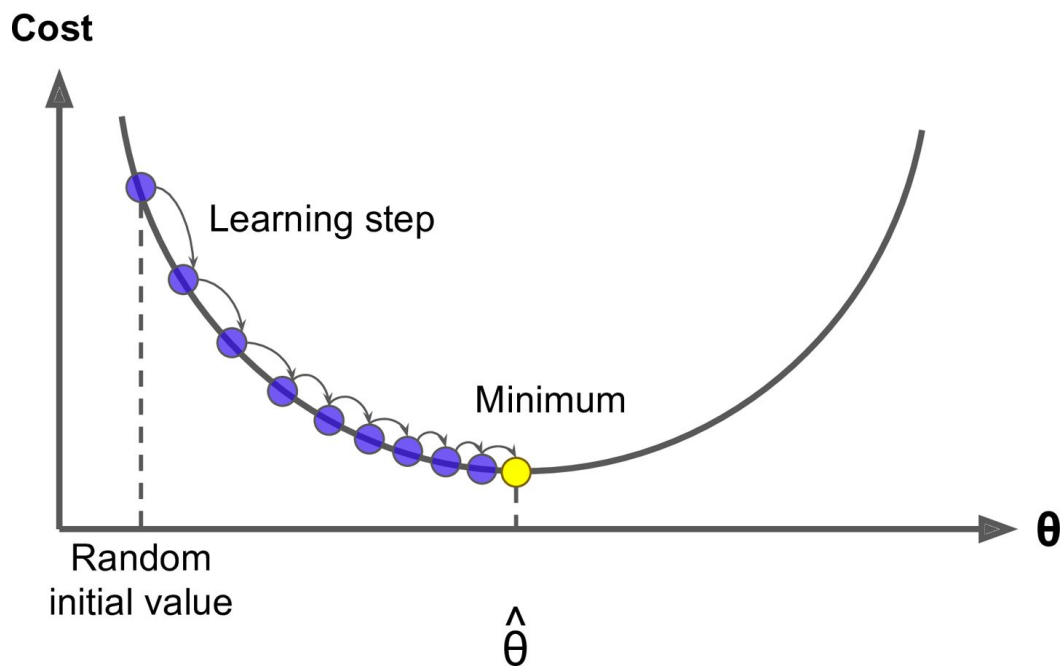
فلو حاولنا نحسب ال error هيطلع معنا 60-100 يعني 40

فتخيل معايا
دلوقتي error
ده قيمته
اتقاطعت مع ال
graph في
نقطة عند
initial weight
معين .

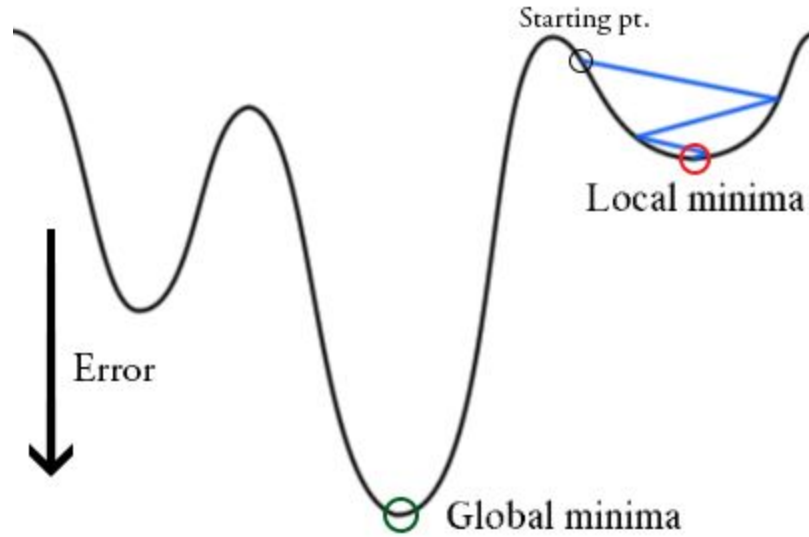


فانت هدفك تخلي النقطة دي تحت عند أقل error اللي هو بنسميه Global Minimum ...
بس هو ايه اللي هيوصلنا لتحت ايه اللي يعرفنا ؟

في الحقيقة Gradient descent هو بالنسبة لك عبارة عن Vector بيوجهك لأقل Error يعني لو بتكلم عن concave function زي دي من الدرجة الثانية فهو لما ناخذ derivative بتاعها هيبقي مجرد vector بيشاور لتحت فانت هتبدأ تنزل ب Rate معين بنسميه Learning Rate أو Alpha



بس Learning rate ليه قصة لازم نفهمها
احنا في بعض Costfunctions هنتكلم عن انها مش مجرد Curve لا دي هي مثلا 3d بالشكل ده



فلو قلت ان ال Step بتاعتك هي Learning rate
تخيل لو رقم صغير فانت هتقدر توصل minimum .. بس في
الحقيقة ممكن تقع في Local minimum << اما لو كبيرة
فهي ممكن تخليك تعمل jump من local لل global
minimum وهو ده المطلوب لانه أقل Error

+ لو هي رقم صغير أوي فممكن متلحقش تتعلم يعني فترة
التدريب تنتهي وانت لسه موصلتش لتحت ... اما لو رقم كبير
فممكن متعرفش تقف بالضبط عند أقل نقطة ...

فالموضوع ليه حلول كتير ومشاكل أكثر هنوضحها فيما بعد
بس تفكر هل انك تبدأ ب learning rate كبير وتصغر بعد كده
ممكن يعالج المشكلة ؟؟

Activation function

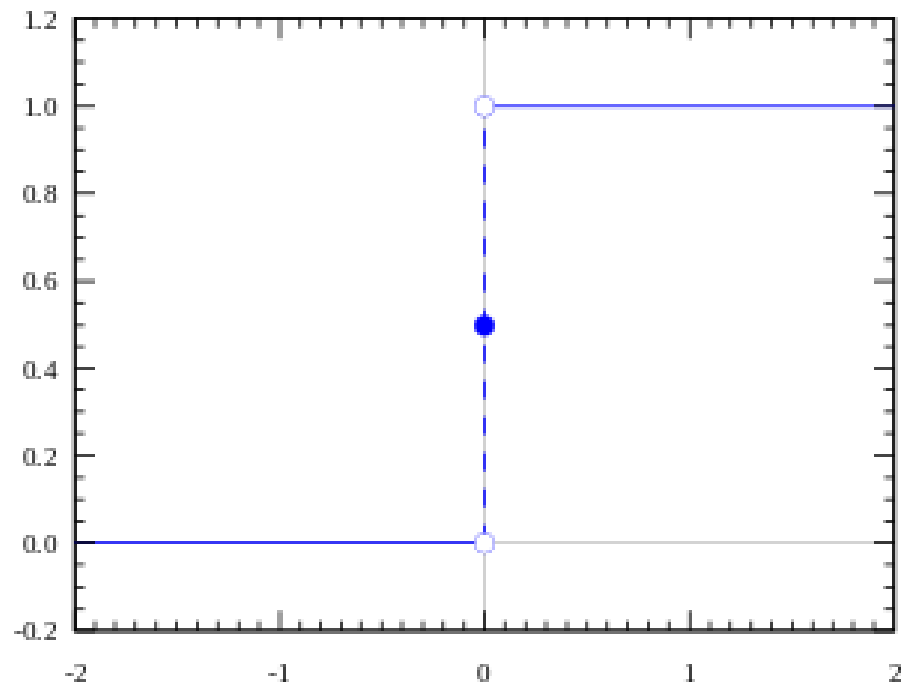
فاضل جزء بسيط اوي هنتكلم عنه وهو لو فاكرين في مخ الإنسان قلنا ان كل Neuron بيبقي متوصل بالاف من Neurons جيرانه يعني ويقدر يتحكم في firing بتاعهم .. فده بيديلك فرصة علي التحكم في مين من Neurons يوصل المعلومة ومين لأ

بالنسبة لنا احنا هو مفهوم مشابه لل Filters

Step Function

احنا لما فكرنا كده قلنا نعمله زي فكرة Threshold بمعنى لو مثلا عندك درجات اكثر من طالب .. ففي طلاب جاييين 40% وفي طلاب جاييين 70% وهكذا .. فانت عايز تعمل ايه ؟ تحط درجة نجاح Threshold يعني اللي يبقى اعلي منها يبقى ناجح يعني 1 واللي أقل منها يبقى ساقط يعني 0 يعني في بعض القيم لما تدخل عندك في Activation function هتطلع ب 0 والباقي ب 1

مثال ثاني يعني زي الرسمة اللي تحت ان لو X input أكبر من الصفر هيطلع الناتج 1 ولو ال X input اقل من 0 هيطلع الناتج 0 بص كويس علي X axis وعلي Y value



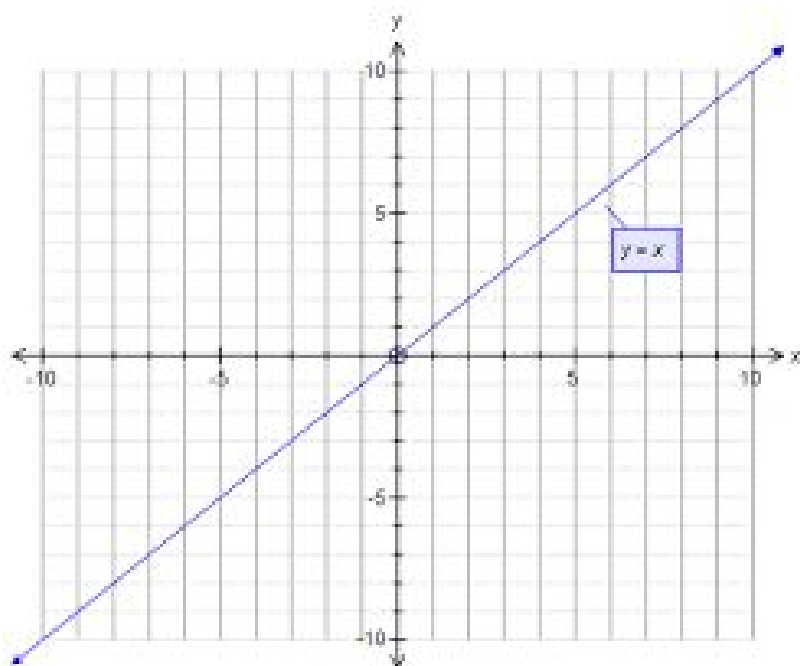
تمام أوي كده انا قدرت اعمل filter وممكن يبقيني مثلا علي output layer ويعمل classification ما بين حاجتين .. بس عيب

حقيقي لل Step Function هي اني مش بقدر أعمل أكثر من Binary Classification يعني Dog/Cat , Male / Female لكن لو عندي أكثر من 2 زي مثلا خمسة Classes ساعتها هيفشل فبدأوا يفكروا في حل ثاني زي Linear Function

Linear function

لان احنا كنا محتاجين Analog output يعني بيختلف معايا فده هيو فرلي احتمالات من $-\infty$ لحد $+\infty$

فانت تقدر تقول ان
هنا كل قيمة لل x
ليها Class مختلف
في y



مثال لو قلنا ان احنا عندنا Classes ليها قيم
Egypt = 0.65 , France 0.9, Germany 0.1

فلو طلع عندك ال 0.5 input ولو افترضنا ان ميل الخط 45 درجة
هنقول كده 0.5 output برده يبقى لو قربنا الرقم لأي Class
هيطلع عندي الناتج Egypt

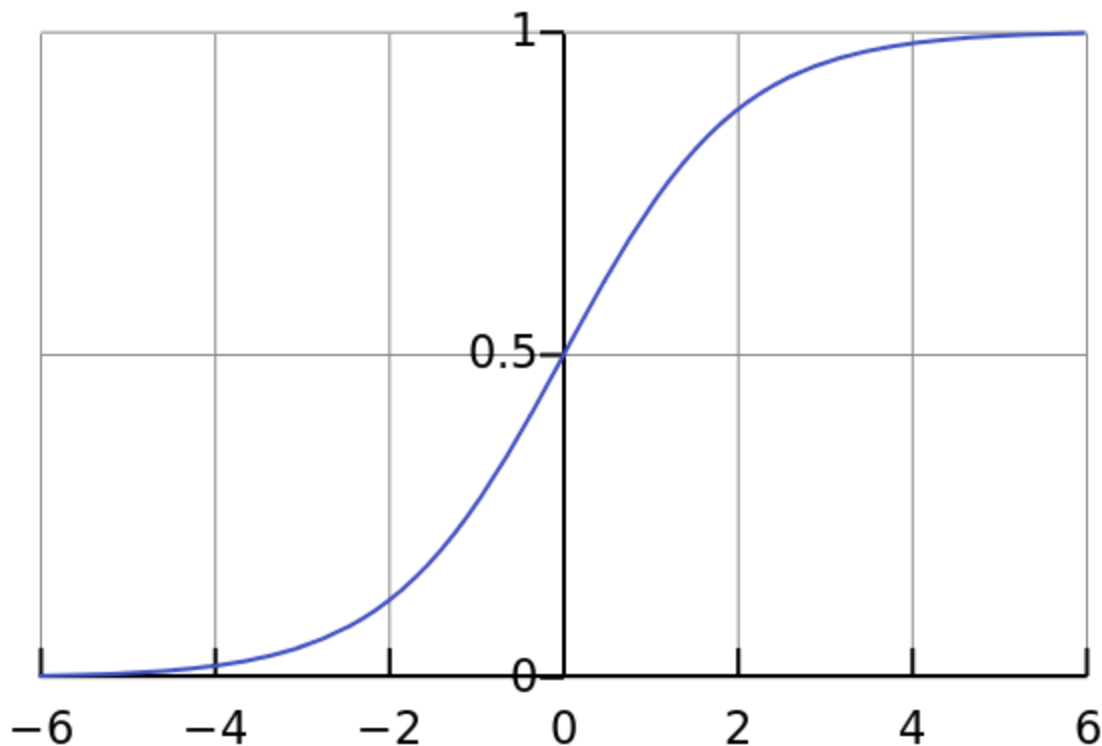
بس ايه العيب ؟

العيب الحقيقي ظهر لما كنا نتكلم عن Gradient Descent ..
فهو في الحقيقة هي عبارة عن Derivative لل Activation
Function فطبعا لو عملت اشتقاق للخط المستقيم الناتج
هيبي ايه ؟ Constant << وبالتالي مفيش حاجة هتعرفني
اتجاه اقل error فين ...

فبدأنا نفكر في نوع ثاني يحل كل المشاكل اللي فاتت دي وهو

Sigmoid function

فهو دي بالنسبة لنا الحل نشوف شكلها كده



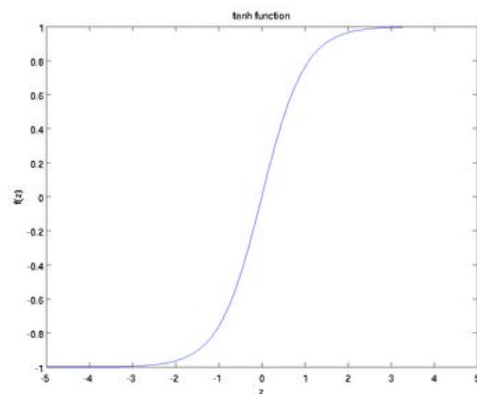
بتوفر لنا موضوع threshold في انها بتنتهي ب 0 و 1 يعني ممكن اعمل filter .. كمان بتوفرلي Variance range هتلاقية مايل شوية وده بالنسبة لي Analog يعني كده تمام اقدر اعمل اكثر من Classification ... واخيرا بقي هي non linear فحتي لو عملت ليها derivative هتبقى vector وهو ده اللي انا عايزه ..

بس هي برده محكومة ب Range ان اخرها 0 ل 1 طيب افرض انا عايز حاجة بالسالب ؟

فضهر نوع ثاني وهو

Tanh function

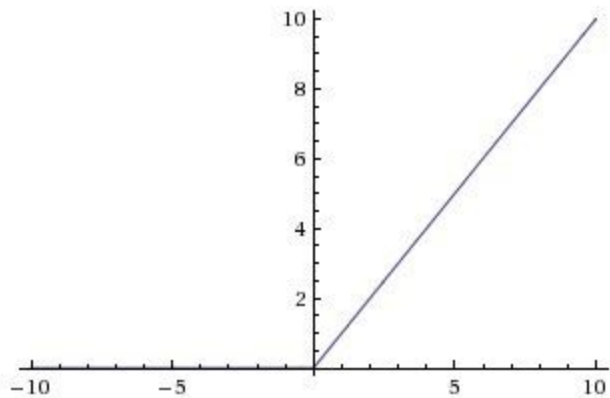
وهي هتوفر لنا Range أوسع من Sigmoid بس ميل هيتلف شوية .. ليها تطبيقات كتير بس مش هنحتاج نخش في تفاصيلها دلوقتي



Relu Function

وهي اخر نوع جابيين نتكلم عنه دلوقتي بياخد ميزة Range ال Linear function وفي نفس الوقت ميزة Filter ودمجهم مع بعض ب Range from Zero to - inf

بالشكل ده طبعا لو فكرت تاخذ اشتقاق الخط الثابت اللي عند zero ده فانت بتعمل اشتقاق ل Constant وطبعا الناتج هيبقى صفر ...



ودي بتوجهنا لمشكلة بنسميها Dying Relu ... هنتكلم عنها فيما بعد بشكل اوضح بس Relu function من أشهر Activation functions المستخدمة

دلوقتي احنا بقينا جاهزين نفهم ايه Process اللي بتحصل عشان Neural Network تتعلم وتوصل ل Solution

وده بيحصل علي مرحلتين

- * Forward Propagation
- * Backward Propagation

Forward Propagation

وهي بالنسبة لنا كل الخطوات اللي بتتم من أول ما بنعمل Random initialization لكل Wiegths لحد مانوصل لل Output مروراً بكل Perceptrons بتاعتنا وال Activation functions والقصة دي كلها

بمعني ان :-

هدف Forward Propagation هو الوصول ل Output بعد سلسلة من العمليات ..

بس هل Output ده صح ؟ يعني هيساوي Actual ؟
أكيد لأ لانه مبني علي Random weights

Backward Propagation

وهي الهدف منها انك تعمل update لل Weights اللي عندك من الآخر للأول يعني بهدف تقليل ال Error وده هيتم أكيد وسط Algorithms من Gradient Descent و parameters Learning Rate ثانية زي

هل بعد ما عملنا Backward propagation كده احنا خلاص يعني جاهزين لو عملنا Forward مرة ثانية ان احنا نوصل للهدف بالضبط ؟

أكيد لأ برده انت بس ممكن تبقي قللت ال error فمحتاج تكرر الموضوع ده مرة و اتنين وتلاتة وهكذا

اللفة المكونة من Forward Propagation + Backward Propagation احنا بنسميها **Epoch**

كل ما بتزود عدد Epochs بيفرق كثير معاك في الوقت والجهد علي Processor ده غير انك برده ممكن تتجه ناحية مشكلة كبيرة أوي وهي **Over fitting**.

لحد هنا ممكن نقول احنا نكتفي بالمعلومات دي بشكل مبدئي في علم Neural Networks وممكن نبدأ نتجه ناحية Coding شوية ونشوف هنقدر نوصل لايه باذن الله

Develop Your First Model With Keras

باذن الله هنشغل Python Programming Language وده نظرا
لأسباب كثيرة جدا منها سهولتها و كثرة Libraries بتاعتها
وانها حصلت علي لقب لغة Deep Learning وده طبعا نظرا
لانها Open Source



و طبعا لكثرة المكتبات
وصعوبة تنزيل بعضها فاحنا
هنتجه باذن الله ل Platform

اسمها Anaconda



هتوفرلنا IDE زي Jupyter
وكمان هتدينا tool ل Library
installation وهي Conda
وكمان هتساعدنا نبني أكثر

ANACONDA®

من Environment بشكل بسيط جدا جدا

Some Installation Tips:-

أول خطوة هي انك تنزل Anaconda سواء انت Linux أو Windows أو Mac

<https://www.anaconda.com/download/>

بعد ما تختلص installation لو انت Windows هتتابع معنا من Anaconda Prompt
أما لو Linux هتتابع معنا من Terminal عادي

لو عايز تتأكد ان كل حاجة نزلت صح أكتب Python في Terminal/Anaconda Prompt

لو ظهر لك

python3.6/ Anaconda-Custom ... etc

اعرف ان انت كده تمام وجاهز انك تبدأ

هنشتغل باذن الله من Jupyter lab

```
meracoda@meracoda-Lenovo-G510: ~  
bash: /opt/ros/lunar/setup.bash: No such file or directory  
meracoda@meracoda-Lenovo-G510:~$ jupyter lab  
[I 13:31:43.616 LabApp] Writing notebook server cookie secret to /run/user/1000/jupyter/notebook_cookie_secret
```

هتختار python3 من Notebook وكده انت جاهز رسميا نبدأ
سوا أول برنامج لينا

Your first Neural Network with Keras (Diabetes Classifier)

-Most of Code are taken from [here-](#)

هنبداً دلوقتى نحاول نعمل كود بيصنف هل العينة اللي قدامه
دي مصابة بمرض السكري ولا لا

ايه الخطوات اللي هنمشي عليها ؟

- 1- Load Data
- 2- Data preprocessing
- 3- Build the Model
- 4- Training
- 5- Prediction

الكود ده مجرد عرض لشكل Neural Network مش الهدف منه
انك تتعلم Coding وتبقي جاهز هو مجرد عرض فقط وتاخذ
فكرة بسيطة عن شكل الكود و Parameters بتاعته

1- Load Data

طبعا عشان تعمل Load لل Data الموضوع بيبقي متوقف علي العميل بتاعك موفرلك Data صيغتها ايه احنا في حالتنا دي متوفرلنا csv file

وهو diabetes.csv .. حمل Data من هنا < [Dataset](#)

أبسط طريقة انك تقرا File ده هي انك تستخدم function اسمها read_csv بس هي مش متوفرة في Python فहतضطر تناديها من library اسمها pandas بالشكل ده

```
import pandas as pd
df = pd.read_csv('diabetes.csv')
df
```

تعالوا نشوف الناتج عامل ازاي ... في الحقيقة قدامنا Table محترم متقسم ل 9 عواميد بالشكل ده .. بس في مشكلة ان Data نفسها جايالنا من غير header فاحنا هنضطر نجيب headers نفسها عشان الدنيا تترتب شوية

The screenshot shows a JupyterLab window with a file named 'Kerasmodel.ipynb'. The output of a code cell is displayed as a table with 12 rows and 11 columns. The first row is the header, and the subsequent rows are data points. The columns represent various features of a dataset, likely the Pima Indians Diabetes Dataset.

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0.0
1	8	183	64	0	0	23.3	0.672	32	1.0
2	1	89	66	23	94	28.1	0.167	21	0.0
3	0	137	40	35	168	43.1	2.288	33	1.0
4	5	116	74	0	0	25.6	0.201	30	0.0
5	3	78	50	32	88	31.0	0.248	26	1.0
6	10	115	0	0	0	35.3	0.134	29	0.0
7	2	197	70	45	543	30.5	0.158	53	1.0
8	8	125	96	0	0	0.0	0.232	54	1.0
9	4	110	92	0	0	37.6	0.191	30	0.0
10	10	168	74	0	0	38.0	0.537	34	1.0
11	10	139	80	0	0	27.1	1.441	57	0.0

وده عن طريق Data Set Details من اللينك ده

Data set Details

فهنعدل في read csv function وهنخليها بالمنظر ده

```
df = pd.read_csv('diabetes.csv', sep=',',
                 names = ["Number of times
pregnant", "Plasma", "blood pressure",
"Triceps skin fold thickness",'insulin','Body
mass','Diabetes pedigree
function','Age','Class variable'])
df
```

Out[3]:

	Number of times pregnant	Plasma	blood pressure	Triceps skin fold thickness	insulin	Body mass	Diabetes pedigree function	Age	Class variable
0	6	148	72	35	0	33.6	0.627	50	1.0
1	1	85	66	29	0	26.6	0.351	31	0.0
2	8	183	64	0	0	23.3	0.672	32	1.0
3	1	89	66	23	94	28.1	0.167	21	0.0
4	0	137	40	35	168	43.1	2.288	33	1.0
5	5	116	74	0	0	25.6	0.201	30	0.0
6	3	78	50	32	88	31.0	0.248	26	1.0
7	10	115	0	0	0	35.3	0.134	29	0.0
8	2	197	70	45	543	30.5	0.158	53	1.0
9	8	125	96	0	0	0.0	0.232	54	1.0
10	4	110	92	0	0	37.6	0.191	30	0.0

2-Data PreProcessing

لحدهنا احنا خلصنا خطوة Loading Data وكمان يعتبر بدأنا في
Pre Processing بخطوة تعديل ال Header

الخطوة اللي جاية دلوقتي ان احنا هحاول نقسم جزء Features
لوحده وهو X وجزء Y لوحده وهو Label

في function عندنا اسمها iloc هتحتاج منك object اللي هو
table /Matrix يعني عشان تبدأ تتحرك فيها وتاخذ جزء
من Matrix وتفصله لوحده لو عايز

فانت دلوقتي عايز تاخذ علي سبيل المثال X
هي أول 8 أعمدة و مثلا اول 760 صف
فهدخله

From 0 : 760 in rows and 0: 8 in columns

اما Y فهدخل

From 0:760 in rows and the 8th column

```
x=df.iloc[0:760,0:8]  
y=df.iloc[0:760,8]
```

تقدر تعمل Check بانك تشوف قيمة X او y ايه هي !

3-Build the Model

فهو في الحقيقة احنا نفترض عايزين نعمل Model مكون من

Input Layer - 8 Input Neurons

First hidden Layer -12 Neurons (Fully connected / Dense)
with relu

Second hidden layer - 8 Neurons (Fully connected/Dense)
with relu

Output layer - 1 Neuron (to tell if its 0 / 1) with Sigmoid

في الحقيقة احنا هنستخدم Keras وهو بيتبع أسلوب بنسبيه Sequential Model نقدر نقول ان احنا بنعمل Sequential Model ونبدأ بعديه نضيف Layer ورا الثانية

```
model = Sequential()
model.add(Dense(12, input_dim=8,
init='uniform', activation='relu'))
model.add(Dense(8, init='uniform',
activation='relu'))
model.add(Dense(1, init='uniform',
activation='sigmoid'))
```

فاحنا ده بالظبط اللي قلنا فوق ماعدا كلمة uniform ممكن نعيدها دلوقتي ونرجعها في Week 2

لكن بالفعل احنا عملنا أول Layer مكونة من 12 نيورون وهي منتظرة من input كام ؟ منتظرة 8 ونوع Activation هو relu وهكذا باقي السطور وطبعاً زي ما قلنا بنعمل Sequential model ونبدأ نضيف Layer ورا الثانية ...

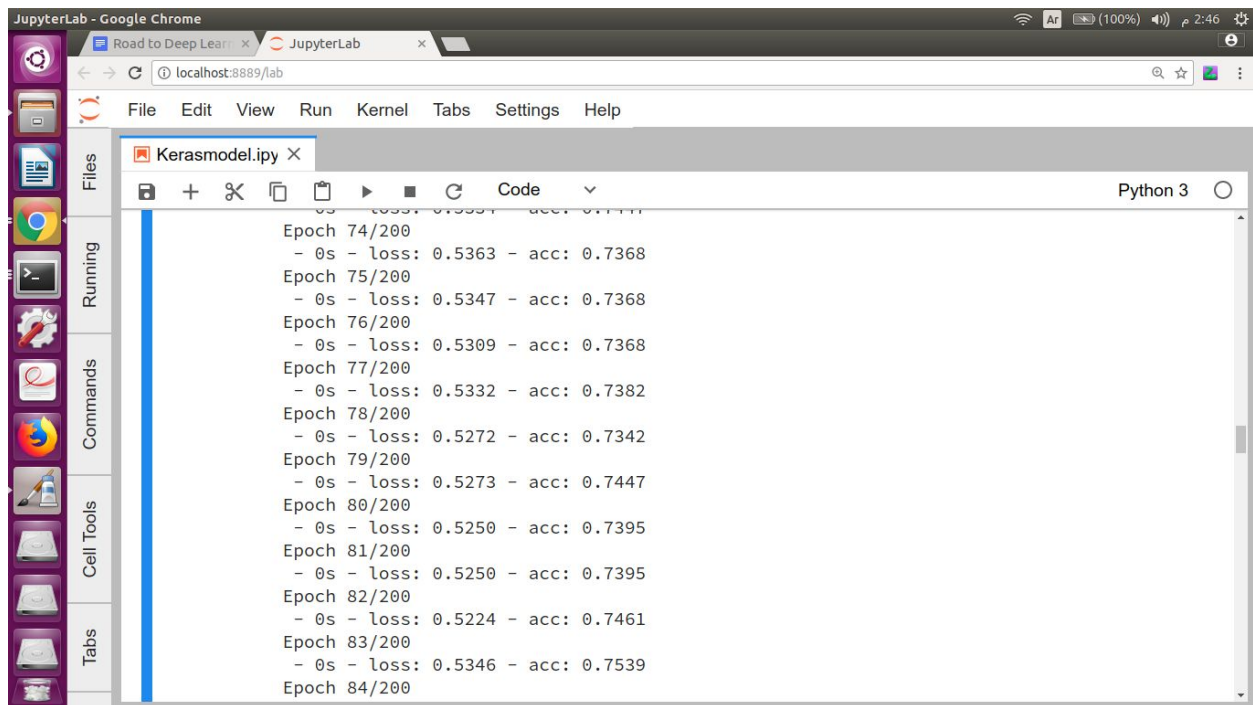
3- Train the Model

دلوقتي هنتنقل لخطوة Training احنا مش interested دلوقتي في كل تفاصيلها اكر من اني بجهر فيها Optimizer بتاعي سواء gradient descent أو غيره وكمان بدخله cost function بتاعتي سواء هي طريقة حساب error العادية او شكل تاني زي binary cross entropy وهنتكلم عنها فيما بعد

وبدخل فيها موضوع Epoch واشوف عايز كام لفة و اعرف ال model شكل X,Y اللي هو features , labels بتاعتي

```
model.compile(loss='binary_crossentropy',  
optimizer='adam', metrics=['accuracy'])  
  
model.fit(x, y, epochs=200, batch_size=32,  
verbose=2)
```

وهيبدأ بالفعل يعمل Training بالشكل اللي تحت ده



```
Epoch 74/200
- 0s - loss: 0.5363 - acc: 0.7368
Epoch 75/200
- 0s - loss: 0.5347 - acc: 0.7368
Epoch 76/200
- 0s - loss: 0.5309 - acc: 0.7368
Epoch 77/200
- 0s - loss: 0.5332 - acc: 0.7382
Epoch 78/200
- 0s - loss: 0.5272 - acc: 0.7342
Epoch 79/200
- 0s - loss: 0.5273 - acc: 0.7447
Epoch 80/200
- 0s - loss: 0.5250 - acc: 0.7395
Epoch 81/200
- 0s - loss: 0.5250 - acc: 0.7395
Epoch 82/200
- 0s - loss: 0.5224 - acc: 0.7461
Epoch 83/200
- 0s - loss: 0.5346 - acc: 0.7539
Epoch 84/200
```

كده فاضلك خطوة اخيرة وهي انك لو حابب تعمل Prediction

4-Prediction

فانت دلوقتي ممكن تاخد أي سطر من Data وابدأ غير في أرقامه ودخلها له في array وشوف هيعمل prediction بكام طبعا لو اقرب لل 0 يبقى هي مريضة لو أقرب لل 1 يبقى هي سليمة

```

Xnew =
numpy.array([[2,82,50,20,0,10.4,0.2,1]])
# make a prediction
ynew = model.predict(Xnew)
# show the inputs and predicted outputs
print("X=%s, Predicted=%s" % (Xnew[0],
ynew[0]))

```

عدل في ال أرقام بتاعت array براحتك وشوف كل مرة
هيطلعلك prediction عامل ازاي
وتقدر تضيف function round للتقريب يعني

```

In [17]: Xnew = numpy.array([[2,82,50,20,0,10.4,0.2,1]])
# make a prediction
ynew = model.predict(Xnew)
# show the inputs and predicted outputs
print("X=%s, Predicted=%s" % (Xnew[0], ynew[0]))

X=[ 2. 82. 50. 20. 0. 10.4 0.2 1. ], Predicted=[0.02140561]

```

لو في أجزاء من الكود مش واضحة ده طبيعي الهدف النهائي
كان أنك بس تشوف الكود بيبقي ماشي ازاي وايه علاقته
باللي شرحناه بشكل عام ...

هنتعلم في Week 2 أو Chapter 2 :-

- ايه هي Deep Neural Networks ؟
- هنتعلم Convolution Neural Networks !
- هنعمل CNN Code on MNIST Data set .
- هنتعلم Tensor Flow .
- و Concepts كتيرة جدا في CNN .
- و اخيرا هنعمل code hardware with Arduino .

END OF WEEK 1

Chapter 2

CNN

- Motivation
- Computer Vision
- DNN
- CNN
- Convolution layer
- Pooling
- FC layer
- Intro to Tensorflow
- CNN with TensorFlow

Motivation

احنا في حياتنا اليومية بنعمل تحليل لاي صورة قدامنا وبنبدأ
نعمل **Predictions** كمان ... طيب مثال علي كده ..



الصورة دي علي سبيل المثال تقدر تقولي انت شايف فيها ايه؟
مممكن تقول ده ولد فرحان وماسك في ايده غطاء بلاسيك و
معجب بالكيكة اللي قدامه و هو قاعد علي كرسي خشب ..

الكلام ده بيبقي بالنسبة لنا عبارة عن ايه ولا وصلنا له ازاي ؟

- احنا قابلنا حاجات كتيرة اوي في حياتنا فبنعتمد علي **Previous experience** بتاعتنا في اننا نحكم علي أي حاجة قدامنا .. فبمجرد ماتشوف مثلا طفل صغير فانت عرفت ان الطراييزة اللي هو قدامها اكيد عالية عليه فهو أكيد دلوقتي قاعد علي كرسي او بينط من علي كرسي ولمجرد بس انك شوفت جزء من اعلي حقة في الكرسي فانت عرفتة بالفعل وحددت كل ده من صورة ... غير طبعا موضوع انك عرفت انه بيحب الكيكة/ التورتة اللي قدامه دي وانه فرحان ... الخ الخ



فلو قلنا ايه ملخص اللي فات ؟ إن احنا عن طريق Previous Experience بنحدد Labels معينة في الصورة وبنبدأ عن

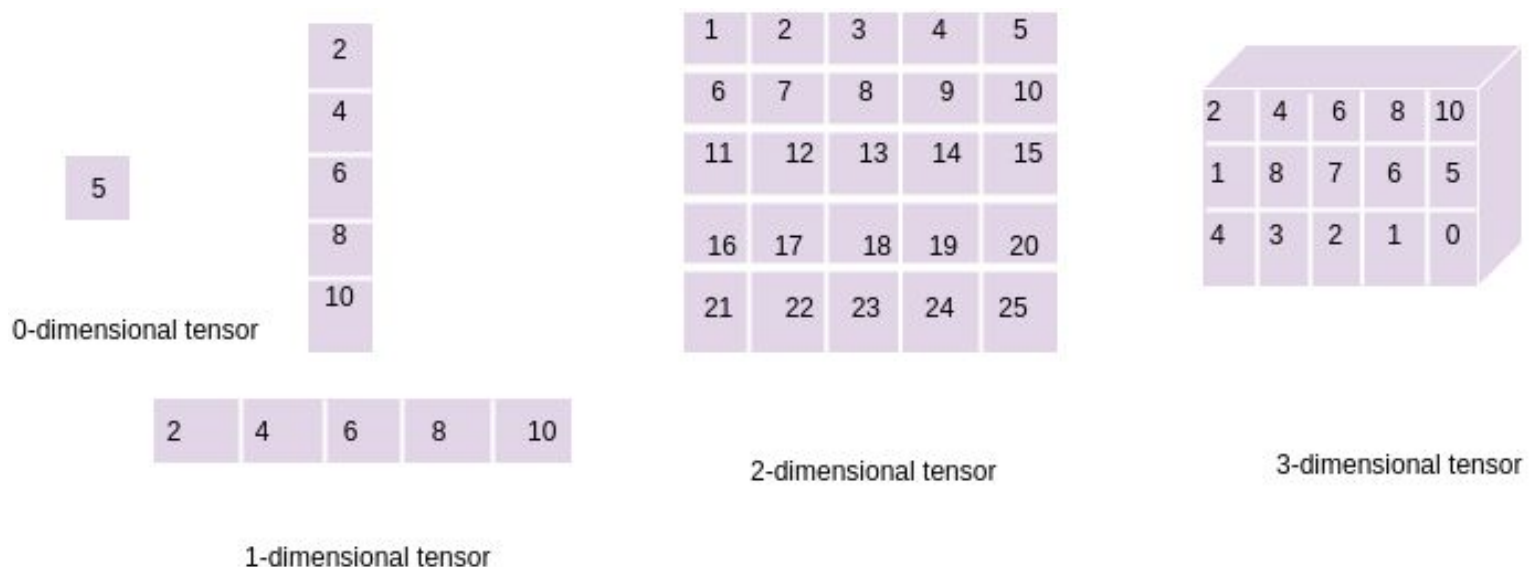
طريقها نستخرج Pattern من الصورة أو معلومة يعني ونقول ان الصورة دي بتتكلم عن كذا وكذا.

Computer Vision

دي رؤيتنا لكل حاجة حولينا ... طيب تفتكر الكمبيوتر بتاعنا بيشف الصورة ازاى ؟

عشان تفهم النقطة دي هتحتاج ترجع لمصطلح احنا بنسميه **Tensor** عارف لما تكون بتتكلم عن data 1 bit ؟ دي احنا بنسميها Zero Dimension Tensor طيب ولو بتتكلم عن

مجموعة من Bits متصلة ببعض ساعات كنا بنسميها Vector لو فاكروا ! احنا بنسميها برده one Dimension Tensor



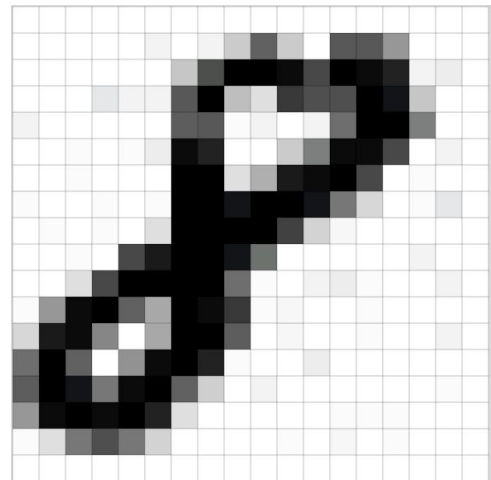
طيب لو بقي عند عدد من Vectors هيكونوا عندي Array او
احنا هنا هنسميهم Two Dimension Tensor ولو عندك

عدد من Arrays هيكونوا Three Dimension Tensor اللي
هي احنا بنسميها احيانا 3d Array

طبعاً تقدر برده تنتقل لل Four Dimension Tensor بنفس الطريقة

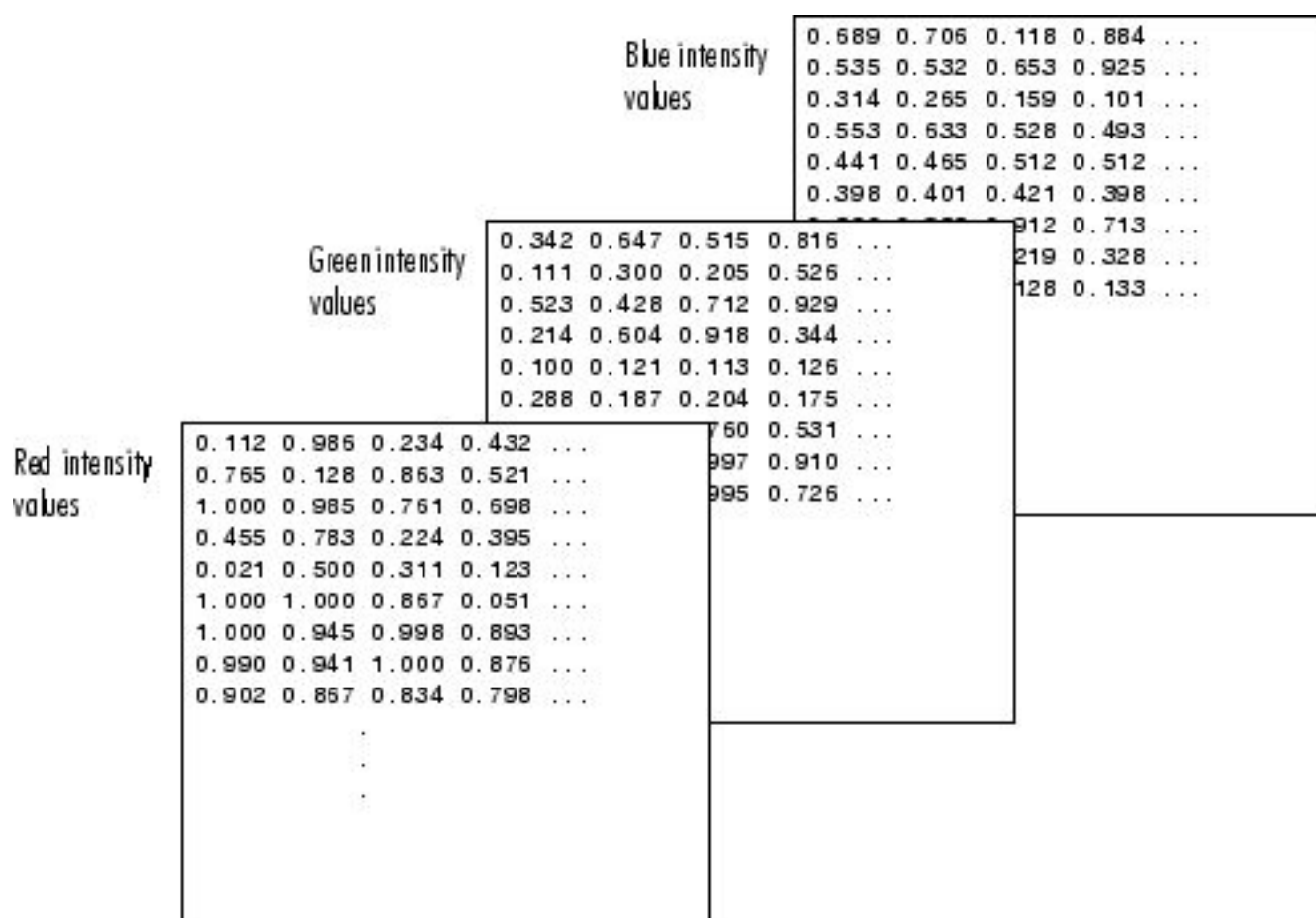
جميل جدا تفكر بقي image هي عبارة عن ايه فيهم ؟ في الحقيقة هي ممكن تكون 2d Array لو هي Gray Scale

يعني لونها رمادي فهي هتبقى ليها width , hight بس
وطبعا كل pixel ليها value

[illegible]

كلمة Gray scale نقصد بيها ان كل pixel عندي هي ما بين الأبيض والاسود في Range من 0 ل 255 تمام ده كلام كويس

في الحقيقة الصورة برده ممكن تبقي 3D Tensor و ده ازاى ؟
 لما تبقي مكونة من Three Channels يعني R G B
 بتبقي حاجة مشابهة ان في Three of 2D Arrays واحدة ورا
 الثانية



طيب في سؤال .. هل ممكن الصورة تبقي 4D ؟ في الحقيقة
 اه وليها حالات كتيرة مثلا ان يكون في Four Channels

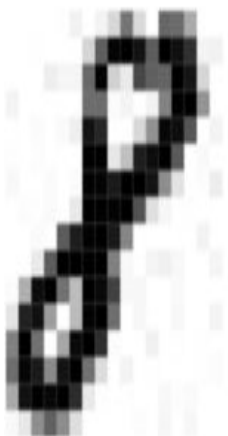
زي RGBA وال A دي بنقصد بيها Alpha channel مسئولة عن Transparency بتاع الصورة اللي هي مقدار الشفافية



او ممكن تكون اصلا الصورة متوزعة علي شكل CMYK زي الطباعة أو ان احنا نكون بنتكلم علي سلسلة من الصور RGB المتكررة ورا بعض ك Frames في Video فيبقى ال 4th Dimension هو Time في حلول كثيرة

بس خلينا نرجع للحل البسيط اوي وهو ان الصورة كانت معناها في شكل Gray Scale اللي هو Two Dimension Tensor

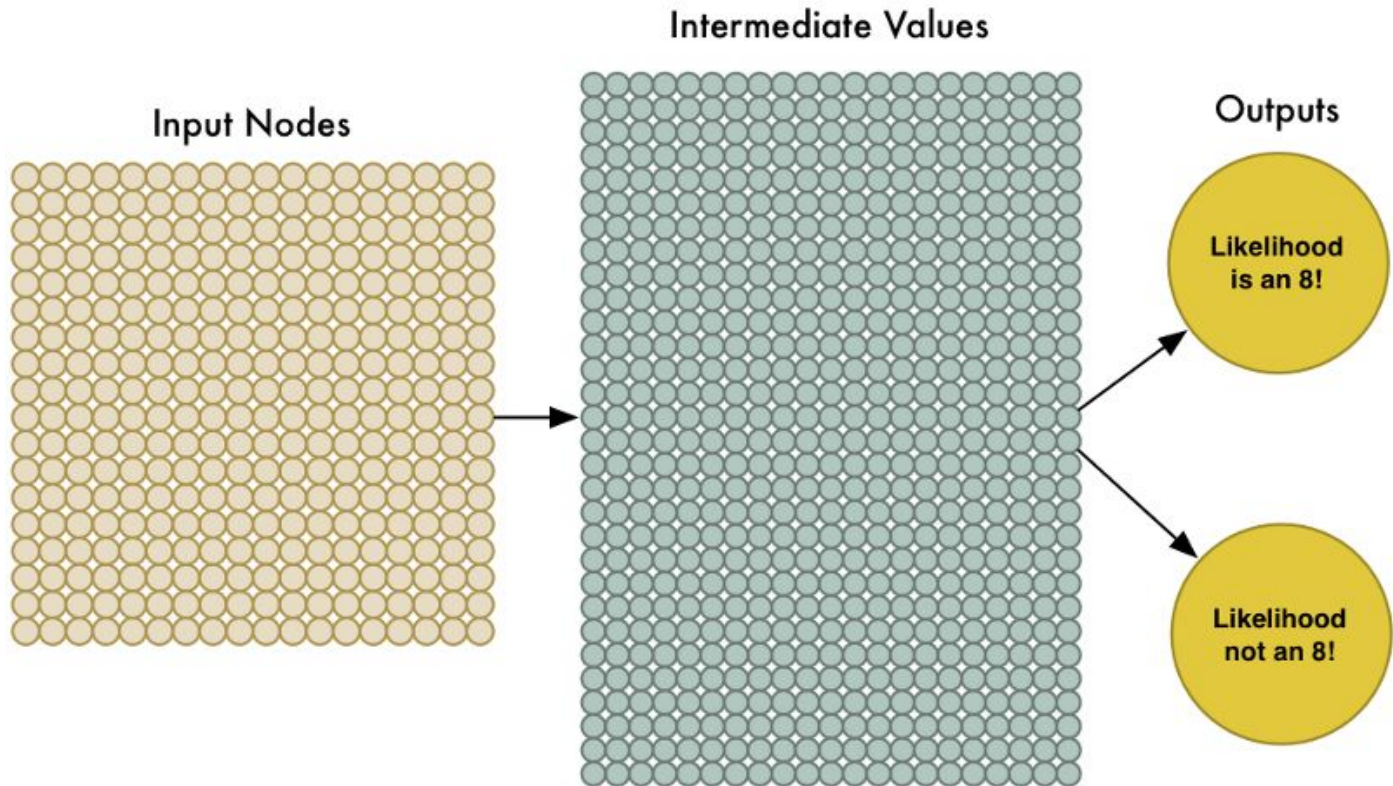
انت متخيل لو الصورة دي لو Size: 18x18 فاحنا هنتعامل معاها ك Array مكون من 324 Number بالشكل ده



=

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 12, 0, 11, 39, 137, 37, 0, 152, 147, 84, 0, 0, 0, 0,
0, 1, 0, 0, 0, 41, 160, 250, 255, 235, 162, 255, 238, 206, 11, 13, 0, 0, 0, 16, 9, 9, 150, 251, 45, 21, 184, 159, 154, 2,
55, 233, 40, 0, 0, 10, 0, 0, 0, 0, 145, 146, 3, 10, 0, 11, 124, 253, 255, 107, 0, 0, 0, 3, 0, 4, 15, 236, 216, 0, 0,
38, 109, 247, 240, 169, 0, 11, 0, 1, 0, 2, 0, 0, 0, 253, 253, 23, 62, 224, 241, 255, 164, 0, 5, 0, 0, 6, 0, 0, 4, 0, 3, 252,
, 250, 228, 255, 255, 234, 112, 28, 0, 2, 17, 0, 0, 2, 1, 4, 0, 21, 255, 253, 251, 255, 172, 31, 8, 0, 1, 0, 0, 0, 0, 4,
0, 163, 225, 251, 255, 229, 120, 0, 0, 0, 0, 11, 0, 0, 0, 0, 21, 162, 255, 255, 254, 255, 126, 6, 0, 10, 14, 6, 0, 0, 9,
, 0, 3, 79, 242, 255, 141, 66, 255, 245, 189, 7, 8, 0, 0, 5, 0, 0, 0, 26, 221, 237, 98, 0, 67, 251, 255, 144, 0, 8, 0, 0,
, 7, 0, 0, 11, 0, 125, 255, 141, 0, 87, 244, 255, 208, 3, 0, 0, 13, 0, 1, 0, 1, 0, 0, 145, 248, 228, 116, 235, 255, 141, 34,
, 0, 11, 0, 1, 0, 0, 0, 1, 3, 0, 85, 237, 253, 246, 255, 210, 21, 1, 0, 1, 0, 0, 6, 2, 4, 0, 0, 0, 6, 23, 112, 157, 114, 32,
, 0, 0, 0, 0, 2, 0, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

فاحنا لو رجعنا لفكرة Neural Networks بتاعتنا فاحنا محتاجين
نكتر عدد Neurons شوية بدل ما كانوا 4 مثلا في input layer
هيزيدوا (((بطريقة ضخمة جدا))) بحيث يقي عدد 324 عشان
كل neuron تشيل قيمة pixel من الصورة



الموضوع كبير أوي صح ؟ بس برده انت جهازك يقدر يتعامل
بسهولة جدا حتي مع Neuron 300 وفي وقت قليل كمان

يعني احنا جاهزين ؟ نعمل Training كده لصور زي دي ؟؟
مبدأيا صورة رقم 8 دي احد الصور المتوفرة في Data

اسمها MNIST DATA SET ودي داتا عبارة عن Hand written Digits يعني أرقام مكتوبة بخط الايد باشكال مختلفة .. الداتا دي Benchmark يعني تقدر تقول عليها انها سليمة ومتوثقة بحيث تنفع تبقي Reference للكود بتاعك Test ليه لو نجحت في انك تعمل Classifier عليها يبقى تمام ... لان ممكن تعمل كود بتاعك علي أي داتا عادية مش Benchmark وساعتها مش هيتم الاعتراف بال Model بتاعك

8 2 7 7 5 7 7 2 8 8 5 7 0 7 1 7 5 9 3 1 0 2 7 9 9 6 9 4 7 4 1 1 4 4 8 8 0 2 6 3
0 0 7 6 3 4 4 4 3 4 2 3 2 8 0 8 2 9 7 6 7 9 0 0 4 2 0 6 6 4 3 3 9 0 4 7 3 2 2 0
2 6 4 6 4 7 5 9 8 7 1 9 0 6 8 7 7 1 9 8 6 5 7 1 0 1 0 8 3 4 7 7 1 3 0 9 6 0 3 8
0 2 8 3 6 5 7 6 6 7 2 6 1 0 2 6 9 7 1 9 5 8 7 0 0 6 1 6 4 4 8 6 2 3 3 1 3 9 9 4
5 1 0 2 1 4 2 2 0 9 9 9 3 1 3 4 1 9 5 5 4 3 9 3 3 5 8 5 0 6 5 1 8 2 6 8 9 2 2 8
4 7 7 7 5 5 0 7 2 2 1 3 5 8 4 8 8 5 2 5 7 1 6 1 8 3 8 0 0 1 0 3 6 7 4 0 8 6 6 2
1 3 3 9 0 4 9 7 5 4 9 5 5 2 6 9 5 3 4 7 3 0 4 6 2 9 4 0 6 2 7 1 0 3 9 1 2 6 0 6
3 4 1 1 9 0 8 2 1 1 9 0 7 5 7 4 2 3 9 9 9 0 2 5 2 1 3 8 3 3 1 6 7 6 0 7 2 0 0 5
7 1 3 1 2 8 8 2 9 4 4 2 4 7 9 8 4 8 0 3 0 7 8 8 3 9 4 7 3 3 1 0 0 8 7 2 1 1 6 2
6 0 1 7 2 3 6 1 6 5 0 7 8 7 8 6 9 2 3 8 8 6 5 1 1 3 2 6 0 6 0 5 9 9 1 0 2 2 1 9

MNIST Data Set

المهم أول ماتبدأ تدخل ال Data فعلا علي Neural Network بتاعتك دي هتبدأ تشوف المشاكل الحقيقية ...

أول مشكلة ان انت لو بس الرقم اللي في الصورة جه تحت علي اليمين او الشمال مثلا هتلاقي الموديل بتاعك مش بيتعرف عليه .. في الحقيقة ده بسبب

انه اتعود علي ان الرقم في النص فبدأ يدي لل pixels اللي في النص weight عالي أكثر من باقي pixels فبدأ يحمل الأطراف وبالتالي مش هيتعرف علي الرقم المكتوب ده !

Test Image #1



Prediction from our network

No ideal?!

Test Image #2



Prediction from our network

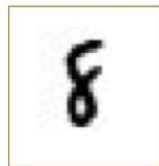
What is this?!@

فبدأنا نفكر في حلول تانية زي ان احنا نعمل مبدأ Sliding window وده زي في ال gif [هنا](#) بتبدأ تمشي ب window صغيرة وت apply ال algorithm بتاعك مع كل step لحد ماتلاقي الرقم مثلا وتتعرف عليه ...

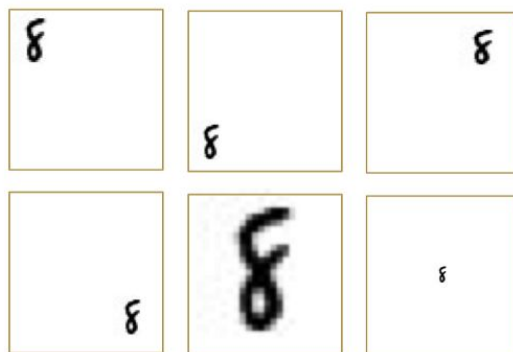
بس طبعا لو الرقم اكبر من Window فبرده هنقع في مشكلة ... فبدأنا نبعد عن الحلول دي كلها ونتجه لفكرة تانية ..

ليه منوريش model الرقم بكل أشكال وهو كبير وهو صغير وهو مقلوب وهو معدول ... فاحتاجنا حجم كبير من Data و شكل أكبر وأعقد من Neural Network

Original Training Image



More training images generated by a script



Welcome to Deep Learning

فبدأ العلم يحتاج احجام اكبر من Neural Networks وهو ده
اللي احنا بنسميه **Deep Neural Networks**

متطلبات DNNs :-

- الداتا تكون حجمها كبير لانه بيعمل بنفسه Feature Extraction
- الداتا تكون Labeled يعني هي عبارة عن inputs ليها Output
- هتحتاج منك Process time كبير غالبا و كمان هتحتاج منك GPU أو Cloud زي ماهنوضح في next week باذن الله

Convolution Neural Networks

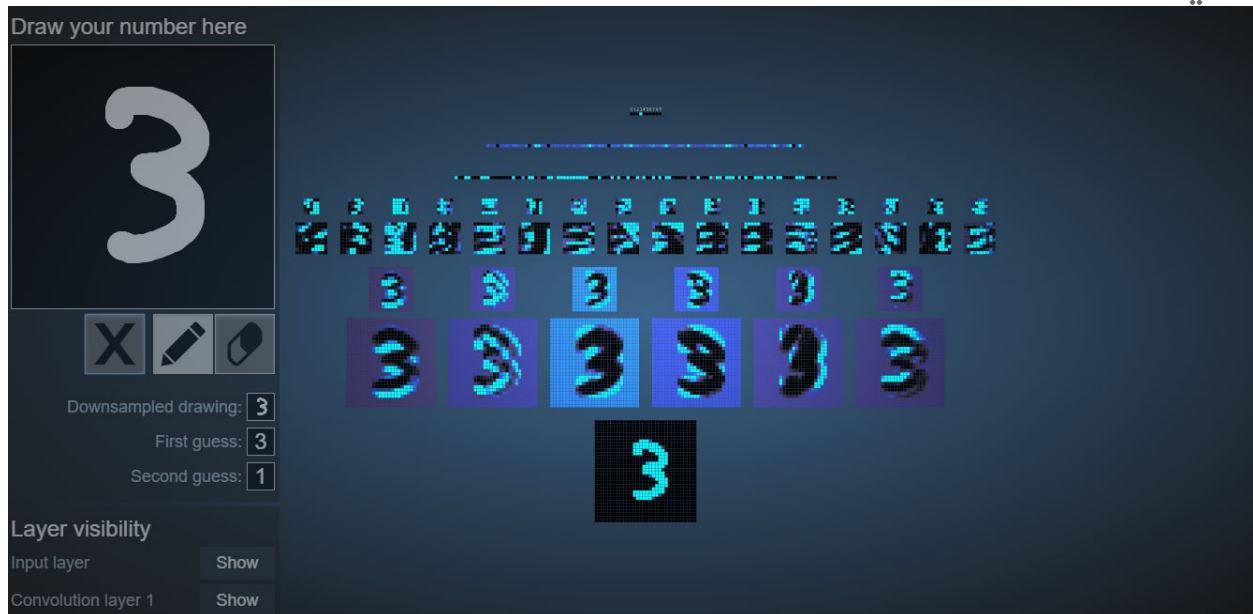
هنبداً بأول نوع وهو CNN ونبدأ نعرف بتكون من ايه

Architecture:-

- Convolution Layer+Relu
- Down sampling Layer
- Flatten Layer
- Dense Layer (FC)
- Softmax

بس قبل مانبدأ بالتكوين محتاجين نفهم كشكل عام اجنا عايزين نوصل لايه ..

[اللينك](#) ده بيوضح ازاي الصورة بتتحول من مرحلة لمرحلة لحد ما CNN تقدر تحدد وتعمل Classification ليها



فهنا بنشوف تحول الرقم 3 من أول input layer وشكله عمال يتغير من مرحلة لمرحلة لحد ماقدام خالص هتلاقية وصل ان model منور رقم 3 والباقي يعتبر اسود

فهو في الحقيقة CNN بتبدأ تبسط الصورة من مرحلة لمرحلة ومع التبسيط ده هتحس ان الصورة بتتشوه بالنسبة لعينك لكن هي في الحقيقة بتبقى أوضح للكمبيوتر لحد مايوصل الصورة لمجرد Vector ليه قيمة يبدأ هو يقربها لل Vectors اللي عنده بالفعل

هنوضح أكثر بعد شرح Layers

Convolution Layer

ودي تعتبر أول خطوة ... لان أول مشكلة بالنسبة لنا هي ان

احنا منعرفش ايه

اللي بيميز الصورة

او ايه هي

Features يعني

فاحنا هحاول

دلوقتي نستخرج






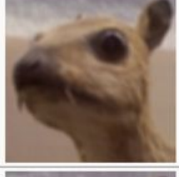
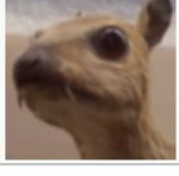
من الصورة أكثر

من معلومة

وعشان اعمل كده

هحتاج Filters

تساعدني

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

في الحقيقة لو

شوفنا تأثير انواع

من Filter علي

الصورة هتلاقي

اختلافات كبيرة

جدا علي حسب

الفلتر نفسه

فاحنا مبدأيا عايزين نفصل ما بين **Input image** , **Filter/window/kernel** , **Activation map/Feature map**
Convolved image اللي هو Result من convolution بين
 الاتنين احنا ممكن نبدأ نشوف ايه العلاقة اللي بتحصل في
 الأول

فلو قلنا ان ال Green Matrix دي هي input image وان
 orange matrix دي هي Filter فال Convolved image هي
 ال pink

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

4		

Image

Convolved
Feature

[Gif link](#)

ايه الملاحظ في الناتج ؟

- ال Convolved image أصغر في الحجم
- أي تعديل في الارقام بداخل filter هتأثر في الناتج
- ان ال filter ماشي بخطوة ثابتة وهي one pixel

الخطوة هي Strides :- ودي نقدر نقول هي مقدار Step اللي بيمشي بيها كل مرة في المثال اللي فوق $Strides = 1$.

لو حببت احافظ علي ان حجم Convolved image = input image ممكن أضيف **Padding** من الاجناب وهو بيبقي عبارة عن اطار من zeros ويمكن يكون عدة اطارات ورا بعض .

في الحقيقة **Convolved image** دي تعتبر معاها Feature عن الصورة يعني معاها معلومة قد تكون مهمة او لأ وقد تكون معقدة ومحتاجة تبسيط برده أو لأ

فهل تفتكر كافي انك تعرف معلومة واحدة ؟ هل كافي انك تستخدم filter واحد؟؟

في الحقيقة لأ احنا بنفضل خاصة في اول Layer ييقي معاك عدد أكبر من Filters وليكن عدد 6 من Filters وبالتالي هيبقي

في 6 من Convolved images كل واحد منهم معاه features معينة .

تعالوا نتكلم بقي دلوقتي عن نقطة مهمة جدا وهي يعني ايه **Depth** كل صورة ليها زي ما تقول Thickness وده بيعتمد علي عدد Channels بتاعت الصورة

- يعني لو صورة Gray Scale يعني One Channel يبقى Depth=1

- ولو صورة RGB Color يعني Three Channel يبقى Depth=3

اما بقي Convolved image فهي ليها over all depth وهو بيساوي عدد convolved images اللي في layer دي أو ممكن نقول بيساوي عدد Filters << نشوف مثال

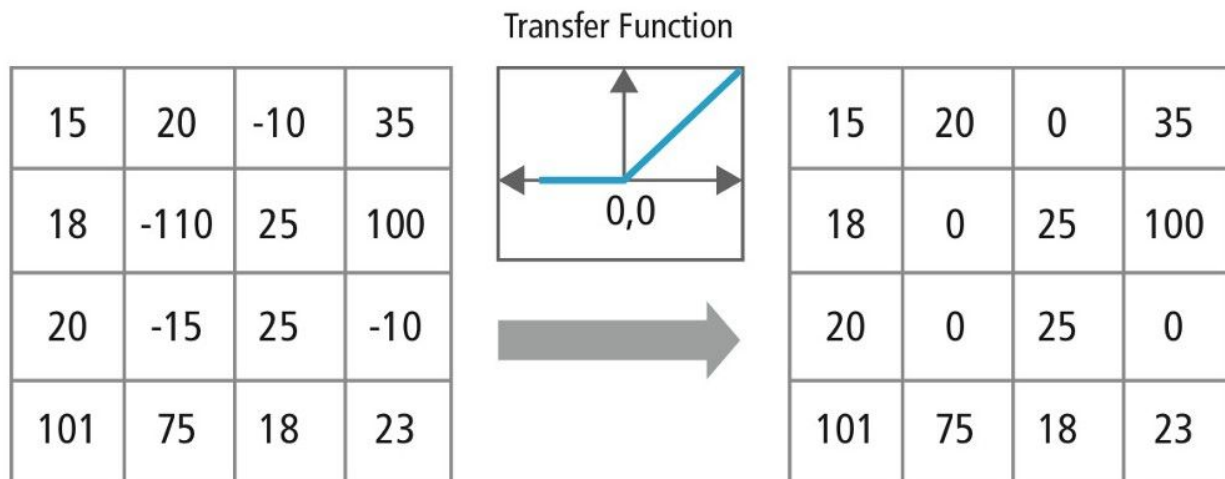
لو معانا صورة 28x28x1 فلو قلنا ان في 32 Filter موجودين في أول Convolution Layer يبقى Depth بتاع Convolved image بكام ؟

الاجابة هي 32

في كام ملاحظة كده عن filter لازم ناخذ بالناس منها :-

- القيم بداخل Filter matrix دي هي weights و بالتالي تقدر تقول انك بتعملها random initialization لانها هتتغير فيما بعد أثناء عملية Training.
- القيم دي ممكن تبقي بالسالب ودي لانها بتفرق في filter لو هو sharpen , edge detector , etc فانت ممكن ينتج عندك في Convolved image يبقا في negative values.

ولان احنا عايزين نتخلص من Negative Values فبنبدأ نطبق Relu Filter على Convolved image ودي بتعتبر آخر خطوة جوة Convolution Layer

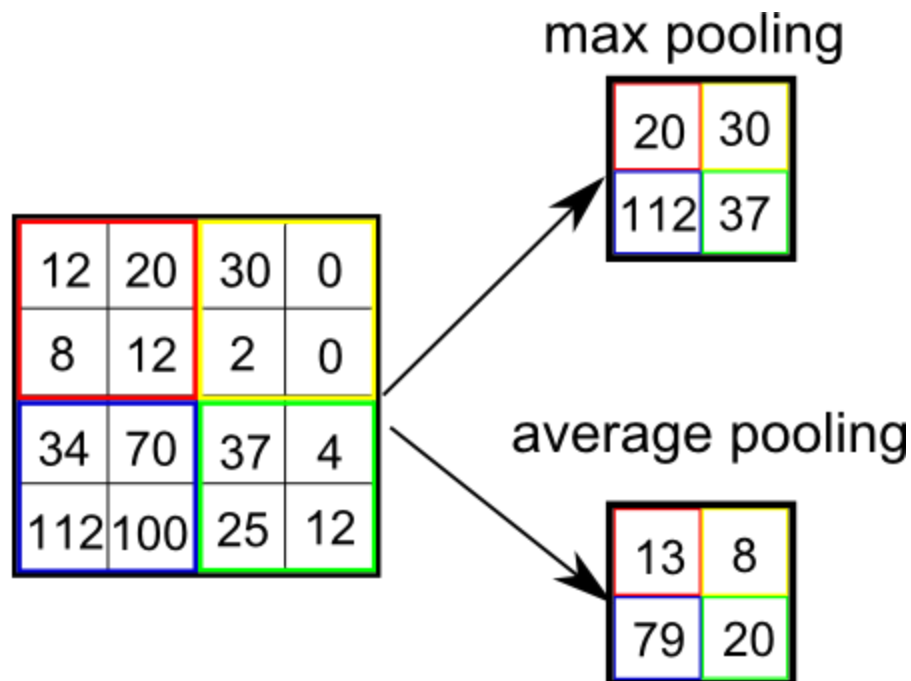


لو حيينا نعمل summary سريع لل Convolution Layer :-

- بيدخلها input image و بنعديها علي عدد من Filters وبينتج عدد من Conv images بنفس عدد Filters.
- ال Size بتاع Conv image بيختلف علي أساس
 - Filter size
 - Padding
 - Strides
- اما Depth بتاع Conv image فهو متوقف علي عدد Filters
- كل Conv image بتبقي معاها معلومة او عدد من features منفصلين عن بعض.
- القيم بداخل Filters ممكن تبقي ب negative نظرا انها Randomly initialized و كمان لانها بتفرق من نوع كل filter
- بنطبق Relu عشان نتخلص من Negative values - وفي هدف ثاني هنعرفه الاسبوع الجاي .-

Downsampling Layer :-

أحيانا بنسُميها Pooling layer وهي الهدف منها انها بتعمل Reduction لحجم الصورة أو كنوع من أنواع التبسيط يعني للكمبيوتر في انه يستخرج اهم المعلومات من الصورة وده بيتم زي الصورة دي كده



لو قلتك ان الصورة اللي فوق دي في window أو Filter يعني حجمه 2x2 وان Strides بتاعته 2 من المفترض انك لو بصيت شوية تفهم ايه الفرق بين اللي فوق واللي تحت

فاحنا نقدر نقول ان مبدأ Downsampling أو Pooling يعتمد علي انك بتنزل مساحة الصورة بشكل كبير جدا وبتعتمد علي Strides , filter size وفي CNN بنفضل استخدام Max Pooling Layer عن Avg .

فكده يعتبر Convolution Layer وصلتني لشوية صور اسمها convolved images كل واحدة منهم معاها معلومة وبعديها عن طريق pooling صغرنا حجم الصورة

طبعا لو الصور دي لسه معقدة يعني فيها تفاصيل كثير ممكن نضيف Conv layer ثانية وراها Pooling layer ثانية و نكرر الموضوع براحتنا ومش هتقدر تعرف العدد optimum غير لما تزود وتجرب Try and Error يعني



Flatten Layer

الملاحظ ان في خطوة مختلفة شوية عن اللي قبلها ان احنا كل الخطوات اللي عدينا عليها كنا في شكل صور 2d Array اما في الخطوة دي انت نزلت لحالة one Dimension Tensor نقدر نقول ان one dimension ده مجمع اهم Features من previous layer ... بس السؤال الأهم ليه بنحتاج Flatten layer ليه بننزل لل one Dimension ...

لان لو فاكرين من Neural Networks كنا عشان نجمع معلومة او نوصل لحل فكنا بنخلي Neurons عندنا Fully connected فانا اكيد هييبقي صعب جدا ومعقد جدا ان اعمل Full

connection مع input layer ولا حتي مع Conv layers بحالتها دي والا هتاخذ مني process time وجهد كبير جدا ...اما ازاي Flatten layer بتتم دي هنعرفها next week باذن الله

Dense Layer (FC Layer):-

وهي دي الخطوة اللي يعتبر بيحصل فيها conclusion لكل اللي فات فاحنا بنربط المعلومات ببعضها وبنحسب كل حاجة بحيث نبقى جاهزين نوصل للخطوة اللي بعدها وهي ان يبغي في Neuron واحد في الآخر اللي هو هيشيل Vector بتاع classification.

Output layer(softmax)

الخطوة اللي فاتت وصلتنا لنقطة كل Neurons الكثيرة جدا دي وصلت في الآخر معايا Neuron واحد بس ... Neuron ده ليه قيمة .. القيمة دي ممكن تبقي علي سبيل المثال 0.68 الرقم ده هيبقي قريب لأحد Classes اللي عندي وعلي أساس هو أقرب لمين هنبداً نعمل لأقرب class ليه Firing والباقي هيبقي أقل وده عن طريق Softmax function

نحاول نفهم Process بشكل نهائي بقي في مرحلة Training:-

1. بنجيب شوية صور ليها Label يعني صور مثلا مختلفة لقطط وكلاب في أشكال مختلفة وانواع مختلفة منهم بس نبقى عارفين طبعا ان دي صورة كلب ودي صورة قطة.
2. نبدأ ندخل صورة صورة .. ولنفترض انها صورة قطة فالقطة دي هتبدأ أول حاجة تعدي علي Conv layer و اول حاجة هتقابلها شوية Filters هينتج عنها Conv images
3. كل Conv image منهم هتبقى معاها معلومات معينة عن القطة .
4. هنطبق بعدها downsampling layer عشان نقلل مساحة ال conv images دي
5. بعد كده هنتحول الصور المنفصلة دي ل vector واحد طويل بيحتوي علي اهم المعلومات عن طريق Flatten layer
6. نبدأ نحلل كل المعلومات دي عن طريق شبكة من Fully Connected layer هتوصلنا كل Neurons ببعض بحيث نجمع كل اللي وصلنا له
7. هنوصل في النهاية ل Neuron واحدة بتحتوي علي قيمة Vector معين ولتكن 0.225 فدي بالنسبة لنا أول قيمة ظهرت للقطة
8. هنكرر نفس ال 7 خطوات دول تاني علي كل صورة عندنا فهيظهر لنا list من القيم للقطة وزياها للكلب علي سبيل

المثال (القطعة : 0.225/0.23 / 0.189/0.21) والكلب)

(0.95/ 0.88 / 0.9/0.87

9. هناخذ Average vector للقطعة مثلا 0.2 وللكلب مثلا 0.8

10. لما نوصل لمرحلة prediction وندخل صورة غريبة

هتعدّي علي نفس الخطوات هتوصل في الآخر منها ل

vector معين قيمته مثلا 0.7 ساعتها هتقول انه كلب وده

عن طريق Softmax function بتاعت اخر layer !

في next week المعلومات هتوضح أكثر عن اهمية Relu
هنتكلم عن ان في filters تانية في النص زي Dropout ونوضح
أكثر حاجات عن Flatten layer بس حتي الان ممكن نبدأ نخش
نشوف الكود هيتكتب ازاى !

Welcome to Tensorflow

لان Keras من أسهل Libraries لأي حد جديد في مجال Deep Learning كان لازم نبدأ بيه في الأول بس من أول ما بدأنا نستعد ان احنا نطور ونتحكم أكثر ونوصل ل Architectures معقدة فكان لازم نتوجه لأسلوب مختلف وده اللي بيقدمهولنا TensorFlow

How to install Tensorflow

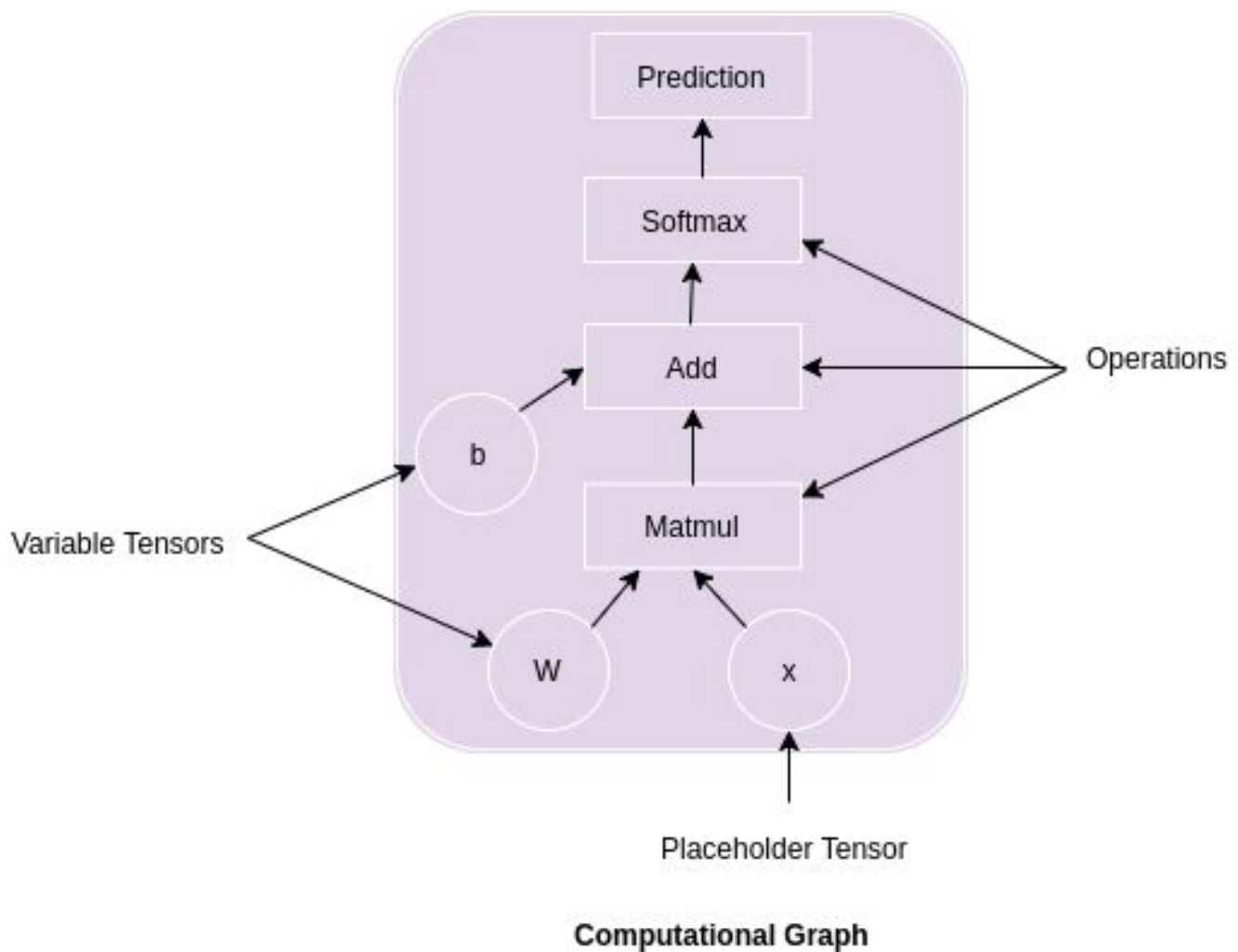
```
conda install tensorflow
```

or/

```
pip install tensorflow
```

Tensorflow Basics :-

أولاً Tensorflow في الحقيقة يتعامل مع كل حاجة واكنها Graph ويبدا عن طريق Graph ده يوصل لحل للمشكلة



Session:-

وده بالنسبة لك الحدث الرئيسي اللي لازم تستدعيه عشان تنفذ الخطوات بتاعتك أو عشان تنادي Variables او عشان تعدل أي تعديل كل ده هيتم داخل session .

Constants :-

ودي بنقصد بيها أي رقم هتخزنه يفضل ثابت معاك طول .process / Session

Variables :-

ودي بقي الأرقام اللي هتتغير معاك زي Weight , Bias وخلافه

Placeholders:-

ودي هي المكان اللي هتدخل فيه Data وتستقبل منه output يعني X , Y

CNN on MNIST Data Set with TensorFlow:-

Code source [here](#)

أول خطوة كالعادة هي ان احنا هحتاج ننزل Data طبعا هي Benchmark Data set فليها شكل مختلف شوية ومش واضح تفاصيلها من جوا لانها متخزنة في Byte File مضغوط تقدر تنزل Fashion Mnist من [هنا](#) :-

بعد ماتفك الضغط وتفتح Jupyter في نفس directory نبدأ بان احنا نستدعي libraries اللي هنستخدمها باذن الله

```
#Import libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.examples.tutorials.mnist
import input_data
%matplotlib inline
import os
```

```
%matplotlib inline
```

السطر ده احنا بنستخدمه عشان graphs اللي هتطلع من
matplotlib تظهر وسط running متبقاش pop up window
برة يعني ك outline

وعلي افتراض انك ماشي معايا بنفس الخطوات فدلوقتي
هنبداً نقرأ Data

```
data =  
input_data.read_data_sets('data/fashion',one_h  
ot=True)
```

لو حابب تتعرف بقي علي شكل الداتا من جوة فمممكن تعمل
print لل Shape بتاعها بالطريقة دي

```
# Shapes of training set  
print("Training set (images) shape:  
{shape}".format(shape=data.train.images.shape))  
print("Training set (labels) shape:  
{shape}".format(shape=data.train.labels.shape))  
  
# Shapes of test set  
print("Test set (images) shape:  
{shape}".format(shape=data.test.images.shape))  
print("Test set (labels) shape:  
{shape}".format(shape=data.test.labels.shape))
```

هتلاقي الناتج بتاعها بالشكل ده

```
Training set (images) shape: (55000, 784)
Training set (labels) shape: (55000, 10)
Test set (images) shape: (10000, 784)
Test set (labels) shape: (10000, 10)
```

فال images نقصد بيها input وال labels نقصد بيها output
55000 نقصد بيها عدد الصور اللي هنتدرب عليها و 784 نقصد
بيها input size اللي هي size بتاع الصورة اللي هو 28x28
784 = اما رقم 10 نقصد بيها عدد labels او Classes اللي عندنا
اللي هما 0 و 1 و 2 و 3....9
رقم 10000 نقصد بيه عدد صور testing اللي هنتبىر فيها
الموديل

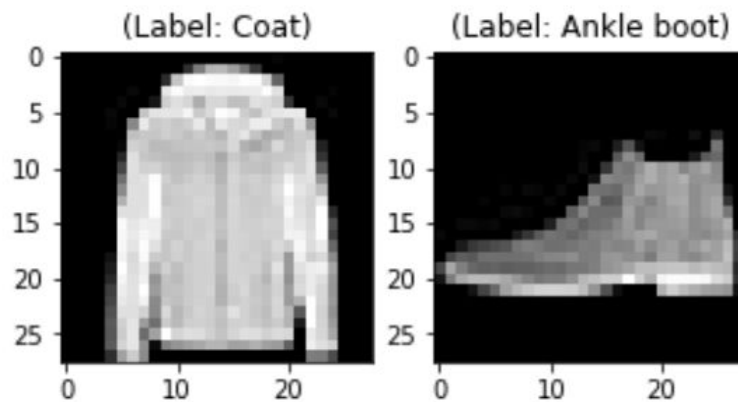
```
# Create dictionary of target classes
label_dict = {
    0: 'T-shirt/top',
    1: 'Trouser',
    2: 'Pullover',
    3: 'Dress',
    4: 'Coat',
    5: 'Sandal',
    6: 'Shirt',
    7: 'Sneaker',
    8: 'Bag',
```

```
9: 'Ankle boot',  
}
```

ولأن احنا بنتعامل مع Fashion MNIST Data set فاحنا بدل ما عايزين نصنف أرقام عايزين نصنف Clothes بشكل عام فعملنا Dictionary لكل Key من 0 لحد 9 ولى Value المقابلة ليهم.

```
plt.figure(figsize=[5,5])  
  
# Display the first image in training data  
plt.subplot(121)  
curr_img = np.reshape(data.train.images[0],  
                        (28,28))  
curr_lbl = np.argmax(data.train.labels[0,:])  
plt.imshow(curr_img, cmap='gray')  
plt.title("(Label: " +  
str(label_dict[curr_lbl]) + ")")  
  
# Display the first image in testing data  
plt.subplot(122)  
curr_img = np.reshape(data.test.images[0],  
                        (28,28))  
curr_lbl = np.argmax(data.test.labels[0,:])  
plt.imshow(curr_img, cmap='gray')  
plt.title("(Label: " +  
str(label_dict[curr_lbl]) + ")")
```

الجزء ده بقي متخصص اكتر في عرض أي صورة عندك في Data كنوع من انواع التجربة يعني وانك تتأكد ان الداتا اتقرت فعلا وكل حاجة ماشية كويس .. الجزء ده مش بيأثر في الكود وهو يعتبر 2 blocks متكررين ممكن تجرب تعدل في الصور عن طريق تغيير الأرقام دي أو تشيل cmap وتشوف الصورة ألوان وا تتحكم في حجم الصورة عن طريق 28 و 28 المكتوبين وكده



لو حبيت برده تشوف أي صورة لك data view يعني تشوف الصورة كأرقام فانت ممكن تكتب

```
data.train.images[0]
```

تقدر تعدل في 0 وتختار صورة تانية او تعدل في train وتجب صورة من test عادي براحتك يعني او حتي images تخليها labels

```
array([[0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.00784314, 0.0509804 ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.00784314, 0.00392157, 0.      , 0.      ,
        0.5137255 , 0.92549026, 0.909804 , 0.87843144, 0.2901961 ,
        0.      , 0.      , 0.00392157, 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.00392157, 0.      ,
        0.      , 0.      , 0.41960788, 0.9176471 , 0.87843144,
        0.8470589 , 0.8980393 , 0.8980393 , 0.21568629, 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , 0.      ,
        0.00784314, 0.      , 0.      , 0.36078432, 0.8000001 ,
        ..... ,
        0.37647063, 0.6745098 , 0.7686275 , 0.81568635, 0.8705883 ,
        0.85098046, 0.8196079 , 0.7843138 , 0.75294125, 0.64705884,
        0.26666668, 0.      , 0.      , 0.      , 0.      ,
        0.      , 0.      , 0.      , 0.      , ], dtype=float32)
```

```
np.max(data.train.images[0])
```

1.0

```
np.min(data.train.images[0])
```

0.0

ودول بيعرفونا أعلي وأقل قيمة موجودين في الصورة
نبدأ بقي في أول خطوة مهمة معانا وهي خطوة Reshaping

```
# Reshape training and testing image
train_X = data.train.images.reshape(-1, 28, 28, 1)
test_X = data.test.images.reshape(-1, 28, 28, 1)
```

مبدأياً احنا بنعمل reshaping لكل input images سواء / train test
وبنديها اول حاجة 1- ونقصد بيه كل ال batch اللي
موجودة في folder يعني في حالة train السالب واحد ده
بيساوي 55000 اما في حالة test فهو هيساوي 10000
لاحظ ان السالب واحد بنقصد بيه كل اللي موجود في folder
اما 28 و 28 نقصد بيهم أبعاد الصورة و ال 1 نقصد بيها
depth بتاع الصورة اللي هي 1 channel يعني gray scale
فكده يعتبر احنا عملنا Resize لكل الصور اللي عندنا

```
# it's already one hot encoded labels  
train_y = data.train.labels  
test_y = data.test.labels
```

اما labels احنا هنسيبها زي ماهي لان احنا خلاص عملناها
one hot encoded فوق وخليناها عبارة عن list من zeros و
one خاص بال Selection عن طريق one hot = True

```
training_iters = 200 #Epoch  
learning_rate = 0.001 # Alpha  
batch_size = 128
```

هنبداً نجهز hyper parameters بتاعتنا

ونعرف variables لأبعاد الصورة ول output classes

```
# MNIST data input (img shape: 28*28)
n_input = 28

# MNIST total classes (0-9 digits)
n_classes = 10
```

وهنحتاج برده نجهز مكان لل input , output وطبعاً احنا قلنا ان دي وظيفة place holder في Tensor flow

```
#both placeholders are of type float
x = tf.placeholder("float", [None, 28,28,1])
y = tf.placeholder("float", [None, n_classes])
```

فال X خاصة بال input وال Y خاصة بال Output

```
def conv2d(x, W, b, strides=1):
    # Conv2D wrapper, with bias and relu
    activation
    x = tf.nn.conv2d(x, W, strides=[1,
    strides, strides, 1], padding='SAME')
    x = tf.nn.bias_add(x, b)
    return tf.nn.relu(x)

def maxpool2d(x, k=2):
    return tf.nn.max_pool(x, ksize=[1, k, k,
    1], strides=[1, k, k, 1],padding='SAME')
```


نبدأ نبني Layers بنفسنا و طبعا هنلاحظ ان في functions جاهزة أصلا للنقطة دي في TF بس احنا عايزين نجهز Conv layer لما نناديها تطبق conv + Relu و كمان ت add bias فبندخل input image اللي هي X و ندخل Strides , Bias , Weight اما كلمة padding = Same فدي معناها انك هتزود padding بمقدار اللي يخلي conv image قد ال input image

```
weights = {
    'wc1': tf.get_variable('W0', shape=(3,3,1,32),
initializer=tf.contrib.layers.xavier_initializer()),
    'wc2': tf.get_variable('W1', shape=(3,3,32,64),
initializer=tf.contrib.layers.xavier_initializer()),
    'wc3': tf.get_variable('W2', shape=(3,3,64,128),
initializer=tf.contrib.layers.xavier_initializer()),
    'wd1': tf.get_variable('W3', shape=(4*4*128,128),
initializer=tf.contrib.layers.xavier_initializer()),
    'out': tf.get_variable('W6', shape=(128,n_classes),
initializer=tf.contrib.layers.xavier_initializer()),
}
biases = {
    'bc1': tf.get_variable('B0', shape=(32),
initializer=tf.contrib.layers.xavier_initializer()),
    'bc2': tf.get_variable('B1', shape=(64),
initializer=tf.contrib.layers.xavier_initializer()),
    'bc3': tf.get_variable('B2', shape=(128),
initializer=tf.contrib.layers.xavier_initializer()),
    'bd1': tf.get_variable('B3', shape=(128),
initializer=tf.contrib.layers.xavier_initializer()),
    'out': tf.get_variable('B4', shape=(10),
initializer=tf.contrib.layers.xavier_initializer()),
}
```

دلوقتي بقي احنا بنجهر قيم Filters واحجامها وزى ما اتفقنا قبل كده هي تعتبر Weights فمثلا رقم 3, 3 ده نقصد بيه أبعاد Filter ورقم 1 نقصد بيه input image depth وال 32 نقصد بيه output depth بتاع first layer ... ونكمل بعدها ال 32 تبقي input depth للي بعدها وهكذا

Hint : depth = 1 = 1 Channel Gray Scale

Depth = 32 = N.o. Filters = Conv images

نكمل بعدها flow

```
def conv_net(x, weights, biases):

    # here we call the conv2d function we had defined above and pass
    the input image x, weights wc1 and bias bc1.
    conv1 = conv2d(x, weights['wc1'], biases['bc1'])
    # Max Pooling (down-sampling), this chooses the max value from a
    2*2 matrix window and outputs a 14*14 matrix.
    conv1 = maxpool2d(conv1, k=2)

    # Convolution Layer
    # here we call the conv2d function we had defined above and pass
    the input image x, weights wc2 and bias bc2.
    conv2 = conv2d(conv1, weights['wc2'], biases['bc2'])
    # Max Pooling (down-sampling), this chooses the max value from a
    2*2 matrix window and outputs a 7*7 matrix.
    conv2 = maxpool2d(conv2, k=2)

    conv3 = conv2d(conv2, weights['wc3'], biases['bc3'])
    # Max Pooling (down-sampling), this chooses the max value from a
    2*2 matrix window and outputs a 4*4.
    conv3 = maxpool2d(conv3, k=2)

    # Fully connected layer
    # Reshape conv2 output to fit fully connected layer input
```

```

    fc1 = tf.reshape(conv3, [-1,
weights['wd1'].get_shape().as_list()[0]])
    fc1 = tf.add(tf.matmul(fc1, weights['wd1']), biases['bd1'])
    fc1 = tf.nn.relu(fc1)
    # Output, class prediction
    # finally we multiply the fully connected layer with the weights
and add a bias term.
    out = tf.add(tf.matmul(fc1, weights['out']), biases['out'])
    return out

```

ونبدأ نبني model بشكل كامل يعني نرتب layers ورا بعض
وندخل weights وال functions اللي بنيناها فوق نستدعيها
هنا ونرتب Conv layers , pooling لحد مانوصل لل output layer

**** متقلقش من الكود لان السيشن الجاية هتبني انت
الاكواد دي بايدك وهتفهم كل تفاصيلها لكن دلوقتي احنا
لسه بنتعرف علي الفكرة العامة ****

```

pred = conv_net(x, weights, biases)

cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logi
ts=pred, labels=y))

optimizer =
tf.train.AdamOptimizer(learning_rate=learning_rate).minimiz
e(cost)

```

هنا بنبدأ نجهز cost function بتاعتها ونوعها Cross entropy
بنستخدم Optimizer نوعه Adam و بنبدأ ندخل learning rate
و نبقى جاهزين عشان نعمل run لل Session

وكده نبقى وصلنا لأهم خطوة وهي خطوة تجميع كل اللي
فات عندنا وهي خطوة Session run

```
with tf.Session() as sess:
    sess.run(init)
    train_loss = []
    test_loss = []
    train_accuracy = []
    test_accuracy = []
    summary_writer = tf.summary.FileWriter('./Output', sess.graph)
    for i in range(training_iters):
        for batch in range(len(train_X)//batch_size):
            batch_x =
train_X[batch*batch_size:min((batch+1)*batch_size,len(train_X))]
            batch_y =
train_y[batch*batch_size:min((batch+1)*batch_size,len(train_y))]
            # Run optimization op (backprop).
            # Calculate batch loss and accuracy
            opt = sess.run(optimizer, feed_dict={x: batch_x,y:
batch_y})
            loss, acc = sess.run([cost, accuracy], feed_dict={x:
batch_x,y: batch_y})
            print("Iter " + str(i) + ", Loss= " + \
                  "{:.6f}".format(loss) + ", Training Accuracy= "
+ \
                  "{:.5f}".format(acc))
            print("Optimization Finished!")

            # Calculate accuracy for all 10000 mnist test images
            test_acc,valid_loss = sess.run([accuracy,cost], feed_dict={x:
test_X,y : test_y})
```

```
train_loss.append(loss)
test_loss.append(valid_loss)
train_accuracy.append(acc)
test_accuracy.append(test_acc)
print("Testing Accuracy:", "{:.5f}".format(test_acc))
summary_writer.close()
```

هنا هتبدأ عملية training وده عن طريق Sess.run اللي هتدخلها batch size , optimizer , cost function وكل ال hyper parameters بتاعتك طبعا ده والداتا والكلام ده كله معاها وبتبدأ تشوف التطور معاك كل epoch وبكده نبقى وصلنا لى target

Next week :-

- Let's Code (Rush coding for 3 hours learn & learn)
- Let's Go For Embedded Systems
- Further details about Adam , cost , flatten , etc
- RCNN Optional

END OF WEEK 1