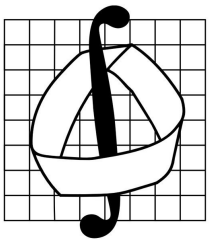
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА
Механико-математический факультет
Кафедра Математической теории интеллектуальных систем



Курсовая работа

Оптимизация количества параметров нейронной сети, решающей задачу классификации изображений

Работу выполнила:
студент 311 группы
Кондаурова Ксения Сергеевна
Научный руководитель:
к. ф.-м. н. Миронов А.М.
Помощник научного руководителя:
студент 311 группы,
инженер-исследователь Skoltech
Семёнов Н.А.

Москва, 2022

Аннотация

С ростом количества информации все острее проявляется необходимость её автоматической обработки. Сейчас одной из самых популярных задач машинного обучения является разделение объектов на непересекающиеся классы. Существует обширный класс методов, решающих данную проблему, поэтому возникает потребность в сравнении эффективности различных алгоритмов. Самой очевидной метрикой качества алгоритма является точность правильных ответов, однако не учитывается важное ограничение - сложность реализации. Для систем, работающих на современных ЭВМ, сложность алгоритма сопряжена с количеством шагов и объёмом затраченной памяти. На практике оказывается, что в подавляющем большинстве случаев обе нагрузки растут полиномиально от количества переменных, параметризующих систему. Соответственно важным пунктом оптимизации алгоритмов является снижение числа их параметров без потери качества работы.

Полный обзор методов машинного обучения выходит за рамки одной статьи и не является целью, поэтому в настоящей работе была поставлена задача оптимизировать количество параметров методов классификации изображений, основанных на сверточных нейронных сетях. Подобный выбор обусловлен высокой актуальностью данной проблемы, а также крайней эффективностью нейронных сетей в этой области.

В настоящей курсовой работе мною были освещены теоритические основы нейронных сетей с полным математическим обоснованием. Эти тезисы позволили сформулировать идею модификации свёрточного слоя, которая заключается в использовании Convolution 3d вместо двумерной свёртки. Развитие этой мысли и желание ещё больше сократить вычислительные затраты без видимой потери точности натолкнули на мысль о создании собственной архитектуры, которая является по сути обобщением обычного полносвязного слоя. Несмотря на то что мне не удалось добиться высоких показателей точности, исследование этой иновации (она упоминается в работе как Meta слой) привело к идеи замены полносвязной головы нейронной сети с относительно большим

числом параметров на комбинацию нескольких полносвязных слоёв с меньшим числом нейронов и последующего соединения их выходов посредством Meta слоя. В результате этой манипуляции не только сократилось в несколько раз количство обучаемых параметров, но и увеличилась точность алгоритма. В связи с этим наблюдением был сделан вывод о том, что в произвольно взятой нейронной сети, в которой присутствует большой полносвязный слой, имеет смысл заменять его на комбинацию нескольких маленьких и Meta.

Содержание

1 Введение

1.1 Задача классификации в машинном обучении

Задача классификации в машинном обучении — это задача отнесения объекта к одному из заранее определенных классов на основании его формализованных признаков. Объект в этой задаче представляется в виде вектора в N-мерном пространстве, каждое измерение в котором является описанием одного из признаков.

Для обучения классификатора необходимо иметь набор объектов, для которых заранее определены классы. Это множество называется обучающей выборкой, её разметка производится вручную, с привлечением специалистов в исследуемой области.

Например, у нас есть множество изображений рукописных цифр, и мы знаем и можем сообщить классификатору, какая именно из десяти цифр представлена на той или иной картинке из данного набора. В этом случае методы машинного обучения позволяют создать алгоритм, распознающий цифру на вновь постувшем изображении.

1.2 Актуальность

На сегодняшний день задача классификации изображений получила широкое практическое применение. Она относится к области научных исследований, называемой компьютерным зрением. Компьютерное зрение представляет собой теорию и технологию создания искусственного интеллекта, который способен получать информацию из двумерных и трёхмерных изображений или видео. Данные могут быть представлены

множеством форм, таких как видеопоследовательность, изображения с различных камер или медицинских сканеров.

В последнее время границы применения машинного зрения быстро расширяются, охватывая всё больше различных сфер жизнедеятельности. Системы компьютерного зрения, использующие нейронные сети, о которых далее пойдёт речь, обладают хорошей устойчивостью к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. На основе сверточных нейронных сетей реализовываются системы для беспилотного управления транспортными средствами, распознавания лиц и выявления различных заболеваний на медицинских снимках. Актуальность данной работы обуславливается тем, что применение методов распознавания объектов на изображении может помочь уменьшить роль человеческого фактора в тех случаях, где первостепенно важным является быстрое реагирование на происшествия, например, на сообщения о пожаре.

В целом, область компьютерного зрения может быть охарактеризована как молодая, разнообразная и динамично развивающаяся.

1.3 Нейронные сети как метод решения задачи классификации

Искусственные нейронные сети являются одним из методов машинного обучения, успешно решающих задачу классификации и распознавания образов. Они построены по принципу организации и функционирования биологических нейронных сетей, то есть сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать работу человеческого организма.

По сей день исследователи черпают идеи для создания новых архитектур из наблюдений биоинженеров, психологов и социологов.

По сути нейронная сеть - это последовательность соединённых между собой определённым образом простых функционалов (искусственных нейронов). Сами по себе нейроны устроены довольно просто: каждый из них имеет дело только с сигналами, которые периодически получает, и с сигналами, которые он посылает другим нейронами с идентичной конструкцией. Так они передают информацию друг другу и, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, локально простые механизмы вместе способны обрабатывать данные разного формата и содержания.

Главным преимуществом нейронных сетей является возможность обучения, которое заключается в настраивании параметров, отвечающих за передачу информации от нейрона к нейрону. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщения и тем самым решать довольно сложные прикладные задачи.

Существенным недостатком алгоритмов нейросетей считается их слабая математическая обоснованность. Главные выводы о "плохой" или "хорошей" архитектуре делаются в большей степени из результатов тестирований. Тем не менее возможность построить строгую математическую теорию есть, и данная курсовая работа служит скорее не демострацией показаний, полученных опытным путём, а попыткой обоснования этих результатов с применением теории вероятности, линейной алгебры, наглядной геометрии и теории случайных процессов.

2 Математические основы нейронных сетей

2.1 Основные понятия и определения

Начнём с общей поставновки задачи обучения с учителем (supervised machine learning).

Определение 1. *Пусть X - множество объектов, Y - множество ответов. Функция зависимости ответов от объетов: $p : X \to Y$ называется целевой функцией (target function).*

Предпологается, что функция p существует, но не известна, а значения целевой функции $p_i = p(x_i)$ заданы только на конечном числе объектов $(x_1, ...x_m),\ x_i \in X$.

Определение 2. *Пары "объект-ответ" (x_i, p_i) называются прецедентами.*

Определение 3. *Совокупность пар $\overline{X} = \{(x_i, p_i) \mid i = \overline{1, m}\}$ называется обучающей выборкой (training sample).*

Задача обучения по прецендентам заключается в том, чтобы по выборке $\overline{X}$ восстановить зависимость p, то есть построить решающую функцию (decision function) $a : X \to Y$, которая бы приближала целевую функцию p, причём не только на объектах обучающей выборки, но и на всём множестве X. Решающая функция a должна допускать эффективную компьютерную реализацию, по этой причине её принято называть алгоритмом.

Если $Y = \{1, ... M\}$, то это *задача классификации* (classification) на M непересекающихся классов. В этом случае множество объектов X разбивается на классы $K_p = \{x \in X \mid p_i = p(x_i),\ i = \overline{1, m}\}$, и алгоритм $a(x)$

должен давать ответ на вопрос: "Какому классу принадлежит объект x?"

Определение 4. *Моделью алгоритмов называется параметрическое семейство отображений* $A = \{g(x, \theta) \mid \theta \in \Theta\}$, *где* $g : X \times \Theta \to Y$ — *некоторая фиксированная функция,* Θ — *множество допустимых значений параметра* θ, *называемое пространством параметров или пространством поиска (search space).*

Определение 5. *Процесс подбора оптимального параметра* θ *модели* A *по обучающей выборке* $\overline{X}$ *обучением (leaning, training) алгоритма* $a \in A$.

Этап обучения наиболее сложен. Как правило, он сводится к поиску параметров модели, доставляющих оптимальное значение заданному функционалу качества.

Определение 6. *Функция потерь (loss function) — это дифференцируемая функция* $L(a, x)$, *характеризующая величину ошибки алгоритма* a *на объекте* x.

Определение 7. *Функционал качества алгоритма* a *на выборке* $\overline{X} : Q(a, \overline{X}) = \frac{1}{m} \sum_{i=1}^{m} L(a, x_i)$. *Функционал* Q *называют также функционалом средних потерь или эмпирическим риском, так как он вычисляется по эмпирическим данным.*

Классический метод обучения, называемый минимизацией эмпирического риска (empirical risk minimization, ERM), заключается в том, чтобы найти в заданной модели A алгоритм a, доставляющий минимальное значение функционалу качества Q на заданной обучающей выборке $\overline{X}$:

$\mu(\overline{X}) = arg \min_{a \in A} Q(a, \overline{X})$.

2.2 Вероятностный подход

В задачах обучения по прецедентам элементы множества X — это не реальные объекты, а лишь доступные данные о них. Данные могут быть неточными, поскольку измерения значений признаков $f_j(x)$ и целевой зависимости $p(x)$ обычно выполняются с погрешностями. Так, одному и тому же описанию x могут соответствовать различные объекты и различные ответы. В таком случае $p(x)$, строго говоря, не является функцией. Устранить эту некорректность позволяет вероятностная постановка задачи.

Вместо существования неизвестной целевой зависимости $p(x)$ предположим существование неизвестного вероятностного распределения на множестве $X \times Y$ с плотностью $\rho(x, y)$, из которого случайно и независимо выбираются n наблюдений $\overline{X} = \{(x_i, p_i) \mid i = \overline{1, m}\}$. Такие выборки называются простыми или случайными одинаково распределёнными (independent identically distributed, i.i.d.).

Вероятностная постановка задачи считается более общей, так как функциональную зависимость $p(x)$ можно представить в виде вероятностного распределения $\rho(x, y) = \rho(x)\rho(y|x)$, положив $\rho(y|x) = \delta(y - p(x))$, где $delta(z)$ — дельта-функция.

При вероятностной постановке задачи вместо модели алгоритмов $g(x, \theta)$, аппроксимирующей неизвестную зависимость $p(x)$, задаётся модель совместной плотности распределения объектов и ответов $\phi(x, y, \theta)$, аппроксимирующая неизвестную плотность $\rho(x, y)$. Затем определяется значение параметра θ, при котором выборка данных $\overline{X}$ максимально правдоподобна, то есть наилучшим образом согласуется с моделью плотности.

Если наблюдения в выборке $\overline{X}$ независимы, то совместная плотность распределения всех наблюдений равна произведению плотностей $\rho(x, y)$ в каждом наблюдении: $\rho(\overline{X}) = \rho((x_1, y_1), ...(x_m, y_m)) = \rho(x_1, y_1)...\rho(x_m, y_m)$. Подставляя вместо $\rho(x, y)$ модель плотности $\phi(x, y, \theta)$, получаем *функцию правдоподобия* (likelihood): $L_\theta(\overline{X}) = \prod_{i=1}^{m} \phi(x_i, y_i, \theta)$.

Чем выше значение правдоподобия, тем лучше выборка согласуется с моделью. Значит, нужно искать значение параметра θ, при котором значение $L_\theta(\overline{X})$ максимально. В математической статистике это называется принципом максимума правдоподобия. После того, как значение параметра θ найдено, искомый алгоритм $a_\theta(x)$ строится по плотности $\phi(x, y, \theta)$.

Вместо максимизации L_θ удобнее минимизировать функционал $- ln\ L_\theta$, поскольку он аддитивен по объектам выборки. Этот функционал совпадает с функционалом эмпирического риска, если определить вероятностную функцию потерь $L(a_\theta, x) = - m\ ln\ \phi(x, y, \theta)$. Такое определение потери вполне естественно — чем хуже пара (x_i, y_i) согласуется с моделью ϕ, тем меньше значение плотности $\phi(x_i, y_i, \theta)$ и выше величина потери $L(a_\theta, x)$.

2.3 Линейная модель нейрона МакКаллока-Питтса

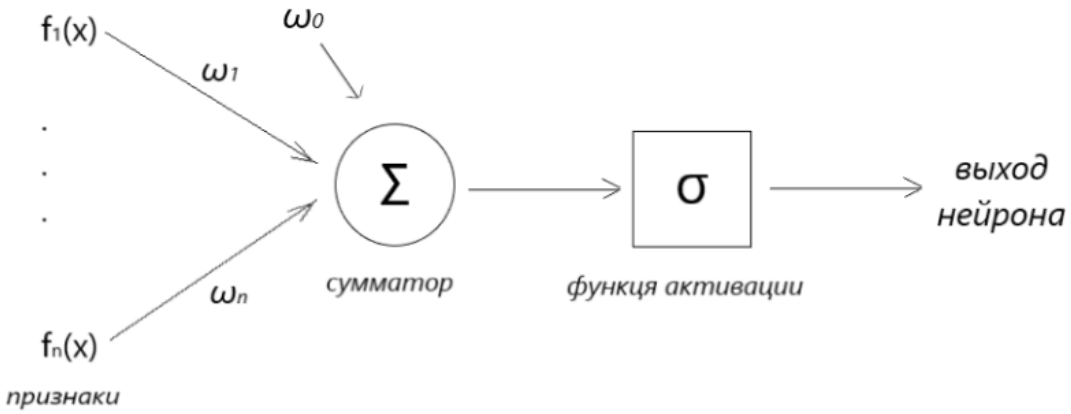
Существует множество методов машинного обучения, которые способны с какой-либо точностью построить алгоритм a. Далее рассматривается один из них, а именно искусственные нейронные сети.

Пусть $f_j : X \to \mathbb{R}$, $j = \overline{1, n}$ - признаковое описание объекта, то есть каждому объекту из X можно поставить в соответствие n числовых признаков $f_j(x)$. Искусственные нейроны представляют из себя функцию

$a(x, w) = \sigma(\sum\limits_{j=1}^{n} (w_j f_j(x) + w_0))$, которая является частным случаем модели *линейной регрессии*. На вход функции подаются значения числовых признаков $f_j(x)$, помноженные на корректируемые параметры – веса w_j. После чего результаты произведения суммируются, добавляется некоторый корректирующий коэффициент w_0, и полученное значение проходит через нелинейную функцию σ, являющуюся показателем силы возбуждения нейрона.

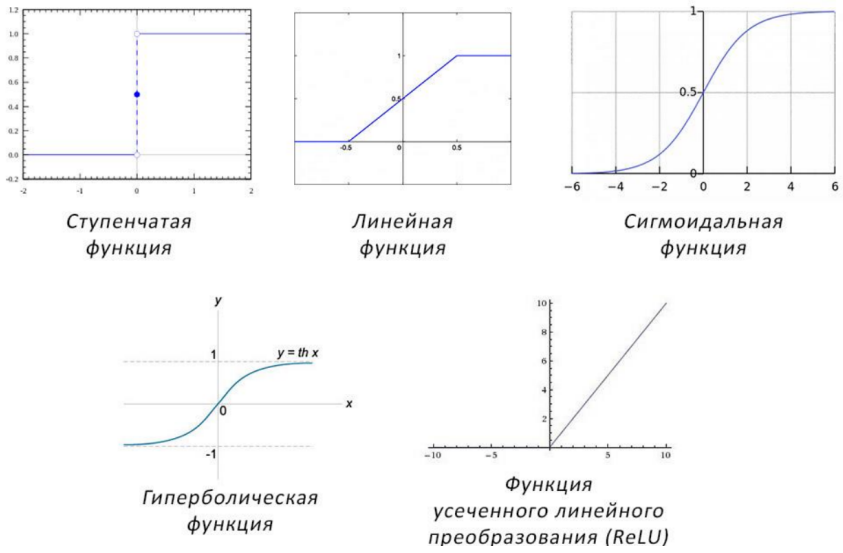
Определение 8. *Коэффициент w_0 называется смещением (bias).*

Определение 9. *Функция σ называется функцией активации (activation function).*



Функция активации определяет, должен ли нейрон быть активирован: она производит нормализацию проходящего через нее сигнала и представляет результат работы нейронной сети в нужном диапазоне. Таким образом, функция активации позволяет выполнять нелинейное преобразование для входных данных, без неё любая архитектура нейронной сети представляла бы из себя обычную модель линейной регрессии и не могла бы решать столь сложные задачи. Разновидностей функций активации огромное количество, основными из них являются: ступенчатая,

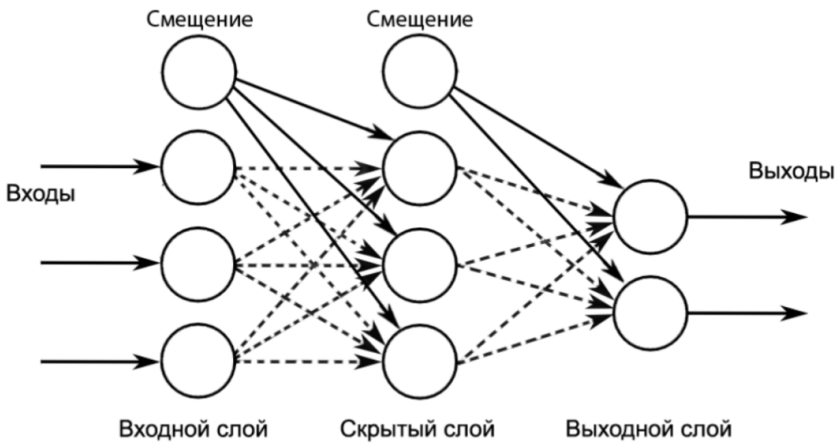
линейная, сигмоидальная, гиперболический тангенс, функция усеченного линейного преобразования (ReLU), которые представлены на рисунке:



Структура искусственной нейронной сети - это направленный граф, в качестве узлов которого выступают нейроны, а в качестве рёбер – связи с весовыми коэффициентами.

Определение 10. *Если в сети нейроны каждого слоя соединены с нейронами следующего слоя, но не связаны друг с другом, то нейронная сеть называется полносвязной. Все слои данной сети, за исключением первого и последнего слоя называются скрытыми.*

Данная структура показана на рисунке:

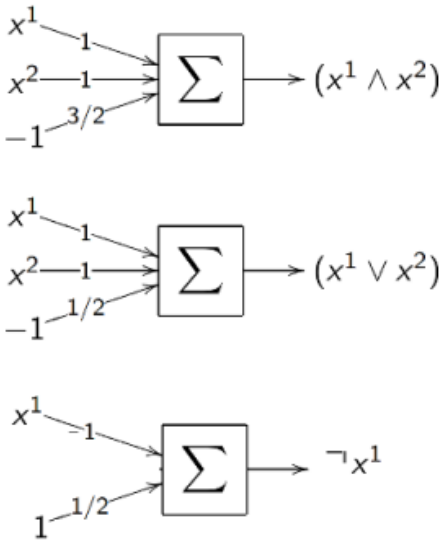


2.4 Проблема полноты

Вспомним, что основной задачей является построение на пространстве X некоторой функции, задающей разделяющую гиперплоскость. Очевидно, что отдельно взятый нейрон, являясь линейной структурой, не способен аппроксимировать произвольную непрерывную функцию. В связи с этим возникает вопрос (который по сей день считается открытым): являются ли нейронные сети методом решения любой задачи классификации? В данном разделе перечислены основные тезисы о полноте функционального класса нейронных сетей.

Теорема 1. *Пусть* $f_j : X \to \{0, 1\}$, $j = \overline{1, n}$ - *признаковое описание объекта. Тогда существует двухслойная нейронная сеть, которая позволяет реализовать любую булеву функцию на заданном пространстве признаков.[13]*

Доказательство: Легко построить нейроны, реализующие логические функции И, ИЛИ, НЕ от бинарных переменных x_1, x_2:



Далее, пользуемя тем фактом, что произвольную булеву функцию от конечного числа переменных можно представить в виде Совершенной Дизъюнктивной Нормальной Формы. Значит, двух слоёв достаточно для построения любой булевой функции. ◁

Из геометрических соображений вытекают утверждения:

Теорема 2. *Двухслойная сеть в* $\mathbb{R}^n$ *позволяет отделить произвольный выпуклый многогранник.[5]*

Теорема 3. *Трёхслойная сеть в* $\mathbb{R}^n$ *позволяет отделить произвольную область, не обязательно выпуклую и не обязательно связную.[5]*

Однако эти результаты не конструктивны, так как ничего не говорят о том, как построить алгоритм, решающий произвольную задачу.

Следующая теорема, доказанная Джорджем Цыбенко в 1989 году, наиболее близка к обоснованию полноты данного функционального класса. Она утверждает, что нейронная сеть с одним скрытым слоем может аппроксимировать любую непрерывную функцию многих переменных с любой точностью. Условиями являются: достаточное количество нейронов скрытого слоя и удачный подбор параметров. В [1] показано, что сети с активационной функцией ReLU с шириной n + 1 достаточны для аппроксимации любой непрерывной функции n-мерных входных переменных.

Определение 11. *Функция* $\sigma : \mathbb{R} \to \mathbb{R}$ *называется сигмоидальной функцией, если пределы* $\lim_{x\to\infty} \sigma(z)$, $\lim_{x\to-\infty} \sigma(z)$ *существуют и равны* $\lim_{x\to\infty} \sigma(z) = 1$, $\lim_{x\to-\infty} \sigma(z) = 0$.

Например, функция $\Theta -$.

Теорема 4. *Пусть $\sigma(x)$ - ограниченнная сигмоидная функция, $f(x)$ - непрерывная функция на $[0,1]^n$. Тогда существуют такие значения параметров $H, \alpha_h \in \mathbb{R}, w_h \in \mathbb{R}^n, w_0 \in \mathbb{R}$, что двухслойная сеть $a(x) = \sum_{h=1}^{H} \alpha_h \sigma(< x, w_h > + w_0)$ равномерно приближает $f(x)$ с любой точностью $\epsilon > 0$: $|a(x) - f(x)| < \epsilon$, для всех $x \in [0,1]^n$./1/*

Доказательство теоремы представлено в публикации [1]. В работе [2] показано, что сети с активационной функцией ReLU с шириной $n+1$ достаточны для аппроксимации любой непрерывной функции n-мерных входных переменных.

Эти результаты показывают, что двух-трёх скрытых слоёв в теории достаточно для решения широкого спектра задач. Поэтому долгое время исследовали ограничивались этим количеством, несмотря на то что биологические нейронные сети, содержат, как правило, 15-20 слоёв.

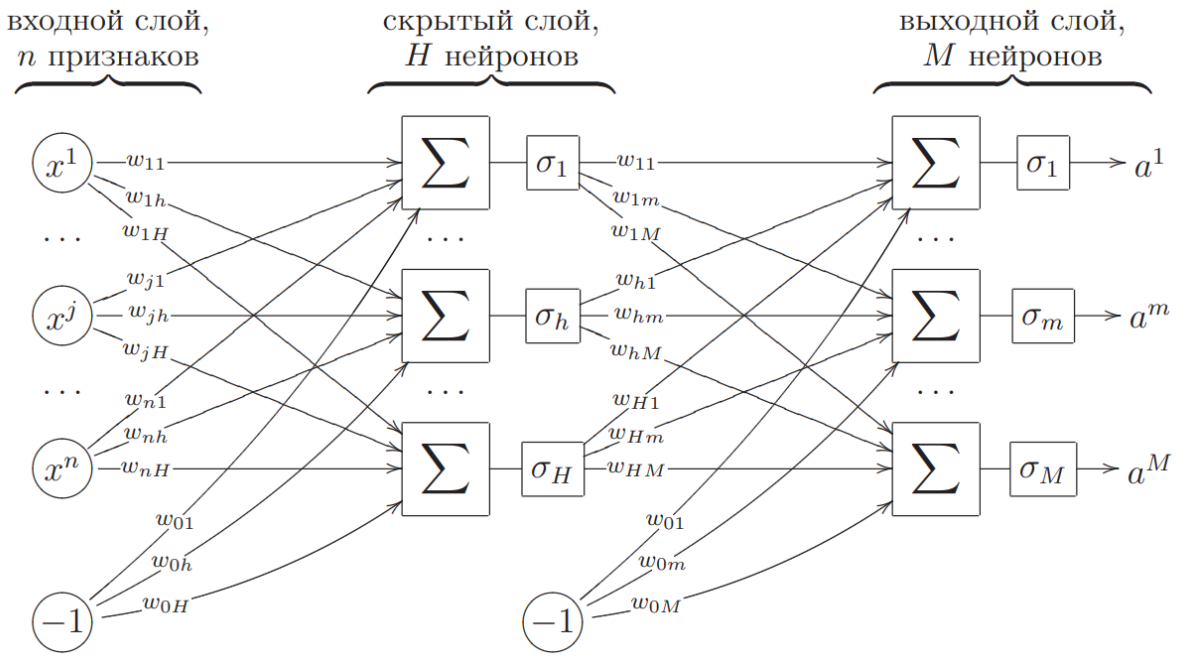
2.5 Метод обратного распространения ошибки

Одной из главных особенностей искусственных нейронных сетей является их обучаемость. Основная цель обучения состоит в том, чтобы минимизировать ошибку между выходными значениями сети, получаемыми при прямом распространении ошибки и ожидаемыми значениями.

Определение 12. *Прямым распространением ошибки (forward propagation) называется процесс получения выходных данных нейронной сети на основе одного экземпляра тренировочных данных.*

Рассмотрим полносвязную сеть. Положим $X = \mathbb{R}^n, Y = \mathbb{R}^M$. Значит, выходной слой состоит из M нейронов с функциями активации σ_M и выходами $a^m, m = \overline{1, M}$. Перед ним находится скрытый слой из H

нейронов с функциями активации σ_h и выходами $u^h, h = \overline{1, H}$. Веса связей между h-м нейроном скрытого слоя и m-м нейроном выходного слоя будем обозначать через w_{hm}. Перед этим слоем будет находиться входной слой признаков. Обозначим через w вектор всех весов сети. Данная конструкция показана на рисунке:



Выходные значения сети объекта x вычисляются как суперпозиция:
$a^m(x) = \sigma_m(\sum_{h=0}^{H} w_{hm} u^h(x)); \quad u^h(x) = \sigma_h(\sum_{i=0}^{n} w_{ih} x^i)$

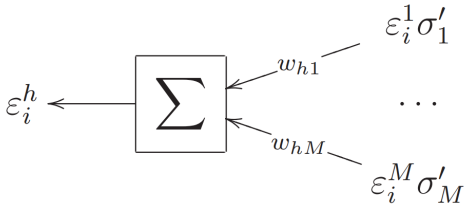
На этом forward propagation заканчивается и начинается этап обучения.

Определение 13. *Процесс корректировки весов с помощью вычисления производной сложной функции называется методом обратного распространения ошибки (back propagation).*

Запишем функционал среднеквадратичной ошибки на признаке x^i фиксированного объекта $x \in X$: $Q_i(w) = \frac{1}{2} \sum_{m=1}^{M} (a^m(x^i) - y^m)^2$. Функционалы подбираются в зависимости от задачи, однако обязательным

условием является наличие легко вычисляемой производной. Функционал среднеквадратичной ошибки без ограничения общности взят в качетсве примера для простоты изложения. Для него запишем частные производные: $\frac{\partial Q_i(w)}{\partial a^m} = a^m(x^i) - y^m = \epsilon_i^m$. Получается, что частная производная функционала ошибки по m-ому выходу равна величине этой ошибки на m-ом выходе.

Воспользуемся правилом дифференцирования сложной функции и запишем частные производные по выходам скрытого слоя: $\frac{\partial Q_i(w)}{\partial u^h} = \sum\limits_{m=1}^{M} \frac{\partial Q_i(w)}{\partial a^m}\sigma'_m(a^m(x^i))w_{hm} = \sum\limits_{m=1}^{M} \epsilon_i^m \sigma'_m w_{hm} = \epsilon_i^h$. Эту величину, по аналогии с ϵ_i^m, будем называть ошибкой сети на скрытом слое и обозначать через ϵ_i^h. Заметим, что ϵ_i^h вычисляется по ϵ_i^m, если запустить сеть «задом наперёд», подав на выходы нейронов скрытого слоя значения $\epsilon_i^m \sigma'_m$, а результат ϵ_i^h получив на входе. При этом входной вектор скалярно умножается на вектор весов w_{hm}, находящихся справа от нейрона, а не слева, как при прямом вычислении (отсюда и название алгоритма — обратное распространение ошибок):



Имея частные производные по a^m и u^h, легко выписать градиент L по весам: $\frac{\partial Q_i(w)}{\partial w_{hm}} = \frac{\partial Q_i(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \epsilon_i^m \sigma'_m u^h(x^i)$, $\frac{\partial Q_i(w)}{\partial w_{ih}} = \frac{\partial Q_i(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{ih}} = \epsilon_i^h \sigma'_h x^i$. Если число скрытых слоёв больше одного, то остальные частные производные вычисляются аналогично обратным ходом по слоям справо налево. Затем делается шаг градиентной оптимизации, веса обновляются и алгоритм повторяется, пока Q не стабилизуется.

2.6 Переобучение сети и выборка валидации

Минимизацию эмпирического риска следует применять с известной долей осторожности. Если минимум функционала $Q(a, \overline{X})$ достигается на алгоритме a, то это ещё не гарантирует, что a будет хорошо приближать целевую зависимость на произвольной контрольной выборке $\overline{X}' = (x'_i, y'_i),\ i = \overline{1,k}$. Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте *переобучения (overfitting)*. При решении практических задач с этим явлением приходится сталкиваться очень часто.

Легко представить себе метод, который минимизирует эмпирический риск до нуля, но при этом абсолютно не способен обучаться. Получив обучающую выборку $\overline{X}$, он запоминает её и строит алгоритм, который сравнивает предъявляемый объект x с обучающими объектами из $\overline{X}$. В случае совпадения алгоритм выдаёт правильный ответ. Иначе выдаётся произвольный ответ. Эмпирический риск принимает наименьшее возможное значение, равное нулю. Однако этот алгоритм не способен восстановить зависимость вне материала обучения. Отсюда вывод: для успешного обучения необходимо не только запоминать, но и обобщать.

Пусть дана выборка $\overline{X} = (x_i, y_i), i = \overline{1,L}$. Разобьём её N различными способами на две непересекающиеся подвыборки — обучающую $\overline{X_n}$ длины l и контрольную $\overline{X'_n}$ длины $k = L - l$.

Определение 14. *Контрольная выборка $\overline{X'_n}$ называется выборкой валидации (cross-validation).*

Для каждого разбиения $n = 1, ..., N$ построим алгоритм $a_n = \mu(\overline{X_n})$ и вычислим значение $Q_n = Q(a_n, \overline{X'_n})$. Среднее арифметическое значе-

ний Q_n по всем разбиениям называется оценкой скользящего контроля: $CV = \frac{1}{N} \sum_{n=1}^{N} Q(\mu(\overline{X_n}, \overline{X'_n}))$. Возможны различные варианты скользящего контроля, отличающиеся способами разбиения выборки. В простейшем варианте разбиения генерируются случайным образом, число N берётся в диапазоне от 20 до 100.

3 Глубокие нейронные сети

Нейронная сеть с более чем одним скрытым слоем называется глубокой нейронной сетью (Deep Neural Network). Выбор архитектуры сети, то есть числа слоёв, числа нейронов и числа связей для каждого нейрона, является, пожалуй, наиболее сложной задачей. Проблема выбора структуры тесно связана с проблемами недообучения и переобучения. Слишком простые сети не способны адекватно моделировать целевые зависимости. Слишком сложные сети имеют избыточное число свободных параметров, которые в процессе обучения настраиваются не только на восстановление целевой зависимости, но и на воспроизведение шума. Так появляется фундаментальный вопрос о *выразительности нейронной сети* (neural network expressivity), то есть о том, как архитектурные свойства сети - глубина, ширина, тип слоя - влияют на функции, которые она может реализовать.

3.1 Вопрос экспрессивности нейронных сетей

Как было показано в пункте 2.4, теоретические результаты, отвечающие на поставленный вопрос, могут оказаться неконструктивными и не применимыми для практических задач. Однако несколько недавних

работ [6]-[11] вносят некоторую конкретику: они были сосредоточены на понимании преимуществ глубины нейронных сетей. Эти результаты убедительны и учитывают современные архитектурные изменения, но они только показывают, что конкретный выбор весов для более глубокой сети приводит к неприменимости для "мелкой" (обычно с одним или двумя скрытыми слоями).

Большим прорывом в изучении экспрессивности стала статья [12]. В ней авторы предложили легко вычисляемые меры выразительности нейронной сети, основанные на изучении класса функционалов, появляющихся на каждом слое. С помощью введённых мер и объединяющего анализа данных, подверженных преобразованиям слой за слоем, была найдена экспоненциальная зависимость точности ответов от глубины сети и степенная зависимости точности от числа параметров. Из чего делается фундаментальный вывод, что глубина сети важнее ширины. Также в работе [12] были приведены доказательства того, что настраеваемые веса не равны по значимости. Авторы обнаружили, что первые слои сильнее подвержены влиянию шума, и нейронные сети лучше работают, когда эти веса хорошо оптимизированы.

Попытка изучить вопрос выразительности была предпринята и мной. В данной работе я исследую зависимость точности классификации изображений от количества параметров, обучаемых сетью и привожу результаты сравнений нескольких свёрточных архитектур с добавлением нестандартных слоёв.

3.2 Свёрточные нейронные сети

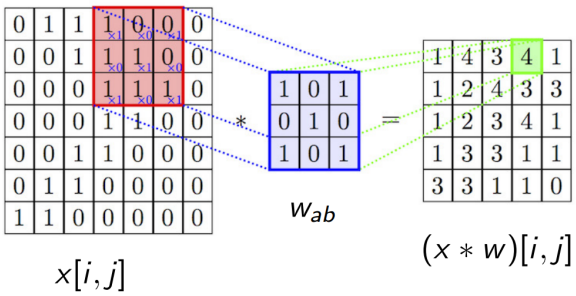
Для анализа изображений чаще всего применяются свёрточные нейронные сети. Основой для данной архитектуры послужил принцип работы зрительной коры головного мозга, который заключается в том, что отдельные группы клеток реагируют только на сигналы от конкретных областей зрительного поля. В целом, архитектура сверточной нейронной сети позволяет выделять во входных данных различные признаки разного уровня сложности.

Пусть X - множество изображений размера $n \times n$, $x[i, j]$ - матрица исходных признаков. Если изображение черно-белое, то $x[i, j]$ - одна матрица, каждая ячейка которой описывается значением яркости в диапазоне от 0 до 255. Цветные изображения представляются тремя матрицами, каждая из которых отвечает за один из каналов цветовой модели RGB, значения матриц описывают интенсивность цвета пикселей.

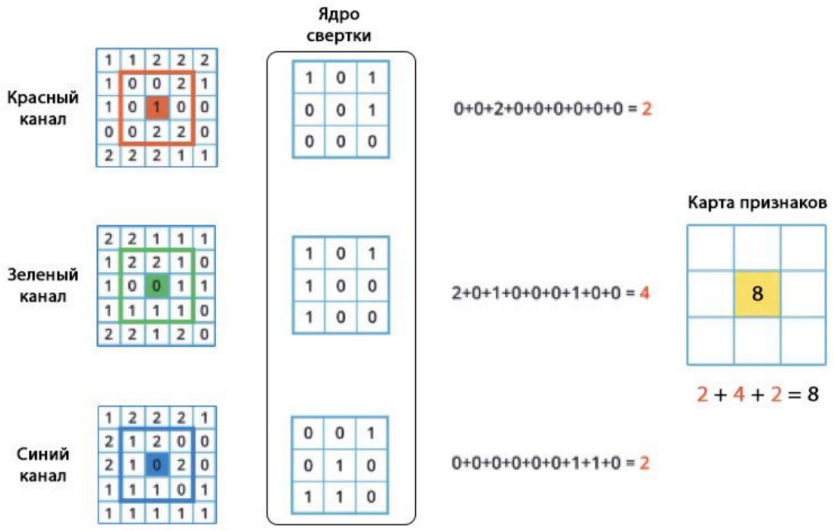
Определение 15. *Матрица* $(w_{ab}), a = -A, ...A, b = -B, ...B, 0 < A < n, 0 < B < n$, *состоящая из обучаемых параметров, называется ядром свёртки (filter).*

Свёрточный слой отличается от полносвязного тем, что у него некоторые веса обнулены, то есть отсутствует часть рёбер вычислительного графа. Таким образом, свёрточный слой аналогичен применению операции свертки, где используется лишь матрица весов небольшого размера (ядро свертки), которую «двигают» по всему обрабатываемому слою.

Определение 16. *Неполносвязный свёрточный нейрон с* $(2A+1)(2B+1)$ *весами, или операция свёртки (convolution), определяется следующим образов:* $(x * w)[i, j] = \sum_{a=-A}^{A} \sum_{b=-B}^{B} w_{ab} x[i + a, j + b]$.



Свертка цветного изображения отличается лишь тем, что к каждому цветовому каналу применяется одна из матриц, составляющих трехмерное ядро свертки, после чего полученные матрицы суммируются в одну результирующую матрицу. Операция свертки для цветного изображения представлена на рисунке:



В выходном терминале может присутствовать линейное смещение, независимое от функций каждого из ядер и свойственное лишь выходному каналу.

Определение 17. *Матрица, полученная в результате свёртки, называется картой признаков.*

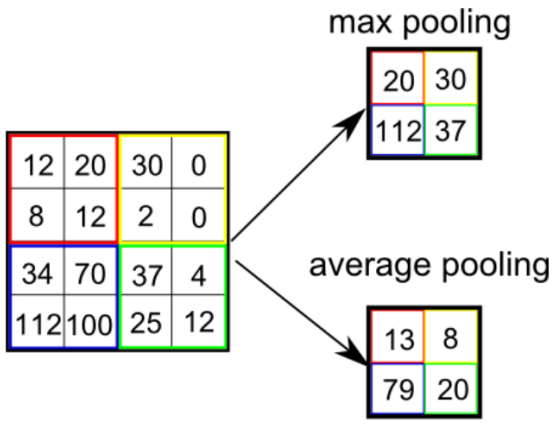
Первый сверточный слой, как правило, выявляет базовые признаки, например, различные прямые и кривые линии, полуокружности, а

последующие сверточные слои позволяют получить признаки все более и более высокого уровня, обобщая информацию на изображении. Сверточные слои чаще всего используют более одного ядра свертки, каждый из которых идентифицирует конкретный признак, поэтому в результате свертки получается набор различных числовых матриц, число которых характеризует глубину карты признаков.

После слоя свертки обычно следует слой пулинга, работающий как объединяющий нейрон.

Определение 18. *Объединяющий нейрон (pooling) - это необучаемая свёртка с шагом* $h > 0$*, агрегирующая данные прямоугольной области* $h \times h$*:* $y[i,j] = F(x[hi, hj], ...x[hi + h - 1, hj + h - 1])$*, где* F *- функция максимума, минимума, среднего и т. д.*

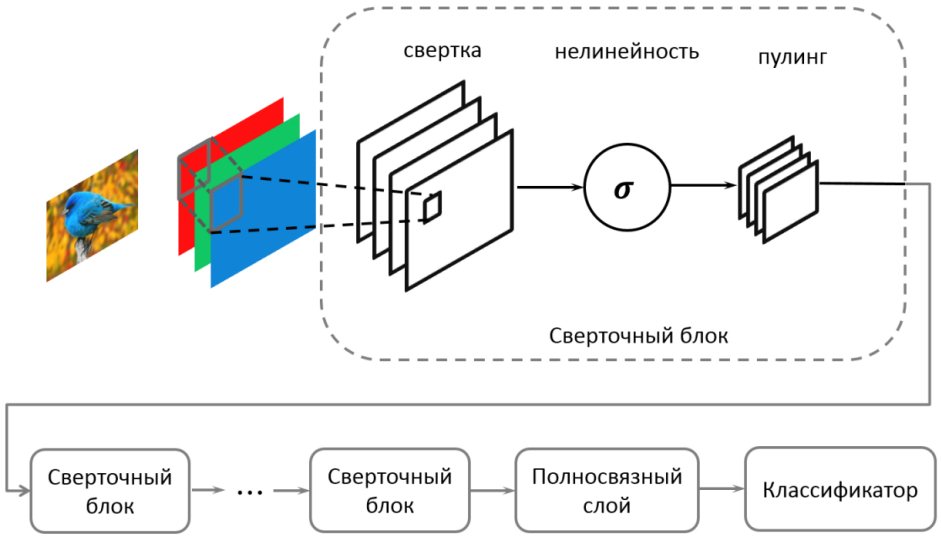
Пример работы max-pooling, который позволяет обнаружить наиболее значимые признаки и удаляет лишние шумы:



Данный слой уменьшает пространственную размерность карты признаков. Необходимость этой операции объясняется тем, что важна именно информация о наличии конкретных признаков на изображении, а не их местоположение, поэтому оно сжимается до менее подробного, состоящего из доминирующих признаков. Кроме того, операция подвыборки

уменьшает количество параметров сверточной сети, тем самым снижая вычислительную нагрузку.

Архитектуру свёрточной нейронной сети схематично можно изобразить так:



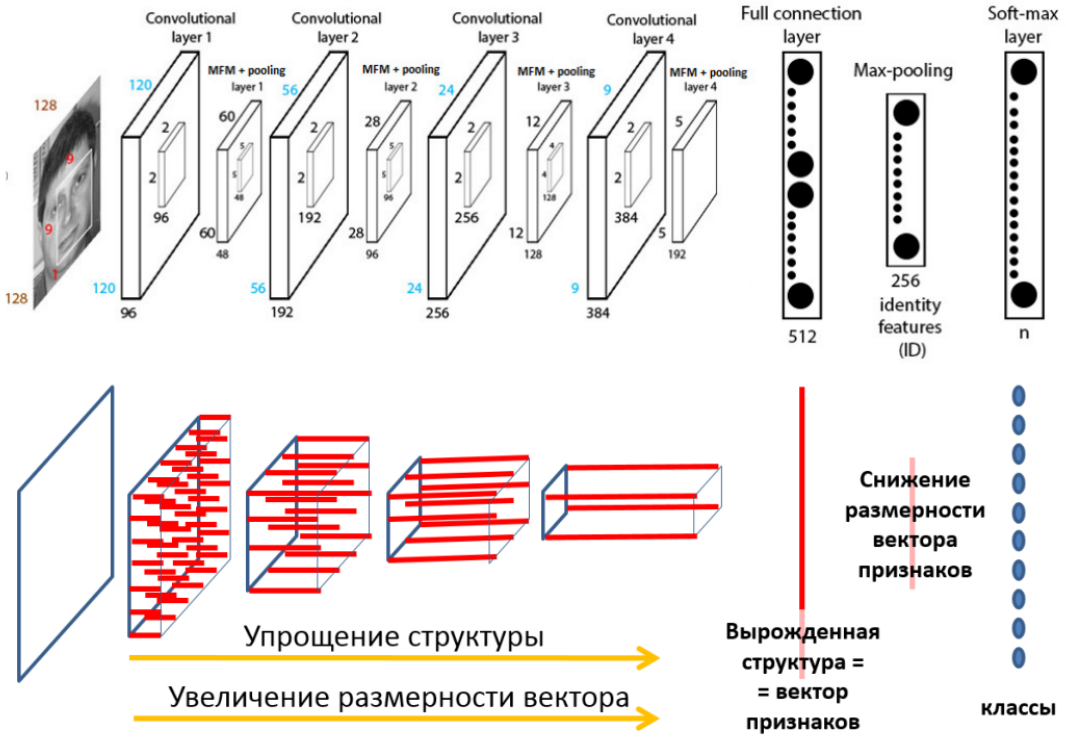
3.3 Векторизация изображения

В 2012 году сверточная нейронная сеть AlexNet победила в конкурсе ImageNet, уменьшив ошибку классификации изображений в полтора раза по сравнению с предыдущими результатами. Это был огромный прорыв в решении задачи классификации. После 2012 года в вышеупомянутом конкурсе искусственных интеллектов всегда используются нейронные сети и методы, основанные на них. Причины успеха кроются в самой структуре данных, получаемых на входе.

Проблема работы с изображениями методами классического ML и многослойными персептронами состоит в том, его надо растянуть в одномерный вектор перед началом работы алгоритма. В результате такого преобразования теряется связь между соседними пикселями. Если говорить точнее - связь остается, но она становится неявной. Модели нуж-

но самой понять, какие пиксели являются соседними, и порождающими какой-то объект, а какие находятся в разных частях изображения. Из-за этого модель становится неустойчивой к поворотам, отражениям, сдвигам и другим преобразованиям, которые на самом деле сохраняют изображённые объекты.

CNN с уверенностью решают эту проблему и не меняют начальной топологии, поэтапно векторизуя изображение и подготавливая его на вход полносвязному слою. С этой точки зрения работа сети устроена следующим образом:



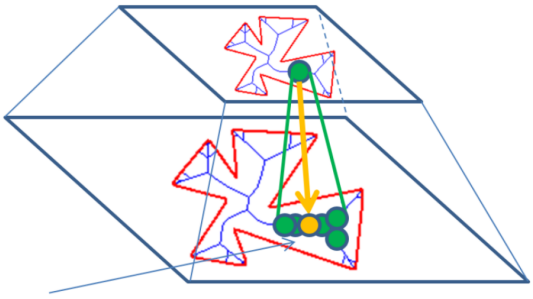
3.4 Идея обобщения CNN на любые структурированные данные

Вышеупомянутое достоинство CNN наталкивает на мысль об их более широком применении, ведь свёрточные сети по сути способны клас-

сифицировать любые данные, на которых возможно ввести топологию, то есть сказать, какие элементы близки к друг другу. Например, посредством CNN можно обрабатывать речевые сигналы, превратив звуковые колебания в изображения с помощью рядов Фурье.

Если ввести на произвольных данных определение окрестности, то далее легко определить свёртку и некоторый аналог пулинга. Таким образом, вне зависимости от природы анализируемого объекта с введённой топологией можно обеспечить векторизацию, то есть постепенно уменьшать сложность структуры, стягивая её слой за слоем в одну точку и вместе с тем увеличивая размерность векторного представления. Этой размерности должно оказаться достаточно для решения тех или иных задач.

Допустим, каждый объект имеет структуру, заданную графом. Определим свёртку как суммирование по локальной окрестности вершины, а пулинг как агрегатор вершин из окрестности. Тогда свёрточная нейросеть будет учиться находить и классифицировать подграфы:

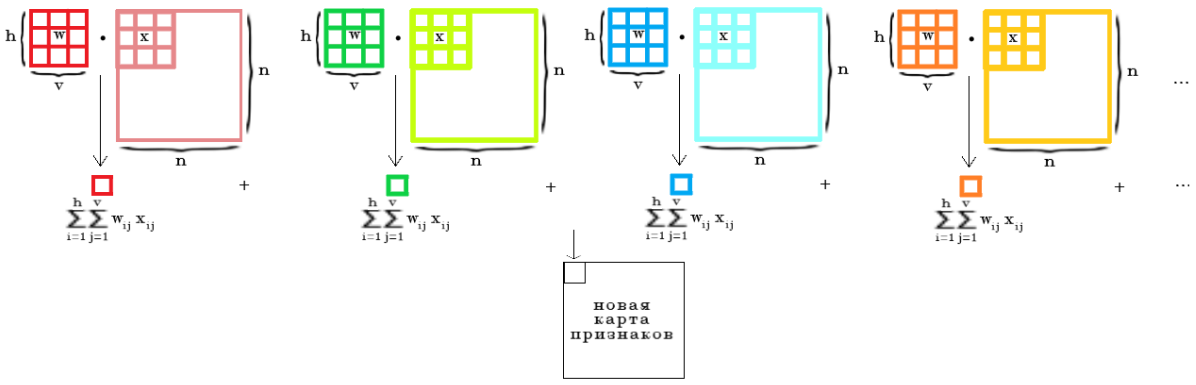


Этот факт наталкивает на мысль о том, что на вход свёрточной архитектуре можно подать выходы предущего слоя. Именно эта идея послужила основой исследований, описанных в данной работе.

4 Оптимизация свёрточного слоя

4.1 Мотивация для использования Convolution 3d

Ещё раз вспомним, как устроен свёрточный слой. Предположим, что ему на вход подаётся C карт признаков размера $n \times n$. Определим фильтр размера $h \times v$, h, v - нечётные, $h < n,\ v < n$. Операция свёртки устроена таким образом, что для каждой карты признаков будет задан свой фильтр с hv обучаемыми параметрами, а пиксель выходного слоя получится путём суммирования результатов применения соответствующего фильтра к карте. Значит, для формирования одной новой признаковой карты используется столько двумерных фильтров, сколько было прошлых признаковых карт, и новый "пиксель"содержит информацию из всех карт признаков предыдущего уровня, так как вычислен по формуле: $\sum_{k=1}^{C} \sum_{i=1}^{h} \sum_{j=1}^{v} w_{kij} x_{kij}$, где w_{kij} - вес k-ого фильтра, x_{kij} - признак в k-ой карте. Этот процесс можно изобразить так:

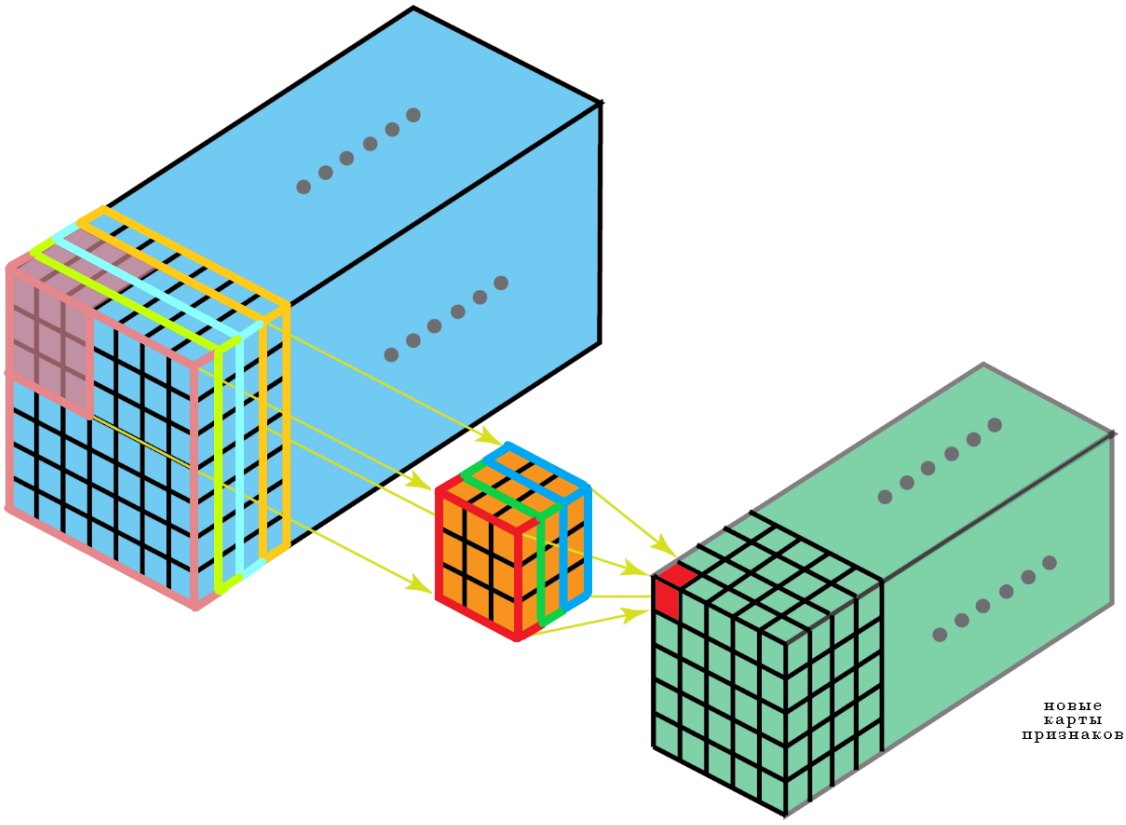


Чтобы получить на выходе D карт признаков, понадобится $(h \cdot v \cdot C + 1) \cdot D$ обучаемых параметров.

Это число можно сократить, если **воспользоваться трёхмерной свёрткой.** Она устроена похожим образом. Её отличие состоит лишь в том, что на вход подаются трёхмерные данные, которые можно себе пред-

ставить как параллелепипед. Фильтр в свою очередь тоже приобретает дополнительную размерность и скользит по параллелепипеду, вычисляя скалярное произведение весов и признаков.

Итак, перед применением трёхмерной свёртки необходимо преобразовать данные. У каждой из C карт признаков введём дополнительную размерность и склеим по ней. Применим Convolution 3d с фильтром размера $h \times v \times u$, $u < C$ и единичным сдвигом по третьей оси. Таким образом, новый признак объединяет информацию из u карт признаков предыдущего уровня, так как вычислен по формуле: $\sum_{k=a}^{a+u} \sum_{i=1}^{h} \sum_{j=1}^{v} w_{kij} x_{kij}$, где w_{kij} - вес k-ого фильтра, x_{kij} - признак в k-ой карте, $a \in \{1, \dots C-u\}$. Изобразим схематично этот процесс:



Тогда, если применяется s трёхмерных фильтров, число обучаемых параметров равно $(h \cdot v \cdot u + 1) \cdot s$, и на выходе получим $(C - u + 1) \cdot s$ карт признаков.

Вычислим число параметров, необходимое для получения D выходных карт при условии использования s, $D \vdots s$ трёхмерных фильтров. Сначала найдём размер фильтра:

$D = (C - u + 1) \cdot s$

$C - u + 1 = D : s$

$C - D : s + 1 = u$

Тогда количество обучаемых весов равно:

$(h \cdot v \cdot (C - D : s + 1) + 1) \cdot s \leq (h \cdot v \cdot C + 1) \cdot D.$

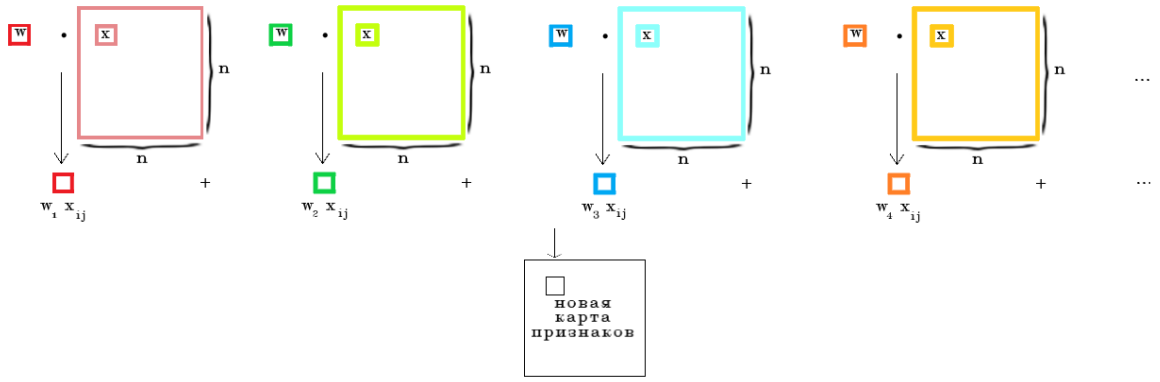
Отметим несколько важных моментов, являющихся мотивацией для тестирования предложенной архитектуры:

- Трёхмерная свёртка позволяет экономить параметры, а, следовательно, снижает вычислительную нагрузку.
- Трёхмерная свёртка, вообще говоря, выявляет другие признаки нежели привычная нам двумерная.
- Двумерная свёртка является частным случаем трёхмерной свёртки при $u = 1,\ s = D$.
- Трёхмерная свёртка олицетворяет собой свёртку карт признаков как целиковых объектов.

4.2 Мотивация для создания Meta слоя

Последний сделанный нами вывод заставляет задуматься об обобщении не только свёрточного, но и полносвязного слоя на любые структурированные данные, в частности, на карты признаков. Такой слой (назовём его Meta) не имплементирован во фреймворках для реализации нейронных сетей, однако его нетрудно написать самостоятельно.

Как и ранее, допустим, что на вход подаётся C карт признаков размера $n \times n$. На выходе хотим получить карту, пиксель которой объединяет информацию со всех карт предыдущего слоя, но в предположении, что все относящиеся к одной карте веса равны. Тогда понадобится всего C обучаемых параметров, и новые признаки будут вычисляться по формуле: $\sum_{k=1}^{C} w_k x_{kij}$, где w_k - вес, присвоенный k-ой карте, x_{kij} - признак в k-ой карте. Этот процесс можно изобразить так:



Если необходимо получить D выходных карт, то потребуется $C \cdot D$ обучаемых параметров.

Далее введём некоторое обобщение. Допустим, что входные карты признаков разделены на N групп, в каждой из которых по $C : N$ карт. Тогда на вход Meta слою будем подавать эти N групп, а на выходе требовать M групп по $C : N$ новых признаковых карт. На алгоритмическом языке слой Meta(N, M) можно задать так:

```
Input:
layers:= list of last outputs
N:= lenght of last outputs list
M:= lenght of new_ouptut list
```

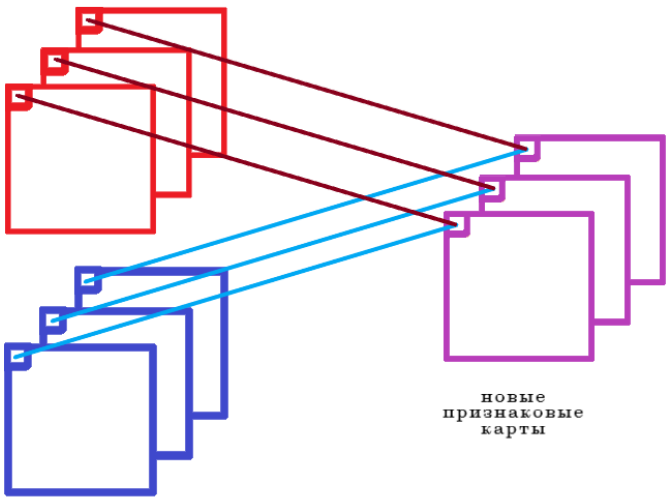
```
Learnable parameters:
weights:= matrix with shape (N, M)

Algorithm:
new_output:= empty list
for i in [1, M]:
    one_output:= 0
    for j in [1, N]:
        one_output += layers[j] * weights[j][i]
    Add one_output to new_output

Output:
new_output
```

Несмотря на то что описанная конструкция даёт большой вычислительный граф, количество требуемых параметров всего лишь $N \cdot M$.

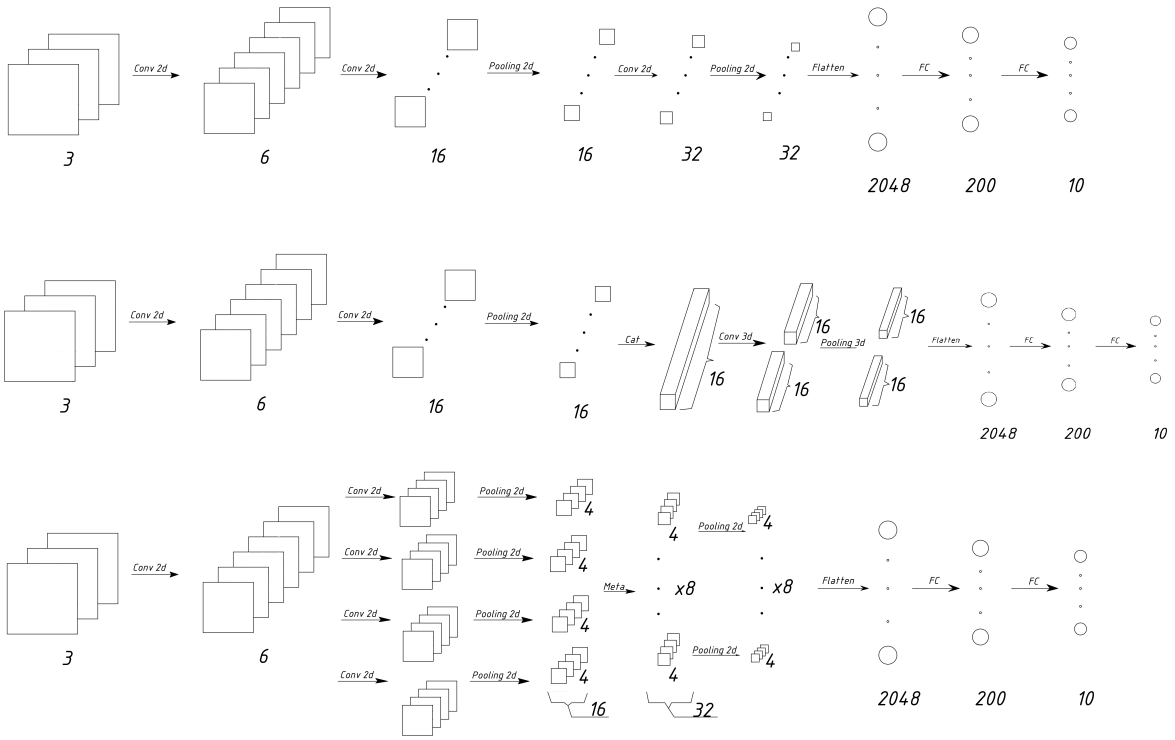
Для $N = 2,\ M = 1,\ C = 6,\ D = 3$ сделаем рисунок, на котором одинаковыми цветами отмечены рёбра с одинаковыми весами:



4.3 Тестирование

Итак, необходимо сравнить результаты работы трёх слоёв: двумерного свёрточного, трёхмерного свёрточного и слоя Meta.

Для тестирования была выбрана база данных (dataset) CIFAR-10, включающая 60 000 цветных изображений размера 32x32. Эти изображения разделены на 10 непересекающихся классов: самолеты, автомобили, птицы, кошки, олени, собаки, лягушки, лошади, корабли и грузовики. Чтобы сравнение методов было корректным, осуществим классификацию изображений с помощью нескольких архитектур с разным количеством параметров. Однако число признаковых карт оставим согласованным. Для первого тестирования предъявим наглядную иллюстрацию всех трёх архитектур:



Каждая нейронная сеть обучалась в одинаковых условиях (batch size, learning rate, количество эпох и так далее). Во всех двумерных свёртках kernel size = 3, padding = 1, во всех двумерных пулингах kernel size = 2, stride = 2, остальные параметры по умолчанию. В сравнительной таблице приведены данные об архитектуре, числе обучаемых параметров и точности:

Архитектура			**Количество параметров**	**Точность**
Conv2d(3,6)→ Conv2d(6,16)→ MaxPool→	Conv2d(16,32)→ MaxPool→	FC(2048,40) → FC(40,10)	88 058	**68,13%**
	Conv3d (1,2,16,(3,3,3))→ Pooling3d ((1,2,2), (1,2,2))→		83 474	66,67%
	4× Conv2d(16,8)→ Meta(4,8) → MaxPool→		83 418	67,26%
Conv2d(3,16)→ Conv2d(16,32)→ MaxPool→	Conv2d(32,64)→ MaxPool→	FC(2048,40) → FC(40,10)	187 874	**71,03%**
	Conv3d (1,2,16,(3,3,3))→ Pooling3d ((1,2,2), (1,2,2))→		169 434	66,62%
	4× Conv2d(32,16)→ Meta(4,8) → MaxPool→		169 378	69,27%
Conv2d(3,16)→ Conv2d(16,32)→ Conv2d(16,32)→ MaxPool→	Conv2d(32,64)→ MaxPool→	FC(2048,40) → FC(40,10)	197 122	**72,47%**
	Conv3d (1,2,16,(3,3,3))→ Pooling3d ((1,2,2), (1,2,2))→		178 686	67,64%
	4× Conv2d(32,16)→ Meta(4,8) → MaxPool→		178 626	70,11%

4.4 Вывод

Проведённые мною эксперименты показали, что замена двумерной свёртки на Convolution 3d или комбинацию нескольких независимых свёрток с меньшим числом карт признаков и Meta слоя привела к ощутимой потере точности на 2-5% при сокращении числа параметров на 5-10%. Худшие результаты показала трёхмерная свёртка, из чего можно сделать вывод, что для формирования признаков более высокого уровня необходимо иметь информацию из всех предыдущих карт.

Однако полученные результаты интересны и могут быть объяснены с точки зрения теории. Идеологически Meta слой осуществляет параметризацию нескольких рёбер вычислительного графа одним числом. Предполагалось, что приравнивание некоторых весов не снизит обобщающую способность модели и уменьшит количество обучаемых параметров. Однако, как оказалось впоследствии, похожая идея уже реализована в свёрточном слое: если сравнить карту признаков свёртки с полносвязным слоем, то будет видно, что многие связи параметризуются одним элементом фильтра. Поэтому использование Meta слоя и Convolution 3d фактически являются новой реализацией ранее известной идеи.

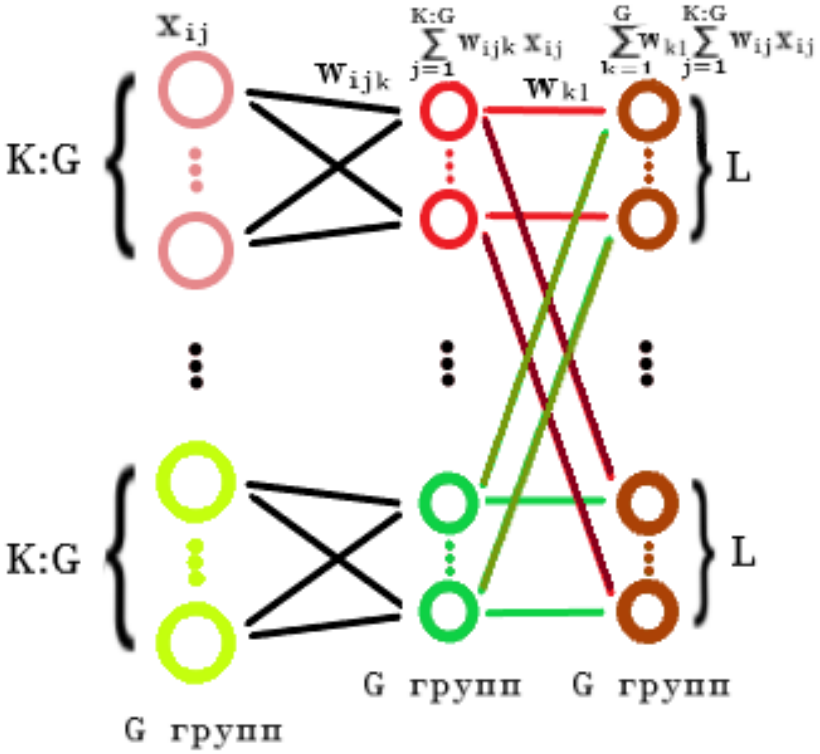
Также отметим, что смысла оптимизировать ранние слои свёртки нет, так как в них задействовано относительно меньше параметров, а их вклад в точность значительный, как показал пункт 3.1.

5 Оптимизация полносвязного слоя

5.1 Мотивация для использования Meta слоя

Многократное сокращение количества параметров при использовании Meta слоя подталкивает к идее оптимизации самых "трудоёмких" слоёв - полносвязных. Допустим, что после прохождения свёрток и преобразований получившихся карт в вектор имеем K признаков. Если бы было необходимо задать полносвязный слой с H признаками на выходе, то понадобилось бы $K \times H$ обучаемых параметров. На практике это число очень велико, поэтому оптимизация первого полносвязного слоя - одна из самых важных задач.

Описанный в предыдущей главе Meta слой позволяет обрабатывать группы признаков, приравнивая некоторые весовые коэффициенты. Предположим, что K признаков теперь разбито на G ($K \vdots G$) групп по $K : G$ в каждой. Для всех групп зададим отдельно полносвязный слой с $L = H : G$ выходами. А затем свяжем их друг с другом с помощью слоя Meta($G,\ G$). Тогда новый k-ый признак в l-ой группе будет вычислен по формуле: $\sum_{k=1}^{G} w_{kl} \sum_{j=1}^{K:G} w_{ijk} x_{ij}$, где w_{ijk} - j-ый вес в i-ой группе, соединяющий k-ый промежуточный признак, x_{ij} - j-ый признак в i-ой входной группе, w_{kl} - вес, соединяющий k-ую промежуточную группу и l-ую выходную. Проиллюстрируем конструкцию:



Таким образом, портебуется $(K:G) \times L + G \times G \leq K \times H$ параметров.

5.2 Тестирование

Как и ранее, было выбрано несколько свёрточных архитектур для сравнения. Каждая нейронная сеть обучалась в условиях, освещённых в главе 4. В таблице ниже приведены данные об архитектуре, числе обучаемых параметров и точности:

Архитектура		Количество параметров	Точность
Conv2d(3,6)→ Conv2d(6,16)→ MaxPool→ Conv2d(16,32)→ MaxPool→	FC(2048,40) → FC(40,10)	88 058	**68,13%**
	4× FC(512,10) → Meta(4,4) → FC(40,10)	26 618	66,62%
Conv2d(3,16)→ Conv2d(16,32)→ MaxPool→ Conv2d(32,64)→ MaxPool→	FC(4096,40) → FC(40,10)	187 874	**71,03%**
	4× FC(1024,10) → Meta(4,4) → FC(40,10)	64 994	70,92%
Conv2d(3,16)→ Conv2d(16,32)→ Conv2d(32,32)→ MaxPool→ Conv2d(32,64)→ MaxPool→	FC(4096,40) → FC(40,10)	197 122	**72,47%**
	4× FC(1024,10) → Meta(4,4) → FC(40,10)	74 242	72,04%
Conv2d(3,16)→ Conv2d(16,32)→ Conv2d(32,32)→ MaxPool→ Conv2d(32,64)→ Conv2d(64,64)→ MaxPool→	FC(4096,40) → FC(40,10)	234050	73,40%
	4× FC(1024,10) → Meta(4,4) → FC(40,10)	111 170	**75,28%**
Conv2d(3,16)→ Conv2d(16,32)→ Conv2d(32,32)→ MaxPool→ Conv2d(32,64)→ Conv2d(64,128)→ MaxPool→	FC(8192,40) → FC(40,10)	434 818	73,70%
	4× FC(2048,10) → Meta(4,4) → FC(40,10)	189 058	**74,72%**

5.3 Вывод

Таким образом, мною было протестировано пять различных комбинаций двумерных свёрточных слоёв. Для каждой из них была совершена

попытка заменить последующий полносвязный слой с количество нейронов от 2048 до 8192 на четыре полносвязных слоя с числом нейронов от 512 до 1024, выходы которых считались независимо и объединялись с помощью Meta слоя. Как видно из таблицы, это действие позволило сократить количество обучаемых параметров в 2-4 раза, причём в некоторых случаях точность алгоритма упала на незначительные 0,5-1,5%, а в других даже увеличилась на 1-2%.

Такой результат можно объяснить идеологически: полносвязный слой параметризуется большим количеством значений, которые позволяют приближать любую функцию от n-мерного вектора, однако на практике целые группы параметров отвечают за схожие признаки объекта, передающиеся и преобразующиеся слой за слоем в архитектуре. Они по сути являются одним и тем же ребром вычислительного графа, но используются несколько раз, а значит, их веса можно приравнять. Именно этот механизм и реализует Meta слой, позволяющий, с одной стороны, снизить вычислительную нагрузку при обучении сети, а с другой - повысить ее обобщающую способность путем использования более удачной параметризации искомой функции.

Таким образом, Meta слой может быть успешно применён для оптимизации полносвязных слоев в большом наборе архитектур, который не стоит ограничивать свёртыми.

6 Направление дальнейших исследований

Несмотря на то что нейронные сети стали за последние годы настолько эффективны, что способны одолеть человека в решении многочисленных задач, мы всё еще не можем строго объяснить причину их успеха.

Поэтому свои дальнейшие исследования я планирую посвятить вопросу экспрессивности нейронных сетей, который, по моему мнению, разрешим с помощью теории случайных процессов.

Список литературы

[1] George Cybenko. Approximation by Superpositions of a Sigmoidal function. Mathematics of Control, Signals, and Systems. 1989.

[2] Lu Z. et al. The expressive power of neural networks: A view from the width //Advances in neural information processing systems. – 2017. – С. 6231-6239.

[3] Goodfellow I. Deep Learning (Adaptive Computation and Machine Learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville - The MIT Press. -2016.-Nov. -С. 800.

[4] Визильтер Ю.В., Горбацевич В.С. Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей. ММРО-2017.

[5] Воронцов К. В.. Математические методы обучения по прецедентам (теория обучения машин).

[6] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. arXiv preprint arXiv:1312.6098, 2013

[7] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Advances in neural information processing systems, pages 2924–2932, 2014.

[8] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. arXiv preprint arXiv:1512.03965, 2015

[9] Matus Telgarsky. Representation benefits of deep feedforward networks. arXiv preprint arXiv:1509.08101, 2015.

[10] James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. In Advances in Neural Information Processing Systems, pages 2877–2885, 2013.

[11] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. Neural Networks and Learning Systems, IEEE Transactions on, 25(8):1553– 1565, 2014.

[12] Maithra Raghu et al. On the Expressive Power of Deep Neural Networks. 2016.

[13] Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1986.