TF & KERAS. РОБОТА із ЗОБРАЖЕННЯМИ

Файл: TF_KERAS_Image_02_001

ШАРИ KERAS Part 1

Посилання:

KERAS

KERAS layers API

Шари є основними елементами, необхідними під час створення нейронних мереж. Послідовні шари відповідають за архітектуру моделі глибокого навчання. Кожен шар виконує обчислення на основі даних, отриманих з попереднього шару. Потім інформація передається далі. Зрештою, останній шар видає потрібний результат.

KERAS (TensorFlow) надає широкий набір типів шарів, які можна використовувати для створення нейромереж. Базові типишарів, доступних у KERAS:

1. Шар Input використовується для створення вхідних тензорів для моделі. Він є точка входу для даних, які будуть оброблятися в моделі.
2. Шар [Flatten] (https://keras.io/api/layers/reshaping_layers/flatten/) - Згладжування даних.
3. Повнозв'язковий шар Dense – це класичний шар нейромережі, в якому кожен нейрон вхідного шару з'єднаний з усіма нейронами вихідного шару. Такий шар може використовуватись для класифікації чи регресії.
4. Шари активації Activation (ReLU, Sigmoid, Tanh) – використовується для додавання нелінійності до нейромережі. Різні функції активації можуть бути використані для різних типів завдань.

Типово для визначення або створення шару Keras потрібна наступна інформація:

- Форма введення: для розуміння структури вхідної інформації
- Кількість: для визначення кількості вузлів/нейронів у шарі
- Ініціалізатор: для визначення ваги кожного входу, що важливо для виконання обчислень
- Активатори: для перетворення вхідних даних у нелінійний формат, щоб кожен нейрон міг навчатися ефективніше
- Обмежувачі: для накладання обмежень на ваги під час оптимізації
- Регулятори: для застосування штрафів до параметрів під час оптимізації

Визначення версії TF

```
import tensorflow as tf # імпорт tensofflow
import numpy as np # імпорт numpy
import pprint as pprint # імпорт пакету посиленого друку
print(tf.__version__) # версія TF
```

```
2.14.0
```

ШАР Dense

Формально Dense реалізує операцію:

output = activation(dot(input, kernel) + bias),

- activation — це поелементна функція активації, яка передається як аргумент активації,
- kernel — це матриця ваг, створена шаром,
- bias — це вектор зміщення, створений за шаром (застосовно, лише якщо use_bias має значення True). Це все атрибути Dense.

Примітка
Якщо вхід до шару має ранг більше 2, тоді Dense обчислює скалярний добуток між входами та ядром уздовж останньої осі входів і осі 0 ядра (за допомогою tf.tensordot). Наприклад, якщо вхідні дані мають розміри (batch_size, d0, d1), тоді створюється ядро з формою (d1, одиниці), і ядро працює вздовж осі 2 вхідних даних на кожному субтензорі форми (1, 1 , d1) (існують такі підтензори batch_size * d0). Вихід у цьому випадку матиме форму (batch_size, d0, одиниці).

Крім того, атрибути шару не можна змінити після того, як шар було викликано один раз (за винятком атрибута trainable).

Приклад 1

```
# Create a `Sequential` model and add a Dense layer as the first layer.
model = tf.keras.Sequential()

# Додавання шару Dense с 32 нейронами и функцией активации ReLU
model.add(tf.keras.layers.Dense(32, activation='relu', input_shape=(784,)))

# Додавання вихідного шару з функцією активаціі softmax
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

У цьому прикладі створюється модель нейронної мережі, яка складається із двох шарів. Перший шар – це шар Dense з 32 нейронами та функцією активації ReLU. Цей шар має три параметри:

1. units - кількість нейронів у шарі. В прикладі обрано 32.
2. activation – функція активації, яка застосовується до виходу кожного нейрона у шарі. Використано ReLU.
3. input_shape – розмірність вхідних даних. У прикладі передаємо вхідні дані розміром 784, що відповідає одновимірному масиву зображення (28 X 28), що містить 784 пікселя.

Другий шар - це вихідний шар, що складається з 10 нейронів та функції активації softmax. Цей шар має лише один параметр:

1. units - кількість нейронів у шарі. У прикладі обрано 10, тому що потрібно визначити ймовірність приналежності зображення до 10 класів (Наприклад 10 цифр від 0 до 9).

Таким чином, шар Dense є класичний шар нейронної мережі, який з'єднує кожен нейрон вхідного шару з усіма нейронами вихідного шару. Цей шар має кілька параметрів, включаючи кількість нейронів у шарі, функцію активації та розмірність вхідних даних. Залежно від конкретного завдання та архітектури нейронної мережі ці параметри можуть змінюватися.

Приклад 2

```
# Create a `Sequential` model and add a Dense layer as the first layer.
model = tf.keras.models.Sequential()
model.add(tf.keras.Input(shape=(16,)))
model.add(tf.keras.layers.Dense(32, activation='relu'))
# Now the model will take as input arrays of shape (None, 16)
# and output arrays of shape (None, 32).
# Note that after the first layer, you don't need to specify
# the size of the input anymore:
model.add(tf.keras.layers.Dense(32))
model.output_shape
```

```
(None, 32)
```

ШАР FLATTEN

Flatten використовується для конвертації вхідних даних у менший розмір.

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(64, 3, 3, input_shape=(3, 32, 32)))
model.output_shape
```

```
(None, 1, 10, 64)
```

```
model.add(tf.keras.layers.Flatten())
model.output_shape
(None, 640)
```

```
(None, 640)
```

ШАР INPUT

tf.keras.Input(shape=None, batch_size=None, name=None, dtype=None, sparse=None, tensor=None, ragged=None, type_spec=None, **kwargs)

Шар використовується в Keras для створення моделі на основі введення та виведення моделі. Він є точкою входу до моделі графа.

Приклад

```
# Створення шару Input
input_layer = tf.keras.layers.Input(shape=(28, 28, 1))

# Створення нгаступного шару
conv_layer = tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3),
activation='relu')(input_layer)

# Створення Keras моделі
model = tf.keras.Model(inputs=input_layer, outputs=conv_layer)
```

У цьому прикладі створено шар Input з розмірністю (28, 28, 1), що означає, що очікуються на вхідні дані з розмірністю 28 на 28 пікселів і 1 канал (чорно-біле зображення). Потім створено наступний шар - згортковий шар Conv2D, який застосовуватиме 32 фільтри розміром 3x3 з функцією активації ReLU до даних, що надходять на вхідний шар.

Нарешті ці шари в об'єднані в модель, використовуючи клас tf.keras.Model. У цьому прикладі шар Input не має додаткових параметрів, але може приймати ряд параметрів, таких як:

- shape: форма вхідних даних
- dtype: тип даних вхідних даних
- batch_size: розмір пакета для навчання
- name: ім'я шару

Наприклад, ми можемо додати параметри batch_size і name у шар Input наступним чином:

```
# Створення шару Input с параметрами batch_size та name
input_layer = tf.keras.layers.Input(shape=(28, 28, 1), batch_size=32,
name='input_layer')
```

У цьому випадку, встановлено batch_size в 32, що означає, що буде використовуватися пакет з 32-х зображень для навчання моделі, і встановлено ім'я шару = 'input_layer'.

Шар Dropout

Шар Dropout (шар регуляризації) використовується для запобігання перенавчанню моделі. Він випадково встановлює вхідні дані рівними нулю із заданою ймовірністю на кожному оновленні в процесі навчання, що дозволяє моделі уникнути сильної залежності від окремих ознак вхідних даних і робить її більш стійкою до шуму та змін у даних.

Приклад

```
# Створення шаруDropout
dropout_layer = tf.keras.layers.Dropout(0.2)

# Визначення моделі
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(784,)),
    dropout_layer,
    tf.keras.layers.Dense(10, activation='softmax')
])

# Компиляції моделі
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=
['accuracy'])
```

У прикладі створено шар Dropout з ймовірністю 0.2, що означає, що на кожному оновленні вхідні дані будуть випадково встановлюватися в нуль з ймовірністю 0.2. Потім визначенамодель, яка починається з повнозв'язкового шару Dense з 64 нейронами та функцією активації ReLU, потім застосовується шар Dropout і завершуємо модель з повнозв'язковим шаром Dense та функцією активації softmax для багатокласової класифікації.

Параметр rate визначає можливість того, що кожна вхідна одиниця буде встановлена в нуль. У прикладі rate = 0.2, що означає, що кожен вхідний нейрон буде встановлений на нуль з ймовірністю 0.2 на кожному оновленні.

Шар Dropout також може приймати низку інших параметрів, таких як:

- noise_shape: форма шуму, що додається до даних
- seed: seed для генерації випадкових чисел

Наприклад, можена встановити параметр noise_shape, щоб додати шум лише у певні ознаки:

```
# Створення шару Dropout с noise_shape
dropout_layer = tf.keras.layers.Dropout(0.2, noise_shape=(None, 1))
```

У цьому випадку встановлюється шум тільки для кожного елемента вхідних даних вздовж першої розмірності, що може бути корисним, якщо бажано додати шум тільки певні ознаки, наприклад, для запобігання перенавчання на викиди в даних.

Шари активації Layer activation functions

Шар активації в нейронній мережі використовується для введення нелінійності у вихідні значення нейронів. Це дозволяє моделі вивчати більш складні та абстрактні залежності між вхідними та вихідними даними.

Одним із типів активації є LReLU (Leaky ReLU), який є варіацією на функції активації ReLU. Функція LReLU визначається так:

LReLU(x) = max(ax, x)

де x – вхідні дані, a – коефіцієнт витоку (leakage coefficient), який зазвичай встановлюється на невелике позитивне значення, наприклад, 0.01.

Коли x<0, LReLU повертає ax замість 0, що дозволяє уникнути проблеми "мертвих нейронів" (dead neurons), яка може виникнути при використанні ReLU. Мертвий нейрон - це нейрон, який завжди повертає 0, і який не може робити свій внесок у вихідні значення нейронної мережі, що може призвести до погіршення продуктивності.

Приклад

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation=tf.keras.layers.LeakyReLU(alpha=0.01),
input_shape=(784,)),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=
['accuracy'])
```

У прикладі створено нейронну мережу, яка починається з повнозв'язкового шару Dense з 64 нейронами та функцією активації LReLU з коефіцієнтом витоку 0.01, потім завершується повнозв'язковим шаром Dense з функцією активації softmax для багатокласової класифікації.

Шар LReLU також може приймати інші параметри, такі як:

- alpha: коефіцієнт витоку, який вказує, наскільки більшим має бути негативне значення x перед тим, як LReLU почне повертати ax замість 0. У встановлено alpha=0.01.