

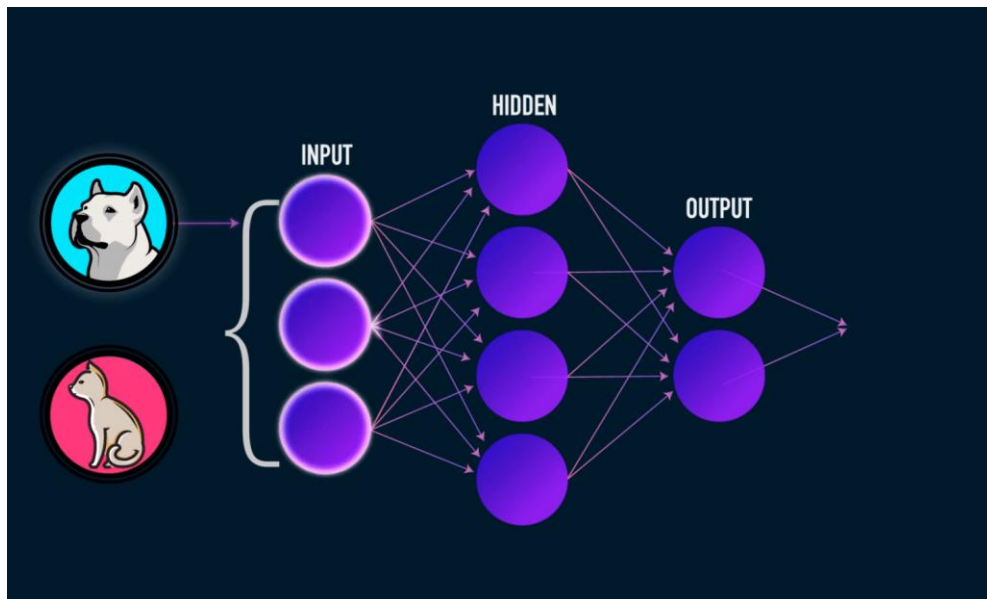
الشبكات العصبية باستخدام Tensorflow و Keras

- خطوات تدريب الشبكة العصبية
- Keras وTensorflow
- تقييم نماذج التعلم العميق
- تحضير البيانات
- تحسين أداء نماذج التعلم العميق
- الإطار البديل PyTorch

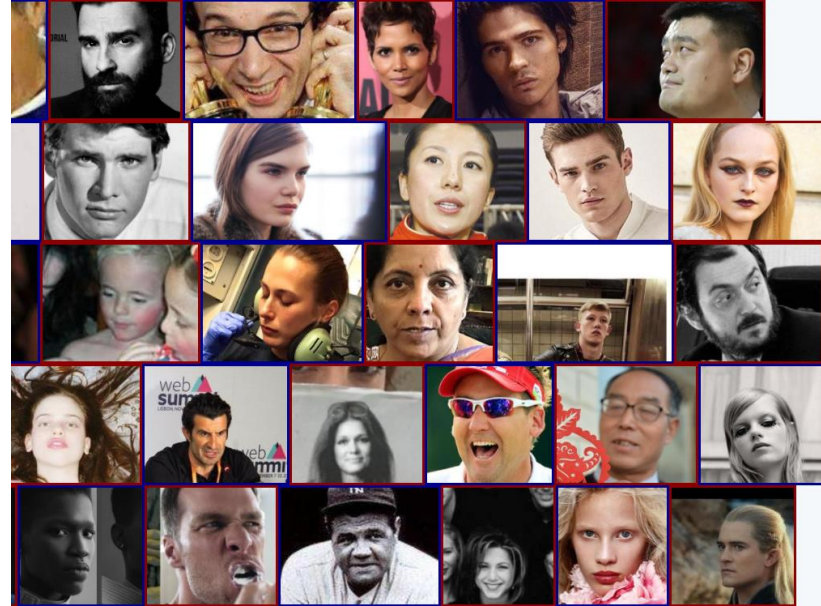
المحتوى

خطوات تدريب شبكات التعلم العميق

خطوات تدريب شبكة التعلم العميق



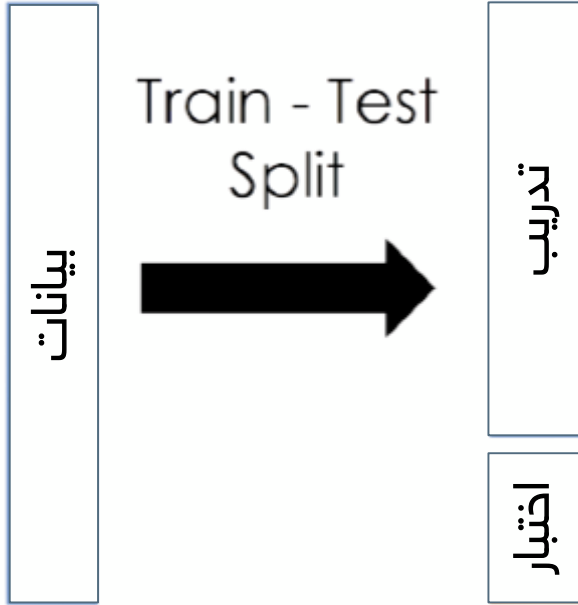
الخطوة 1 : تحصيل مجموعة البيانات



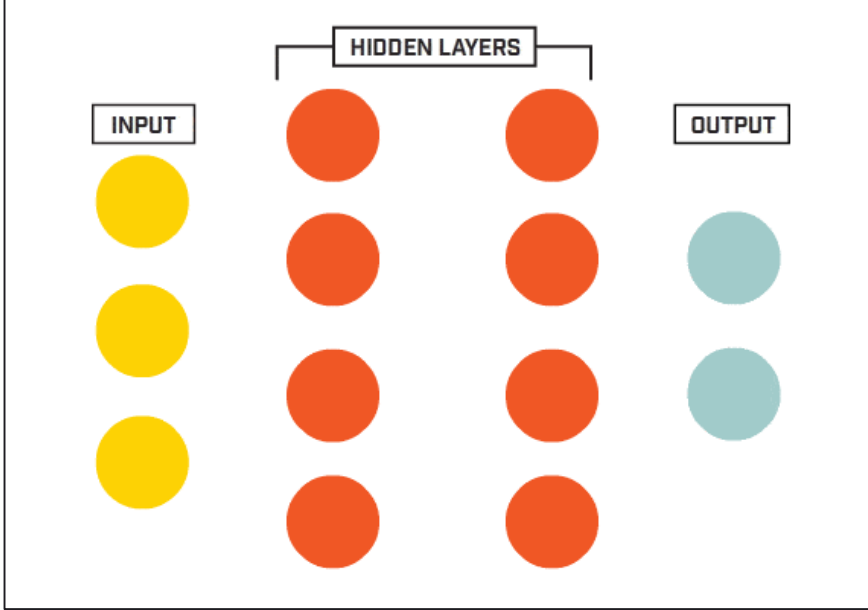
الخطوة 2 :تقسيم مجموعة البيانات

نحن بحاجة إلى الشبكة لتكون قادرة على التعميم.
قسّم مجموعة بيانات التدريب:

- مجموعة التدريب
- مجموعة الاختبار



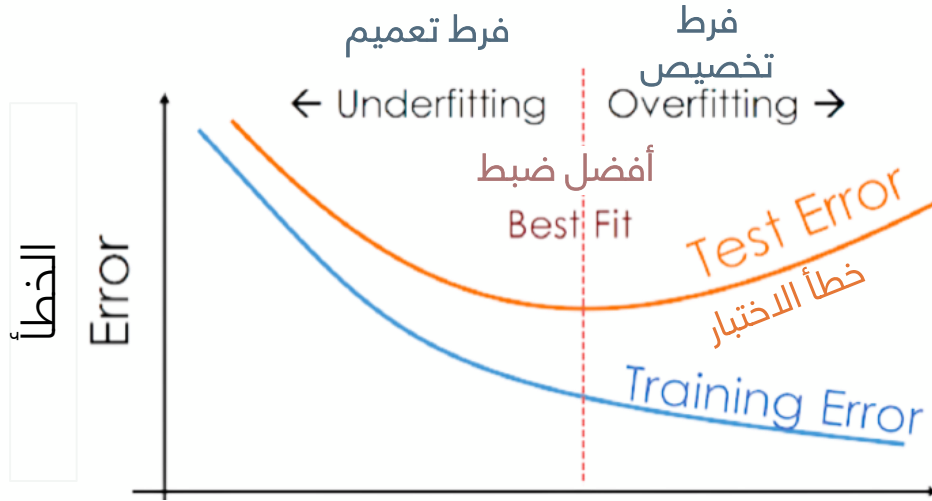
الخطوة 3 :بناء وتدريب الشبكة العصبية



- اختيار إطار / مكتبة التعلم الآلي
- مراحل سير وتحضير البيانات
- بنية النموذج
- معاملات ضبط الشبكة:
- معدل التعلم
- دوال التنشيط
- حجم الحزمة
- عدد الدورات
- ...
- تدريب الشبكة

الخطوة 4 :تقييم النموذج

- فرط التعميم
- فرط التخصيص
- الحالة المثلى لضبط النموذج



خطأ التدريب

الخطوة 5: التعديل عند اللزوم

- تعديل هيكل النموذج (طوبولوجيا الشبكة): إضافة /إزالة طبقات، وإضافة /إزالة الخلايا العصبية، وما إلى ذلك.
- تغيير دوال التنشيط: سيجمويد (sigmoid)، ظل زائدي (tanh)، وحدة خطية مصححة (ReLU)، ...
- تعديل معاملات الضبط (معدل التعلم، الزخم، ...)
- تعديل أحجام الحزم وعدد الدورات.
- تحصيل كمية أكبر من البيانات.
- تدريب النموذج من البداية والتكرار من الخطوة 3.

Abstract geometric shapes in the top corners of the slide, including triangles and polygons in shades of blue, purple, and orange.

Keras 9Tensorflow

TensorFlow



- مكتبة بايثون للحوسبة الرقمية السريعة.
- تم إنشاؤها وإصدارها بواسطة Google.
- مفتوحة المصدر

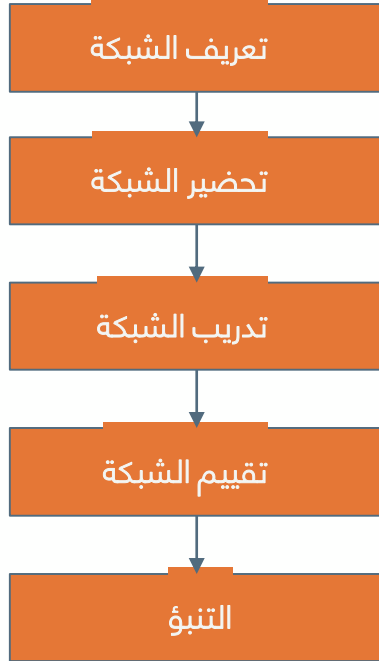
`pip install tensorflow`

Keras



- مكتبة بايثون المبسطة للتعلم العميق.
- يمكن تشغيلها فوق الـ Theano أو الـ TensorFlow (الإصدار الأول)
- تم دمجها رسميا مع الإصدار الثاني من الـ TensorFlow.
- مفتوحة المصدر.

دورة حياة نموذج Keras



فيما يلي نظرة عامة على الخطوات الخمس في دورة حياة نموذج الشبكة العصبية في Keras:

1. تعريف الشبكة Define Network
2. تحضير الشبكة Compile Network
3. تدريب الشبكة Fit Network
4. تقييم الشبكة Evaluate Network
5. استخدام الشبكة للحصول على تنبؤات Make Predictions

الخطوة 1 :تحديد البنية

يجب أن تحدد الطبقة الأولى في الشبكة عدد المدخلات المتوقعة.
دوال التنشيط المختلفة لطبقة الإخراج:

- الانحدار :Regression: دالة تنشيط خطي – عدد الخلايا العصبية لا بد وأن يطابق عدد النواتج.
- التصنيف الثنائي (Binary Classification فئتان): دالة تنشيط سبحمود (sigmoid)، وخلية عصبية واحدة في طبقة الإخراج.
- تصنيف متعدد الفئات (Multiclass Classification أكثر من فئتين): دالة تنشيط سوفتماكس (softmax) وخرج عصبي واحد بقيمة الاحتمال لكل فئة، بحيث ترجح فئة واحدة عن باقي الفئات بأعلى احتمال ومجموع الاحتمالات لكل الفئات لأي إخراج يكون 100%.

الخطوة 2 :تحضير الشبكة

يحوّل التجميع أو التحضير التسلسل البسيط للطبقات التي حددناها إلى سلسلة عالية الكفاءة من تحويلات المصفوفة بتنسيق مخصص ليتم تنفيذه على وحدة GPU أو CPU الخاصة بك.

يتطلب التجميع و التحضير هنا تحديد عدد من المعاملات:

- خوارزمية التحسين optimization algorithm لاستخدامها في تدريب الشبكة.
- دالة الخسارة loss function المستخدمة لتقييم الشبكة أثناء التدريب حيث يتم تصغيرها بواسطة خوارزمية التحسين.

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

الخطوة 2 :تحضير الشبكة

دوال الخسارة المختلفة:

- **الانحدار**: متوسط الخطأ التربيعي (MSE – Mean Squared Error).
- **التصنيف الثنائي**: الخسارة اللوغارتمية، يطلق عليها أيضا الانتروبيا التقاطعية الثنائية (Binary Cross-entropy).
- **التصنيف متعدد الفئات**: الخسارة اللوغارتمية للفئات المتعددة، يطلق عليها أيضا الانتروبيا التقاطعية للفئات (Categorical Cross-entropy).

الخطوة 3 :تدريب الشبكة

يتطلب تدريب الشبكة بيانات التدريب :المدخلات X والمخرجات Y .

`model.fit(X, Y, batch_size=10, epochs=100)`

يتم تدريب الشبكة باستخدام خوارزمية الانتشار العكسي وبها يتم تحسين دالة الخسارة وفقاً لخوارزمية التحسين حيث يتم تحديدهما معاً أثناء تحضير النموذج.

الخطوة 4 :تقييم الشبكة

بعد تدريب النموذج ، نحتاج إلى تقييمه للتحقق من أدائه .يتم ذلك في مجموعة بيانات منفصلة تماما!

$$loss = model.evaluate(X_test, Y_test)$$

الخطوة 5 :مرحلة التنبؤ

predictions = model.predict(X)

- الانحدار :يتم إرجاع النتائج في الشكل الأساسي للمخرجات التي تم التدريب عليها.
- التصنيف الثنائي :التوقع هو احتمالية الفئة الأولى ويمكن تحويلها إلى 1 أو 0 بالتقريب.
- تصنيف متعدد الفئات :مصفوفة من الاحتمالات التي يجب تحويلها إلى تنبؤ معبر عن فئة واحدة باستخدام دالة `argmax`.



Keras: النماذج الدالية مقابل التسلسلية

الاسلوب التسلسلي: (Sequential API)

- إنشاء نماذج بواقع طبقات مكدسة فوق بعضها البعض.
- محدود -لا يمكنه إنتاج نماذج تتشارك وبعضها البعض في الطبقات أو لها طبقات متعددة للإدخال أو الإخراج.

الاسلوب الدالي: (Functional API)

- طريقة بديلة لإنشاء النماذج.
 - توفر مرونة أكثر، وذلك يشتمل على إنشاء نماذج أكثر تعقيدا.
- يتم تعريف النماذج من خلال إنشاء الطبقات وربطها مباشرة ببعضها البعض في أزواج ، ثم إنشاء نموذج يحدد من الطبقات المدخلات والمخرجات.

Keras: النماذج الدالية مقابل التسلسلية

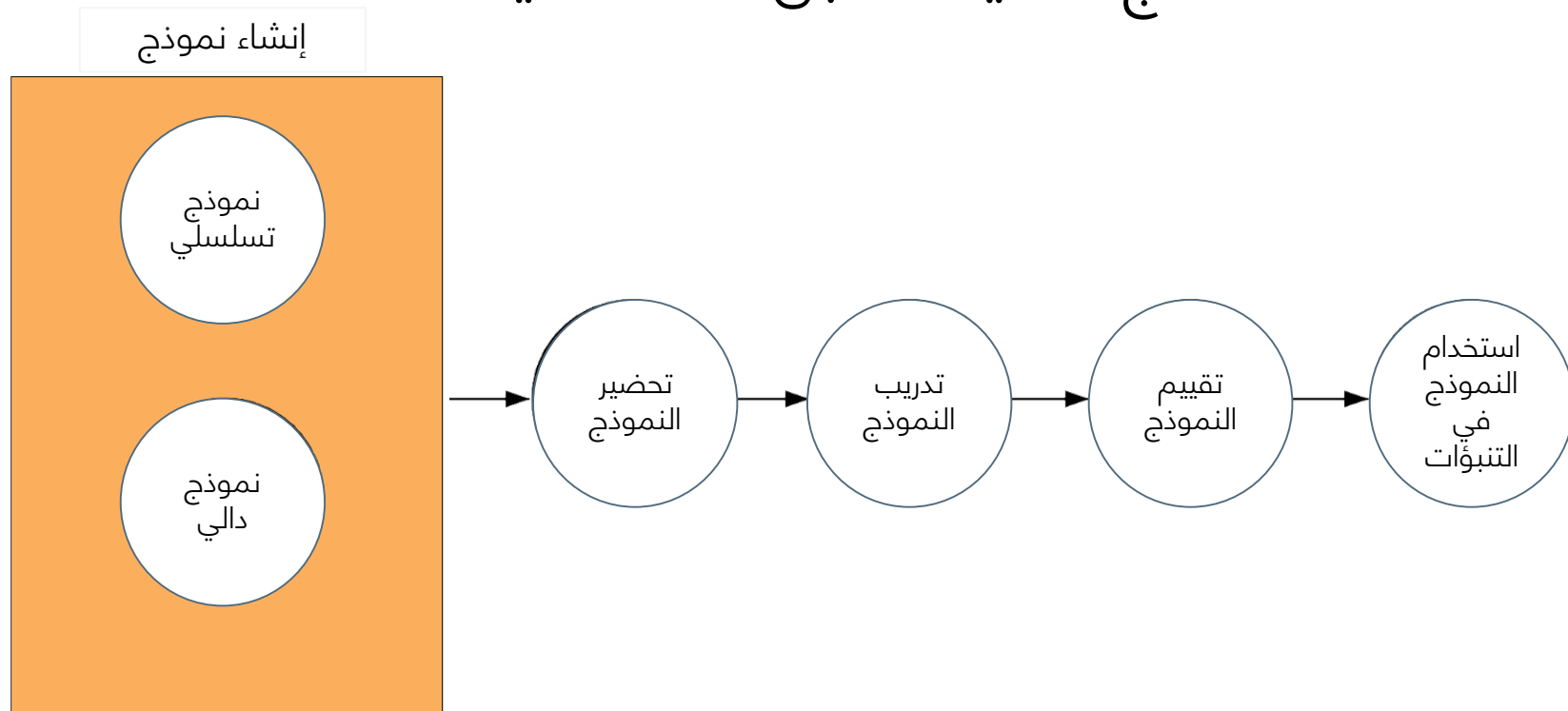
#تسلسلي

```
model = Sequential()  
model.add(Dense(10, input_dim = 3, activation = 'relu'))  
model.add(Dense(10, activation = 'relu'))  
model.add(Dense(1, activation = 'sigmoid'))
```

#دالي

```
visible = Input(shape=(3,))  
hidden1 = Dense(10, activation = 'relu')(visible)  
hidden2 = Dense(10, activation = 'relu')(hidden1)  
outlayer = Dense(1, activation = 'sigmoid')(hidden2)  
model = Model(inputs = visible, outputs = outlayer)
```

Keras: النماذج الدالية مقابل التسلسلية



تدريب عملي:

النموذج التسلسلي مقابل الدالي في Keras
Sequential vs. Functional API with Keras

تقييم نماذج التعلم العميق

تقسيم البيانات

- استخدم مجموعة بيانات للتحقق التلقائي.

تدريب النموذج

```
model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10)
```

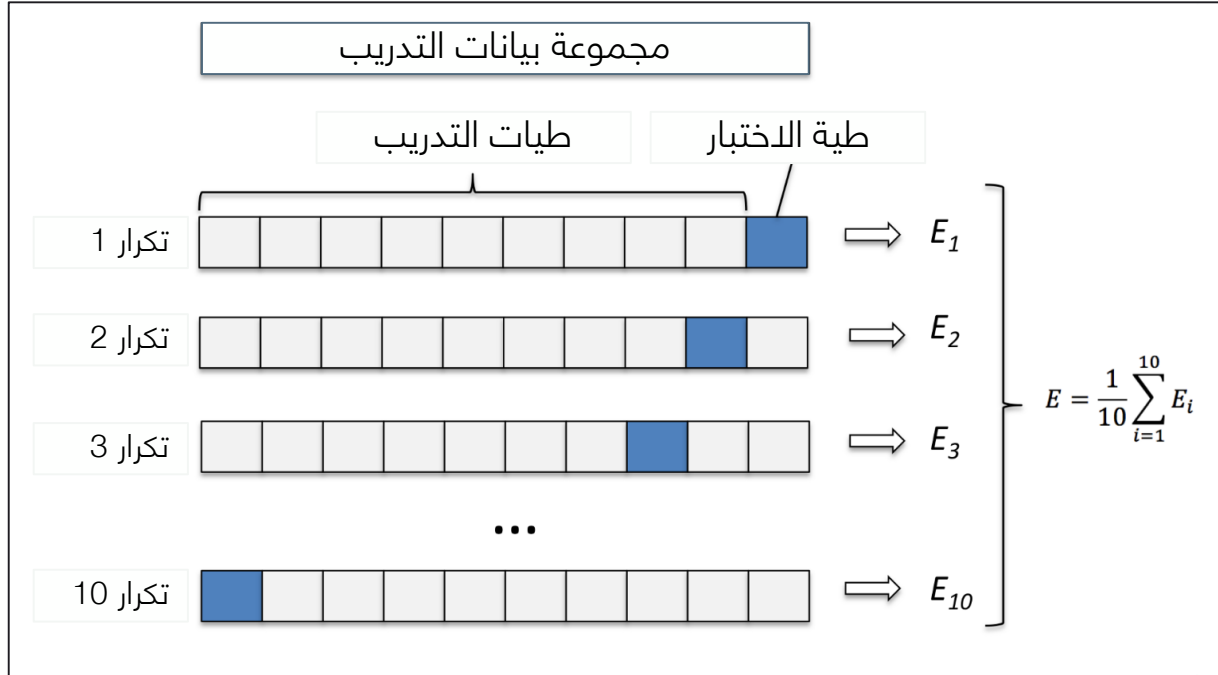
- استخدم مجموعة بيانات للتحقق اليدوي.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=seed)
```

تدريب النموذج

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=150, batch_size=10)
```

التحقق التقاطعي Cross Validation



Keras و Scikit-learn

كيف يمكن تغليف نماذج keras بحيث يتم استخدامها مع مكتبة التعلم الآلي scikit-learn؟

توفر مكتبة keras غلافًا wrapper مناسبًا لنماذج التعلم العميق لاستخدامها كمقدّرات estimators أو انحدار في scikit-learn.

KerasClassifier — KerasRegressor

عليك أن تحدد دالة تنشئ نموذجًا وتقوم بتمريره إلى مغلفات نماذج keras.

التحقق التقاطعي بمكتبة scikit-learn

- قم بتعريف دالة `create_model` التي تنشئ نموذج باستخدام `keras`.
- أنشئ نموذج `KerasClassifier`.

```
model = KerasClassifier(build_fn=create_model, epochs=150, batch_size=10, verbose=0)
```

- استخدم مثيل من `StratifiedKFold` من مكتبة `scikit-learn`.

تعريف مثيل لطى البيانات بعد ترتيبها عشوائيا #

```
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
```

- قم بتقييم النموذج باستخدام التحقق التقاطعي ب 10 طيات لمجموعة البيانات.

```
results = cross_val_score(model, X, Y, cv=kfold)
```

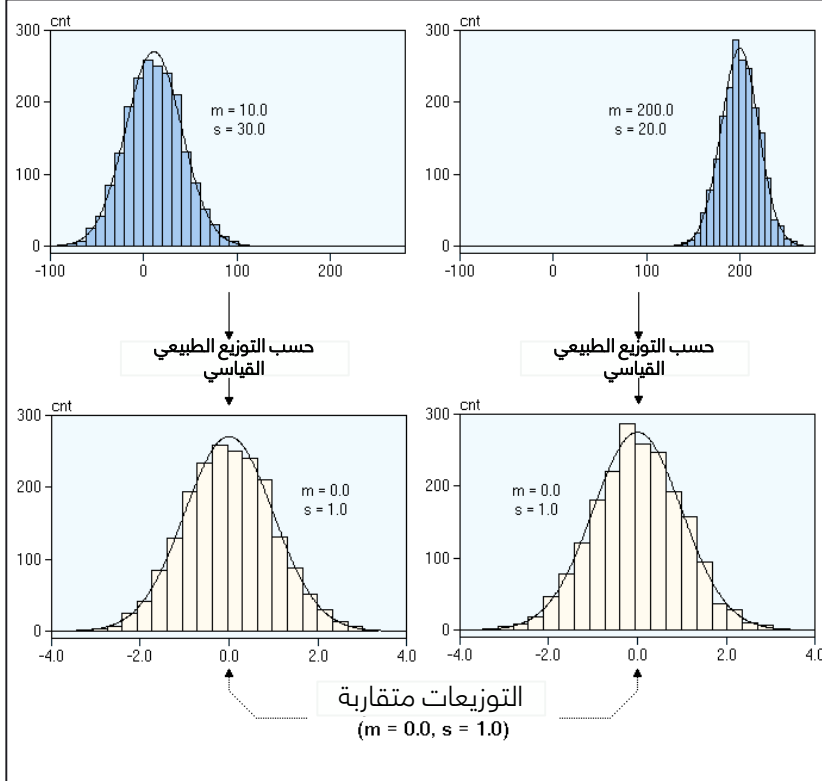
تحضير البيانات

تحضير البيانات

القيم الرقمية (العمر، الطول، السعر، ...) قم بتعديل مقاييس البيانات:

○ مقاييس البيانات حسب قيمتها الكبرى والصغرى MinMaxScaler.

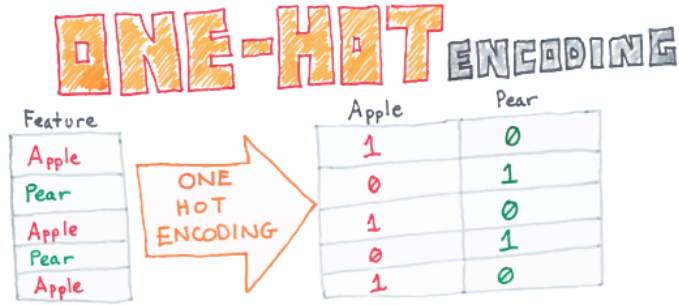
○ مقاييس البيانات حسب منحنى التوزيع الطبيعي القياسي StandardScaler.



تحضير البيانات

قيم فئوية أو منفصلة (النوع، اللون، الكائن، ...)

ترميز الواحد النشط one-hot encoding ○

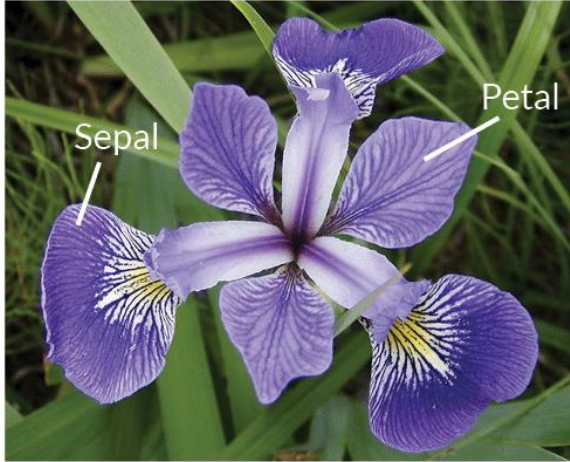


One-hot encoding allows us to turn nominal categorical data into features with numerical values, while not mathematically imply any ordinal relationship between the classes.

ChrisAlbon

مجموعة بيانات أزهار السوسن (Iris Dataset)

عدد المخرجات : 3 = مسألة تصنيف متعدد الفئات!



سوسن فيرسيكولور
Iris Versicolor



سوسن سيتوسا
Iris Setosa



سوسن فيرجينيكا
Iris Virginica

ترميز الواحد النشط (One-hot encoding)

سوسن فيرجينيكا	سوسن فيرسيكولور	سوسن سيتوسا
0	0	1
0	1	0
1	0	0

```
# ترميز القيم الفئوية كأعداد صحيحة
```

```
encoder = LabelEncoder()
```

```
encoder.fit(Y)
```

```
encoded_Y = encoder.transform(Y)
```

```
# ترميز الأعداد الصحيحة إلى قيم ثنائية (0 أو 1) على أساس موقعها
```

```
dummy_y = np_utils.to_categorical(encoded_Y)
```

مراحل سير عمل نموذج التعلم الآلي (scikit-learn)

اتباع مراحل سير العمل لضمان تطبيق الطرق القياسية للتعلم الآلي في تسلسل تلقائي.

التحقق التقاطعي للنموذج الأساسي باستخدام خط أنابيب يقوم بتعديل مجموعة البيانات حسب التوزيع الطبيعي القياسي تلقائياً

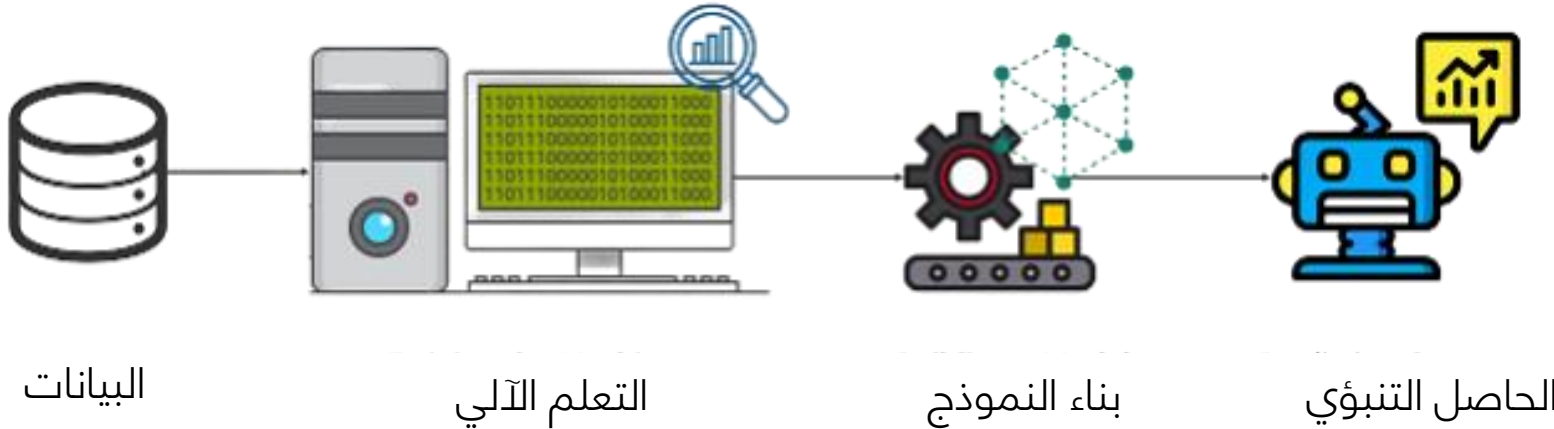
```
# evaluate baseline model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_baseline, epochs=100,
    batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
```



تحسين نماذج التعلم العميق: الطرق المتعلقة بالبيانات

أهمية البيانات

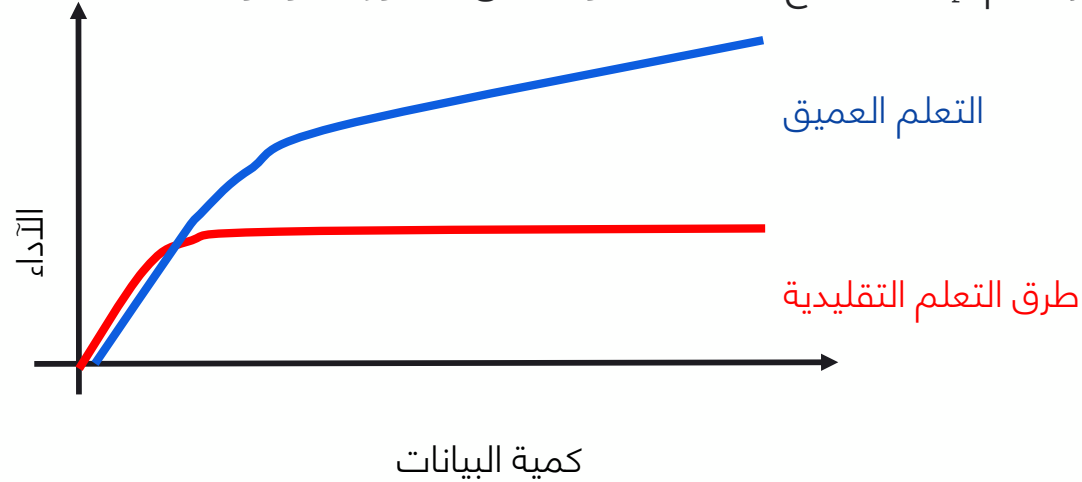
البيانات هي جوهر أي نموذج تعلم عميق نريد بناءه.



زيادة البيانات

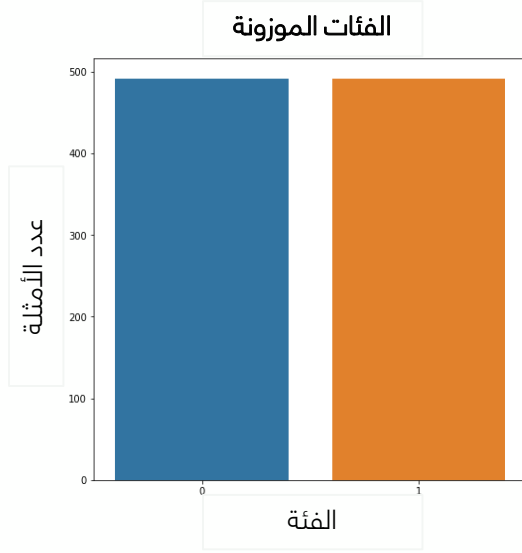
إذا لم تتمكن من الحصول على المزيد من البيانات بشكل معقول، يمكنك إنشاء المزيد من البيانات:

- متجه الأرقام Vectors of numbers؟ إنشاء نسخ معدلة عشوائيًا من القيم الموجودة.
- الصور؟ قم بإنشاء نسخ معدلة عشوائيًا من الصور الموجودة.



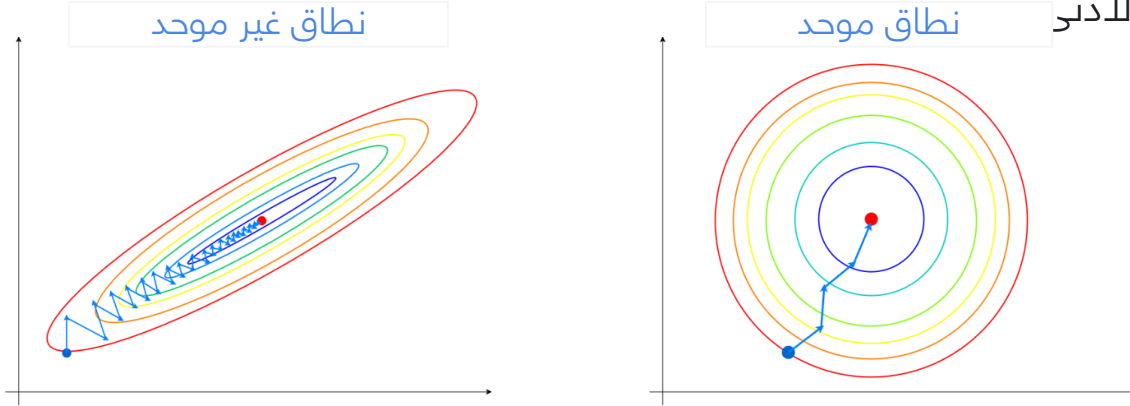
موازنة مجموعة البيانات

- تقليل عينات فئة الأغلبية: بإمكانك موازنة توزيع الفئات عن طريق سحب عدد أقل من أمثلة فئات الأغلبية.
- زيادة عينات فئة الأقلية: يمكن استخدام السحب التعويضي للعينات لزيادة نسبة فئة الأقلية بمجموعة البيانات.



تعديل مقياس البيانات Rescaling Data

- الخصائص التي يتم قياسها بمقاييس مختلفة لا تؤثر بشكل متساوٍ في التحليل وقد ينتهي الأمر بإدخال التحيز على النموذج إلى خصائص معينة دوناً عن أخرى.
- يمكن أن يساعد وجود خصائص على نطاق مشابه في أن يتقارب النزول الاشتقاقي بسرعة أكبر نحو الحد الأدنى



تعديل مقياس البيانات

لتعديل مقياس البيانات هناك طريقتان:

- التسوية (Normalization).
- التوحيد القياسي (Standardization).

$$x_{\text{standardized}} = \frac{x - \mu}{\sigma}$$

$$x_{\text{normalized}} = \frac{x}{x_{\text{max}}}$$



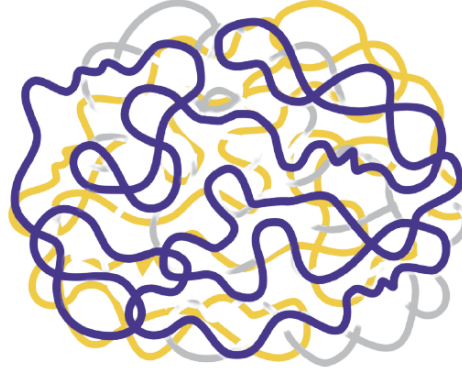
أحيانًا ما يتم استخدام مصطلحات التوحيد القياسي والتسوية بالتبادل

اختيار الخصائص

- اختيار الخصائص: اختيار الخصائص التي تساهم بشكل أكبر في الناتج النهائي من حيث:



تسريع عملية التدريب



تقليل التعقيد



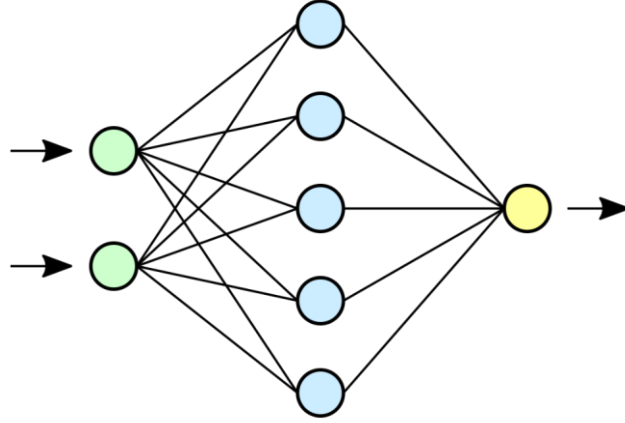
تحسين الدقة

تحسين نماذج التعلم طرق تصميم: العميق النماذج

The top corners of the slide are decorated with various geometric shapes. On the top-left, there is a brown-to-orange gradient rectangle and a dark blue triangle. On the top-right, there is a blue-to-teal gradient rectangle and a dark blue triangle. In the center-top, there are several dark blue hexagons and a diamond shape.

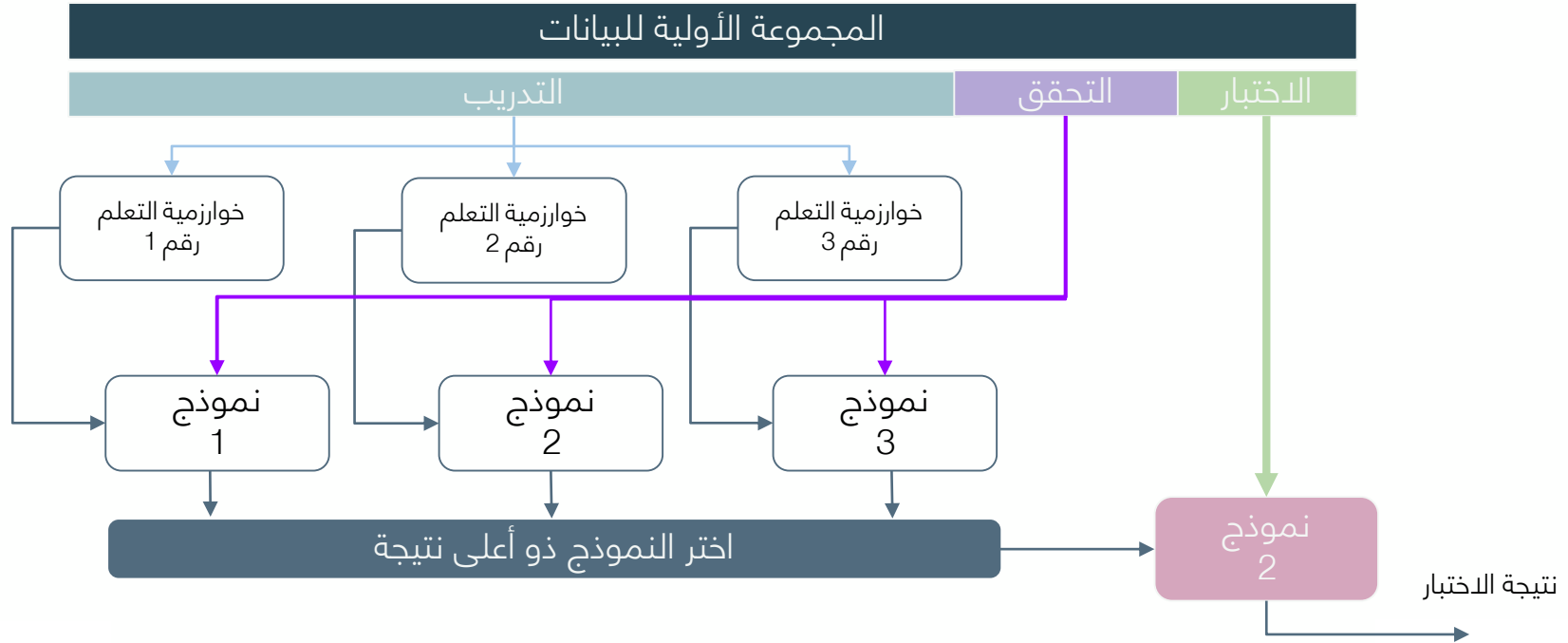
ضبط معاملات الضبط Hyperparameter Tuning

ضبط معاملات الضبط Hyperparameter Tuning



- هي عملية البحث عن أفضل قيم لمعاملات ضبط خوارزمية التعلم.
- أمثلة عن معاملات الضبط بالشبكات العصبية:
 - عدد الطبقات
 - عدد الخلايا العصبية بكل طبقة
 - معدل التعلم
 - معامل التنظيم (Regularization Factor)

ضبط معاملات الضبط Hyperparameter Tuning



تغيير معدل التعلم

Learning Rate

معدل التعلم

الاشتقاق القيمة الحالية للمعامل

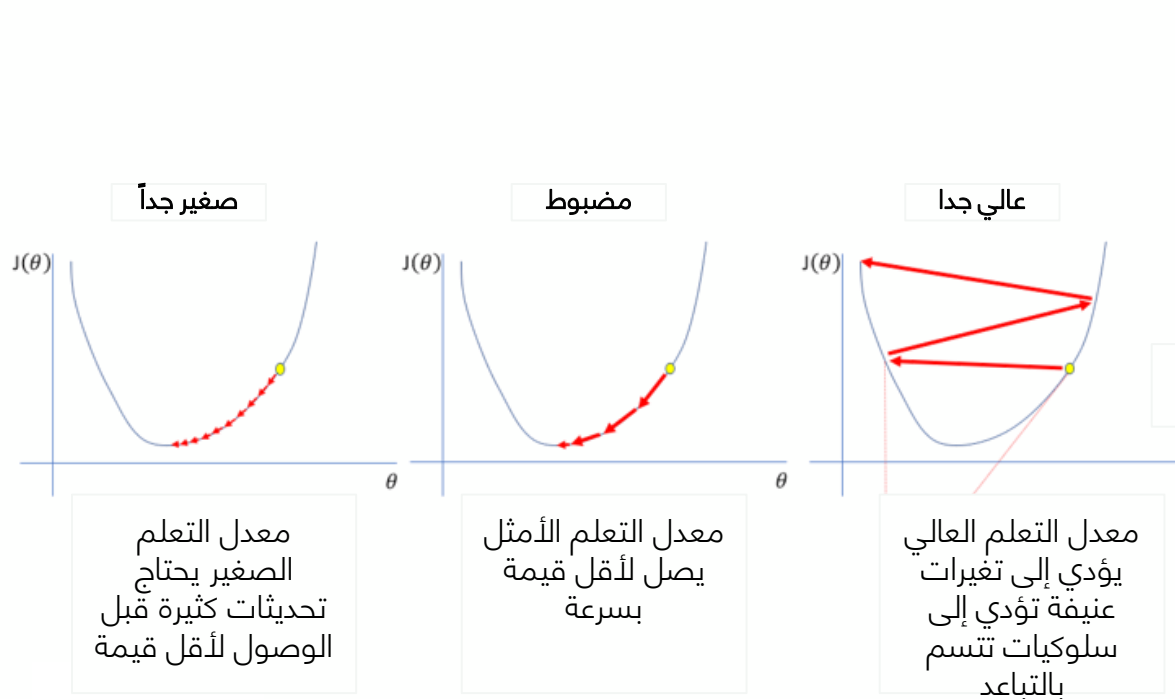
$$W_{t+1} = W_t - \alpha \frac{\partial L}{\partial W_t}$$

Current Parameter value *Gradient*

Updated Parameter value *Learning Rate*

قيمة المعامل بعد التعديل معدل التعلم

معدل التعلم



$$W_{t+1} = W_t - \alpha \frac{\partial L}{\partial W_t}$$

Diagram illustrating the update rule for the parameter W based on the current value and the derivative of the loss function L with respect to W .

Labels in the diagram:

- الاشتقاق (Derivative)
- القيمة الحالية للمعامل (Current parameter value)
- Learning Rate (معدل التعلم)
- Updated Parameter value (قيمة المعامل بعد التعديل)

معدل تعلم مرّن

يمكن أن يؤدي تكييف معدل التعلم –بمسألة التحسين بطريقة النزول الاشتقاقي –إلى زيادة الأداء وتقليل وقت التدريب!

- معدل تعلم يعتمد على الوقت: Time-based Learning Rate Schedule

$$LearningRate = LearningRate \times \frac{1}{1 + decay \times epoch}$$

معدل التعلم

 $=$

معدل التعلم

 \times

$\frac{1}{1 + \text{التآكل} \times \text{الدورة}}$

- معدل تعلم يعتمد على الإسقاط: Drop-based Learning Rate Schedule

معدل التعلم

 $=$

معدل التعلم الابتدائي

 \times

معدل الإسقاط

$\left(\frac{1 + \text{الدورة}}{\text{عدد الدورات لكل إسقاط}} \right)^{\text{عدد صحيح}}$

$$LearningRate = InitialLearningRate \times DropRate^{\text{floor}\left(\frac{1+epoch}{epochdrop}\right)}$$

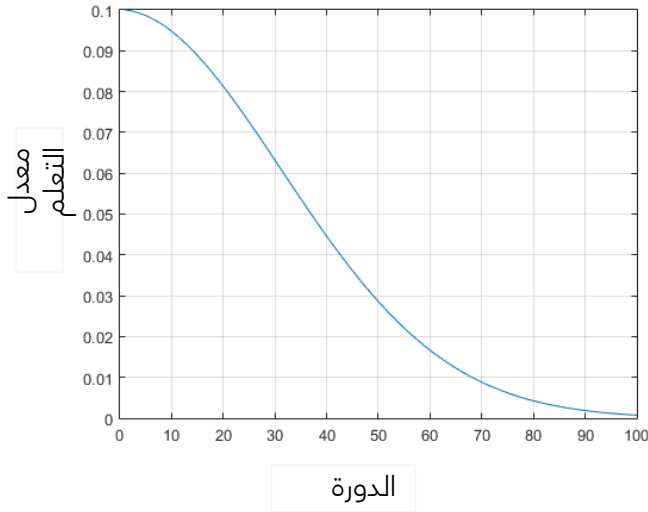
معدل تعلم مرّن

عامل التضاؤل

مثال — $0.1 = \text{معدل التعلم}$ — $0.001 = \text{التضاؤل}$

$$\text{LearningRate} = \text{LearningRate} \times \frac{1}{1 + \text{decay} \times \text{epoch}}$$

$$\boxed{\text{معدل التعلم}} = \boxed{\text{معدل التعلم}} \times \frac{1}{1 + \boxed{\text{التضاؤل} \times \text{الدورة}}}$$



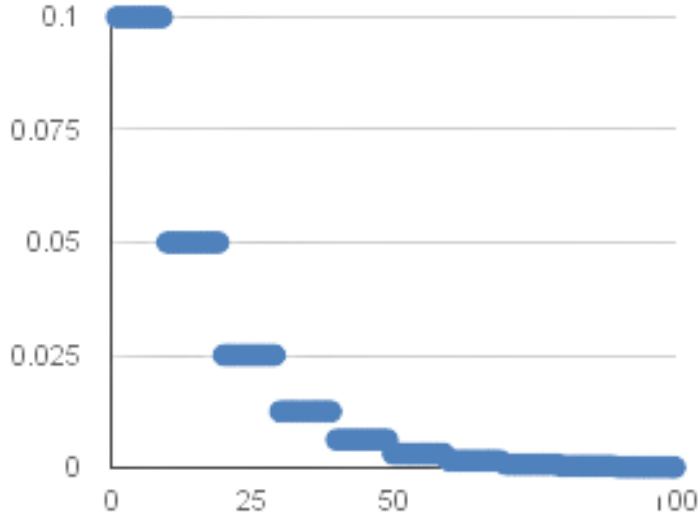
معدل تعلم مرّن

قم بإسقاط معدل التعلم بشكل منهجي في أوقات محددة أثناء التدريب.

LearningRateScheduler Callback

مثال:

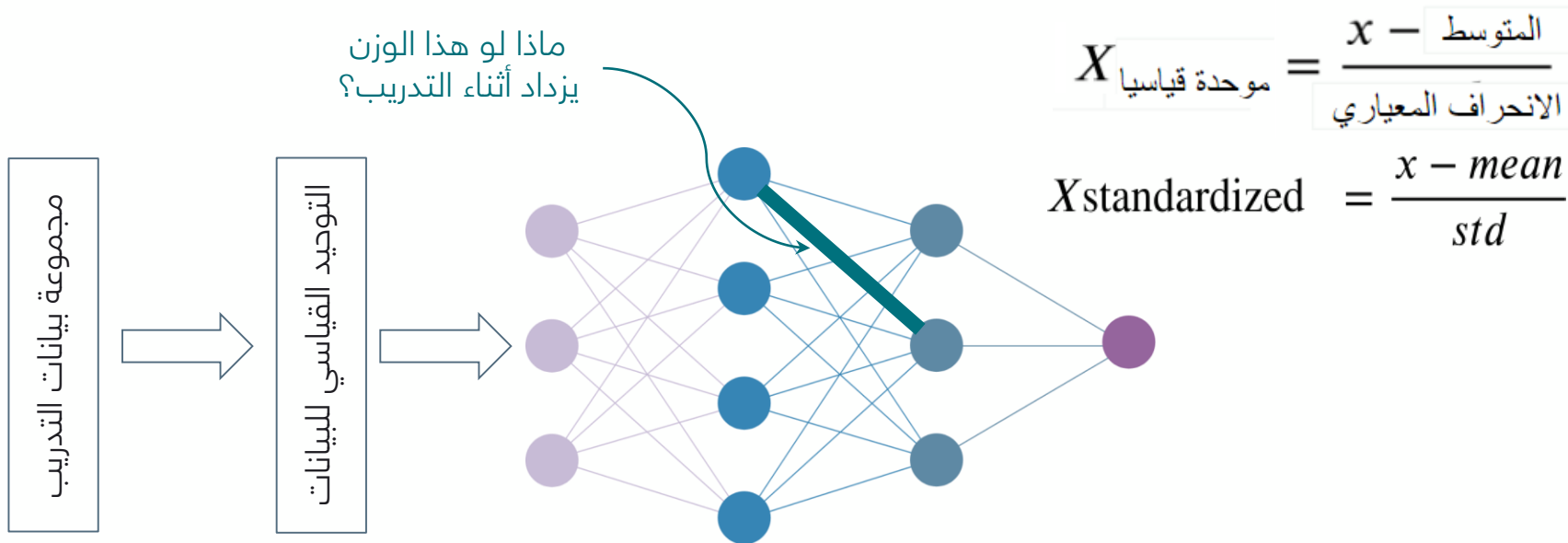
معدل التعلم 0.1 = ويتم إسقاطه بمعدل 50% كل 10 دورات.



The top left corner of the slide features several overlapping geometric shapes, including a large orange-to-red gradient rectangle, a smaller dark blue triangle, and a larger dark blue rectangle. The rest of the slide background is a solid dark blue color.

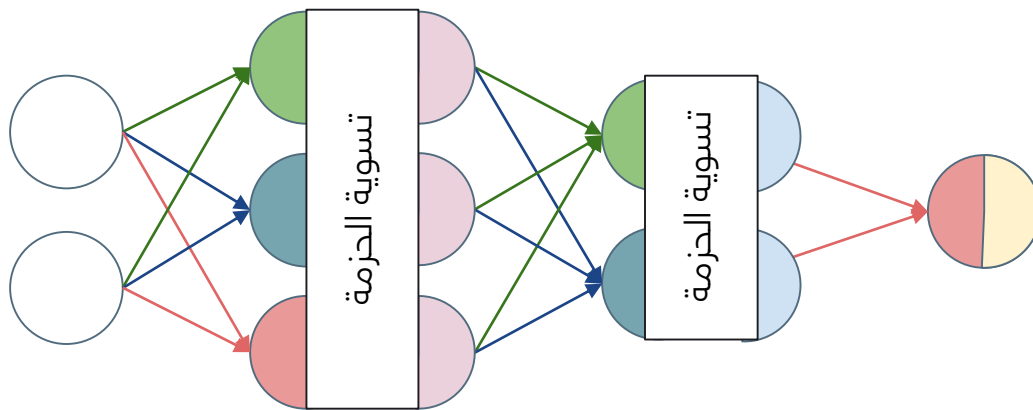
تسوية الحزمة Batch Normalization

تسوية الحزمة



تسوية الحزمة

تعمل تسوية الحزمة على توحيد نطاقات النواتج الصافية للخلايا العصبية عند تغذية حزمة إلى الشبكة.



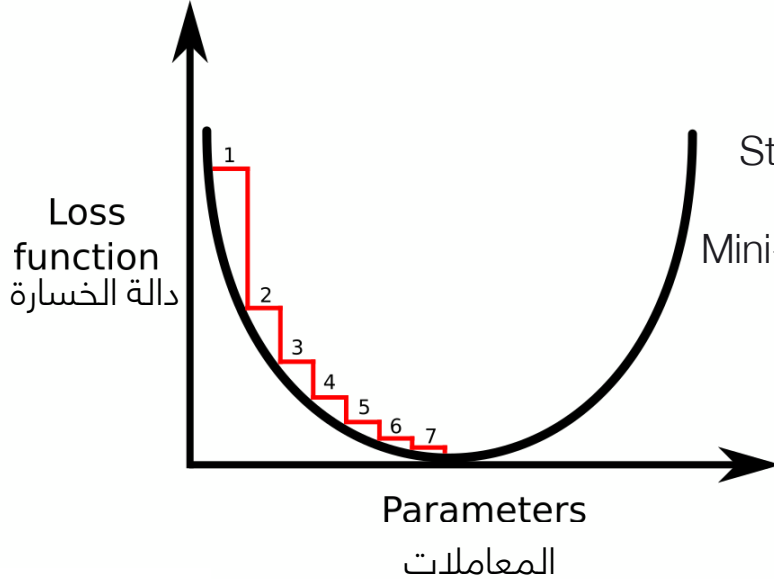
تحسين أداء نماذج التعلم العميق: طرق التحسين

1. أنواع النزول الاشتقاقي

أنواع النزول الاشتقاقي (التحسين)

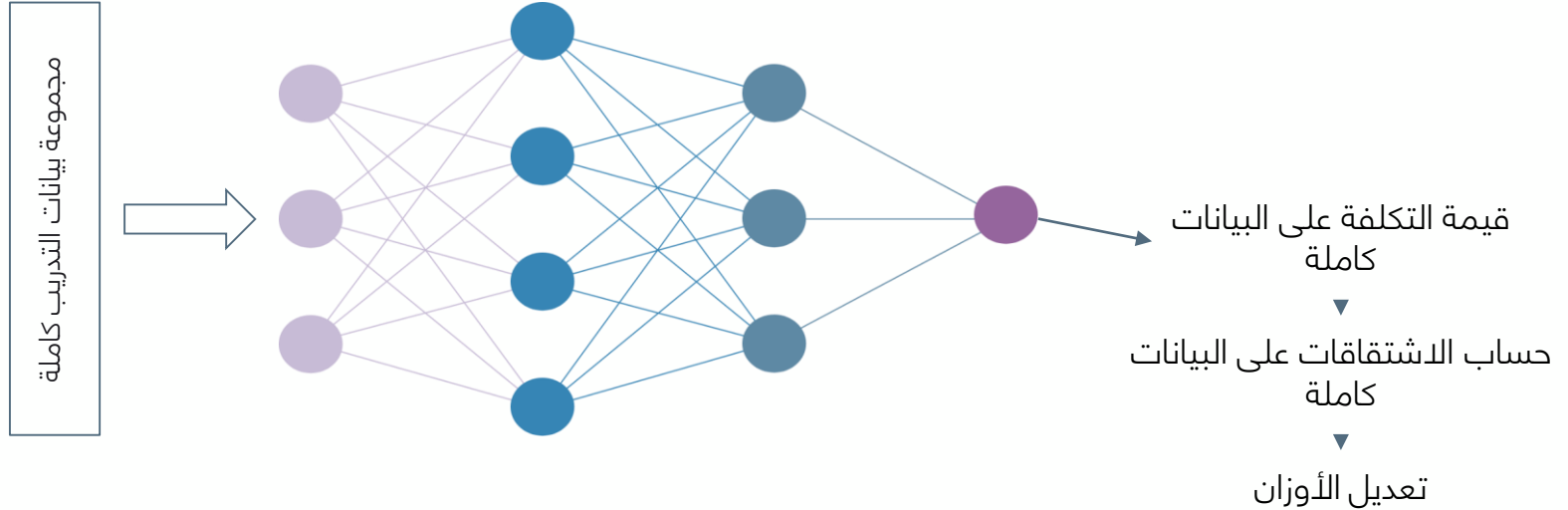
هناك ثلاثة أنواع من النزول الاشتقاقي:

- النزول الاشتقاقي الكلي Batch gradient descent
- النزول الاشتقاقي العشوائي Stochastic gradient descent
- النزول الاشتقاقي بحزم صغيرة Mini-batch gradient descent



النزول الاشتقاقي الكلي

- تحسب هذه الطريقة الأوزان الجديدة بالنزول الاشتقاقي لدالة التكلفة باستخدام مجموعة بيانات التدريب بأكملها.



النزول الاشتقاقي الكلي

الإيجابيات

- يضمن التقارب إلى الحد الأدنى.

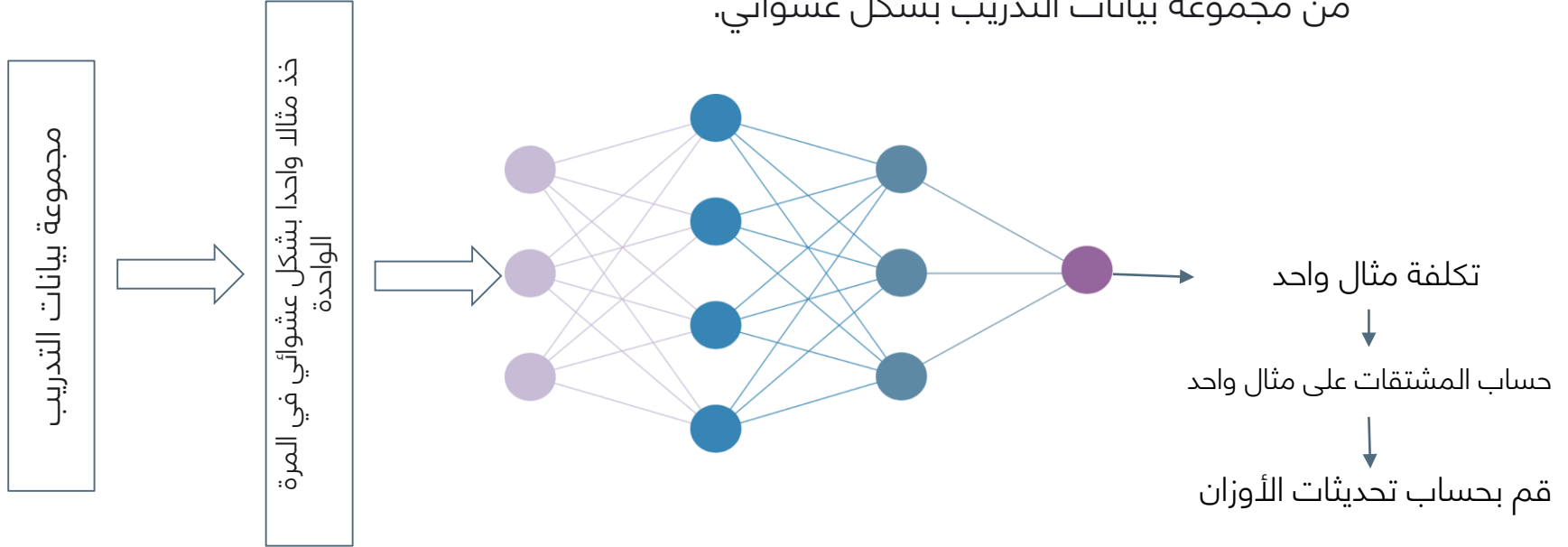
السلبيات

- يمكن أن يكون بطيئاً جداً.
- من الصعب استخدامه لمجموعات البيانات التي لا تتناسب مع حجم الذاكرة.
- لا يسمح لنا بتحديث نموذجنا عبر الإنترنت.



النزول الاشتقاقي العشوائي

- هذا النوع من النزول الاشتقاقي يقوم بتعديل المعاملات المدربة عند كل مثال يتم اختياره من مجموعة بيانات التدريب بشكل عشوائي.



النزول الاشتقاقي العشوائي

الإيجابيات

- عادةً ما يكون أسرع بكثير من النزول الاشتقاقي الكلي.
- يمكن استخدامه للتعلم عبر الإنترنت.

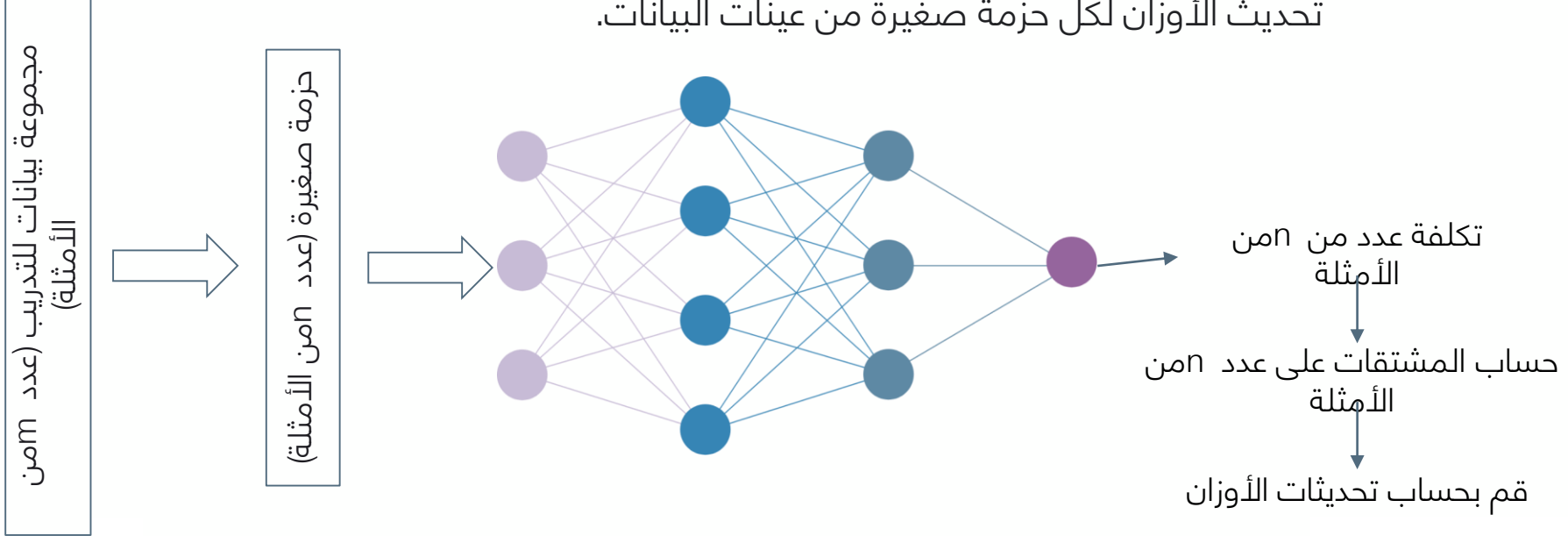
السلبيات

- يقوم بإجراء تحديثات متكررة مع تباين احصائي كبير يتسبب في تقلب دالة التكلفة بشكل كبير.



النزول الاشتقاقي بحزم صغيرة

- تجمع هذه الطريقة بين أفضل ما في طريقتي النزول الاشتقاقي الكلي والعشوائي بحيث يتم تحديث الأوزان لكل حزمة صغيرة من عينات البيانات.



النزول الاشتقاقي بحزم صغيرة



الإيجابيات

- يقلل من تباين تحديثات المعاملات.
- يمكن أن يؤدي إلى تقارب أكثر استقرارًا.



السلبيات

- علينا ضبط حجم الحزمة الصغيرة.

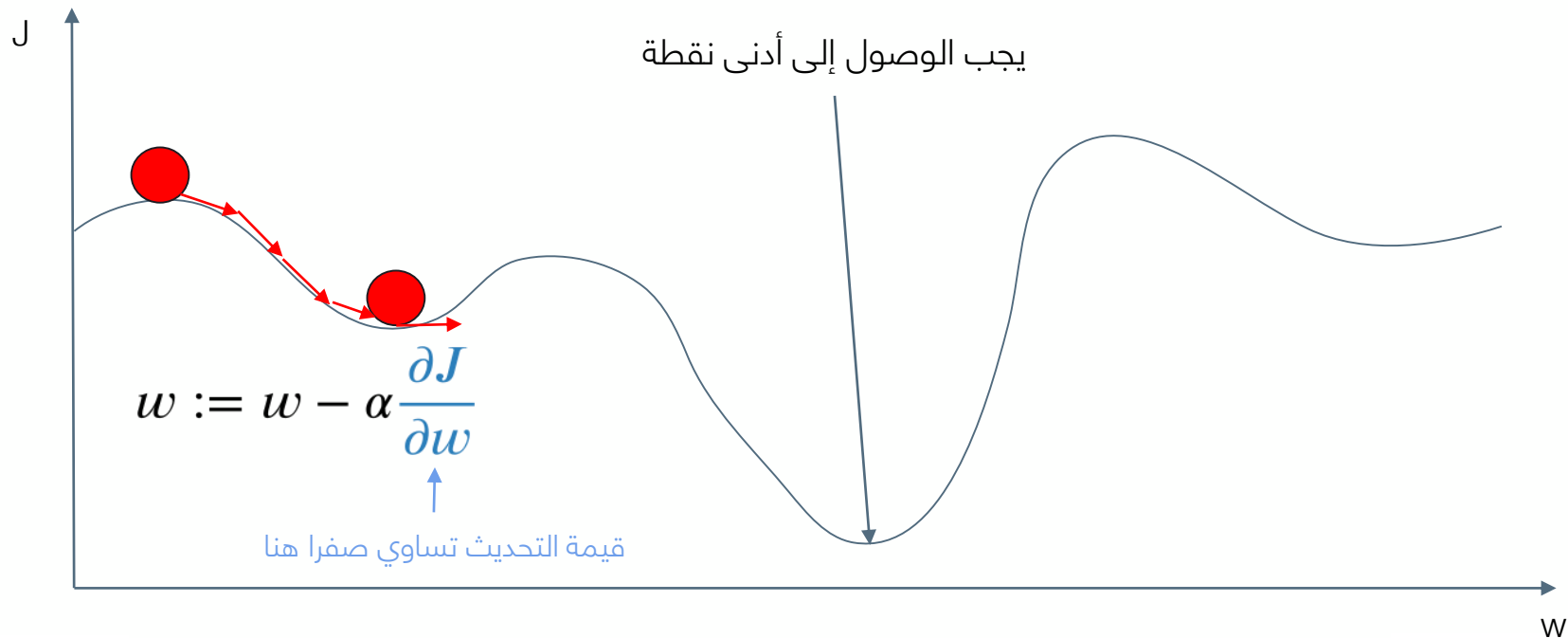
مقارنة الطرق الثلاثة



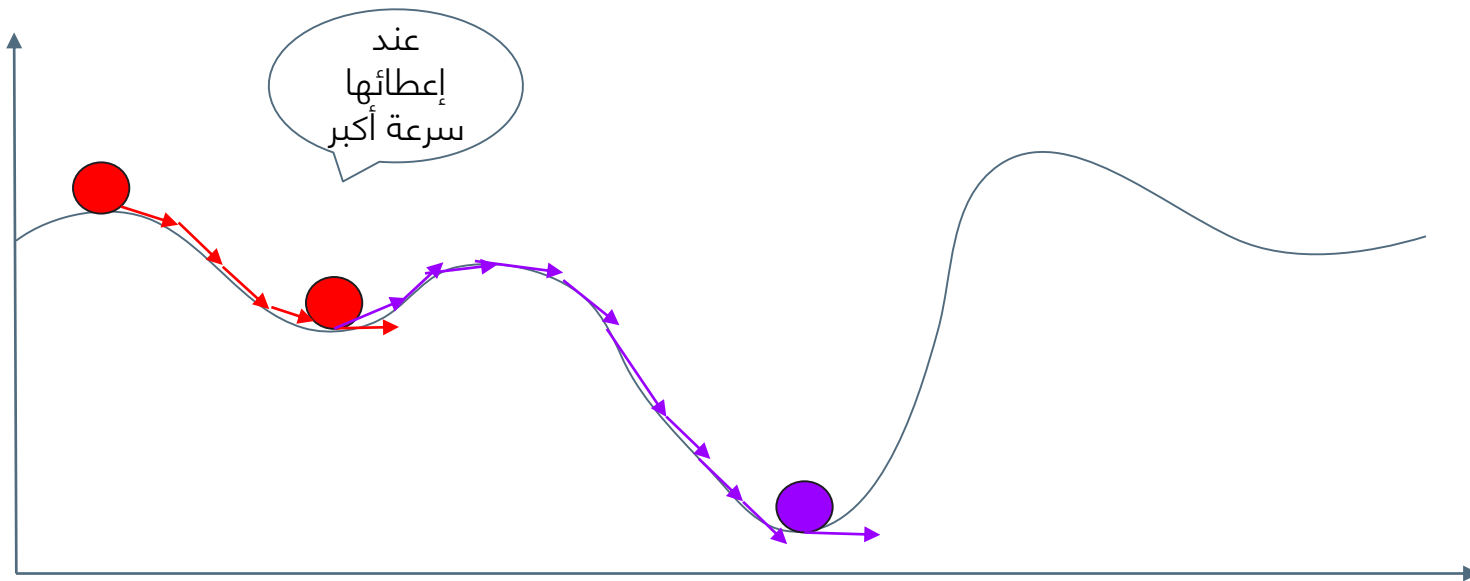
- Batch gradient descent الضبط الاشتقاقي الكلي
- Mini-batch gradient Descent الضبط الاشتقاقي بحزم صغيرة
- Stochastic gradient descent الضبط الاشتقاقي العشوائي

2. خوارزميات تحسين دالة النزول الاشتقاقي

خوارزميات تحسين دالة النزول الاشتقاقي



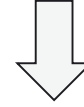
خوارزميات تحسين دالة النزول الاشتقاقي



خوارزميات تحسين دالة النزول الاشتقاقي

قاعدة تحديث النزول الاشتقاقي:

$$w := w - \alpha \frac{\partial J}{\partial w}$$



قاعدة تحديث خوارزميات أخرى:

$$w := w - \alpha f \left(\frac{\partial J}{\partial w} \right)$$



النزول الاشتقاقي بالزخم Momentum

إنها خوارزمية تضيف جزءًا من متجه التحديث في الخطوة الزمنية السابقة إلى متجه التحديث الحالي.

$$V_w := \beta V_w + (1 - \beta) \cdot \frac{\partial J}{\partial w}$$

المتوسط المتحرك
للمشتقات:

$$w := w - \alpha V_w$$

قاعدة التحديث:

إذا أصبحت هذه القيمة كبيرة، فقد نتجاوز
القيمة الأدنى

RMS Prop

عبارة عن خوارزمية تم تطويرها لحل مشكلة زخم التدرجات المتراكمة.

المتوسط المتحرك لقيمة
الاشتقاق المربّعة

$$S_w := \beta S_w + (1 - \beta) \cdot \left(\frac{\partial J}{\partial w} \right)^2$$

قاعدة التحديث

$$w := w - \alpha \cdot \frac{1}{\sqrt{S_w} + \epsilon}$$

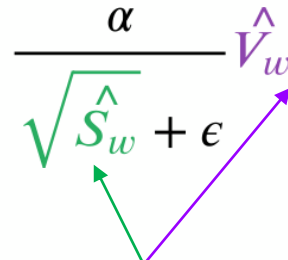
إذا أصبحت هذه القيمة كبيرة ، فسيصبح الكسر قريباً من الصفر ، ولن يتم تحديث الوزن.

Adam

يمكن الاستشهاد بـ Adam كأفضل مُحسِّن لأنه يجمع بين فوائد RMSProp والزخم.

$$V_w := \beta_1 V_w + (1 - \beta_1) \cdot \frac{\partial J}{\partial w} \quad \hat{V}_w = \frac{V_w}{1 - \beta_1}$$

$$S_w := \beta_2 S_w + (1 - \beta_2) \cdot \left(\frac{\partial J}{\partial w} \right)^2 \quad \hat{S}_w = \frac{S_w}{1 - \beta_2}$$

$$w := w - \frac{\alpha}{\sqrt{\hat{S}_w} + \epsilon} \hat{V}_w$$


هاتان القيمتان تزدادان سويا وبالتالي الكسر لن يصل للصفر أو لما لا نهاية له

3. الضبط Regularization

الضبط

- الشبكات العصبية هي مثال للنماذج المعقدة.
- يتم استخدام الضبط في الشبكات العصبية لتقليل تعقيدها.
- يقوم بذلك عن طريق تقليل قيم الأوزان المختلفة.

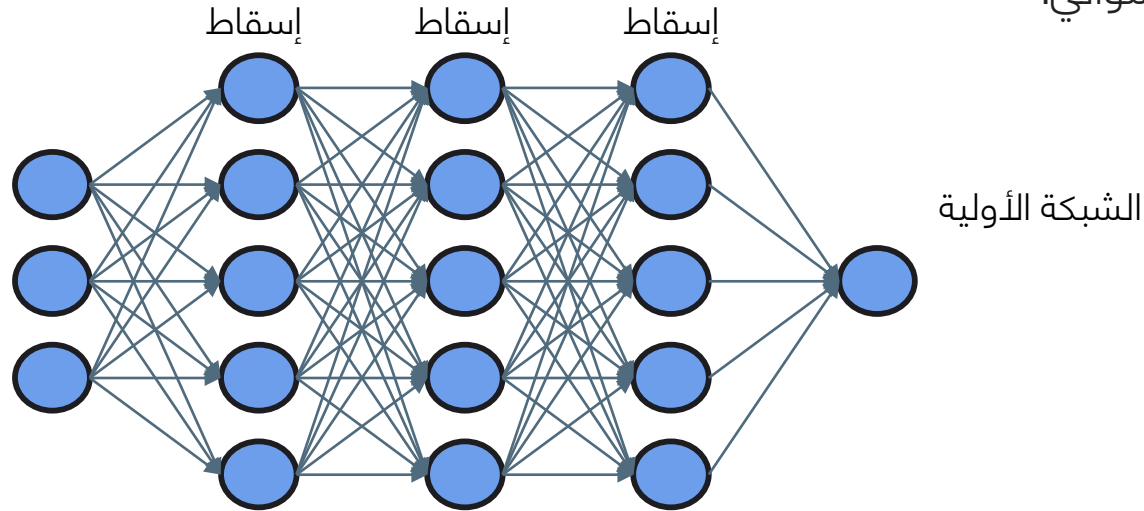
$$\text{ضبط } L2 \rightarrow \frac{\text{التكلفة الجديدة}}{\text{التكلفة الجديدة}} = J + \frac{\lambda}{2m} \sum ||w||^2$$

$$\text{ضبط } L1 \rightarrow \frac{\text{التكلفة الجديدة}}{\text{التكلفة الجديدة}} = J + \frac{\lambda}{2m} \sum ||w||$$

معيار مصفوفة الأوزان : $||w||$

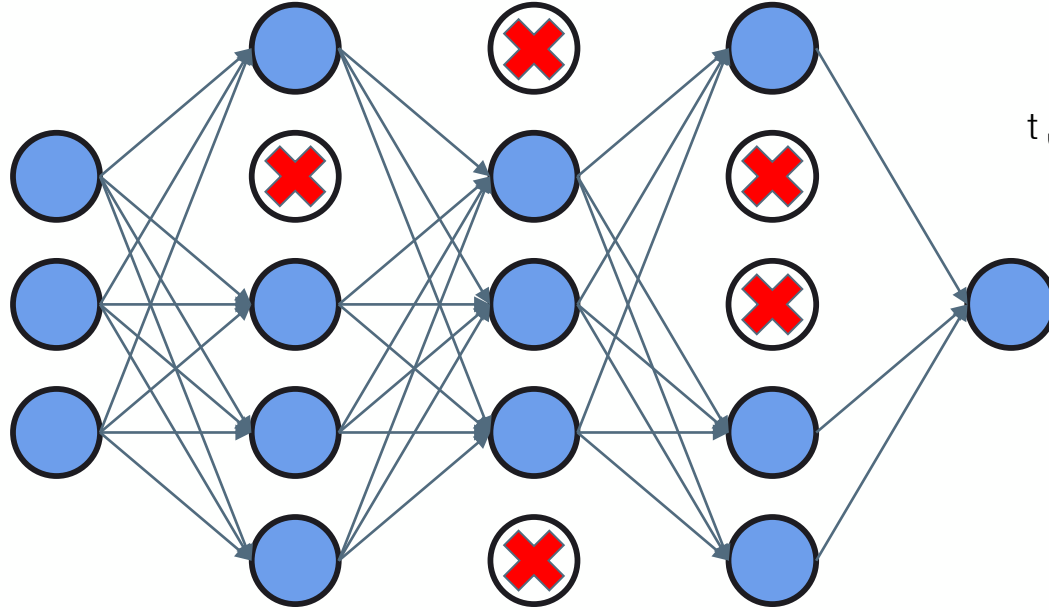
ضبط الإسقاط Dropout Regularization

- في كل تكرار، يتم إزالة نسبة معينة من الخلايا العصبية من خلال طبقة الإسقاط بشكل عشوائي.



الإسقاط

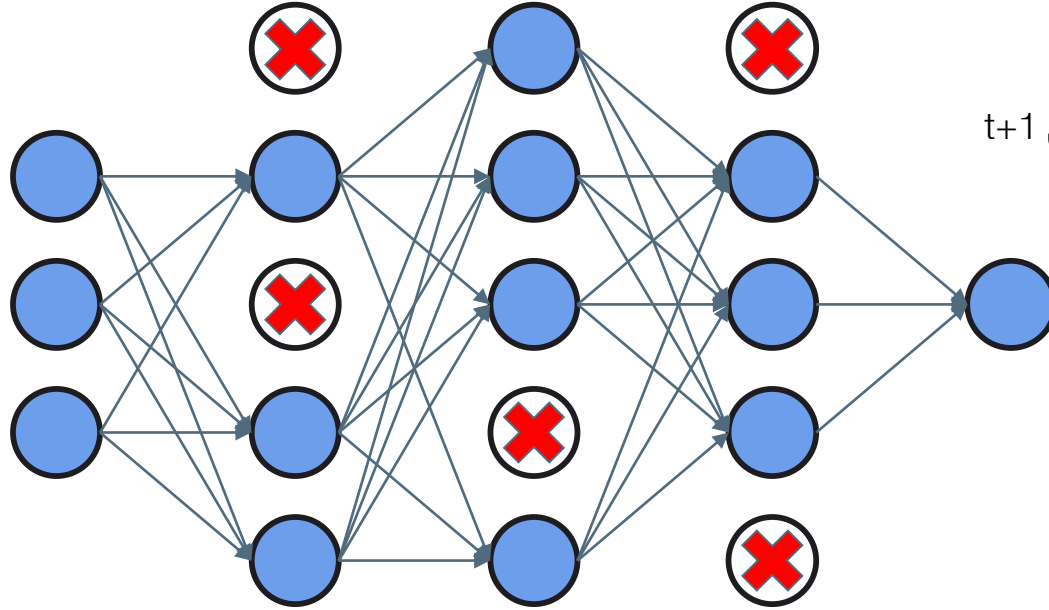
- في كل تكرار، يتم إزالة نسبة معينة من الخلايا العصبية من خلال طبقة الإسقاط عشوائي.



الشبكة عند مثال t

الإسقاط

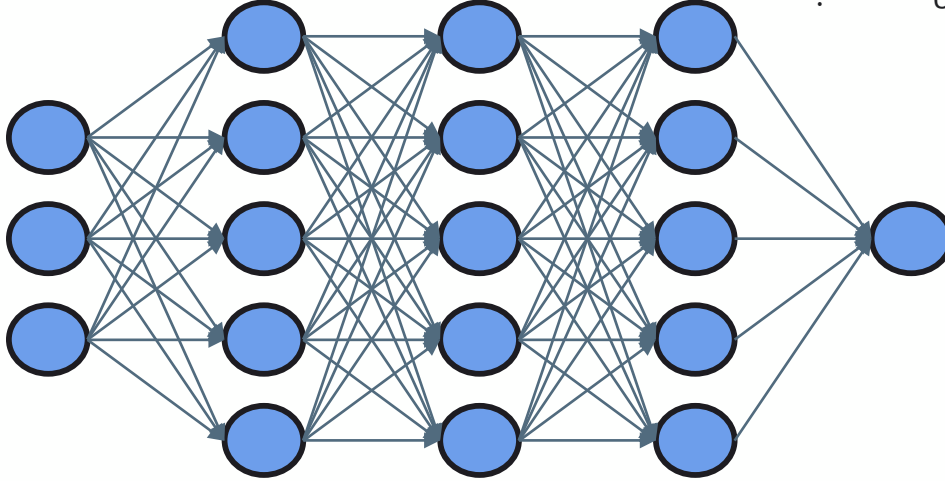
- في كل تكرار، يتم إزالة نسبة معينة من الخلايا العصبية من خلال طبقة الإسقاط عشوائي.



الإسقاط

لتطبيق الإسقاط، نحتاج إلى:

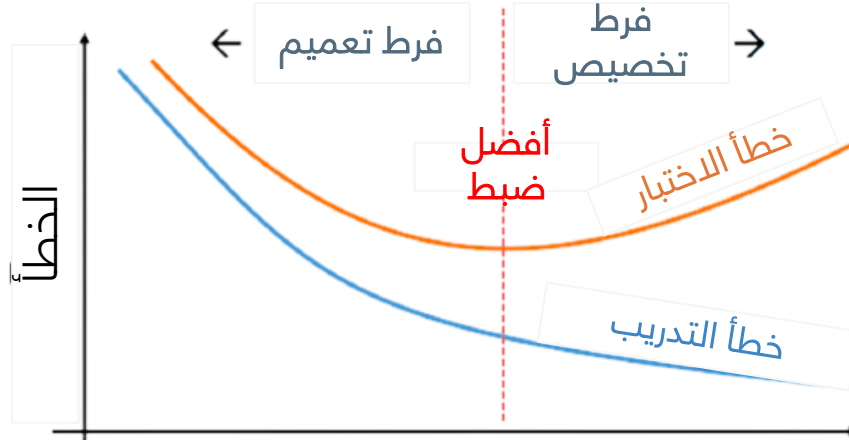
- تحديد الطبقات التي يمكن أن يحدث فيها الإسقاط.
- تحديد معدل الإسقاط لهذه الطبقة.



4. التوقف المبكر Early Stopping

التوقف المبكر

- التوقف المبكر هي تقنية تسمح لنا بإيقاف التدريب بمجرد توقف تحسين القيمة التي تتم مراقبتها مثل خسارة التحقق من الصحة `val_loss`



The background is a solid dark blue color. It is decorated with several geometric shapes: a large orange-to-red gradient rectangle on the top left, a teal-to-blue gradient rectangle on the top right, and several dark blue triangles and diamonds scattered across the upper half of the image.

تدريب عملي: تصنيف فصائل أزهار السوسن Iris Specie Detection

إطار بديل PyTorch

مقدّمة

ما هو PyTorch ؟

- PyTorch هو مكتبة بايثون
- تم تطويره بواسطة مختبر أبحاث الذكاء الاصطناعي التابع لفيسبوك
- مكتوب بلغة بايثون و C++
- يوفر الوصول إلى الأدوات والمكتبات وبنى النموذج
- يوفر أدوات مكتوبة مسبقاً تستخدم في رؤية الحاسب، معالجة اللغة الطبيعية...



ما هو PyTorch ؟

- بسيط وسهل الاستخدام
- يتم باستخدام بايثون
- معروف باستخدام الذاكرة بكفاءة
- يمكن المستخدم من التوسع في وحدات معالجة الرسومات المتعددة GPUs
- يمكن تثبيته محلياً أو في Google Colab



PyTorch خصائص

يوفر PyTorch خاصيتين رئيسيتين:

- Tensors التي يمكن تشغيلها على وحدات معالجة الرسومات على عكس صفائف Numpy arrays
- الرسم البياني الحسابي المرن Dynamic computational graph

PyTorch خصائص

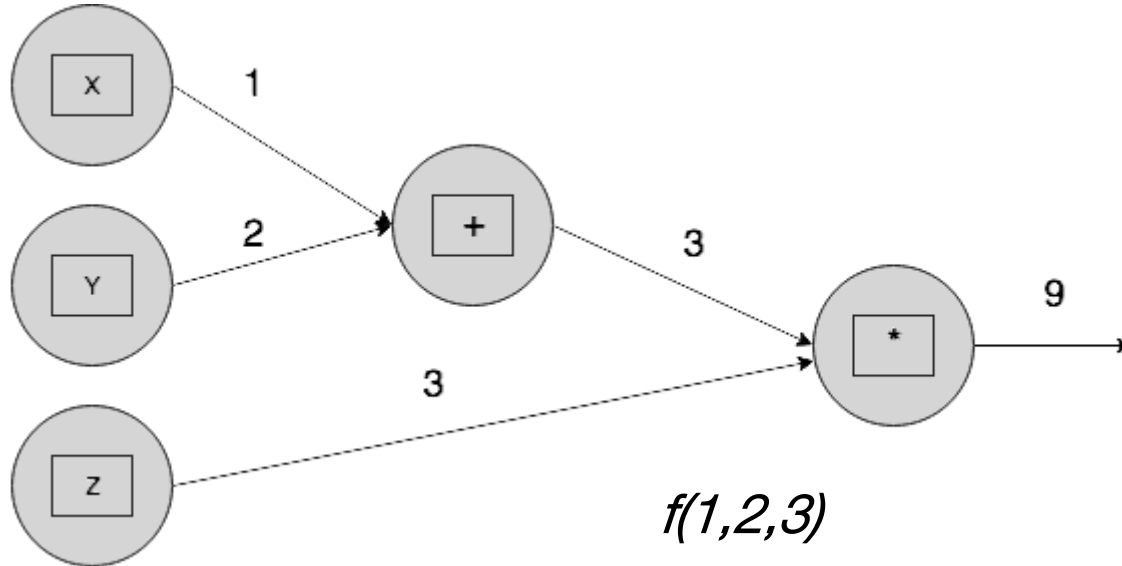
يوفر PyTorch خاصيتين رئيسيتين:

- **Tensors** التي يمكن تشغيلها على وحدات معالجة الرسومات على عكس صفائف Numpy arrays
- الرسم البياني الحسابي المرن Dynamic computational graph

PyTorch خصائص

يوفر PyTorch خاصيتين رئيسيتين:

- Tensors التي يمكن تشغيلها على وحدات معالجة الرسومات على عكس صفائف Numpy
- الرسم البياني الحسابي المرن



PyTorch خصائص

يوفر PyTorch خاصيتين رئيسيتين:

- Tensors التي يمكن تشغيلها على وحدات معالجة الرسومات على عكس صفائف Numpy
- الرسم البياني الحسابي المرن



$$\begin{aligned}f(x) &= e^x \\g(x) &= \sin x \\h(x) &= x^2 \\f(g(h(x))) &= e^{g(h(x))}\end{aligned}$$

PyTorch خصائص

الرسوم البيانية الحسابية يمكن أن تكون:

- ثابتة: لن تتغير بمجرد تعريفها. يتم إعداد الرسم البياني ومن ثم تنفيذه عدة مرات
- مرنة: تتضمن القدرة على التكيف مع كميات مختلفة في البيانات المدخلة.

A graph is created on the fly

```
from torch.autograd import Variable
```

```
x = Variable(torch.randn(1, 10))  
prev_h = Variable(torch.randn(1, 20))  
W_h = Variable(torch.randn(20, 20))  
W_x = Variable(torch.randn(20, 10))
```

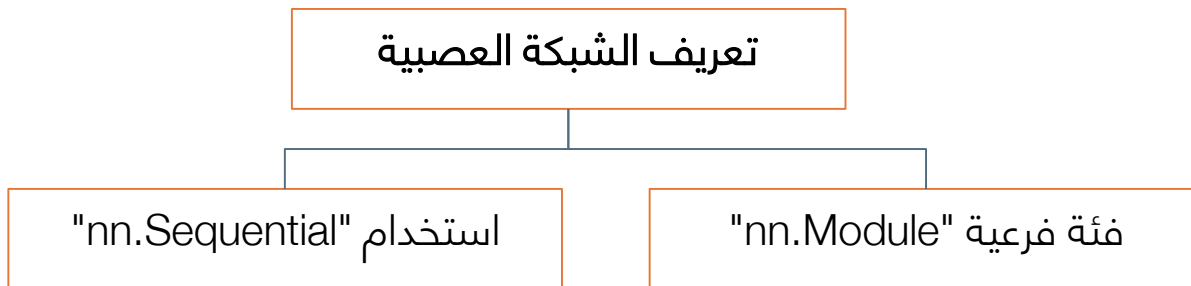


إنشاء شبكات عصبية باستخدام PyTorch

الشبكات العصبية في PyTorch

- يوفر PyTorch وحدة `torch.nn` لتعريف الشبكات العصبية

```
import torch
import torch.nn as nn
```



تعريف الشبكات العصبية في PyTorch

باستخدام "nn.Sequential":

- nn.Sequential عبارة عن حاوية
- يتم تكديس الطبقات بالتسلسل.
- مفيد للشبكات التي تتدفق فيها البيانات خطيًا

```
import torch.nn as nn

model = nn.Sequential(
    nn.Linear(in_features=64, out_features=128),
    nn.ReLU(),
    nn.Linear(128, 10),
    nn.Softmax(dim=1)
)
```

تعريف الشبكات العصبية في PyTorch

من خلال التصنيف الفرعي "nn.Module":
● بنية معقدة أو أفضل

```
import torch.nn as nn

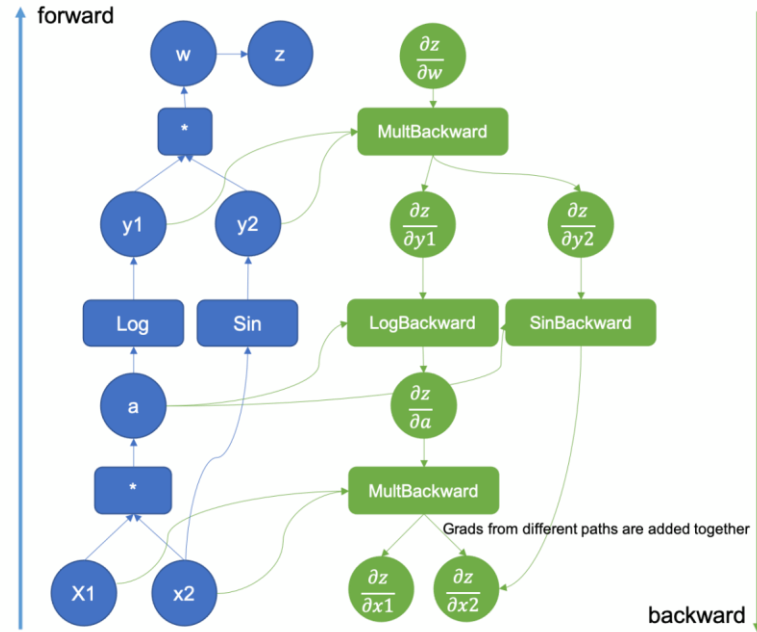
class CustomNet(nn.Module):
    def __init__(self):
        super(CustomNet, self).__init__()
        self.layer1 = nn.Linear(64, 128)
        self.layer2 = nn.Linear(128, 10)

    def forward(self, x):
        x = nn.ReLU()(self.layer1(x))
        return nn.Softmax(dim=1)(self.layer2(x))

model = CustomNet()
```

تدريب الشبكات العصبية باستخدام PyTorch

تدريب الشبكات العصبية في PyTorch



تدريب الشبكات العصبية في PyTorch

1. تحديد الشبكة
2. اختيار دالة الخسارة والمحسن
3. حلقة التدريب:
 - a. تغذية البيانات إلى الشبكة (إنتشار أمامي)
 - b. احتساب الخسارة باستخدام دالة الخسارة المختارة.
 - c. إزالة أنواع المشتقات للنموذج باستخدام `optimizer.zero_grad()`.
 - d. احتساب المشتقات بالنسبة للخسارة (إنتشار عكسي) باستخدام `Loss.backward()`.
 - e. تحديث أوزان الشبكة باستخدام المحسن المختار، على سبيل المثال، `Optir.step()`.

تدريب الشبكات العصبية في PyTorch

```
loss_fn = nn.BCELoss() # binary cross entropy
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
n_epochs = 100
batch_size = 10

for epoch in range(n_epochs):
    for i in range(0, len(X), batch_size):
        Xbatch = X[i:i+batch_size]
        y_pred = model(Xbatch)
        ybatch = y[i:i+batch_size]
        loss = loss_fn(y_pred, ybatch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

شكراً لكم

Thank you



SDAIA

الهيئة السعودية للبيانات
والذكاء الاصطناعي
Saudi Data & AI Authority