Sequential

API модели Sequential

Для начала прочтите это руководство по модели Keras Sequential.

Методы последовательной модели

compile

```
compile(optimizer, loss=None, metrics=None,
loss_weights=None, sample_weight_mode=None,
weighted_metrics=None, target_tensors=None)
```

Настраивает модель Keras Sequential для обучения.

Аргуметы

- **optimizer**: Строка (имя оптимизатора) или экземпляр оптимизатора. См. оптимизаторы.
- **loss**: Строка (название объективной функции) или объективная функция или экземпляр потерь См. Ошибки. Если модель имеет несколько выходов, можно использовать разные потери на каждом выходе, передавая словарь или список потерь. Значение потерь, которое будет минимизировано моделью, будет представлять собой сумму всех индивидуальных потерь.
- **metrics**: Перечень метрик, которые будут оцениваться моделью во время обучения и тестирования. Обычно вы будете использовать metrics= ['accuracy']. Чтобы задать различные метрики для разных выходов мульти-выходной модели, можно также передать словарь, такой как metrics= {'output_a': 'accuracy', 'output_b': ['accuracy', 'mse']}. Вы также можете передать список (len = len(outputs)) списков метрик, таких как metrics=[['accuracy'], ['accuracy', 'mse']] or metrics=['accuracy', ['accuracy', 'mse']].
- **loss_weights**: Необязательный список или словарь с указанием скалярных коэффициентов (Python floats) для взвешивания вкладов в потери

различных выходов модели. Значение потерь, которое будет минимизировано моделью, будет затем взвешенной суммой всех индивидуальных потерь, взвешенных по коэффициентам loss_weights. Если список, то ожидается, что на выходы модели будет наложено отображение 1:1. Если необходимо, то ожидается, что на выходе будут отображены имена (строки) по скалярным коэффициентам..

- **sample_weight_mode**: Если вам необходимо выполнить взвешивание образца по времени (2D веса), установите это значение в «временное». По умолчанию весов образца (1D) нет. Если модель имеет несколько выходов, можно использовать разные режимы sample_weight_mode на каждом выходе, передавая словарь или список режимов.
- **weighted_metrics**: Список метрик, которые должны быть оценены и взвешены по sample_weight или class_weight во время обучения и тестирования.
- **target_tensors**: По умолчанию Keras создаст плацдарм для целевого объекта модели, который будет поставляться с целевыми данными во время обучения. Если вместо этого Вы хотите использовать собственные тензоры целей (в свою очередь, Keras не будет ожидать внешних данных Numpy для этих целей во время обучения), Вы можете указать их с помощью аргумента target_tensors. Это может быть один тензор (для модели с одним выходом), список тензоров или диктат, отображающий выходные имена целевых тензоров.
- ****kwargs**: При использовании бэкендов Theano/CNTK эти аргументы передаются в функцию K.function. При использовании бэкенда TensorFlow эти аргументы передаются в функцию tf.Session.run.

Исключения

- **ValueError**: В случае недействительных аргументов для optimizer, loss, metrics or sample_weight_mode.

---

fit

fit(x=**None**, y=**None**, batch_size=**None**, epochs=1, verbose=1, callbacks=**None**, validation_split=0.0, validation_data=**None**, shuffle=**True**, class_weight=**None**, sample_weight=**None**, initial_epoch=0, steps_per_epoch=**None**, validation_steps=**None**, validation_freq=1, max_queue_size=10, workers=1, use_multiprocessing=**False**)

Обучает модель Keras Sequential для фиксированного количества эпох (итераций на наборе данных).

Аргументы

- x: Входные данные. Возможно:
- Массив Numpy (или массив типа), или список массивов (в случае, если модель имеет несколько входов).
- В случае, если модель имеет несколько входов, диктуем отображение имен входных данных на соответствующие массивы/тензоры.
- Генератор или keras.utils.Sequence (входы, цели) или (входы, цели, веса сэмплов).
- Нет (по умолчанию), если питание осуществляется от собственных тензоров фреймворка (например, тензоры данных TensorFlow).
- y: Целевые данные. Как и входные данные x, это могут быть либо массив(ы) Нампи, независимый от структуры тензор(ы), список массивов Нампи (если модель имеет несколько выходов) или нет (по умолчанию), если питание осуществляется от независимого от структуры тензора (например, тензоры данных TensorFlow). Если выходные слои в модели имеют имена, можно также передать в Numpy массивы имена выходных слоев, отображенные в словаре. Если x — генератор, или keras.utils.Sequence экземпляр, то y указывать не следует (так как цели будут получены из x).
- **batch_size**: Целое или нет. Количество образцов на обновление градиента. Если не указано, то по умолчанию параметр batch_size будет равен 32. Не указывайте размер переменной batch_size, если ваши данные представлены в виде символических тензоров, генераторов или экземпляров последовательности (поскольку они генерируют партии).
- **epochs**: Целое. Количество эпох для обучения модели. Эпоха — это итерация по всем предоставленным данным x и y. Обратите внимание, что в сочетании с initial_epoch, эпохи должны пониматься как «конечная эпоха». Модель не тренируется для ряда итераций, заданных эпохами, а только до тех пор, пока не будет достигнута эпоха индексов.
- **verbose**: Целое. 0, 1 или 2. Вербозный режим. 0 = бесшумный, 1 = шкала прогресса, 2 = одна строка в эпоху.
- **callbacks**: Список keras.callbacks.callback экземпляров. Список обратных вызовов, которые должны применяться во время обучения и проверки (если). См. список обратных вызовов.
- **validation_split**: Поплавок между 0 и 1. часть данных об обучении, которая будет использоваться в качестве данных для проверки. Модель выделит эту часть тренировочных данных, не будет тренироваться на ней, и оценит потери и любые метрики модели по этим данным в конце каждой эпохи. Валидационные данные выбираются из последних выборок в предоставленных данных x и y, перед перетасовкой. Этот аргумент не поддерживается, когда x является генератором или экземпляром последовательности.

validation_data: Данные, по которым оцениваются потери и любые показатели модели в конце каждой эпохи. Модель не будет обучена работе с этими данными.

данные validation_data будут переопределять данные validation_split. данные validation_data могут быть: — кортеж (x_val, y_val) нулевых массивов или тензоры — кортеж (x_val, y_val, val_sample_weights) нулевых массивов — набор данных или итератор набора данных.

Для первых двух случаев необходимо указать batch_size. В последнем случае должны быть предоставлены validation_steps.

•

shuffle: Булевы (чтобы перетасовать тренировочные данные перед каждой эпохой) или str (для «batch»). «Batch» — это специальная опция для работы с ограничениями данных HDF5; он тасуется в кусках пакетного размера. Не имеет эффекта, когда steps_per_epoch не None.

- **class_weight**: Необязательное отображение индексов класса (целых чисел) по словарю в значение веса (плавающего), используемое для взвешивания функции потерь (только во время обучения). Это может быть полезно для того, чтобы сказать модели «обратить больше внимания» на выборки из недопредставленного класса.
- **sample_weight**: Опциональный массив весов Numpy для сэмплов обучения, используемый для взвешивания функции потерь (только во время обучения). Можно либо передать плоский (1D) Numpy-массив с той же длиной, что и входные образцы (1:1 отображение между весами и образцами), либо, в случае временных данных, передать 2D-массив с формой (образцы, sequence_length), чтобы к каждому временному отрезку каждого образца применить разный вес. В этом случае в функции compile() необходимо обязательно указать sample_weight_mode=»temporal». Этот аргумент не поддерживается, если генератор x или экземпляр Sequence вместо этого предоставляет sample_weights в качестве третьего элемента x.
- **initial_epoch**: Целый. Эпоха, с которой следует начинать обучение (полезно для возобновления предыдущего тренировочного заезда)..
- **steps_per_epoch**: Целое или нет. Общее количество шагов (партий образцов) перед объявлением одной эпохи законченной и началом следующей. При обучении с помощью входных тензоров, таких как тензоры данных TensorFlow, значение по умолчанию None равно количеству отсчетов в наборе данных, деленному на размер партии, или 1, если это невозможно определить.
- **validation_steps**: Актуально только в том случае, если указан параметр steps_per_epoch. Общее количество шагов (партий образцов) для проверки перед остановкой.

- **validation_steps**: Актуально только в том случае, если данные validation_data предоставлены и являются генератором. Общее количество шагов (партий сэмплов) перед остановкой при выполнении валидации в конце каждой эпохи.
- **validation_freq**: Актуально только в случае предоставления данных проверки. Целое число или список/тройка/набор. Если целое число, то оно определяет, сколько эпох должно пройти обучение перед новым запуском валидации, например, valid_freq=2 запускает валидацию каждые 2 эпохи. Если список, кортеж или набор определяет эпохи, в которых будет проходить проверка, например, validation_freq=[1, 2, 10] запускает проверку в конце 1-й, 2-й и 10-й эпох.
- **max_queue_size**: Целый. Используется только для генератора или keras.utils.Sequence входа. Максимальный размер для очереди генератора. Если не указано, то по умолчанию max_queue_size равен 10.
- **workers**: Целый. Используется только для генератора или keras.utils.Sequence входа. Максимальное количество процессов для раскрутки при использовании потоковой обработки на основе процессов. Если не указано, то по умолчанию workers будут равны 1. Если 0, то генератор будет выполняться в главном потоке.
- **use_multiprocessing**: Булев. Используется только для генератора или keras.utils.Sequence входа. Если Правда, используйте потоковую обработку на основе процесса. Если не указано, то по умолчанию use_multiprocessing будет False. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, вы не должны передавать генератору непиклеющиеся аргументы, так как они не могут быть легко переданы дочерним процессам.
- ****kwargs**: Используется для обратной совместимости.

Возврат

Объект History. Его атрибут History.history представляет собой запись обучающих значений потерь и метрик в последующие эпохи, а также значений потерь проверки и метрик проверки (если применимо).

Исключения

- **RuntimeError**: Если модель Keras Sequential так и не была скомпилирована.
- **ValueError**: В случае несоответствия между предоставленными входными данными и тем, что ожидает модель.

---

evaluate

evaluate(x=**None**, y=**None**, batch_size=**None**, verbose=1, sample_weight=**None**, steps=**None**, callbacks=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**)

Возвращает значение потерь и метрики для модели Keras Sequential в тестовом режиме.

Вычисление производится партиями.

Аргументы

- x: Входные данные. Возможно:
- Массив Numpy (или массив типа), или список массивов (в случае, если модель имеет несколько входов).
- В случае, если модель имеет несколько входов, диктуем отображение имен входных данных на соответствующие массивы/тензоры.
- Генератор или keras.utils.Sequence return (входы, цели) или (входы, цели, веса дискретов).
- Нет (по умолчанию), если питание осуществляется от собственных тензоров фреймворка (например, тензоры данных TensorFlow).
- y: Целевые данные. Как и входные данные x, это могут быть либо массив(ы) Нампи, независимый от структуры тензор(ы), список массивов Нампи (если модель имеет несколько выходов) или нет (по умолчанию), если питание осуществляется от независимого от структуры тензора (например, тензоры данных TensorFlow). Если выходные слои в модели имеют имена, можно также передать в Numpy массивы имена выходных слоев, отображенные в словаре. Если x — генератор, или keras.utils.Sequence экземпляр, то y указывать не следует (так как цели будут получены из x).
- batch_size: Целое или Нет. Количество отсчетов на обновление градиента. Если не указано, то по умолчанию параметр batch_size будет равен 32. Не указывайте размер переменной batch_size, если ваши данные представлены в виде символических тензоров, генераторов или экземпляров keras.utils.Sequence (поскольку они генерируют партии).
- verbose: 0 или 1. вербозный режим. 0 = беззвучный, 1 = шкала прогресса.
- sample_weight: Дополнительный массив весов Numpy для тестовых образцов, используемый для взвешивания функции потерь. Можно либо передать плоский (1D) Numpy-массив с той же длиной, что и входные образцы (1:1 отображение между весами и образцами), либо, в случае временных данных, передать 2D-массив с формой (образцы, sequence_length), чтобы к каждому временному шагу каждого образца применить разный вес. В этом случае в функции compile() необходимо обязательно указать sample_weight_mode=»temporal».

- steps: Целое или нет. Общее количество шагов (партий отсчетов) перед объявлением раунда оценки завершенным. Игнорируется значением по умолчанию None.
- callbacks: Список keras.callbacks.callback экземпляров. Список обратных вызовов, которые будут применяться во время оценки. См. список обратных вызовов.
- max_queue_size: Целое. Используется только для ввода генератора или keras.utils.Sequence. Максимальный размер для очереди генератора. Если не указано, по умолчанию max_queue_size будет равен 10.
- workers: Целое число. Используется только для ввода генератора или keras.utils.Sequence. Максимальное количество процессов для раскрутки при использовании потоковой обработки процессов. Если не указано, то по умолчанию рабочие будут равны 1. Если 0, то генератор будет выполняться в главном потоке.
- use_multiprocessing: Булева. Используется только для ввода генератора или keras.utils.Sequence. Если Правда, используйте потоковую обработку на основе процесса. Если не указано, то по умолчанию use_multiprocessing будет False. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, вы не должны передавать генератору невыбираемые аргументы, так как они не могут быть легко переданы дочерним процессам.

Исключеия

- **ValueError**: в случае недействительных аргументов.

Возврат

Скалярные тестовые потери (если модель имеет один выход и не имеет метрик) или список скаляров (если модель имеет несколько выходов и/или метрик). Атрибут model.metrics_names даст вам метки отображения для скалярных выходов.

---

predict

predict(x, batch_size=**None**, verbose=0, steps=**None**, callbacks=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**)

Генерирует предсказания для входных сэмплов.

Вычисление производится партиями.

Аргументы

- x: Входные данные. Возможно:
- Массив Numpy (или массив типа), или список массивов (в случае, если модель имеет несколько входов).
- Словарь, сопоставляющий имена входных данных с соответствующими массивами/тензорами, если модель имеет имена входных данных.
- Генератор или keras.utils.Sequence return (входы, цели) или (входы, цели, веса дискретов).
- None (по умолчанию), если питание осуществляется от собственных тензоров фреймворка (например, тензоры данных TensorFlow).
- batch_size: Целое или нет. Количество образцов на обновление градиента. Если не указано, то по умолчанию параметр batch_size будет равен 32. Не указывайте размер переменной batch_size, если ваши данные представлены в виде символических тензоров, генераторов или экземпляров keras.utils.Sequence (поскольку они генерируют партии).
- verbose: Режим вербозности, 0 или 1.
- steps: Общее количество шагов (партий образцов) перед объявлением раунда предсказания законченным. Игнорируется значением по умолчанию None.
- callbacks: Список keras.callbacks.callback экземпляров. List of callbacks to apply during forecast (Список обратных вызовов, применяемых во время прогнозирования). См. список обратных вызовов.
- max_queue_size: Целое. Используется только для ввода генератора или keras.utils.Sequence. Максимальный размер для очереди генератора. Если не указано, по умолчанию max_queue_size будет равен 10.
- workers: Целое число. Используется только для ввода генератора или keras.utils.Sequence. Максимальное количество процессов для раскрутки при использовании потоковой обработки процессов. Если не указано, то по умолчанию рабочие будут равны 1. Если 0, то генератор будет выполняться в главном потоке.
- use_multiprocessing: Булева. Используется только для ввода генератора или keras.utils.Sequence. Если Правда, используйте потоковую обработку на основе процесса. Если не указано, то по умолчанию use_multiprocessing будет False. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, вы не должны передавать генератору невыбираемые аргументы, так как они не могут быть легко переданы дочерним процессам.

Возврат

Numpy массив(ы) предсказаний.

Исключения

- **ValueError**: В случае несоответствия между предоставленными входными данными и ожиданиями модели, или в случае получения статистической моделью количества отсчетов, не кратного размеру партии.

---

train_on_batch

train_on_batch(x, y, sample_weight=**None**, class_weight=**None**, reset_metrics=**True**)

Выполняет обновление одного градиента на одном пакете данных.

Аргументы

- x: Numpy массив обучающих данных, или список массивов Numpy, если модель имеет несколько входов. Если все входы в модели имеют имена, Вы также можете передать словарь, отображающий имена входов в массивы Numpy.
- y: Массив целевых данных или список массивов Numpy, если модель имеет несколько входов. Если все выходы в модели именованы, Вы можете также передать Numpy массивам словарь, отображающий имена выходных данных.
- sample_weight: Необязательный массив той же длины, что и x, содержащий веса, которые будут применяться к потерям модели для каждой выборки. В случае временных данных, Вы можете передать 2D массив с формой (samples, sequence_length), чтобы применить к каждому временному шагу каждого образца разный вес. В этом случае в функции compile() необходимо обязательно указать sample_weight_mode=»temporal».
- class_weight: Необязательное словарное отображение индексов классов (целых чисел) к весу (float) для применения к потерям модели для образцов из этого класса в процессе обучения. Это может быть полезно для того, чтобы сказать модели «обратить больше внимания» на выборки из недопредставленного класса.
- reset_metrics: Если переменная True, то возвращаемые метрики будут только для данной партии. Если False, то метрики будут статистически накапливаться по партиям.

Возврат

Скалярные потери при обучении (если модель имеет один выход и не имеет метрик) или список скаляров (если модель имеет несколько выходов и/или метрик). Атрибут model.metrics_names даст вам метки отображения для скалярных выходов.

---

test_on_batch

test_on_batch(x, y, sample_weight=**None**, reset_metrics=**True**)

Тестируем модель Keras Sequential на одной партии образцов.

Аргументы

- **x:** Numpy массив тестовых данных, или список массивов Numpy, если модель имеет несколько входов. Если все входные данные в модели имеют имена, Вы также можете передать словарь, отображающий имена входных данных в массивы Numpy.
- **y:** Массив целевых данных или список массивов Numpy, если модель имеет несколько входов. Если все выходы в модели именованы, Вы можете также передать Numpy массивам словарь, отображающий имена выходных данных.
- **sample_weight**: Необязательный массив той же длины, что и x, содержащий веса, которые будут применяться к потерям модели для каждой выборки. В случае временных данных, Вы можете передать 2D массив с формой (samples, sequence_length), чтобы применить к каждому временному шагу каждого образца разный вес. В этом случае в функции compile() необходимо обязательно указать sample_weight_mode=»temporal».
- **reset_metrics:** Если переменная True, возвращаемые метрики будут только для этого пакета. Если False, то метрики будут статистически накапливаться по партиям.

Возврат

Скалярные тестовые потери (если модель имеет один выход и не имеет метрик) или список скаляров (если модель имеет несколько выходов и/или метрик). Атрибут model.metrics_names даст вам метки отображения для скалярных выходов.

---

predict_on_batch

predict_on_batch(x)

Возвращает прогнозы для одной партии образцов.

Аргументы

- **x: Входные сэмплы, как массив Numpy.**

Возврат

Numpy массив(ы) предсказаний.

---

fit_generator

fit_generator(generator, steps_per_epoch=**None**, epochs=1, verbose=1, callbacks=**None**, validation_data=**None**, validation_steps=**None**, validation_freq=1, class_weight=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**, shuffle=**True**, initial_epoch=0)

Поставляет модель Keras Sequential на основе данных, сгенерированных пакетно по пакетам генератором Python (или экземпляром Sequence).

Генератор работает параллельно с моделью, для большей эффективности. Например, это позволяет выполнять увеличение данных на изображениях на CPU в реальном времени параллельно с обучением модели на GPU.

Использование keras.utils.Sequence гарантирует упорядочение и однократное использование каждого входа в эпоху при использовании use_multiprocessing=True.

Аргументы

- generator: Генератор или экземпляр объекта Sequence (keras.utils.Sequence), чтобы избежать дублирования данных при использовании многопроцессорной обработки. Выходной сигнал генератора должен быть либо
- кортеж (inputs, targets)
- кортеж (inputs, targets, sample_weights).
- Этот кортеж (один выход генератора) составляет одну партию. Поэтому все массивы в этом кортеже должны иметь одинаковую длину (равную размеру этой партии). Различные партии могут иметь разные размеры. Например, последняя партия эпохи обычно меньше остальных, если размер набора данных не делится на размер партии. Предполагается, что генератор будет перебирать данные бесконечно. Эпоха заканчивается, когда модель видит партии steps_per_epoch.
- Step_per_epoch: Целое. Общее количество шагов (партий образцов) для выхода из генератора до объявления одной эпохи законченной и начала следующей. Обычно оно должно быть равно ceil(num_samples / batch_size) опционально для Sequence: если не указано, будет использоваться len(generator) как количество шагов.

- epochs: целое. Количество эпох для обучения модели. Эпоха — это итерация по всем предоставленным данным, определяемая с помощью функции step_per_epoch. Обратите внимание, что в сочетании с файлом initial_epoch, эпохи должны пониматься как «конечная эпоха». Модель не обучена для ряда итераций, заданных эпохами, а только до тех пор, пока не будет достигнута эпоха индексов.
- verbose: Целочисленный. 0, 1 или 2. Вербозный режим. 0 = беззвучный, 1 = шкала прогресса, 2 = одна строка на эпоху.
- callbacks: Список keras.callbacks.callback экземпляров. Список обратных вызовов для применения во время тренировки. См. список обратных вызовов.
- validation_data: Это может быть либо
- generator или объект Sequence для данных проверки.
- кортеж (x_val, y_val)
- кортеж (x_val, y_val, val_sample_weights)
- по которым оцениваются потери и любые метрики модели в конце каждой эпохи. Модель не будет обучена работе с этими данными.
- validation_steps: Актуально только в том случае, если данные validation_data являются генератором. Общее количество шагов (партий сэмплов) для получения данных из генератора valid_data до остановки в конце каждой эпохи. Обычно оно должно быть равно количеству отсчетов вашего набора данных валидации, деленному на размер партии. Необязательно для Sequence: если не указано, будет использоваться len(validation_data) в качестве ряда шагов.
- validation_freq: релевантно только в том случае, если данные валидации предоставлены. Целочисленный или сборный экземпляр контейнера (например, список, кортеж и т.д.). Если целое число, определяет, сколько эпох обучения должно быть выполнено перед новым запуском валидации, например, valid_freq=2 запускает валидацию каждые 2 эпохи. Если Контейнер, укажите эпохи, в которых будет проходить валидация, например, validation_freq=[1, 2, 10] запускает валидацию в конце 1-й, 2-й и 10-й эпох.
- class_weight: Необязательное отображение индексов класса (целых чисел) по словарю в значение веса (float), используемое для взвешивания функции потерь (только в процессе обучения). Это может быть полезно для того, чтобы сказать модели «обратить больше внимания» на выборки из недопредставленного класса.
- max_queue_size: Целое. Максимальный размер для очереди генератора. Если не указано, то по умолчанию max_queue_size будет равен 10.
- workers: Целое число. Максимальное количество процессов для раскрутки при использовании многопоточности на основе процессов. Если не указано, то по умолчанию рабочие будут равны 1. Если 0, то генератор будет выполняться в главном потоке.

- use_multiprocessing: Булева. Если Правда, используйте многопотоковую обработку на основе процессов. Если не указано, то по умолчанию use_multiprocessing будет False. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, вы не должны передавать генератору непередаваемые аргументы, так как они не могут быть легко переданы дочерним процессам.
- shuffle: Булева. Следует ли тасовать порядок партий в начале каждой эпохи. Используется только с экземплярами последовательности (keras.utils.Sequence). Не имеет эффекта, когда steps_per_epoch не None.
- initial_epoch: Целое. Эпоха, с которой следует начать тренировку (полезно для возобновления предыдущего тренировочного цикла)..

Returns

A History object. Its History.history attribute is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values (if applicable).

Исключение

- **ValueError**: В случае, если генератор выдает данные в некорректном формате.

Пример

```
def generate_arrays_from_file(path):
  while True:
    with open(path) as f:
      for line in f:
        # создавать Numpy массивы входных данных
        # и метки, из каждой строки в файле.
        x1, x2, y = process_line(line)
        yield ({'input_1': x1, 'input_2': x2}, {'output': y})

model.fit_generator(generate_arrays_from_file('/my_file.txt'),
```

steps_per_epoch=10000, epochs=10)

---

evaluate_generator

evaluate_generator(generator, steps=**None**, callbacks=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**, verbose=0)

Оценивает модель Keras Sequential на генераторе данных.

Генератор должен возвращать данные того же типа, что и test_on_batch.

Аргументы

generator: Генератор, выдающий кортежи (входы, цели) или (входы, цели, sample_weights) или экземпляр объекта Sequence (keras.utils.Sequence), чтобы избежать дублирования данных при использовании многопроцессорной обработки.

steps: Общее количество шагов (партий образцов) для выхода из генератора до остановки. Необязательно для Sequence: если не указано, будет использоваться len(generator) в качестве нескольких шагов.

callbacks: Список экземпляров keras.callbacks.callback. List of callbacks to apply during training (Список обратных вызовов, применяемых во время тренировки). См. callbacks.

max_queue_size: максимальный размер очереди генератора.

workers: Целостный. Максимальное количество процессов для раскрутки при использовании многопоточности на основе процессов. Если не указано, то по умолчанию рабочие будут равны 1. Если 0, то генератор будет выполняться в главном потоке.

use_multiprocessing: если значение True, использовать многопоточность на основе процессов. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, не следует передавать генератору непикируемые аргументы, так как они не могут быть легко переданы дочерним процессам.

verbose: verbosity mode, 0 или 1.

Возврат.

Скалярные тестовые потери (если модель имеет один выход и не имеет метрик) или список скаляров (если модель имеет несколько выходов и/или метрик). Атрибут model.metrics_names даст вам метки отображения для скалярных выходов.

Исключения

- **ValueError**: В случае, если генератор выдает данные в некорректном формате.

---

predict_generator

predict_generator(generator, steps=**None**, callbacks=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**, verbose=0)

Генерирует предсказания для входных образцов из генератора данных.

Генератор должен возвращать данные того же вида, что и принятый в predict_on_batch.

Аргументы

- **generator:** Генератор, выдающий партии входных отсчетов или экземпляр объекта Sequence (keras.utils.Sequence), чтобы избежать дублирования данных при использовании мультипроцессорной обработки.
- **steps:** Общее количество шагов (партий образцов) для получения из генератора до остановки. Необязательно для Sequence: если не указано, будет использоваться len(generator) в качестве нескольких шагов.
- **обратные вызовы:** Список экземпляров keras.callbacks.callback. Список обратных вызовов, применяемых во время обучения. См. callbacks.
- **max_queue_size:** Максимальный размер очереди генератора.
- **сотрудники:** Целочисленный. Максимальное количество процессов для раскрутки при использовании многопоточности на основе процессов. Если не указано, то по умолчанию рабочие будут равны 1. Если 0, то генератор будет выполняться в главном потоке.
- **use_multiprocessing**: Если верно, использовать многопотоковую обработку на основе процесса. Обратите внимание, что поскольку эта реализация основана на многопроцессорной обработке, не следует передавать генератору непикируемые аргументы, так как они не могут быть легко переданы дочерним процессам.
- **verbose:** verbosity mode, 0 или 1.

Возврат

Numpy массив(ы) предсказаний.

Исключения

- **ValueError**: В случае, если генератор выдает данные в некорректном формате.

---

get_layer

get_layer(name=**None**, index=**None**)

Восстанавливает слой на основе его имени (уникального) или индекса.

Если указаны и name, и index, то приоритет будет отдан индексу.

Индексы основываются на порядке обхода горизонтального графика (снизу вверх).

Аргументы

- **name**: Строка, название слоя.
- **index**: Целое, индекс слоя.

Возврат

Экземпляр слоя

Исключения

- **ValueError**: В случае неправильного названия или индекса слоя.