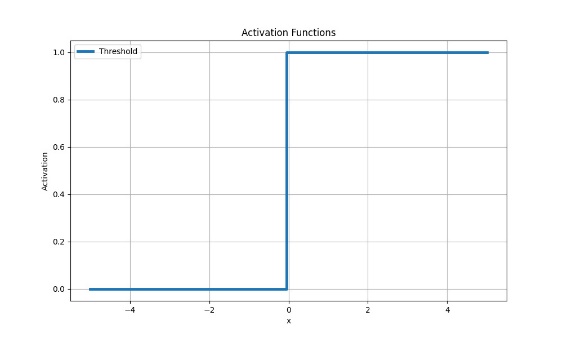
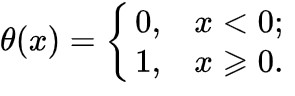
**Функции активации**

Обратите внимание, что и смещение, и функция активации вместе влияют на активацию нейрона. Смещение позволяет установить базовый уровень активации, в то время как функция активации определяет, как будет изменяться активация нейрона в зависимости от входов и смещения.

*Пороговая (ступенчатая) функция*

Самая простая функция активации. Иногда ее называют функцией Хевисайда. Если значение ступенчатой функции превышает некоторый порог, нейрон считается активированным:

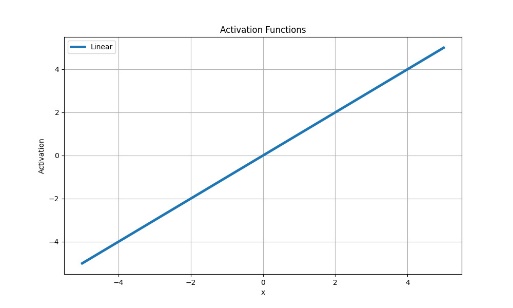


Недостатки. Пороговая функция непрерывна и недифференцируема, в результате чего мы получаем две проблемы, из-за которых в нейросетях с глубоким обучением эта функция почти не используется:

первая проблема связана с алгоритмом обратного распространения ошибки Backpropagation — механизмом обучения нейронной сети. Благодаря ему настраиваются веса связей между нейронами сети на основе вычисленных ошибок предсказания. Вычислить градиент, необходимый для обновления весов в процессе обучения сети, можно только с дифференцируемыми функциями активации (градиент вычисляется с помощью производной функции активации по входу нейрона);

вторая проблема — скачки в функции активации Хевисайда. Они приводят к непрерывным изменениям выходных значений нейронов при небольших изменениях входных данных. В результате сеть может «застревать» в определенных участках нейронной сети, где выходные значения нейронов прыгают между двумя разными значениями. Такое застревание затрудняет обновление весов нейронов в этих регионах, поскольку небольшие изменения весов не приводят к непрерывным изменениям выходов нейронов. В результате сеть может не сместиться в нужном направлении и не достичь оптимальных значений весов.

*Линейная функция*

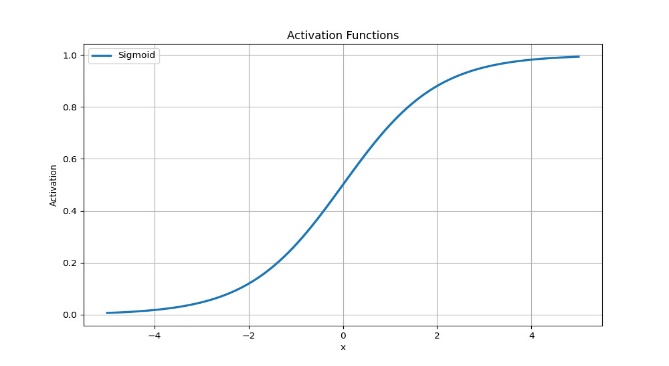
https://robotdreams.cc/ckeditor/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh/%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B02.pngЕще одна простейшая функция активации, в которой изменения входного сигнала пропорционально влияют на выход без каких-либо нелинейных преобразований:

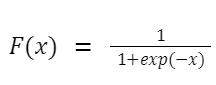
В связи с этим применение линейной функции ограничено задачами регрессии, где необходимо предсказание непрерывного значения. Например, в задаче предсказания цены на недвижимость с учетом разных признаков она может быть применена к выходному слою. Также функция может быть применена к отдельным линейным слоям сети, которые выполняют линейные операции над входами. В этом случае функция активации просто передает значения без нелинейных преобразований.

Пример использования. Предположим, мы решаем задачу прогнозирования продаж товаров на основе их рекламных затрат. Дано: сведения о сумме денег, потраченных на рекламу в различных медиаканалах (раскрутка на телевидении/радио, в социальных сетях); соответствующие сведениям объемы продаж товаров. Конечная цель этой задачи — построение модели, которая бы прогнозировала объем продаж на основе входных данных. Задействуем нейронную сеть с одним линейным слоем и линейной функцией активации. Входными данными будут значения рекламных затрат в различных каналах, а выходными данными — прогнозируемый объем продаж. Нейронная сеть с линейной функцией активации позволит установить линейную зависимость между рекламными затратами и объемом продаж. Модель будет стремиться найти оптимальные веса для каждого канала рекламы, чтобы наилучшим образом предсказывать объем продаж на основе данных о затратах.

Недостатки. Линейная функция активации пригодна только для простых моделей или в качестве заключительного слоя в нейронной сети для решения задач регрессии, где нужно выполнить прогнозирование числового значения.

*Сигмоидная функция (логистическая)*

Логистическая функция нелинейна и сжимает входные значения от 0 до 1 по формуле:

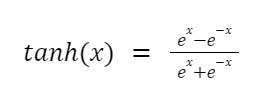
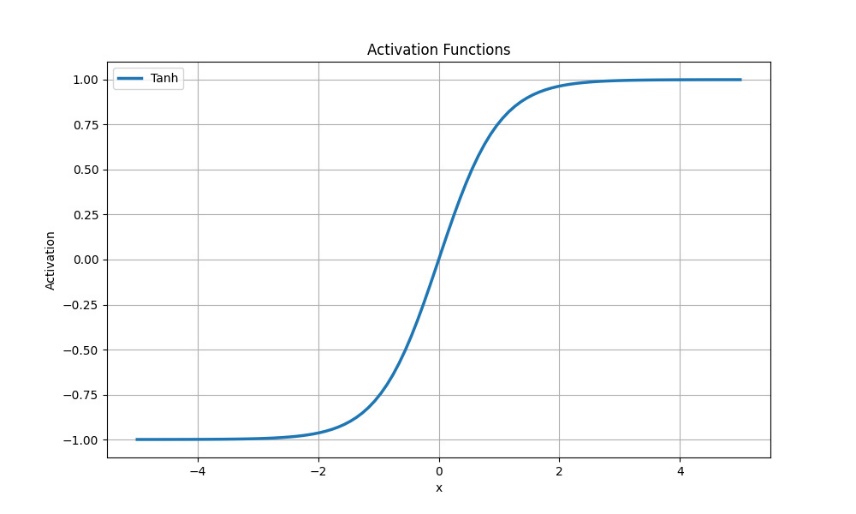
Чаще всего сигмоидная функция активации применяется в задачах бинарной классификации, где нужно предсказать вероятность принадлежности к одному из двух классов. Она конвертирует входные значения в вероятности, которые могут быть интерпретированы как вероятность принадлежности к положительному классу.

Пример использования. Допустим, есть набор данных, состоящий из физических характеристик пациентов. Нам нужно построить модель, которая предскажет, есть ли у пациента болезнь сердца. Для этого мы можем использовать нейронную сеть с одним выходным нейроном и сигмоидной функцией активации: входные данные могут включать возраст пациента, уровень холестерина, давление и другие характеристики; выходной нейрон будет предсказывать вероятность наличия болезни сердца. Сигмоидная функция активации преобразует взвешенную сумму входных значений и весов в вероятность. Например, если выходное значение сигмоидной функции равно 0.89, это означает, что с вероятностью 89 % модель предсказывает наличие проблем с сердечно-сосудистыми заболеваниями.

Недостатки логистической функции. В области сильных положительных или отрицательных значений производная логистической функции принимает малые значения. Это приводит к появлению эффекта затухания градиента при обратном распространении ошибки и слишком малом его значении, что крайне неэффективно для обновления весов нейронов и обучения глубоких нейронных сетей с большим числом слоев.

*Гиперболический тангенс (Tahn)*

Тангенс похож на сигмоиду, ее можно представить так:

https://robotdreams.cc/ckeditor/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh/%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B05.png

С сигмоидой у тангенса есть и схожие особенности, и слабые моменты: они оба нелинейные и сжимающие, и оба ограничены в диапазоне от −1 до 1.

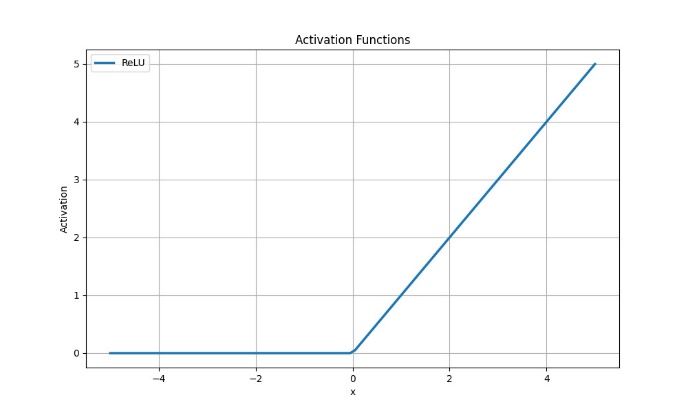
При этом (по сравнению с сигмоидной функцией активации) tanh(x) обладает более заметным градиентом вблизи нуля. Это может быть полезным для моделирования данных с симметричным распределением возле нуля, отрицательные значения которых важно сохранить.

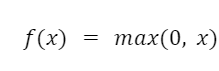
https://robotdreams.cc/ckeditor/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh/%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B07.pnghttps://robotdreams.cc/ckeditor/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh/%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B06.pngНедостатки гиперболического тангенса. Недостатки tanh(x) тоже связаны с проблемой затухания градиента. В процессе использования backpropagation локальный градиент вычисляется путем перемножения производных функций активации слоев нейронной сети. Максимальная производная гиперболического тангенса всегда меньше единицы и вычисляется по формуле: Максимальное значение производной логистической функции активации в точке 0 равна ¼ и вычисляется так:

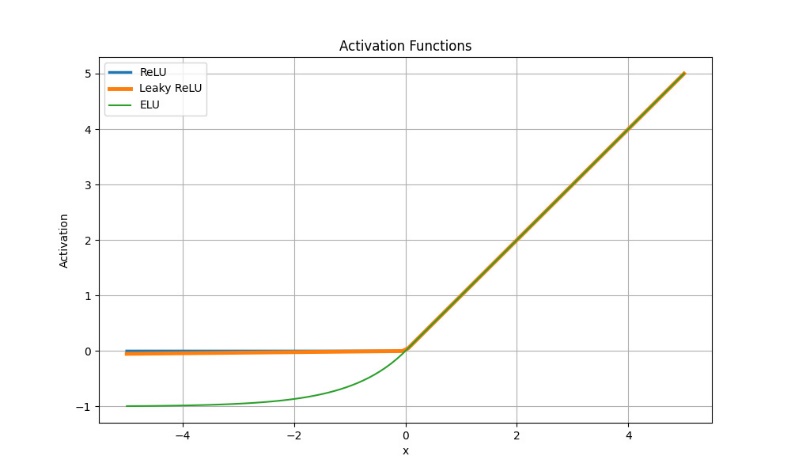
При перемножении производных функций активации для n слоев максимум локального градиента будет уменьшаться с каждым слоем и составит 1/(2n). При увеличении числа слоев в модели сети и соответствующем уменьшении локального градиента веса в более ранних слоях могут оставаться практически без изменений, так как значение градиента становится очень маленьким. Из-за затухания градиента в более ранних слоях обновление весов будет происходить преимущественно в более глубоких слоях, и обучение будет идти с конца нейронной сети. В результате активации седьмого-восьмого слоев мы получим меньший вклад в общий процесс обучения по сравнению с более глубокими слоями, где градиенты сохраняются в более высоких значениях. Вывод: и логистическая функция активации (сигмоидная), и гиперболический тангенс часто не рекомендуются для глубоких нейронных сетей, особенно при использовании большого числа слоев.

Пример использования гиперболического тангенса. Рассмотрим задачу из категории предсказаний временных рядов (временной ряд — это последовательность данных, упорядоченных по времени). Допустим, есть данные о температуре воздуха на протяжении определенного периода времени, и нужно построить модель, которая будет прогнозировать температуру на следующий день. Мы применяем рекуррентную нейронную сеть (RNN), где в качестве функции активации взят tanh(x). У RNN есть память, которая позволяет ей учитывать предыдущие значения и предсказывать будущие значения на основе этой истории.

*Функция активации ReLU*

ReLU (Rectified Linear Unit) — одна из самых популярных функций активации, используемых как в сетях с низким числом слоев, так и в моделях deep learning.

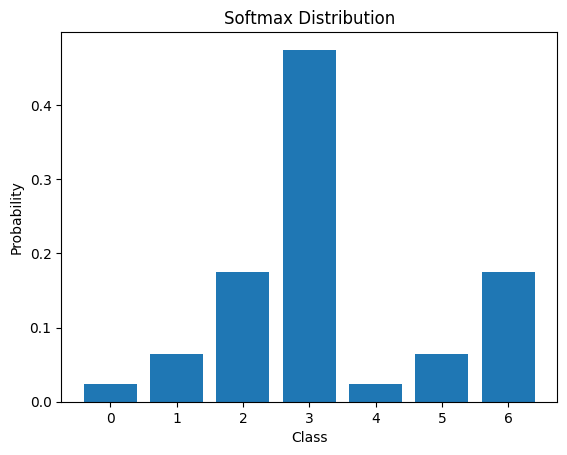
Эта функция активации (равно как и существующие ее модификации) устраняет проблемы с исчезающим градиентом. Градиенты для значений больше нуля остаются ненулевыми, что обеспечивает эффективное распространение градиентов и обновление весов в процессе тренировки.

Полезно знать! У модифицированных функций Leaky ReLU, Parametric ReLU (PReLU) и Exponential Linear Unit (ELU) тоже есть свойства, которые способствуют предотвращению затухания градиента. Например, Leaky ReLU добавляет небольшой наклон для отрицательных значений, а у ELU есть экспоненциальная зависимость отрицательных значений.

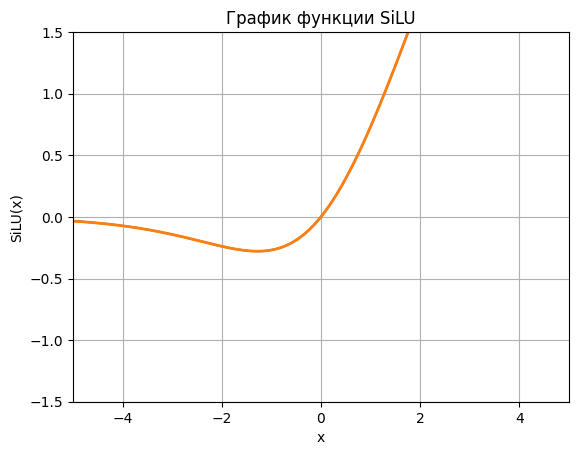
Хотя на графике заметна ее частичная линейность, функция ReLU нелинейна и комбинации с ней также нелинейны. Функция универсальна и пригодна для любых задач, не требуя больших вычислительных мощностей. Все отрицательные значения функция заменяет на ноль и передает положительные значения без изменений. Градиенты для положительных значений остаются ненулевыми, что позволяет эффективно распространять градиенты и обновлять веса во время обучения. Функция ReLU активирует только отдельные нейроны, оставляя остальные неактивными. Благодаря этому сеть может обучаться на более разреженных представлениях данных — и снижается риск переобучения.

Недостатки функции ReLU. Очевидный минус функции активации ReLU — ее ненормализуемость. Так же как и у сигмоидной функции активации, выходное значение ReLU всегда находится в интервале от 0 до ∞, что может быть ограничивающим в некоторых случаях. Как решить проблему: нормализовать данные перед применением функции активации ReLU. Из-за больших значений весов или несбалансированности данных в ReLU (немодифицированных версиях) может проявляться проблема «мертвых нейронов», снижающая производительность в моделях deep learning. Это значит, что если веса для определенных входных данных отрицательны или близки к нулю, то функция ReLU будет активировать нейрон только для положительных значений входа, а остальные останутся пассивными.

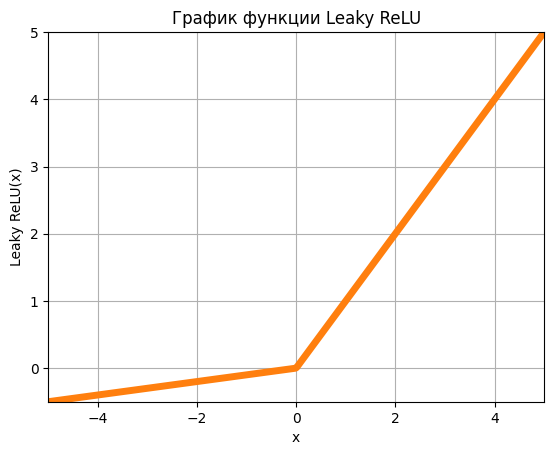
*Softmax*

Функция Softmax используется для преобразования вектора значений в вероятностное распределение, которое суммируется до 1. Она особенно полезна в многоклассовой классификации, где необходимо определить вероятности для каждого класса. График функции Softmax представляет собой гладкую кривую, начинающуюся от 0 и заканчивающуюся на 1, что соответствует сумме вероятностей всех элементов вектора. Кривая функции Softmax имеет свойство, что вероятность любого элемента вектора увеличивается, если значения других элементов уменьшаются, что позволяет использовать эту функцию для многоклассовой классификации. Хотя функция Softmax имеет множество применений в машинном обучении, она также может иметь недостатки, такие как чувствительность к выбросам и несбалансированным данным, что может приводить к неверным вероятностным оценкам.

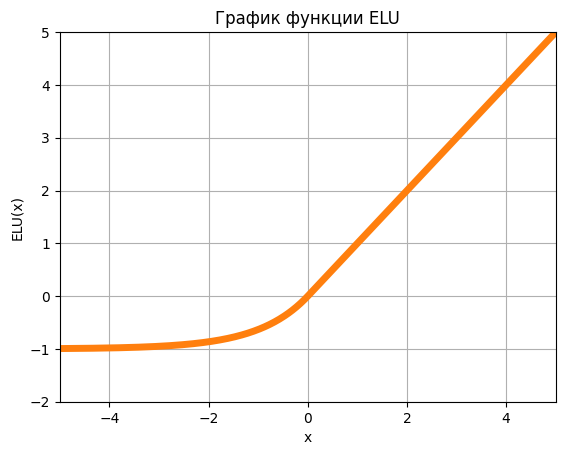
*SiLU (Swish?)*

SiLU - это функция активации, которая решает проблемы "затухание градиентов" и "эффект насыщения". Она является гладкой, монотонно возрастающей и не имеет "эффекта насыщения" как у сигмойдной функции, что позволяет модели обучаться более эффективно и быстро сходиться к оптимальному решению. В области компьютерного зрения SiLU часто используется в сверточных нейронных сетях (CNN), где она может помочь увеличить точность и скорость обучения моделей. Например, её используют в модели YOLOv8. Но, к сожалению, этой модели нет в библиотеки TensorFlow, и использовать её просто так, у вас не получиться. 

*Leaky ReLU*

Обычный ReLU принимает входные значения и преобразует их, оставляя только положительные значения без изменения, а все отрицательные значения заменяет на 0. Однако у этого метода есть один недостаток, а именно "умирание ReLU". Это происходит в том случае, если входное значение отрицательное, то нейрон не будет активироваться и не будет вносить вклад в выходную функцию. Для решения этой проблемы был разработан Leaky ReLU. В отличие от ReLU, Leaky ReLU возвращает само значение при положительном входном значении, а при отрицательных значениях возвращает линейную функцию от входа, умноженную на небольшой коэффициент, называемый отрицательным уклоном (leak). Таким образом, у нейрона всегда есть возможность вносить вклад в выходную функцию, даже если входные данные отрицательны. Преимуществом Leaky ReLU является устойчивость к "умиранию" нейронов и лучшая сходимость в процессе обучения, что приводит к более быстрому и точному обучению нейронных сетей.

*ELU*

Она представляет собой измененную версию ReLU (Rectified Linear Unit), которая помогает ускорить обучение глубоких нейронных сетей и справляется с проблемой "мертвых нейронов" (dead neurons). ELU работает так же, как и ReLU, возвращая исходное значение входа, если он больше нуля. Однако, если значение входа меньше или равно нулю, то ELU использует экспоненциальную функцию, чтобы получить значение, которое ближе к нулю, чем значение, возвращаемое ReLU. Это позволяет избежать "мертвых нейронов" и ускорить обучение глубоких нейронных сетей. Кроме того, ELU имеет свойство гладкости, которое так же помогает избежать проблемы "взрывающегося градиента" (exploding gradient), которая может возникать при использовании других функций активации, таких как ReLU. Это делает ELU более стабильной и более эффективной функцией активации для обучения глубоких нейронных сетей. Однако, как и любая другая функция активации, ELU не подходит для всех задач и может давать неоптимальные результаты в некоторых случаях. Поэтому при выборе функции активации необходимо учитывать особенности конкретной задачи и проводить эксперименты для определения оптимальной функции. 

*Вывод*

Функция активации определяет выходной сигнал нейрона на основе его входа. Мы просто подставляем в нее суммированное значение произведения входных нейронных сигналов и коэффициентов весов — и получаем выходной сигнал нейронной сети. Ниже указаны преимущества и недостатки разных функций активации:

* Пороговая функция бинарна и используется редко по причине своей непрерывности и отсутствия градиента.
* Линейная функция активации передает входной сигнал без изменений, поэтому применяется в простых моделях и не годится для представления сложных нелинейных зависимостей.
* Гиперболический тангенс (tanh) сжимает входные значения в диапазоне от −1 до 1. Он часто используется в скрытых слоях нейронных сетей, сохраняя отрицательные значения и обеспечивая нелинейность.
* Сигмоида преобразует входные значения в диапазоне от нуля до единицы. Она часто используется в задачах классификации, но страдает от проблемы затухающего градиента при обучении глубоких нейронных сетей.
* Функция активации ReLU (Rectified Linear Unit) активирует нейрон, если вход положительный, иначе — выходит 0. Она широко применяется в глубоких нейронных сетях благодаря простоте и отсутствию затухающего градиента, но может вызывать «мертвые нейроны» с отрицательным входом.

*Как выбрать подходящий слой активации?*

Выбор подходящего слоя активации зависит от задачи машинного обучения, типа данных и модели, которую вы хотите создать. Вот несколько рекомендаций, которые могут помочь в выборе подходящего слоя активации:

* Для задач классификации, используйте Softmax, если вы хотите получить вероятности классов в качестве выходных данных. Используйте Sigmoid или Tanh, если вы хотите получить двоичный вывод.
* Для задач регрессии, используйте ReLU или его модификации, такие как LeakyReLU или ELU. Эти функции обычно дают лучшую производительность в задачах регрессии.
* Для моделей глубокого обучения, ReLU является общим выбором для скрытых слоев, так как она может ускорить обучение, но можно также использовать другие функции, например, PReLU или Swish.
* Для рекуррентных нейронных сетей, обычно используются функции активации Tanh.
* Если вы не уверены, какую функцию активации использовать, попробуйте использовать несколько функций активации и сравните их производительность на валидационном наборе данных.
* Кроме того, при выборе функции активации необходимо учитывать свойства функции, такие как производная, способность обеспечивать нелинейность и способность предотвращать затухание градиента.

*Как правильно выбрать функцию активации?*

Все зависит от решаемой задачи прогнозирования. Далее — эксперименты. Как правило, начинать можно с универсальной функции ReLU. И пробовать другие варианты, если ReLU не дает оптимальных результатов обучения НС, например процент ошибки на тестовых данных. Краткие рекомендации по выбору функций активации:

* ReLU следует использовать только для скрытых слоев
* Sigmoid и Tahn наоборот не рекомендуются к использованию в скрытых слоях. Это приводит к проблеме исчезающего градиента
* Swish используется в сетях, где количество слоев больше 40

Для выходных слоев выбор функции активации зависит от решаемой задачи:

* линейная функция — для задач регрессии
* двоичная и многоуровневая (не взаимоисключающие классы) классификации — Sigmoid
* классификация из нескольких классов (взаимоисключающие классы) — Softmax

Для скрытых слоев выбор будет зависеть от архитектуры:

* CNN (сверточная сеть) — ReLU
* RNN (рекуррентная сеть) — Sigmoid и/или Tahn

ReLU - самая часто используемая функция активации. Выбранная функция активации практически неизменна во всех скрытых слоях. Неправильно выбранная функция активации может быть причиной исчезающего градиента или его взрыва. Для выходного слоя выбор функции активации зависит от задачи. Регрессия — ReLU, классификация — Softmax или Sigmoid.