Оглавление

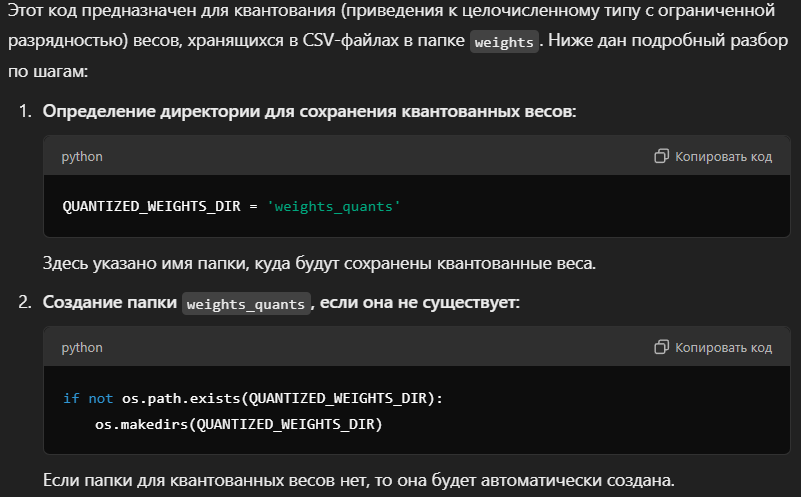
[Первичное описание кода квантизации весов: 1](#_Toc185008663)

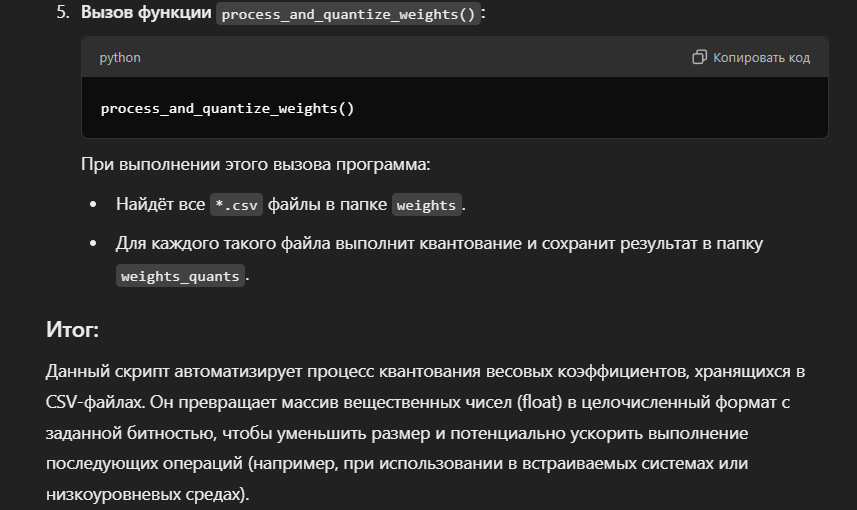
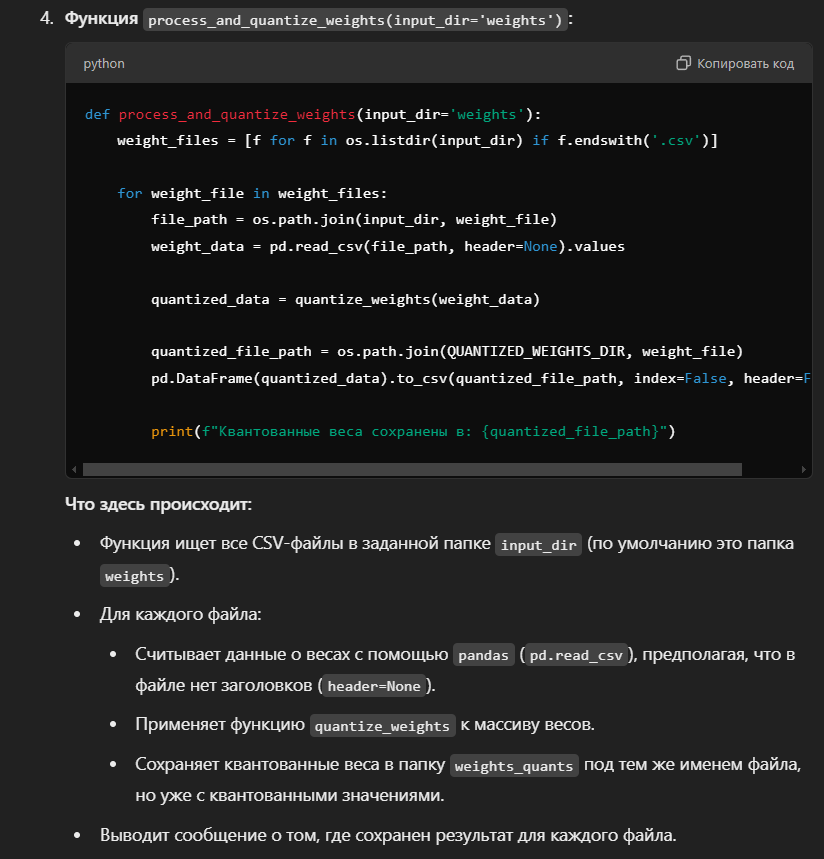
[Детальное описание, отвечающие на некоторые вопросы: 4](#_Toc185008664)

[Пример на реальном значении: 6](#_Toc185008665)

[Ответ на вопрос: «А если веса уже в диапазоне [-1;1]?» 8](#_Toc185008666)

# Первичное описание кода квантизации весов:





# Детальное описание, отвечающие на некоторые вопросы:

1. Я хочу знать числа какого типа и из какого диапазона она преобразует в числа какого типа и из какого диапазона?

2. Для чего нужен масштабный коэффициент?

3. Каким образом определяется масштабный коэффициент, если найдешь научные статьи, где описано как находить его - будет классно

4. Каким образом непосредственно происходит масштабирование, при помощи каких функций и по какой логике?

5. Для чего обрезать значения и при каких условиях это нужно делать?

Ниже приводится детальный разбор работы функции quantize\_weights, отвечая на заданные вопросы по пунктам:

**1. Типы и диапазоны входных и выходных чисел**

Изначально веса (входные данные в weight\_array) представляют собой массив чисел с плавающей точкой (тип float в NumPy, обычно float32 или float64). Их значения могут быть произвольными по величине: от очень маленьких дробных чисел до сравнительно больших.

Функция quantize\_weights преобразует эти вещественные числа в целочисленный формат с заданной разрядностью bit\_depth. Для bit\_depth = 8 допустимый целочисленный диапазон:

* Минимальное значение: -128 (т.е. -2^(8-1))
* Максимальное значение: 127 (т.е. 2^(8-1) - 1)

Иными словами, функция берёт массив вещественных чисел (float) из потенциально широкого диапазона и преобразует их в целочисленные значения (int), умещающиеся в диапазон [-128, 127] для 8 бит.

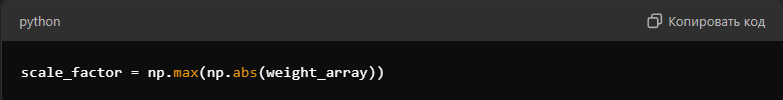
**2. Для чего нужен масштабный коэффициент?**

Масштабный коэффициент (scale\_factor) необходим для того, чтобы отобразить вещественные числа из исходного диапазона в целочисленный диапазон квантования. Поскольку у нас есть ограниченный набор целых чисел (например, от -128 до 127), нам нужно сжать (линейно масштабировать) исходный диапазон весов так, чтобы они поместились внутрь доступного целочисленного диапазона.

Если мы просто округлим вещественные числа, не масштабируя, то большинство значений либо окажутся за пределами допустимого целочисленного диапазона, либо потеряют разрешающую способность. Масштабирование позволяет эффективно использовать весь диапазон целых чисел для представления исходных вещественных значений.

**3. Как определяется масштабный коэффициент?**

В данном коде масштабный коэффициент определяется как:



То есть берётся максимальное абсолютное значение среди всех весов. Таким образом, при делении каждого веса на scale\_factor, мы гарантируем, что нормализованные значения будут лежать в диапазоне [-1, 1]. Затем, умножив на максимальное целое значение (max\_val, например 127), мы растягиваем этот диапазон до [-127, 127].

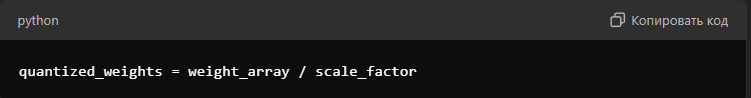
Существует множество научных статей, описывающих различные стратегии выбора масштабирующего коэффициента для квантования. Одна из ключевых работ — "Quantizing deep convolutional networks for efficient inference: A whitepaper" (N. Krishnamoorthi, 2018), доступна на arXiv: <https://arxiv.org/abs/1806.08342>. В ней и других работах по квантованию нейросетей часто рассматривают способы нахождения оптимального масштабирующего фактора, чтобы минимизировать ошибку квантования. Часто применяют следующие подходы:

* Максимальное значение (как в данном коде).
* Среднеквадратичное отклонение или статистики распределения весов.
* Обучаемые масштабирующие факторы (обучение квантованных параметров).

**4. Как происходит масштабирование?**

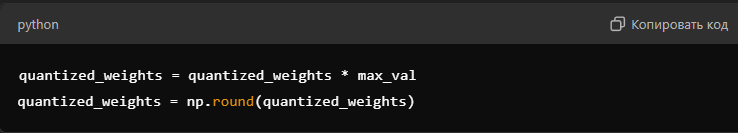
Масштабирование делается в два шага:

1. Нормализация к диапазону [-1, 1]:



Здесь scale\_factor — максимум абсолютных значений. Делением каждого элемента мы приводим значения к диапазону примерно от -1 до 1.

1. Растяжение к целочисленному диапазону:



Теперь мы умножаем нормализованный массив на max\_val (например, 127), чтобы значения заняли весь доступный целочисленный диапазон. После умножения производится округление (np.round) до ближайшего целого числа.

Логика проста: максимальное абсолютное значение становится границей, соответствующей max\_val. Все остальные значения пропорционально сжимаются.

**5. Для чего обрезать значения (np.clip)?**

После округления некоторые значения теоретически могут выйти за пределы целочисленного диапазона из-за погрешностей округления или иных эффектов. Использование np.clip(quantized\_weights, min\_val, max\_val) гарантирует, что все значения останутся в допустимых границах. Это особенно важно, когда значения очень близки к границе диапазона.

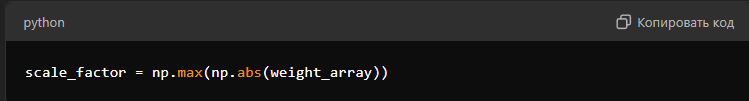
Ситуация, когда это необходимо:

* Если после масштабирования и округления какое-то число получилось 128 (а максимальный допустимый — 127), clip гарантирует, что число станет 127.
* То же самое для нижней границы.

Таким образом, clip обеспечивает соответствие результирующего массива заданному целочисленному диапазону. Без clip можно получить неожиданные выходы за пределы разрешённого диапазона, что недопустимо при реализации квантования.

Ниже разберём процесс квантования на примере, опираясь на тот код, который вы привели, и покажем пошаговые вычисления для нескольких исходных значений. Предположим, что квантование происходит с глубиной 8 бит, как указано в вашем коде bit\_depth=8. Принцип квантования следующий:

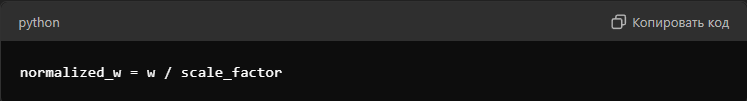
1. **Определение диапазона целочисленного представления**:  
   Для 8 бит знаковый целочисленный диапазон:
   * min\_val = -2^(8-1) = -128
   * max\_val = 2^(8-1) - 1 = 127
2. **Поиск масштабного коэффициента (scale\_factor)**:  
   В коде применяется следующий подход:



То есть scale\_factor — это максимальное по модулю значение из всего массива весов. Предположим, что для данного набора весов максимальное абсолютное значение оказалось равным scale\_factor. (Вы его можете вычислить, взяв максимум из всех абсолютных значений в вашем массиве.)

**Масштабирование весов к диапазону [-1, 1]**:

Каждый вес w нормализуем:



**Перенос в целочисленный диапазон**:

Чтобы использовать весь диапазон целых чисел, умножаем нормализованное значение на max\_val (127 для 8 бит):

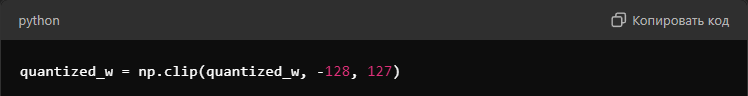


**Округление до целого**:



**Обрезка значений (clip)**:

На случай, если из-за округления некоторые значения вышли за пределы [-128, 127], применяем:



# Пример на реальном значении:

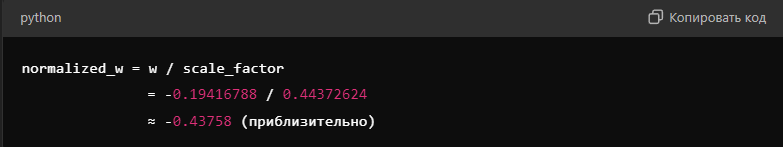
**Пример расчёта на конкретном значении:**

Возьмём одно из первых чисел из исходного набора, например:  
Исходный вес: w = -0.19416788

**Шаг 1: Предположим, что мы уже нашли scale\_factor.**

Для примера допустим, что максимальное абсолютное значение среди всех весов равно scale\_factor ≈ 0.44372624 (как мы примерно оценили по вашим данным).

**Шаг 2: Нормализуем w:**



**Шаг 3: Масштабируем к целочисленному диапазону:**



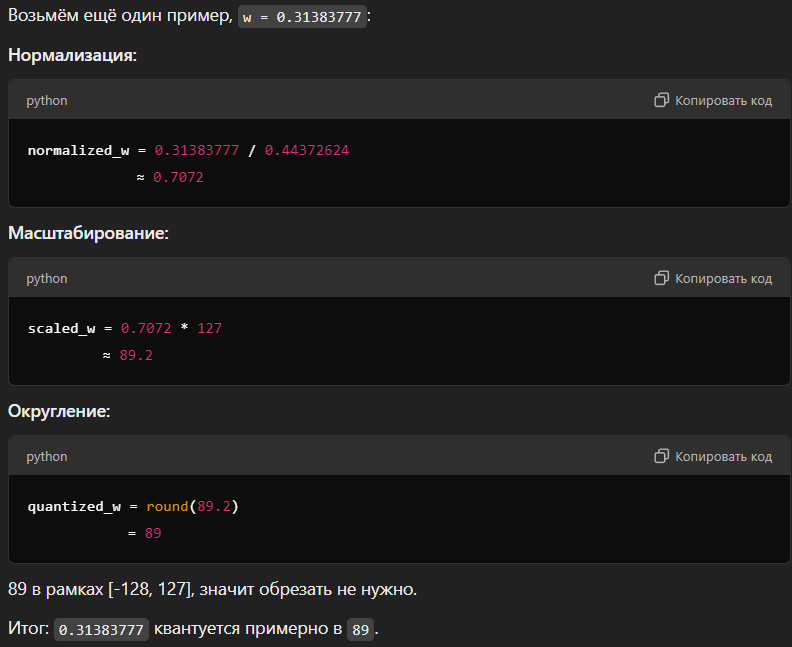
**Шаг 4: Округление:**



**Шаг 5: Обрезка:**

-56 уже находится в диапазоне [-128, 127], так что ничего менять не нужно.

Итог: -0.19416788 квантуется примерно в -56.



# Ответ на вопрос: «А если веса уже в диапазоне [-1;1]?»

Если исходные веса уже лежат в диапазоне [-1; 1], то формально нормализация не меняет их расположения по отношению к этому диапазону. Однако смысл в процедуре масштабирования остаётся, так как квантование рассчитано на использование всего целочисленного диапазона, доступного для выбранной битности.

Вот несколько моментов, почему всё же есть смысл выполнять данный шаг:

1. **Универсальность кода:**  
   Код для квантования обычно пишут универсальным, без привязки к конкретному распределению входных данных. Если однажды ваши веса выходят за пределы [-1;1], метод всё равно будет работать корректно. Таким образом, нормализация — часть общего алгоритма, не зависящая от предположений о диапазоне весов.
2. **Использование полного целочисленного диапазона:**  
   Предположим, даже если веса уже в [-1;1], их максимальное абсолютное значение меньше или равно 1. Допустим, scale\_factor = max(|w|) приблизится к 1. Тогда умножение на max\_val (например, 127 для 8 бит) позволит эффективно растянуть значения на весь доступный целочисленный диапазон. Без этого шага вы могли бы получить менее эффективное использование диапазона квантования (например, если веса уже в [-1;1], просто округление к целым даст значение в {-1,0,1}, что очень грубо).
3. **Корректировка при меньших максимумах:**  
   Если максимальный вес намного меньше 1, скажем 0.1, то без нормализации большая часть целочисленного диапазона останется неиспользованной. Масштабирование же позволит "растянуть" эти значения, чтобы они заняли больше места в целочисленном диапазоне, обеспечивая лучшую разрешающую способность при квантовании.

**Вывод:** Даже если веса уже в [-1;1], масштабирование по scale\_factor всё равно полезно. Оно гарантирует, что будет максимально эффективно использован весь доступный диапазон целочисленных значений, улучшая точность представления весов после квантования.