ال Artificial Neural Network هى أساس ال Deep learning الى هو إمتداد لل Machine Learning ولكنة more powerful و scalable وده ظهر فى الابليكشن المختلفة الى كان صعب التعامل معاها قبل كده زى تصنيف ملايين الصور.

بداية ال Deep learning كان بتعريفة لل Shallow Network والى هى نقدر نقول بتتكون من عدد 2 و 3 من ال Hidden layers الى هنبتدى نعرف عنها قدام أكتر بعد كده ابتدا يظهر عندى Deep NN أو الى بنسمية Multi Layer Perceptron والى من خلالة بقدر أتعامل مع Data set
معقدة وفيه Patterns كتيرة وبيقدر هو يتعرف على ال Patterns لمختلفة ويؤدى لنتايج كويسة والتعامل فى ال Area دية بقا من خلال مكتبات بتساعدنا نعمل Design ل Arcticture مختلفة وفى نفس الوقت بتوفر لينا APIs جاهزة للإستخدام وتم بنائها من خلال واجهة استخدام زى Keras الى مبنية على
Tensorflow, Theano and microsoft Computational Network Toolkit (cntk).
ال Keras APIs بتوفر كتير من ال Architecture المناسبة لكثير من المشاكل.

From BioLogical To Artificial Neurons

بداية ال ANN ظهر فى بحث فى 1943 عن إزاى ال BioLogical Neurons ممكن تشتغل مع بعضها داخل المخ عشان تعمل عمليات معقدة جدا فى جزء من الثانية وبمرور الزمن ابتدا يظهر ANN مختلفة لكن مكنش فيه اهتمام كبير فى الناحية ديه لانها كانت مكلفة جدا فى حين انى كان فى حاجة زى ال SVM الى تم ظهورة فى 1990 وبيدى نتايج كويسة لكن بعد ظهور العدد الضخم ده من ال Applications والى ادى لظهور Massive data واشكال مختلفة من الداتا ومعظمة Unstructured data غير الموجود وهو ال Strcure data الى بتكون مرتبة فى جداول وغيره بقا عندى نوع جديد من الداتا زى الصور زى ال Text زى ال Sensors of IOT كل ده وغيره ولد أنواع جديدة من الداتا وبكمية كبيرة جدا جدا ومعالجتها بقا مهم جدا وده كله خلانا نرجع لل ANN ولانها طلعت Output كويس مع الداتا الكبيرة ديه فى نفس الوقت الى بقا الكومبيتر قادر انه يعمل عمليات معقدة وبقا فيه Multi core computer مش كده وبس وجود ال Cloud والى خلانى قدر اعمل process لكل حاجه ومش لازم يكون عندى الامكانيات ديه انا ممكن استخدمها مقابل مبلغ صغير عشان اعمل التاسك بتاعى وكمان تغير ال ANN بعد 1990 وانها بقت تؤثر بشكل كبير فى مجالات كتيرة خلى الاتجاة الاكبر بقا ليها لان بقت تؤدى لنتايج مزهلة زى Self-driving car زى ال Chat-bots وغيرة كتير.

Logical Computations with Neurons

فى بداية معالجة ال Biological Neurons الى كان فى سنة 1943 عن طريق McCulloch and Pitts كان هو انى ال output بيكون Active فقط لما عدد من ال Neurons يكون Active ومن خلال ده ممكن احسب اى logical proposition زى:

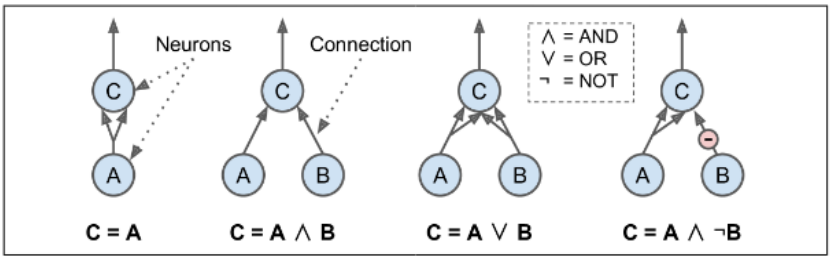


Figure 10-3. ANNs performing simple logical computations

The Perceptron

بدل الإعتماد على موضوع ال Active neurons الى هو off/on ال Perceptron ابتدء يظهر ك Articture مختلف وبسيط من ال ANN والى بقا التعامل فيه من خلال الارقام والارقام ديه بقت هى ال Input features بتاعت الداتا بتاعتك وعن طريق Threshold Logical Unit بنقدر Compute the weighted sum of input ومن ثم بتيجى ال Step function عشان تدينى ال output.
ال Step function ديه كانت بتقول لو z <0 then 0 and z>0 then 1 وكان فى نوع تانى وهو ال Heaviside function وكان نفسها بس لما ال z<0 then 0, z=0 then 0 , z >0 then 1 - 1.

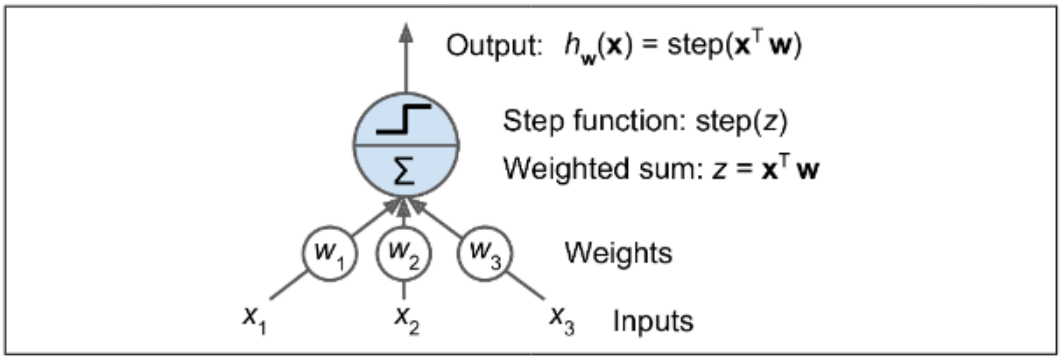


Figure 10-4. Threshold logic unit

طبعا واضح انى ال Simple perceptron ده ممكن يستخدم مع Binary problem فى حالة انى ال output اما 1 او 0 ولكنه بيتكون فقط فى الاخر من layer واحدة من ال TLU ولكن لما يكون فى اكتر من Layer من ال TLU وكل ال Inputs برضه بيروح لكل ال Layers ديه بقا عندى شبكة اعقد وققدر احل مشاكل اكتر وبقا عندى ما يسمى Fully connected layer or Dense Layer وهو ال Connections between neurons in different layers are fully connected.

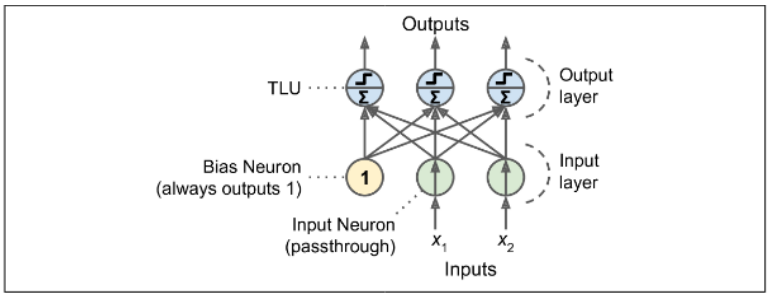


Figure 10-5. Perceptron diagram

الفكرة من ال Perseptron كانت معتمدة على حاجة اسمها Hebbs Rule وهو انى فى ال Biological Neurons لما يحصل Trigger ل Neurons بتؤثر على ال Connection ده ولما اكتر بيعملوا Trigger بيحصل Trigger لل Neurons لمرتبطين بيهم "Cell That Fire together Wire together" .
طبعا كل TLU ليها Computed Wieghted Sum الخاص بيها ولكن ال Linear Algebra خلانى ققدر احسب ده كله فى خطوة واحدة.

Equation 10-2. Computing the outputs of a fully connected layer

$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b})$$

- As always, **X** represents the matrix of input features. It has one row per instance, one column per feature.
- The weight matrix **W** contains all the connection weights except for the ones from the bias neuron. It has one row per input neuron and one column per artificial neuron in the layer.
- The bias vector **b** contains all the connection weights between the bias neuron and the artificial neurons. It has one bias term per artificial neuron.
- The function ϕ is called the *activation function*: when the artificial neurons are TLUs, it is a step function (but we will discuss other activation functions shortly).

لما واحدة من ال Biological neurons بتعمل trigger لوحدة تانية ال Connections بينهم بيكون اقوى
Cell that fire together, wire together
بمعنى بيكون فيه بينهم ارتباط اقوى ده بيسمى Hebbs Rule لكن ال Perceptron مختلف شوية عن كده لانه بيحاول يحسب ال error الى الشبكة بتعملة لما تعمل prediction وهنا ال Perceptron فى learning بيعيد تعزيز ال connections الى بتساعد انها تقلل ال error ده خاصة مع ال wrong prediction بيحاول يعززها مع ال connections الى ساعدت انها تطلع correct predictions.

Equation 10-3. Perceptron learning rule (weight update)

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i$$

- $w_{i,j}$ is the connection weight between the i^{th} input neuron and the j^{th} output neuron.
- x_i is the i^{th} input value of the current training instance.
- $\hat{y}_j$ is the output of the j^{th} output neuron for the current training instance.
- y_j is the target output of the j^{th} output neuron for the current training instance.
- η is the learning rate.

لكن هنا ال Decision boundary of output neuron is linear لذلك مع ال complex problem مش هيقدر يجيب complex pattern لكن لو كانت ال data is linear separale نقدر نلاقى حل كويس .

بعد كده بعض المشاكل الى بتكون معقدة قليلا زى ال XOR ال Perceptron مش بيقدر يحلها وابتدى فى ناس تتجاهل ال Perceptron لكن بعد كده لما حصل combine لاكتر من perceptron architecture مع بعض بقينا نقدر نحل مشاكل معقده وده ما سمى ال Multi layer perceptron .

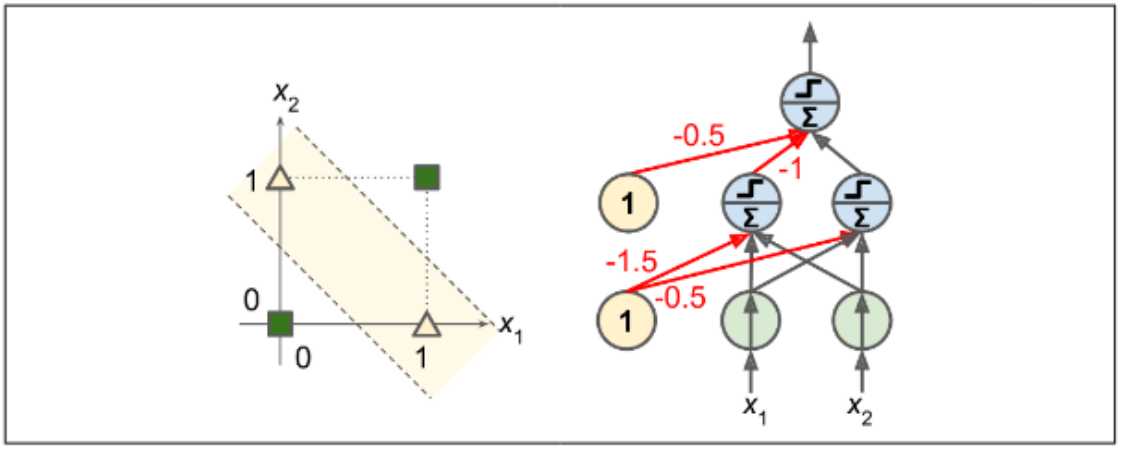


Figure 10-6. XOR classification problem and an MLP that solves it

Multi-Layer Perceptron and Backpropagation

بعد ظهور ال perceptron الى كان بيعمل map علطول من input to output كانه linear model ابتدا ظهور ال multilayer perceptron لكنه كان فقط عبارة عن forward path بعد كده فى 1986 كان Geoffrey Hinton and his friends نشروا Groundbreaking paper الى بتتكلم عن ازاى نعمل Train ل DNN مع Backpropagation بظبط زى ال Grident Decsent بس بدل Forward path بقا فيه Backward path كمان نقدر من خلاله نقلل ال errors من خلال ال Back Propagation انا بقدر احسب ال error الى ال Network عملته مش بس كده هو بيقدر يعرف ال Connections weights and bias for each layer كل منها شارك باية فى ال Errors ديه عشان يقدر يعمل Tweek لل Parameters ديه بطريقة effient.

عملية ال Backprobagation ديه بتبقا عباره عن Chain across all layers وعن طريق الاشتقاق بيقدر يعمل tweeks لل params ديه ويشوف كل منها شارك ب اية بس هنا ال Step function متنفعش لانها ملهاش اشتقاق اصلا لانها Flaten function بمعنى z>0 then 1 z <0 then -1 فبنستخدم activation functions تانية زي Relu or Tanch or Sigmoid.

Regression MLP

ال Activation function فى ال Output تختلف شوية عن المستخدمة فى hidden layers وده بسبب طبيعة ال output نفسه هل هو continuous or discrete فى حالة ال discrete انا بحتاج funtion تعمل لل bound ال output تخلية فى range محدد زى ال sigmoid بين 0 و 1 بينما فى حالة زى ال continuos ال range بتاع ال output غير محدد فى حالة زى كده ممكن استخدم reul فى حالة انى ال output مش negative values.

فى حالة زى ال Regression MLP كمان ال Cost function نفسها بتختلف فممكن استخدم Mean Square Error فى حالى انى مفيش outliers فى ال data بتاعتى لكن لو فيه بستخدم حاجة تكون Less sensitive for outliers زى Mean Absolute Error وفى حاجة بتشتغل مع الاتنين Huber loss.

Classification MLP

هنا بقا زى ما كنا بنقول محتاج ال output يكون bounded فى range معين وهنا برضه تختلف من Binary problem ممكن استخدم معاها Sigmoid او Multi classifcation problem ف استخدم معاها Softmax.

Implementing MLP with Keras

ال Keras فى الاخر عبارة عن APIs زى Sklearn معتمد على Backend Librires زى Tensor Flow and Theano وكمان بقا فيه دلوقت Keras معتمد فقط فى ال Backend على Tensorflow .

بإعتبار انى شغال على image dataset

Sequential Model In Keras

اول model يعتبر فى Keras ابسطهم هو ال Sequntial model لان عبارة عن انك بتعمل Stack of layers وكل Layers ليها ال Paramters بتاعتها.

```
Model = Keras.models.Sequential()
```

كده انا عملت create ل sequential model دلوقت المفروض ابتدى اضيف ليه layers او stack of layers .

```
model.add(Keras.layers.Flatten(input_shape[28, 28]))
```

هنا انا بقولة بكلمة Flaten ديه انتا جايلك عدد غير محدد من ال instances بمعنى مثلا هنا الصور بس كل صورة محتاج تعملها map from 28 * 28 to flatten بمعنى ادق تعمل reshape فتكون 784.

```
model.add(Keras.layers.Dense(300, activation="relu"))
```

هنا بقا انا بقول اعمل Dense layer يعنى fully connected layer بمعنى انى كل nuron فى ال input هيكون متصل بكل ال neruons فى ال hidden layer ديه بمعنى انا عندى فى ال input 784 feture يعنى 784 neuron غير ال x0=1 الى هيكون مرتبط ب Bias or intercept الجزء المقطوع من محور الصادات ^^
طيب نعرف ازاى عدد ال connections weights and bias :
عدد ال neurons الى فى ال layer الى انتا فيها ضرب عدد ال neruons الى فى ال layer الى قبلها + عدد ال Bias unit الى هما عدد ال neurons الى فى ال layer الى انتا فيها :
الى انا فيها فيها 300 والى قبلها فيها 784 يبقا 784*300 + الى عدد ال neurons فى الى انتا فيها يبقا + 300

Weights1+bais1 = 300*784 + 300 = 235500

```
model.add(Keras.layers.Dense(100, activation="relu"))
```

بنفس الطريقة نحسب ال Weights and bais معنى كده الى انا فيها عباره عن 100 neurons والى قبلى كانت 300 لان لو لاحظت عباره عن stack of layer + عدد ال neruns بتاعت ال Bias.

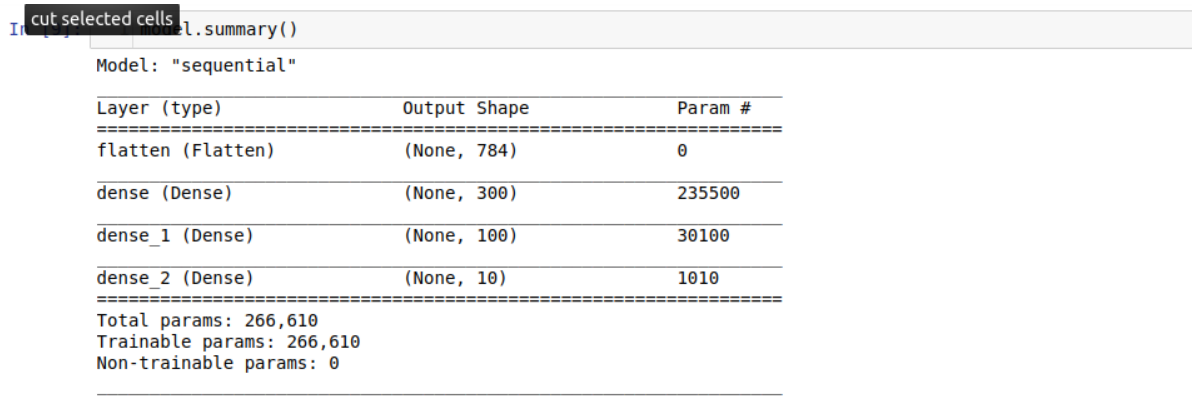
Weights2+bais2 = 100*300 + 100 = 30100

```
model.add(Keras.layers.Dense(10, activation="softmax"))
```

هنا بقا بما انى بتعامل مع dataset of images وال classes المختلفة هما 10 روحت عملت بقا output dense layer of 10 neurons و softmax عشان يكون مجموع الاحتمالات كله 10 وطبعا ال class الى هيتوقع صح هو اعلى احتمال نحسب تانى برضه ال Weights and bias .

Weights3+bais3 = 10*100 + 10 = 1010

لو دلوقت جيت جمعت ال params ديه او روحت قولت model.summary هنشوف النتيجة زى كده



كيراس بيدى اسم لكل layer لو انتا مدتهاش اسم.

Compiling the Model

model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd", metrics="accuracy")

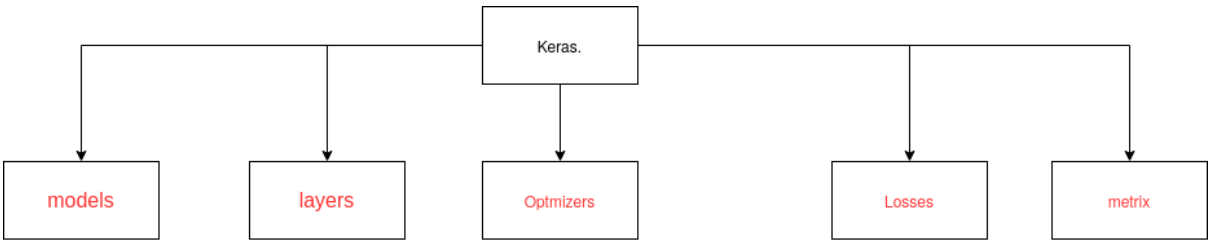
دلوقت ال model بتاعى بقا جاهز انى اعمل compile عشان ققولة انتا هتستخدم انهى optimizer وانتا بتتعلم وانهى cost function عشان تختلف باختلاف ال problem الى انا فيها وال output نفسه بتاع ال problem الاتنين دول هما اهم حاجه وطبعا من غيرهم مش هيرن بعد كده ممكن تحط option params زى ال accuracy .

هنا ال loss function بتختلف فى ال output كمان لو كان عبارة عن one-hot vector بيعبر عن ال class الواحد ولا عباره عن رقم بيعبر عن ال output بمعنى انا عندى 10 كلاس مختلفين هل ال classes ديه من 0 ل 9 ولا كل class عباره عن vector of 10 values وفقط رقم ال index المقابل لل class بيكون 1 والباقى اصفار فهنا هحتاج يكون عندى matrix of 10 * 10 .

كمان وانا بعمل compile لو انا بصيت ال optimizer ك string زى sgd هنا بياخد default value على عكس لو عايز اغير او استخدم range of values ممكن اروح ققوله

keras.optimizer.sgd(lr=)

وهكذا بالنسبه لل losses or metrics او اى params تانية.



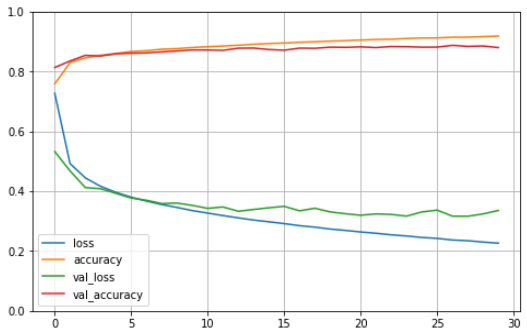
Train & Evaluate Model

history = model.fit(X_train, y_train, epochs=30, validation_data=(X_valid, y_valid))

دلوقت محتاج فقط اعمل train لل model وزى ما بقدر اعمل cross validation هنا بعمل Validation فى اخر كل epochs وال epoch الواحده هى عبارة عن انك بتعمل train على كل ال instances فى الداتا لكن عن طريق SGD بس هو هنا بدام محددش بيعمل instances 32 فى كل مرة لحد ما يخلص كل instances بعد كده يبدء epoch الى بعدها وهكذا.

ال History ده بقا فى كل حاجة خاصة بالموديل مش بس كده لكل layer من الموديل ومن خلاله قدر اجيب weights an bais بتاعت layer محددة وقدر اشوف كل المراحل بتاعت ال model وهو بيعمل train كمان ممكن استخدمه فى انى اشوف ال learning rate بتاع الموديل.

```
In [16]:  1 pd.DataFrame(history.history).plot(figsize=(8,5))
          2 plt.grid(True)
          3 plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
Out[16]: (0.0, 1.0)
```



كمان keras بتوفر class_weight and sample_weight فى حالة انى الداتا بتاعتك فى كلاس ما زيادة كتير عن class تانى فهنا فيه bias ناحية كلاس معين او انى يكون تجميع الداتا نفسة فى بعض ال sampels كان عن طريق ناس خبرة فى المجال والباقى متجمع عادى .

كمان لو جيت عملت fit تانى لنفس الموديل keras بتكمل من عند اخر point هى وقفت فيها بمعنى انها مش بتروح تعمل weights من اول وجديد وتبتدى التعلم من اول وجديد لا بتروح تكمل.

ال Learning rate هو واحد من اهم ال Heper Parameters لذلك اشوف الاول هل هو ممكن يحل المشكلة الى انا فيها زى ال over fitting ولا لا ولو عدلت فى اى hyper parameters تانية يستحسن اروح اعدل ال Learning rate بما يتناسب مع التعديل الى عملته اهم حاجة اشوفة هو الاول قبل اى حاجة.

Using the model to make prediction

دلوقت خلاص انا جاهز انى اتوقع داتا جديدة عن طريق predict لكنها بتجيب ال probability او predicit_classes لى بترجعلى ال output علطول.

Building regression MLP using sequential APIs

الفرق فقط هيكون فى ال output انى بدل ما كانت softmax هتكون relu او اى functions مش بتعمل bound لل output لكن relu مناسبة فى حالة انى مفيش negative classes كمان هى مجرد neuron واحده فقط بتعمل ال output .

ال Sequential model مناسب لبعض ال casses وكمان سهل الاستخدام هو مجرد stack of layers لكن ساعات فى المشاكل اكثر تعقيدا بحتاج architecture مختلف من ال Network لما يكون عندى اكتر من نوع من ال output او عايز ال input نفسه يمشى فى اكتر من طريق فيكون عن multi input.

Building Complex Model Using Functional APIs

واحد من ال Non-Sequential Apis هو ال Functional Apis هو عبارة عن تطبيق لفكرة Wide & Deep Path الى تم نشر بيبر بيها فى 2016 بمعنى ساعات ال model مع ال Deep layers بيحصل فيه تشوة لل Features بتاعت ال Data او انه ميخدش بالة من بعض ال Features فى حاجة زى كده وعن طريق ال Functional APIs انا ممكن اعمل Concat لل input features مع ال output layer ممكن كل ال input او جزء منه وممكن ققسم ال input نفسه على طريقين ال deep path وال wide path .

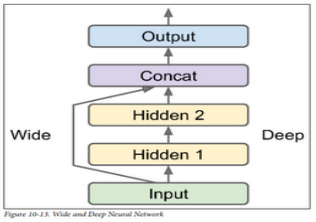


Figure 10-13. Wide and Deep Neural Network

```
Input_ = Keras.layers.Input(shape=X_train.shape[1:])
```

هنا انا بقولة خد كل ال features بتاعت ال input layer

```
Hidden1 = keras.layers.Dense(30, activation='relu')(input_)
```

بظبط كانها function تاخد بعض ال Argument وال Argument هنا هو ال input الى هيمر فى ال Deep path ولذلك بتسمى Functional APIs .

```
Hidden2 = keras.layers.Dense(30, activation='relu')(hidden1)
```

```
Concat = keras.layers.concatenate([input_, hidden2])
```

كده عملت concat لل features الى مرت فى ال deep path مع ال features الاصلية بقا الموديل كانه متعلم من خلال deep path وفى نفس الوقت معاها ال features الاصلية عشان لو فقد بعض المعلومات فى ال deep path .

```
Output = keras.layers.Dense(1)(Concat)
```

```
Model = keras.Model(input=[input_], output=[output])
```

كده انا بقا اخيرا عملت Create لل model وقلتلة انتا هتستخدم ايه ك input وهتستخدم ايه ك output بمعنى انتا هتعمل path لل input ده لما تيجى تتعلم فى ال deep path بتاعك وفى نفس الوقت الى هتتعلمه ده هتعمله concat مع نفس ال input ده تانى عشان ال output بتاعك يكون مبنى عليه .

برضة انا ممكن اروح قسم ال input بتاعى بين الى ماشى فى ال deep path والى هيحصله concat فى ال output وممكن بعض ال features يحصلها overlap بمعنى انها تستخدم فى الاتنين فى ال deep and wide path وده بيسمى multi input . وهنا لازم فى ال fit او ال prediction امشى بنفس النظام.

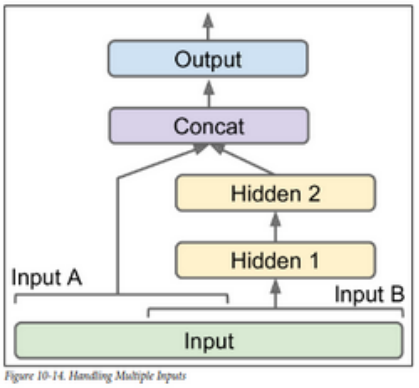


Figure 10-14. Handling Multiple Inputs

```
input_A = keras.layers.Input(shape=[5], name="wide_input")
input_B = keras.layers.Input(shape=[6], name="deep_input")
hidden1 = keras.layers.Dense(30, activation="relu")(input_B)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.concatenate([input_A, hidden2])
output = keras.layers.Dense(1, name='output')(concat)
model = keras.Model(inputs=[input_A, input_B], outputs=[output])

# As I have just 8 features
X_train_A, X_train_B  = X_train[:, :5], X_train[:, 2:] # :5 means 5 features, 2: means 6
features
X_valid_A, X_valid_B  = X_valid[:, :5], X_valid[:, 2:]
X_test_A,  X_test_B   = X_test[:, :5],  X_test[:, 2:]
X_new_A,   X_new_B        = X_test_A[:3], X_test_B[:3]

model.compile(loss="mean_squared_error", optimizer='sgd')

history = model.fit((X_train_A, X_train_B), y_train, epochs=10,
                    validation_data=((X_valid_A, X_valid_B ), y_valid))

mse_test = model.evaluate((X_test_A,  X_test_B), y_test)
y_pred = model.predict((X_new_A,   X_new_B ))
print(mse_test)
print(y_pred)
```

زى ما هو واضح فى كل حاجة لازم اعمل path ل 2 inputs عشان ال Artecture بتاع ال Network لى انا بنيته .

كمان انا ممكن نفس ال output يطلع مرتين ممكن من خلال ال Deep path لوحده ومن خلال انى اعمل Concat لل Wide and deep path وممكن اخلى كل واحد فيهم يعمل output لحاجة معينة زى ال Multi output problem وده بيبقا عبارة عن انى اعرف هل فعلا ال Deep path ده اتعلم شىء مفيد ولا لا من غير اى مساعده عن طريق ال Concat وبيسمى نوع ال output ده Auxiliary output كمان ده ممكن يخلينى اعمل combine بين ال classification and regression من خلال انى واحد فيهم فى الاخر بيعمل bound لل output والتانى شغال على continuous values.

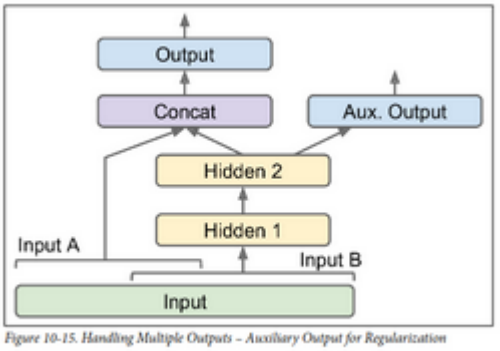


Figure 10-15. Handling Multiple Outputs – Auxiliary Output for Regularization

وبرضة هنا هحتاج وانا بعمل fit or evaluation او اى حاجة بعمل path فيها لل input او ال output انى ققسمهم.

```
input_A = keras.layers.Input(shape=[5], name='wide_input')
input_B = keras.layers.Input(shape=[6], name='deep_input')
hidden1 = keras.layers.Dense(30, activation='relu')(input_B)
hidden2 = keras.layers.Dense(30, activation='relu')(hidden1)
concat  = keras.layers.concatenate([input_A, hidden2])

# All of the above lines as before with multi input idea
output          = keras.layers.Dense(1, name='main_output')(concat)

# The new output aux output and its not complex as the other one and maybe it
#can be used as regularization part
aux_output = keras.layers.Dense(1, name='aux_output')(hidden2)

# Now the model has two output and when compile need output y for each of them
model           = keras.Model(inputs=[input_A, input_B], outputs=[output, aux_output])
```

كمان انا ممكن احدد ال loss لكل output منهم وده هيكون مفيد فى حالة انى شغال مثلا على classification and regression problems

```
model.compile(loss=['mse', 'mse'], loss_weights=[0.9, 0.1], optimizer='sgd')
```

وهنا انا بخلى ال weights مهمه اكتر لل output الى بيعمل concat او على حسب منا عايز.

```
history = model.fit([X_train_A, X_train_B], [y_train, y_train], epochs=20,
        validation_data=([X_valid_A, X_valid_B], [y_valid, y_valid]))
```

هنا زى ما شايف كل حاجة بحددله فيها ال 2 inputs و 2 outputs على اساس ال arcticutre بتاع الشبكة الى عندى.

Using Subclass APIs to build Dynamic Models

ال Sequential and Functional APIs بتوع Keras حاجة Predefined وانتا بتبنى بيها الشكل المناسب بالنسبة ليك بس فى حدود معينة فمثلا فى ال Sequential انتا عارف انك هتبنى Stack of layers وانتا بتختار ال Connections بين ال Layers وبعضها وفقط بتروح تعمل feed لل data وتشتغل علطول ، ده كل بيساعد فى انك تحتفظ بكل ال Weights and hyper parameters بسهولة وتعمل share لل trained model وغيره لان كل حاجه متعرفة او بمعنى اصح static وكمان سهل انك تعمل debug لكن على العكس تماما لما تكون محتاج Dynamic Behavior انتا الى تتحكم فيه عن طريق انك محتاج تبنى Design جديد لشبكة تتعامل مع تجربة جديدة انتا بتحاول تشوفها فهنا تقدر تستخدم ال SubClass عن طريق ال Inheritance من ال Superclass الى معرفاه Keras وتبنى الشبكة الخاصة بيك ولكن ده بيكون طبعا مكلف جدا كمان مش بقدر احتفظ بكل ال hyper parameters وبيكون مكلف جدا لانه Dynamic model فمش سهل انى اعمل Save for weights او اعيد استخدام الموديل وغيرة زى ما كان فى ال Sequential and Functional APIs .

Saving and Restoring the model

بقدر برضه من خلال Keras انى اعمل Save لل model architecture فى كل مراحل ال learning وبقدر احتفظ بكل حاجة اتعلمها سواء parameters or hyper parameters or optimizations or losses كل حاجة استخدمتها الموديل بيحتفظ بيها عن طريق H5 extension بقدر بقا اعمل Save للموديل.

```
model.save('models/first_keras_model.h5')
```

لكن ال deep model وتعاملة مع ال millions of paramters ديه فى ال forward وانى اعمل updates ليها فى ال Backward ممكن ياخد ساعات او ايام وبطبيعة الحال ممكن يحصل عندى مشكلة زى ال Crash بتاع الكومبيتر او النور يقطع مثلا زى عندنا فى مصر وغير فبحتاج انى ال model لما يوصل point معينة يعمل save للى اتعلمة وكده كده لما اجى اعمل fit تانى للموديل هيرجع من النقطة ديه ويكمل كمان ممكن يحتفظ بافضل weights فى كل ال check points الى عملها save او بناء على loss معين وغيرة ودة ما يسمى بال Callbacks .

Using Callbacks

ال call backs بتكون مهمة جدا لأنها بتقدر تحافظ على ال Model بتاعى فى حالة انى الوقت الى بياخدة كبير جدا او حصل عطل او انى محتاج اوقف الموديل لما يوصل ل loss معين وكمان قدر زى ال Subclass بتاع انى اعرف شبكة جديدة قدر استخدم subclass بتاع ال callback عشان اعرف حاجة خاصة بيا للموديل.

الطبيعى لما اجى اعمل save for check points هنا keras بتعمل save لل weights فى نهاية كل epochs وبتحتفظلك بال weights ديه فى نهاية كل epochs لكن ممكن عن طريق Save_best_only احتفظ بأحسن weights جابت validation كويس .
وانا بعمل fit بعمل path لل callback الى عرفتها

```
checkpoints_cb = keras.callbacks.ModelCheckpoint("check_points_model.h5",
save_best_only=True)

history = model.fit(X_train, y_train, epochs=100, validation_data=(X_valid, y_valid),
                    callbacks=[checkpoints_cb])
```

كمان ممكن انى اعرف اكتر من callback

```
earlyStopping_cb = keras.callbacks.EarlyStopping(patience=10,
restore_best_weights=True)
```

هنا انا بقولة من خلال patience=10 لما تلاقى انى ال losses مش بتتغير او يكاد يكون فى اصلا تغير فى 10 epochs ورا بعض بمعنى انى مفيش اى progress بيحصل فى ال learning وقف ال model ومتكملش train زى فى حالة انى مثلا كنت فى local minimum .

```
checkpoints_cb = keras.callbacks.ModelCheckpoint("check_points_model.h5",
save_best_only=True)
earlyStopping_cb = keras.callbacks.EarlyStopping(patience=10,
restore_best_weights=True)
model = keras.models.Sequential([
keras.layers.Dense(30, activation='relu', input_shape=X_train.shape[1:]),
keras.layers.Dense(1)])

model.compile(loss='mean_squared_error', optimizer='sgd')
history = model.fit(X_train, y_train, epochs=100, validation_data=(X_valid, y_valid),
                    callbacks=[checkpoints_cb, earlyStopping_cb])
```

Using Tensorflow for Visualization

انك تشوف المراحل المختلفة للموديل بتاعك عن طريق graphs يوضحلك التاثيرات المختلفة سواء لل hyper parameters او انك تشوف ال losses فى ال training وفى ال validation الى بيخليك تعرف هل فيه overfit وكمان ده مناسب اكتر لان العين بتقدر تلقط الحاجه من الصور سريعا ، كل ال Analysis ده موجود فى ال Tensorboard ولكن عشان استخدمه هحتاج احتفظ بالمراحل المختلفة ديه عشان ال Tensorboard يقدر يستخدمها.

Fine Tune HyperParamters

كمية ال hyper parameters الى بتكون موجودة ال Deep models بتكون كبيرة جدا او حتى فى ال Machine Learning models وبيكون التعامل معاها او انى اعمل Combination بينها صعب جدا وبرغم انى

فى حاجات بتساعد انها تعمل كل ال Combinations المختلفة لل Hyper Parameters ديه الا انه ده مكلف جدا جدا وبياخد وقت رهيب انى اجرب كل ال combinations للقيم المختلفة اصلا لكل params منهم وهنا فيه فكرة انى اعمل زى Zoom out كده الاول عن طريق انى اجرب ranges مختلفة لل hyper params ديه واشوف افضلها هو ايه لكل واحد وبعد كده اعمل zoom in لحد ما اوصل ل ranges معينة ومحددة من خلالها ممكن اعمل بقا ال combinations ديه بس فى range معين .

كمان keras بتوفر ليا Wrapper class ققدر استخدم منه scikit-learn وهنا مهم جدا عشان استخدم حاجه زى ال Grid search او Random search فهنا انتا بتستخدم الاتنين مع بعض يعنى الحاجات الخاصة ب keras هتستخدم عن طريق keras والحاجات الخاصة ب scikit-learn هتستخدم من خلالها.

```
def build_model(n_hidden=1, n_neurons=30, learning_rate=3e-3, input_shape=[8]):
    model = keras.models.Sequential()
    model.add(keras.layers.InputLayer(input_shape=input_shape))
    for layer in range(n_hidden):
    model.add(keras.layers.Dense(n_neurons, activation='relu'))
    model.add(keras.layers.Dense(1))
    model.compile(loss='mse', optimizer=keras.optimizers.SGD(lr=learning_rate))
    return model

keras_reg = keras.wrappers.scikit_learn.KerasRegressor(build_model)
```

دلوقت انا عملت Design لل model بتاعى وبصيته لل Keras wraper الى بيتخدم sklearn دلوقت بقا ققدر استخدم sklearn عادى جدا

```
param_distribs = {"n_hidden": [0, 1, 2, 3],
                  "n_neurons": np.arange(1, 100),
                  "learning_rate": reciprocal(3e-4, 3e-2),}

rnd_search_cv = RandomizedSearchCV(keras_reg, param_distribs, n_iter=5, cv=3)
```

Number of Hidden layer to use

كل ال hyper parameters بتيجى عن طريق انى بجرب وبشوف هل تمام ولا اجرب قيم اخرى وهكذا حاجة زى ال Hidden layers كل ما كانت deep اكتر بمعنى استخدام layers اكتر كل ما هقدر اجيب pattern اكتر ولكن فيه فرق فى ال layers الى فى الاول او ما يسمى بال low-level وانها بتحاول تجيب ال edges بتاعت ال features زى مثلا لو بعمل classification ل faces هتحاول انها تشوف ال horizontal or vertical or diagonal lines وبعد كده فى ال medium layers بتحاول انها تجمع ال lines ديه عشان تبنى اجزء مختلفة من الوجة مثلا بعد كده فى ال high level مع ال output layer بتحاول اننها تتعرف على الوجة وهنا انا حتى لو استخدمت layers زيادة وطلع فيه over fiting هقدر اعالجة مثلا عن طريق ال dropout وغيرة.

Number of neurons in each Hidden layer to use

عدد ال neurons فى ال input وال output بيكون متعرف عن طريق ال problem نفسها لكن مع ال hidden layers بحتاج انى اعرفه فهل استخدم عدد واحد لكل ال layers ولكن ده مش practical على قد استخدام عدد مختلف وكل اما يكون عندى layer اكتر عدد ال neurons قدام بيبتدى يقل عشان يقدر يعمل combine لل patterns مع بعض سواء الى اتعلمتها فى ال low-level او medium level ولكن لو ال input عندى 3-d مروحش اعمل اول hidden layer عباره عن 2 neurons لانى كده هفقد معلومات من الاول اصلا ومش هقدر اجبها حتى لو اصبح عندى 100 من ال layers فلازم اخلى بالى.

كمان فيه فكرة بتاعت انك بتشترى هدوم لو جبتها واسعه هتقدر تضيقها على العكس تماما ف احنا ساعت بنبنى شبكة كبيرة ونبتدى نقلل لما نلاقى مثلا over fitting لكنى فى الاخر هقدر اتعامل مع الشبكة واظبطها .

طبعا ال hyper parameters بقا زى ال learning rate وغيرة زى Batch وغيره بيحتاج منى انى اشوف القيم المناسبة برضه عن طريق تجارب مختلفة .

تم بحمد اللَّه