ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»
или УрФУ

Глубокое обучение систем компьютерного зрения
Глава 1

Михаил Владимирович Ронкин, Василий Викторович Зюзин, Сергей Владимирович Поршнев

г. Екатеринбург, 2021 год.

Содержание

1 Глава 1. Обзор ретроспективы глубокого обучения сверточных нейронных сетей

1.1 Первые теории нейронной сети 1940-1990

1.1.1 Модели линейного персептрона средины 20-го века

Как было сказано во введении основной подход к решению задач компьютерного зрения на сегодня – это подход на основании глубокого обучения сверточных нейронных сетей. Однако, прежде чем разобраться со сверхточными сетями, следует рассмотреть базовый подход к глубокому обучению – так называемые полносвязные нейронные сети.

Полносвязные искусственные нейронные сети возникли еще в 1940-е и были вдохновлены гипотезами об устройстве мозга (нейронов в мозге и их связями). Аналогом каждого нейрона было предложена считать **модель персептрона** (от perceptio – восприятие). Такую модель называют **модель МакКаллока-Питтса** [1]. МакКаллоком и Питтсом было предположено, что каждый персептрон, по существу, представляет собой суммирование всех входных данных с регулируемыми весами и взятие какой-либо нелинейной функции от результата этого суммирования.

$$y = f\left(\sum_{i=0}^{N-1} w_n x_n\right) \tag{1}$$

где y - результат работы персептрона; w_n - это набор коэффициентов взвешивания каждого входного значения; x_n - это входные данные; $f(\cdot)$ - это какая либо нелинейная функция. Часто нелинейная функция называется: **функция активации**. Отметим, что модель МакКаллока-Питтса аналогична работе логистической регрессии.

Таким образом можно представить персептрон так, как это показано на рисунке 1. В самом общем случае выбор функции активации может быть практически

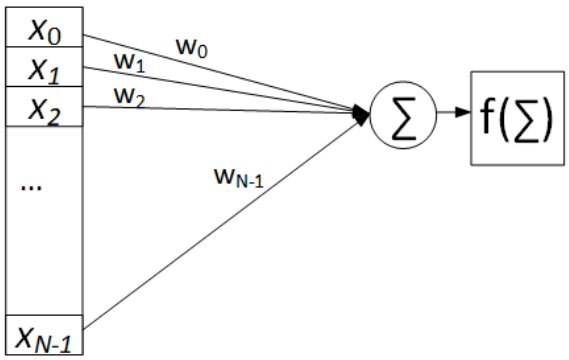

Рисунок 1 – иллюстрация персептрона

любым. В самых простых случаях, в том числе в модели МакКаллока-Питтса это простая **ступенчатая функция (пороговая функция)** (1, если $x \geq 0$ 0 или -1 если $x < 0$). Однако, на практике к этой функции предъявляется ряд требований, ограничивающих возможности для выбора таких функций. Например, часто полезно, чтобы функция активации имела непрерывную первую производную.

В отдельных случаях вид функции активации может быть получен аналитически. Так для задачи т.н. бинарной классификации (например, принятия решений о выполнении или не выполнении одной гипотезы, например, есть кошка на изображении или ее нет) – можно получить функцию, называемую сигмоид. Такая функция также часто называется логистической, а соответствующая ей модель персептрона логистической регрессией. Функция сигмоид имеет следующий вид:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2}$$

В приложении А будет показано как получается такая функция. В оригинальной работе МакКаллока-Питтса не предлагалось конкретного алгоритма обучения - то есть автоматического подбора весовых параметров. В 1949 году был предложен алгоритм **обучение Хебба** (правила Хебба) для итерационного обучения персептрона вида 1 [2]. Для правил Хебба Новиком позже было доказана сходимость за конечное число шагов (**теорема Новика**) [3]. Отметим, что термин обучение в случае правил Хебба в означает **обучение с учителем (supervised learning)**. Сама **процедура обучения** предполагает автоматический подбор весовых параметров w_i таким образом, чтобы персептрон давал результат с минимальной ошибкой по некоторому критерию, заданному исследователем. Для обучения с учителем предполагается, что есть некоторая обучающая выборка, содержащая набор входные данных и набор результатов для каждого экземпляра (выходных данных, например меток класса (номеров класса) - labels). Тогда **обучение с учителем состоит** в том, чтобы установить значения весовых параметров w_i так, чтобы средняя ошибка между результатам работы персептрона и ожидаемым результатом (например, меткой класса) была минимальна. В случае правил Хебба ошибка рассчитывалась как разность между ожидаемым и полученным ответом. Если ожидаемый ответ был 0, а система давала 1, значения весов нужно было увеличить (чтобы $\sum_i w_i x_i$ стала больше). В обратном случае значения весов нужно было уменьшить. Коэффициент изменения значений весов устанавливался перед обучением [2]. В современном понимании такой коэффициент называется **скорость обучения (learning rate)**.

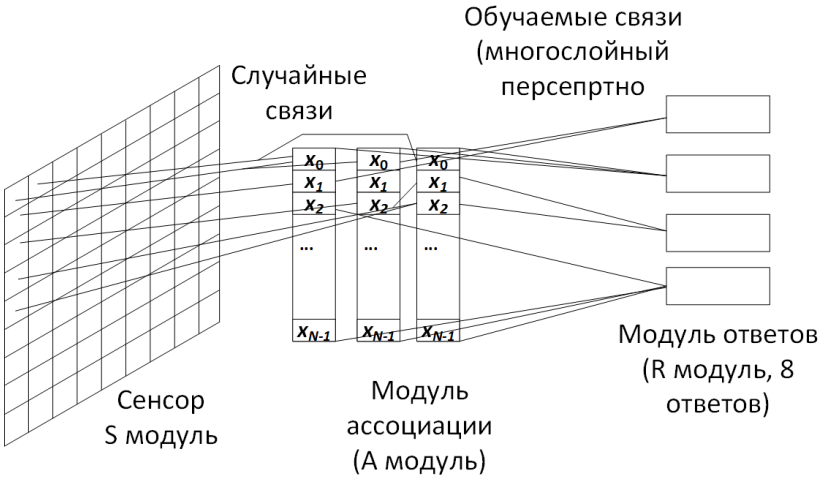

Рисунок 2 – иллюстрация машины "Mark 1 Perceptron", реализующей персептрон Розенблатта

Развитие идей МакКаллока-Питтса и Хебба в технике было предложено в модели **персептрон Розенблатта** [4, 5]. По существу, предложенное Розенблаттом решение представляет собой полносвязную нейронную сеть с одним скрытым слоем состоящим из одного персептрона - многослойный персептрон. Схема персептрона была впервые реализована аппаратно под названием "Mark 1 Perceptron" схема показана на рисунке 2. В данной машине использовался 400 пиксельный сенсор (модуль S), слой проекции на вход персептрона (модуль ассоциации - модуль A) и один обучаемый персептрон с 8 выходами типа "да-нет" (модуль R, например "да" $+1$, "нет" -1). Связи между модулями S и A устанавливались случайно. Связи в персептроне (связывает модули A и R) обучались [6].

Розенблаттом также была предложена **итерационная процедура обучения**. В каждой итерации использовался набор входных данных для каждого из которых был известен результат (тренировочная выборка данных). Таким образом можно было вычислить ошибку, которая могла быть 0, ± 1 или ± 2. На каждой итерации все данные подавались в сеть по одному в любом порядке. Итерации продолжались до тех пор, пока сеть не научится выдавать правильные ответы для всех входных примеров. При этом веса персептрона обновлялись после каждого примера, обратно пропорционально полученной ошибке, если ответ был некорректным. Таким образом для положительной ошибки веса регулировались для снижения выходного значения, для отрицательного значения ошибки веса повышались. Все обновления значений весов персептрона происходили с предварительно выбранным коэффициентом - известным как **скорость обучения (learning rate, lr)**. Такой набор правил позже стал известен как правила обучения персептрона [7]. Формально данные правила можно записать следующим образом:

$$w_i^{t+1} = w_i^t + \eta[y^t - \hat{y}^t]x_i^t, \tag{3}$$

где w_i^{t+1}, w_i^t - значения весового коэффициента i до и после обновления весов на примере t; $y^t - \hat{y}^t$ - ошибка работы персептрона (y^t - целевое значение; $\hat{y}^t$ - результат работы сети); x_i^t - i-тый компонент входного воздействия для примера t; η - предустановленная скорость обучения.

Следует отметить, что в данном случае действует следующий принцип выбора скорости обучения - при слишком низкой скорости обучения изменение средней ошибки по каждой итерации будет незначительным. Если скорость обучения будет слишком высокой, то ошибка по каждой итерации будет как-бы колебаться в положительную и отрицательную стороны, слабо изменяясь в каждой стороне - то есть не будет доходить до минимума. Типичные порядки скорости обучения $0,01 - 0,0001$ в зависимости от задачи.

В обсуждаемой модели все веса инициализировали случайными значениями. Розенблаттом было доказана **теорема о сходимости персептрона**. При обучении методом коррекции ошибок персептрон Розенблатта сходится к правильным значениям весов за конечное число итераций независимо от выбора начальных значений весов если допустить, что такой набор весов существует. Под правильными значениями здесь понимается ситуация когда сеть выдает правильные ответы для каждого тренировочного воздействия. Проблемой данного утверждения является то, что обучение может занимать достаточно долго времени и при этом вопрос о существовании "правильного набора весов" будет оставаться открытым. [7]. Практически параллельно Розенблатту была предложена модель персептрона под названием **ADALINE (adaptive linear neuron - адаптивный линейный нейрон)**, для обучения которого предлагалось использовать **метод наименьших квадратов (МНК - LMS least mean square)** [8]. Главными отличиями данной модели от модели МакКаллока и Питтса являлись: отказ от пороговой функции и процедура обучения персептрона с использованием квадрата ошибки при обновлении значений весов (**расстояния Евклида, Mean Square Error - MSE**). Таким образом правило обновления весов можно записать как:

$$w_i^{t+1} = w_i^t + \eta[y^t - \sum_{i=0}^{N-1} w_i^t x_i^t]x_i^t, \tag{4}$$

Для обучения ADALINE использовались те же правила, что и для персептрона Розенблата. Однако, в данной нейронной сети обучения могли производится не только с целочисленными ошибками, но и с дробными значениями. Таким образом ADALINE позволяет решать не только задачи бинарной классификации, но и задачи регрессии. В задаче бинарной классификации есть только два ответа (± 1 или $0 - 1$).

В задаче регрессии необходимо дать ответ в дробных числах. [7].

Нейронная сеть ADALINE с несколькими персептронами известна как MADALINE Network. Такая сеть наиболее приближена к современному подходу к архитекторам нейронных сетей. Однако такая сеть была реализована гораздо позже оригинальной работы ADALINE - в силу развития средств вычислительной техники [7].

Основной проблемой, с которой можно столкнуться при наличии только одно персепторна (персептрон МакКаллока-Питтса, Розенблата и ADALINE) является проблема принятия решений в нелинейных ситуациях: это ситуации в которых нельзя одним простым правилом разделить данные на классы (то есть дать правильный ответ или принять правильное решение). Если в задаче можно разделить данные на классы одной линией - то такая задача будет называться **задача с линейно-разделяемыми признаками (линейная задача).Линейная задача может быть решена при помощи одного персептрона**. В противном случае будет **нелинейная задача**. В нелинейной задаче требует более сложная кривая для разделения значений признаков на классы. В нелинейной задаче требуется иметь несколько персептронов прежде чем будет принято решение. Таким образом, в нелинейной задаче необходимо иметь три слоя: входной, скрытый слой с несколькими персептронами и выходной слой - принятие решений. [9].

В самом простом случае нелинейная задача известна как **XOR проблема** (проблема исключающего ИЛИ) [10]. Допустим, что есть задача классифицировать автомобили на популярные и непопулярные. При этом для каждого автомобиля есть два признака: цена (разделим на категории - дорого, дешево) и класс (внедорожник или легковой) тогда результаты классификация должна быть такой, как это показано на рисунке 3. Для того, чтобы разделить автомобили в данном случае потребуется провести две линии (по первому классу и по второму) - таким образом будет два условия для популярного автомобиля: если цена высокая, то класс внедорожник, если цена низкая, то класс легковые автомобили.

Описанная задача (см. иллюстрацию на рисунке 3) может показаться элементарной читателю, однако, ее решение недостижимо для любого линейного алгоритма (МакКаллока-Питтса, Розенблата и ADALINE), в том числе одного персептрона. Это объясняется тем, что **каждый персептрон может провести только одну разделительную линию между признаками**. Фактически это эквивалентно одному условию (например $x > 0$). Если нужно провести две линии, как в примере на рисунке 3, то потребуется два персептрона. Обнаружение XOR проблемы в 1969 году стало концом первого поколения развития нейронных сетей. При этом большинство ученых перенесли силы в область символьных методов искусственного интеллекта [9].

Рисунок 3 – иллюстрация XOR проблемы

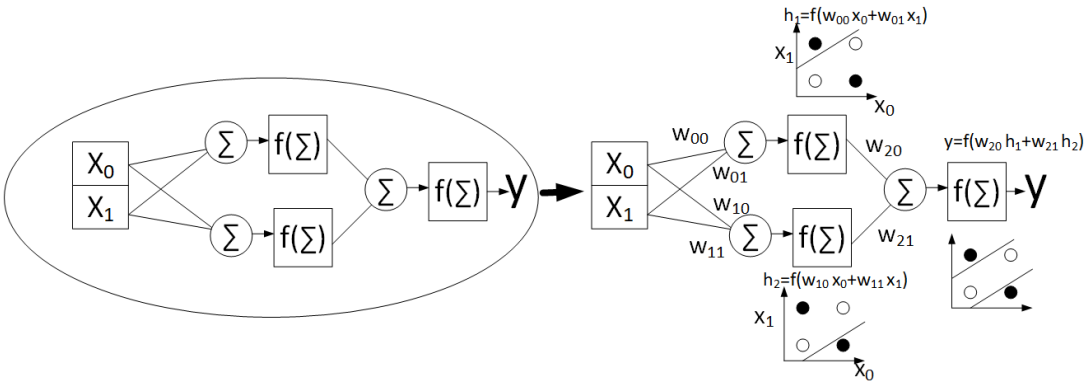

Рисунок 4 – иллюстрация нейронной сети для решения XOR проблемы

Важно отметить, что окончание перовой волны интереса к нейронным сетями было связано исключительно с уровнем технического развития прогресса. Само по себе решение XOR проблемы не представляет трудностей в настоящее время и было реализовано в процессе развития компьютерной техники. [10].

Пример такого решения с использованием нейронной сети, имеющей один скрытый слой с двумя нейронами показан на рисунке 4.

Общее решение таких нелинейных проблем как XOR проблемы было предложено советским математиком А.Г. Ивахненко в 1971 г. [11]. Предложенный Ивахненко метод группового учета данных (GMDH, group method of data handling) является одними и обсуждаемых методов для реализации в современных глубоких нейронных сетях [12, 7].

Отметим также, что строго говоря термин "линия" применим к двух-мерному рассмотрению задачи. Двухмерная задача - это когда рассматривается два признака, например, один по оси абсцисс (скажем x), другой по оси ординат (скажем, y). В более общем случае - когда признаков больше аналогом линии становится **гиперплоскость** (в случае с тремя признаками просто плоскость). Также следует отметить, что такое абстрактное представление пространства образованного признаками называется **признаковое пространство**.

1.1.2 Многослойный персептрон Румельхарта 1980-е

Идея использования большого количества нейронов объединенных вместе в многослойную нейронную сеть была развита в первой половине 1980-х (**идея коннекционизма**). Технически основание коннекционизма лежит в развитие распределенных систем параллельной обработки данных. Также идеей коннекционизма является идея представления данных в виде набора признаков и формирование каждого выхода нейронной сети принимая во внимание все входные признаки [9]. Идеи коннекционизма были развиты группой PDP (parallel distributed processing, параллельная распределенная обработка) во главе с Д.Е. Румельхартом. Данной группой в 1985-1986 были предложены архитектура нейронной сети - **многослойный персептрон Румельхарта** [13] и метод обучения многослойного персептрона - **метод обратного распространения ошибки (backpropagation)** [14] (авторы Rumelhart D.E., Hinton G.E., Williams R.J.). Также важно отметить, что метод обратного распространения ошибки предлагался много раз до 1985 г., однако популярность приобрел только после публикации [14]. В частности, метод предлагался в работе советского математика А.И. Галушкина, посвященной многослойным системам распознавания образов [15]. Также метод обратного распространения ошибки был независимо предложен Барцевым С. И., Охониным В. А. в 1985 г [16]. Архитектура нейронной сети - персептрон Румельхарта часто называется **многослойный персептрон - Multilayer Perceptronб MLP** или **полносвязная нейронная сеть (Full-Connected Neural Netwrok, FCNN)**. Иллюстрация FCNN архитектуры приведена на рисунке 5.

Отдельно следует остановиться на методе обратного распространения ошибок. Данный метод обучения многослойного персептрона наиболее популярный и лежит в основе обучения современных глубоких нейронных сетей. В том числе сверточных нейронных сетей. Математическим аспекты вариантов реализации данного метода будут рассмотренных в соответствующих разделах данной книги. Однако, тут следует также пояснить суть данного метода "на пальцах".

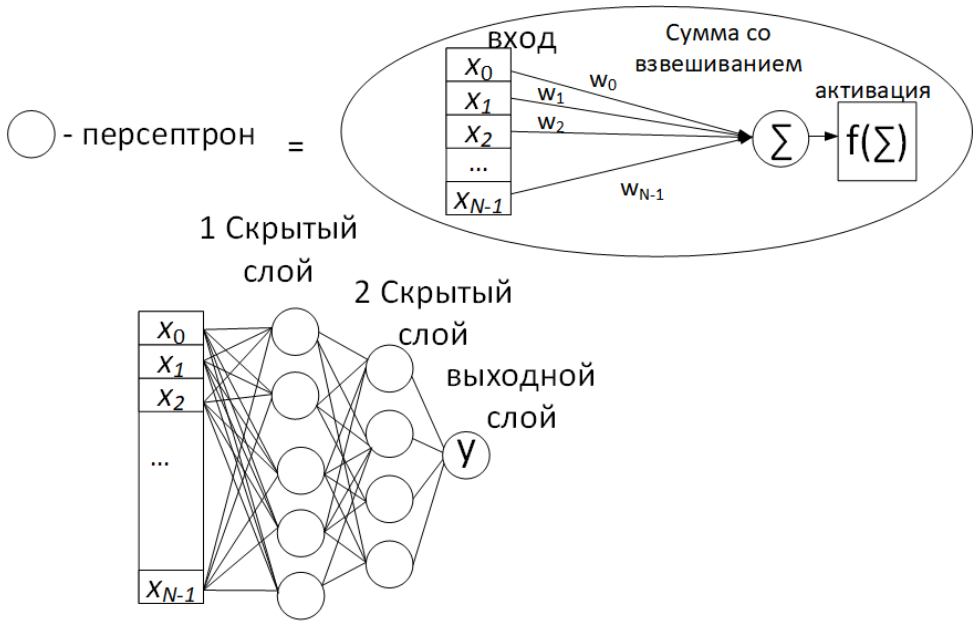

Рисунок 5 – иллюстрация многослойного персептрона

В основе метода обратного распространения ошибки лежит метод градиентного спуска. На самом деле частный случай данного выражения уже был приведен 4 для ADALINE персептрона. Суть метода заключается в обновлении значений весов нейронной сети обратно пропорционально ошибки - почти как правила обучения Хебба, но для работы с дробными значениями. Вывод выражения для градиентного спуска приведен в приложении А.

Рассмотрим идею градиентного спуска более подробно. В самом простом случае градиент это производная. Допустим, что ошибка предсказания нейронной сети как средне-квадратичная ошибка:

$$err_i = L(y_i, \hat{y}_i) = (y_i - f(\sum_{j=0}^{N-1} w_j x_{ij}))^2 = (y_i - f(W^T X_i))^2, \tag{5}$$

где:

- err_i - средне-квадратичная ошибка для примера i. Такую ошибку принято называть **функция потерь (loss function)**.
- $L(y_i, \hat{y}_i)$ - обозначение функции потерь как функции от ожидаемого результата y_i и полученного $\hat{y}_i$. Таким образом функция потерь это и есть некоторая мера сравнения ожидаемого и полученного результатов. Мера может быть выбрана практически любой. Однако, когда речь идет об ошибки - подразумевается, что мера имеет нулевое значение, когда y_i и $\hat{y}_i$ равны друг другу.
- $W = w_0, w_1, ..., w_j, ...w_{N-1}$ - вектор весовых параметров.

- $X = x_0, x_1, ..., x_j, ...x_{N-1}$ - вектор входных параметров (например признаков для одного экземпляра данных).
  Отметим также, что $\sum_{j=0}^{N-1} w_j x_{ij} \equiv W^T X_i$ по определению скалярного произведения векторов.

- $f(\cdot)$ - некоторая функция активации.

Набор производных $L(y_i, \hat{y}_i)$ по всем w_j - будет называться **градиент функции** в данном случае случае градиент функции потерь:

$$\nabla L = \left\{ \frac{\partial L}{w_0}, \frac{\partial L}{w_1}, ..., \frac{\partial L}{w_j}, ..., \frac{\partial L}{w_{N-1}} \right\} \tag{6}$$

Направление градиента будет направлением роста функции, а модуль градиента покажет скорость роста функции. Таким образом для достижения минимума функции потерь требуется следующее.

- Определить выражение ∇L (6) - это покажет направление изменения значений в сторону роста функции потерь.

- Изменить значения веса в обратную сторону.

- Коэффициент изменения значений весов будет называть **скорость обучения.** Выбор скорости обучения очень важен, о чем писалось выше. Легко представить себе ситуацию когда из-за слишком большой скорости обучения значения оптимальных весов будут пропущены.

Общее выражение для обновления весов

$$W^{\{t+1\}} = W^{\{t\}} - \eta \nabla L(\hat{y}^{\{t+1\}}, y^{\{t+1\}}), \tag{7}$$

где $W^{\{t+1\}}, W^{\{t\}}$ - значения весов на шаге обучения $t+1$ и t соответственно; $\hat{y}^{\{t+1\}} = (W^{\{t\}})^T X^{\{t+1\}}, y^{\{t+1\}}$ - значения входных и выходных данных для шага $t+1$; $L(\cdot)$ - функция потерь; ∇L - градиент функции потерь; η - скорость обучения.

В выражении 7 функция потерь может быть любой, для которой существует первая производная. При этом нет необходимости в аналитическом расчете значений ∇L, процедура расчета значений градиента выполняется исключительно численными методами [17].

Для иллюстрации принципа работы градиентного спуска рассмотрим функцию одной переменной $err(w)$ вида (5). Для такой функции потерь производная по w будет иметь вид

$$\frac{\partial err(w)}{\partial w} = \frac{\partial (y_i - w x_i)^2}{\partial w} = 2(y_i - w x_i) x_i. \tag{8}$$

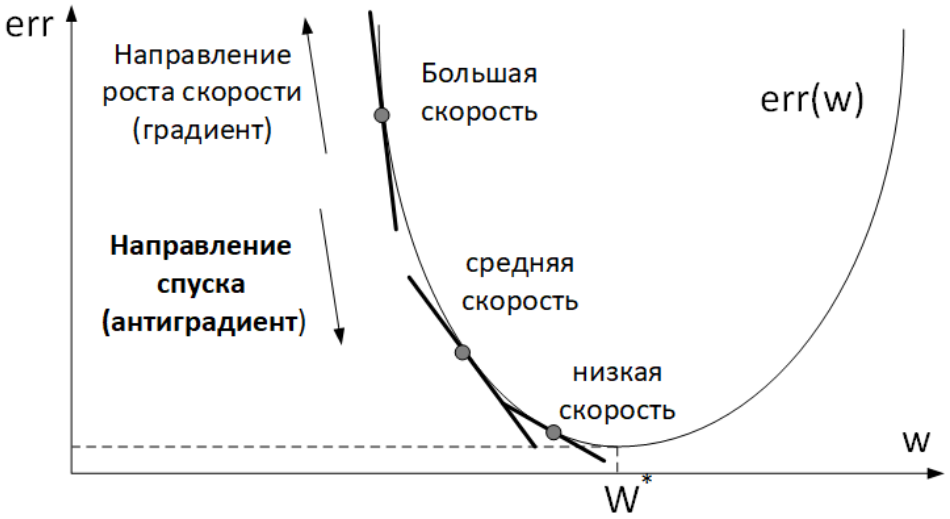

Рисунок 6 – иллюстрация функции потерь err для одного параметра w и ее градиента.

Иллюстрация для такого вида функции потерь и ее градиента показан на рисунке 6. На рисунке показано, что скорость изменения значения функции потерь снижается с приближением к ее минимальному значению err_{min}. Минимальному значению функции потерь соответствует оптимальное значение веса w^*. Направление изменения весов к w^* - называется **направление градиентного спуска** или направление антиградиента.

Метод градиентного спуска относится к методам оптимизации. Существует достаточно много модификаций метода, в том числе позволяющих избежать проблему выбора скорости обучения. Однако, не все они применимы к обучению нейронных сетей. Позже наиболее популярные модификации методов градиентного спуска будут рассмотрены в приложении к глубокому обучению.

Отметим, что задача градиентного спуска характерна для многих приложений регрессии. В том числе данным методом может быть решена и задача логистической регрессии. Кроме того следует отметить, что для случая линейной регрессии задача оптимизации решается в явном виде - в закрытой форме, то есть в виде аналитического выражения. Однако, если параметров в регрессии достаточно много, то также рекомендуется использовать метод градиентного спуска.

В случае когда обучению подлежит многослойный персептрон метод градиентного спуска в своей прямой реализации становится достаточно сложным с вычислительной точки зрения.

Рассмотрим персептрон Румельхарта с одним скрытым слоем. Для такой нейронной сети есть два набора весовых параметров (первый набор: вход - скрытый слой и второй набор: скрытый слой - выход). расчет градиента для первого набора - потребует расчет вектора весов. Расчет градиента для второго слоя потребует расчета матрицы и т.д. Однако процедура может быть упрощена если представить что каждый слой (скрытый и выходной) независимы. Тогда отдельно можно представить скрытый слой как персептрон с большим количеством выходов и отдельно выходной слой, для которого выход скрытого слоя будет входными данными. В этом случае значение функции потерь на выходе нейронной сети можно пересчитать на скрытый слой и обновить значения весов по каждому слою отдельно. Данная идея лежит в основе метода обратного распространения ошибки.

Формально метод обратного распространения ошибки может быть записана следующим образом, для многослойного персептрона с K слоями ($k = 0, ..., K$, $k = 0$ - вход) для $k > 0$:

$$\begin{cases} W_k^{\{t+1\}} = W_k^{\{t\}} - \eta \frac{1}{m_k} \frac{\partial L}{\partial W_k}; \\ \\ \frac{\partial L}{\partial W_k} = \left(e_k \odot \frac{\partial f(z_k)}{\partial z_k} \right) O_{k-1}^T; \\ \\ e_k = e_{k+1} \left(\left[W_{k+1}^{\{t\}} \right]^T \odot \frac{\partial f(z_{k+1})}{\partial z_k} \right), \ 1 \leq k < K; \\ \\ e_K = \frac{\partial L}{\partial \hat{y}}; \end{cases}, \tag{9}$$

где

- $W_k^{\{t+1\}}, W_k^{\{t\}}$ - значения вектора весов для слоя с номером k на шаге обучения $t + 1$;
- m_k - число входных параметров для слоя с номером k;
- η - коэффициент скорости обучения;
- e_k - значение ошибки для слоя с номером k;
- $z_k = (W_k^{\{t\}})^T O_{k-1}$;
- O_{k-1} - выход $k - 1$ слоя (вход слоя k);
- $\frac{\partial f(z_k)}{\partial z_k}$ - производная функции активации.

Таким образом, алгоритм вычисления значений обновлений весовых параметров многослойной нейронной сети методом обратного распространения ошибки следующий [17].

- На каждом шаге ее обучения повторить две стадии: прямое и обратное распространение.
- Прямое распространение (forward propagation), при котором на вход подаются данные и получается ответ на выходе.
- Обратное распространение (backward propagation), при котором происходит:
  - расчет ошибки работы нейронной сети и ее градиента,
  - пересчет градиента функции ошибки на результат для каждого слоя (пропорционально значениям весов с которыми учтен каждый нейрон).
  - обновление значений весов для каждого слоя в соответствие суммарным градиентом ошибки его работы.

Отметим, что в основе использования метода обратного распространения ошибки лежит т.н. **правило цепочного дифференцирования**. По существу, метода обратного распространения ошибки, является всего лишь вычислительным представлением данного правила. Для запуска алгоритма обратного распространения ошибки необходимо про-инициализировать значения всех весовых параметров нейронной сети некоторыми случайными значениями. Каждая итерация в алгоритме обратного распространения ошибки называется **эпоха**. Число эпох может быть задано из некоторых априорных соображений (опыта). Однако, чаще всего обучения прерывают по некоторому условию. Типичные условия для остановки процедуры обучения нейронных сетей будут рассмотрены в книге далее.

Одним из ключевых требований к использованию метода обратного распространения ошибок является требование существование производных функций активации и функции потерь. В моделях персептрона первой волны (МакКаллок-Питтс и Розенблатт) как правило использовалась пороговая функция активации. Однако, пороговая функция не могла быть использована с методом обратного распространения ошибок. Пороговую функцию можно заменить на логистическую или на гиперболический тангенс. Другие, менее тривиальные требования к методу обратного распространения ошибок будут рассмотрены позднее.

Важно отметить, что в многослойном персептроне, как правило, вид функции потерь будет отличаться от показанного на рисунке 6, и будет иметь ряд локальных минимумов помимо одного глобального. Задачи, когда имеется только один минимум (**глобальный минимум**) называются задачами выпуклой оптимизации. В нейронных сетях рассматриваются задачи не выпуклой оптимизации, когда имеется ряд локальных минимумов и один глобальный. Другими словами **ландшафт функции потерь** может быть как выпуклым, так и не выпуклым. Пример не

выпуклого ландшафта функции потерь показана на рисунке 7. В пределе число локальных минимумов может быть равно числу нейронов. Возникающие при этом проблемы связаны с тем, как избежать попадания в локальные минимумы и тем, как не пропустить глобальный минимум. [17]. Отметим, что в ряде работ, в том числе [18] утверждается, что вероятность попадания в локальный минимум при градиентном спуске снижается с ростом числа параметров в нейронной сети (в случае если число параметров нейронной сети больше чем необходимое число признаков для однозначного решения задачи) [18].

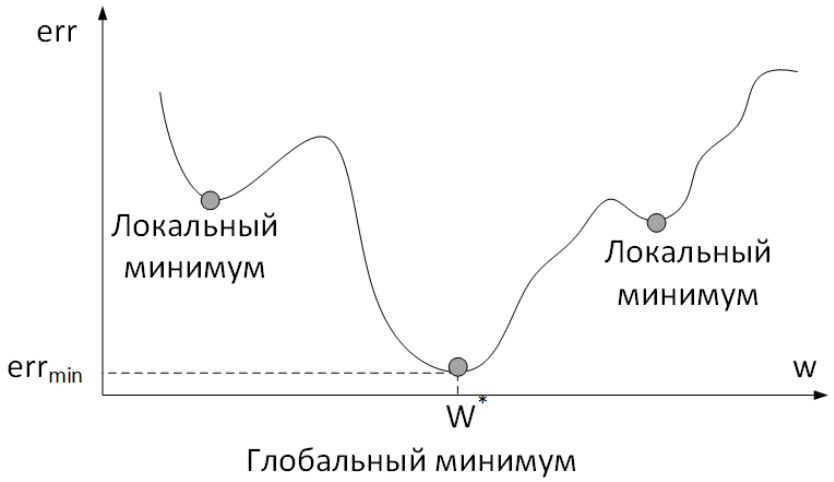

Рисунок 7 – иллюстрация типичного ландшафта функции потерь многослойного персептрона в двухмерной плоскости

С точки зрения концепции параллельных распределенных вычислений удобно представить многослойный персептрон в виде вычислительного графа. [17]. **Вычислительный граф** - это разбиение сложной процедуры вычислений на элементарные операции. Каждая операция представляет собой узел графа. Пример такой процедуры для многослойного персептрона с одним скрытым слоем проиллюстрирован на рисунке 8.

При использовании такого подхода метод обратного распространения ошибки может быть сведен к вычислению производных по каждому ребру графа - что позволяет легко проводить вычисления, необходимые для обучения нейронной сети на любых устройствах, позволяющих распараллеливание операций. [17].

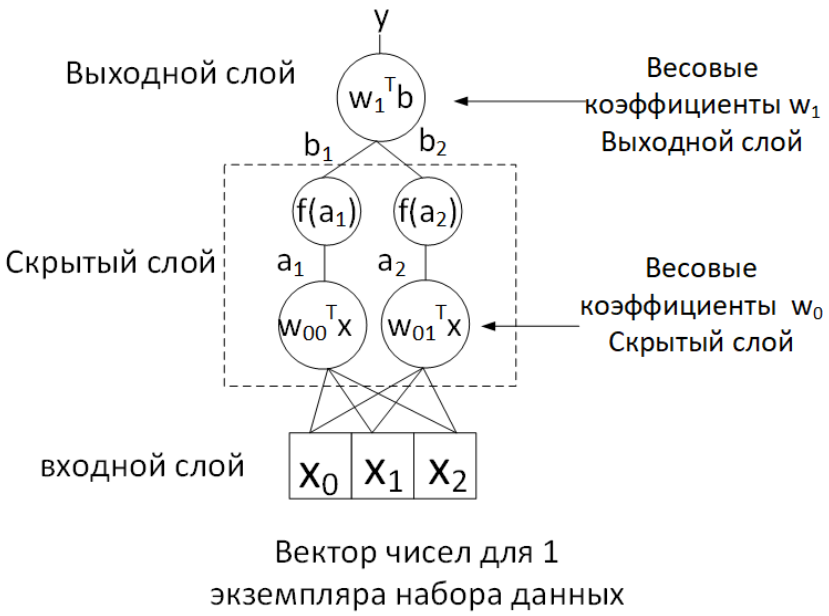

Рисунок 8 – иллюстрация вычислительного графа, $w^T x = \sum wx$

1.1.3 Фундаментальные основы функционироания нейронных сетей 70-е - 90-е

по ссылкам ниже про Вапника и обобщающую способность В 1989 г. Джоржем Цыбенко была доказана универсальная теорема аппроксимации - утверждающая, что достаточно плоносвязной нейронной сети с одним скрытым слоем для аппроксимации любой функции и с любой точностью (зависящей от числа нейронов в скрытом слое). При этом сеть должна иметь вид [19]:

$$\hat{y} = \sum_{j=0}^{N-1} w_{1j} f(\sum_{i=0}^{M-1} w_{ij} x_i + \theta_j), \tag{10}$$

где $\hat{y}$ - результат аппроксимации; $\{x_i\}_{i=0}^{M-1}$ - набор входных данных (вектор M признаков для одного экземпляра); $\{w_{1j}\}_{j=0}^{N-1}$ - набор (вектор) весов скрытого слоя (N нейронов); $\{\theta_j\}_{j=0}^{N-1}$ - набор смещений входных данных для скрытого слоя (вектор из N смещений); $\{w_{ij}\}_{i,j=0}^{M-1,N-1}$ - двухмерный набор весов для входного слоя; $f(\cdot)$ - функция активации, такая, что $f(z) \to 1$ если $z \to \infty$ и $f(z) \to 0$ если $z \to -\infty$, например логистическая функция ($f(z) = 1/[1 + exp(-z)]$).

В 1991 Куртом Хроником результат 10 был обобщен на случай произвольной дифференцируемой функции активации [20]. Также рядом авторов были приведены доказательства универсальная теорема аппроксимации для других частных случаев архитектур полносвязных нейронных сетей, в том числе многослойных нейронных сетей [21] и нейронных сетей с т.н. остаточными связями [22] (данный тип связей

будет подробно рассмотрен и объяснен в книге далее). Также следует отметить, что результат 10 основан на теореме Колмогорва-Арнольда (1956, 1957 гг), которая утверждает, что функцию многих переменных можно представить в виде набора функций одной переменной, как [23, 24]:

$$\hat{y}(x_0, ..., x_{M-1}) = \sum_{j=0}^{2M} F_j(\sum_{i=0}^{M-1} G_{ij}(x_i)), \tag{11}$$

где $F_j(\cdot)G_{ij}(\cdot)$ - наборы некоторых функций одной переменной; $y(x_0, ..., x_{M-1})$ - функция M переменных.

Важным следствием результата 11 является наличие в нейронной сети с одним скрытым слоем, в котором не менее чем $2M+1$ нелинейных узлов (нейронов) для M признаков, описывающих данные. При этом, интуитивно понятно, что тут речь идет о независимых признаках - то есть признаках не коррелирующих (не связанных) друг с другом. Только в этом случае данные образуют M-мерное признаковое пространство. Если два признака во входных данных будут иметь одинаковое поведение, то задача будет недоопределенной.

Другим следствием результата 11 является то, что решение задачи для M признаков требует не менее, чем $(2M + 1) \cdot M \approx 2M^2$ параметров для полносвязной нейронной сети. Также интуитивно понятно, что для вычисления $(2M + 1) \cdot M$ параметров необходимо иметь не менее, чем $(2M + 1) \cdot M$ экземпляров данных. Это следует из аналогии с системами уравнений, в том числе с системами линейных уравнений. При этом речь идет о данных в которых каждый признак представлен независимо. Например, если стоит задача отличить кошек от собак - то надо взять $(2M+1) \cdot M/2$ как можно более разных фотографий собак и столько же кошек. Если есть вероятность, что часть данных тренировочной выборки коррелирует, то выборку надо увеличить. На практике такая вероятность есть всегда, поэтому выборку надо брать предельно большой или использовать специальные меры по регуляризации решения слабоопределенных задачи. Позже в книге будут рассмотрены методы регуляризации обучения, характерные для компьютерного зрения.

Отметим также, что требования по наличию $2M + 1$ параметров в скрытом слое и M параметров во входном слое для задачи требующей M следует из теории Вапника-Червоненкиса (советские математики, 1971 г.) [25]. Авторами было введено понятие размерности Вапника-Червоненкиса. Данная размерность показывает максимальное число способов классификации (разделения) данных. Для линейных систем составляет $M + 1$ гиперплоскостей. Для нелинейных систем типа многогранник размерность составляет $2M + 1$. Таким образом, если есть есть M экземпляров данных для которых выделено M признаков - то такая задача классификации решается линейной системой. В этом случае остается, например, перебрать возможные способы классификации и выбрать лучший. Если

для решения задачи необходимо $2M + 1$ признаков, то задача должны решаться нелинейным классификатором. Если необходимо более чем $2M + 1$ - задача не имеет однозначного решения для обсуждаемых условий. Таким образом в случае архитектуры нейронной сети, соответствующей выражению (10) необходимо взять M линейных персептронов и построить многоугольник из гиперплоскостей, затем на данном многоугольнике необходимо выбрать классификатор из $2M + 1$ вариантов. ОТСЮДА УЖЕ ДОЛЖЕН ИДТИ ЦЫБЕНКО - ЧТО ЭТОГО ДОСТАТОЧНО. ВАПНИК - НЕОБХОДИМО, ЦЫБЕНКО ДОСТАТОЧНО.

Также следует отметить, что в ряде работ, в частности, [26, 27] было показано, что чем больше размерность Вапника-Червоненкиса тем больше данных требуется для обучения алгоритма и тем выше потенциальная обобщающая способность алгоритма. Кроме того для фиксированного размера тренировочной выборки разность ошибок на тестовых и на тренировочных данных будет выше для алгоритма с большей размерностью.

Также Вапником было показано для каждой задачи может существовать оптимальная размерность с точки зрения выше описанных свойств [28]

также из анализа разности ошибок на обучение и тесте следует условие оптимального обучения - то есть ситуация когда разность таких ошибок минимально. В других случаях речь идет о переобученности.

Из теории Вапника также следует, что чем больше тренировочная выборка и чем больше размерность алгоритма (его емкость), тем меньше ошибка на тренировочной выборке. То есть потенциально и тем меньше ошибка на тестовой выборке - то есть тем больше обобщающая способность.

Также надо отметить, что из размерности Вапника-Червоненкиса следует, например, что XOR проблема не решается линейным алгоритмом, но решается нелинейным (для 2 признаков размерность 3 для лин. алго и 5 для нелин.).

Размерность нелинейного алго можно повысить!? она растет полиномиально?! Принцип минимизации сруктурного риска!?

Как правило, в задачах компьютерного зрения не удается выделить (формализовать, описать) все необходимые признаки для ее решения. Например, достаточно сложным будет выделить все необходимые признаки для распознавания лица человека на изображении с произвольного ракурса. Таким образом проблема определения числа необходимых параметров нейронной сети и проблема размера выборки данных для обучения остается открытом вопросом. Однако, в пределе можно сказать, что число признаков равно размеру каждого экземпляра данных, например изображение размером 320×240 предельно будет содержать 76800 признаков, хотя на само деле их скорее всего будет меньше, но сколько определить нельзя. При этом, нейронная сеть вида 10 для такого количества признаков потребует наличие $\sim$ 12 миллиардов параметров и столько же входных данных -

что является достаточно большой цифрой даже для современной вычислительной техники (2021 год). Поэтому требует каких-либо использование методов снижения числа параметров. Проблема снижения числа параметров нейронной сети приводит в задачах компьютерного зрения к использованию сверточных нейронных сетей, изучению которых посвящена данная книга.

Важно отметить, что когда речь идет об оценки точности выражения 10 или 11, то не затрагивается вопрос обобщающей способности нейронной сети. Другими словами высокая точность работы нейронной сети для тренировочного набора данных не гарантирует высокую точность на данных отличных от тренировочных. С этим связана **проблема переобучения нейронных сетей (overfitting)** - то есть ситуации, когда нейронная сеть при обучении начинает запоминать искажения, шумы и т.д. в данных как новые признаки. Снижение влияния этой проблемы будет также подробно обсуждаться в данной книге. Также в продолжение темы обобщающей способности отметим, что в 1997 году была опубликована теорема "об отсутствии бесплатных завтраков" (No-Free-Lunch Theorme) [29]. По существу данная теорема утверждает, что для набора данных вероятность сделать правильное заключение одинакова для любых алгоритмов если они не обучены на подобных данных.

В ранних работа по многослойному персептрону предлагалось в каждом нейроне использовать логистическую функцию активации - такой выбор обоснован аналитическими соображениями, кроме того из выражения 10 следует сходимость нейронной сети с такими функциями активации. В 1991 году в работе [30] было показана одна из самых больших проблем метода обратного распространения ошибки при использовании функций активации с насыщением - **вымывание градиента (gradient vanishing)**. Суть данной проблемы заключается в следующем. Из выражения (9) следует необходимость вычисления производной функции активации $(\partial f(z_{k+1})/\partial z_k)$ и использование вычисленного значения при распространении ошибки по нейронной сети (пересчете для каждого слоя). Однако, если в одном из слоев k значение $f(z_k)$ слишком большое, то производная будет нулевой, в таком случае все веса слоев ниже k перестанут обновляться - то есть обучаться. Графики логиситческой функции активации и ее производной $sigma'(z)$ приведены на рисунке 9. Приведем данные выражения:

$$f(z) = \text{sigma}(z) = 1/(1+\exp(-z)) \quad \text{sigma}'(z) = \text{sigma}(z)(1-\text{sigma}(z)). \tag{12}$$

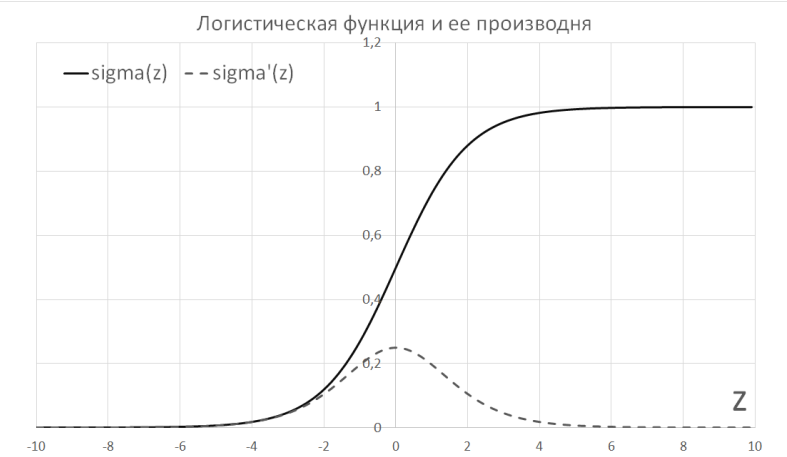

Рисунок 9 – график логиистической функции ($sigma(x)$) и ее производной ($sigma'(x)$)

Вероятность проблемы вымывания градиента растает с ростом числа слоев в многослойном персептроне, если в большинстве случаев $\partial f(z)/\partial z < 1$ (эффект геометрической прогрессии). Это приводит к тому, что начальные слои персептрона перестают обучаться. Если в большинстве случаев $\partial f(z)/\partial z > 1$ - происходит обратный вымыванию случай - **взрыв градиента (gradient explosion).** Данный эффект ведет к слишком большому значению обновлений значений весов нейронной сети, это также приводит к быстрому переходу значений функций активации в область насыщения. Таким образом, в идеале $\partial f(z)/\partial z = 1$ всегда. Однако, соблюдение данного условия достаточно сложно гарантировать, особенно при увеличении числа слоев в нейронной сети. Позже в книге будут показаны, меры, позволяющие снизить вероятность вымывания и взрыва градиента. Отметим, что в настоящее время предложены нейронные сети, имеющие до 1001 слоя - при обучении которых не возникает (обходят) соответствующие проблемы.

В окончании данного раздела следует отметить, что в настоящее время в компьютерном зрении полносвязные нейронные сети (многослойный персептрон) в своей оригинальной реализации как таковые не используются. Однако, они используются в качестве **завершающих слоев (head layers)**, например, в задачах классификации, а также многих других. При этом, в последнее время развивается альтернативный сверточномым сетям подход к компьютерному зрению, основанный на т.н. архитектурах - **трансформерах (vision transformer)**. Данный тип архитектур часто включает только полносвязные слои. Обзор архитектур данного типа будет дан в конце книги.

В начале 1990-х интерес к нейронным сетям упал в силу успехов развития классических методов машинного обучения и выявленных на тот момент технических ограничений в использовании нейронных сетей. В частности, одним из наиболее перспективных на тот момент стал метод опорных векторов (support vector machine, SVM) и его ядерная модификация - kernel SVM Отметим, что метод метод опорных векторов и его ядерная модификация разработаны Вапником (уже упоминавшимся в книге) [??]. Данный метод в настоящее время может быть классифицирован как частный случай многослойного персептрона с одним скрытым слоем радиально-базисной функцией активации и особым методом обучения на основе градиентного спуска. Метод имеет сравнительно низкую вычислительную сложность и часто позволяет получить неплохую точность. Однако, современные глубокие нейронные сети обходят данный метод по точности и обобщающей способности в большинстве задач, в том числе в задачах компьютерного зрения.

Техническими ограничениями связанными с использованием нейронных сетей на начало 1990-х являлись: проблема вымывания градиента, требования к обучению на достаточно большой выборке данных (а также поиск таких выборок и их хранение данных и) и сравнительно высокое время работы многослойных персептронов и необходимость хранить достаточно большой массив параметров (весовых коэффициентов) для их работы [7]. В настоящее время данные проблемы в той или иной степени решены, что и обуславливает популярность нейронных сетей сегодня.

1.2 Идеи сверточных нейронных сетей в 1980е - 1990е

1.2.1 Неокогнитрон и предпосылки к сверточным сетям

В 1975 г К. Фукушимой была предложена архитектура оригинальной искусственной нейронной сети - когнитрон [31]. Данная архитектура базировалась на работах по изучению принципов функционирования глаза, Одной из наиболее известных работ по этой теме является работа D. Hubbel [32]. Также стоит отметить, что в работах D. Hubel 1959 [33] и L.G.Roberts [34] была впервые предложена идея создания систем искусственного зрения как таковых.

Когнитрон стал прообразом использования искусственных нейронных сетей в задачах компьютерного зрения. Когнитрон представлял собой многослойную сеть, каждый слой которой состоял из т.н. ускоряющих и замедляющих нейронов (excitatory, inhibitory). Каждый возбуждающий нейрон следующего слоя был связан с близлежащими нейронами предыдущего слоя. А каждый замедляющий нейрон имел входы с близлежащих нейронов своего слоя и выходные связи с возбуждающими нейронами следующего слоя. Вход каждого возбуждающего нейрона определялся отношением суммы воздействий возбуждающих к тормозящим нейронам предыдущего слоя. Иллюстрация связей двух слоев когнитрона приведена на рисунке 10.

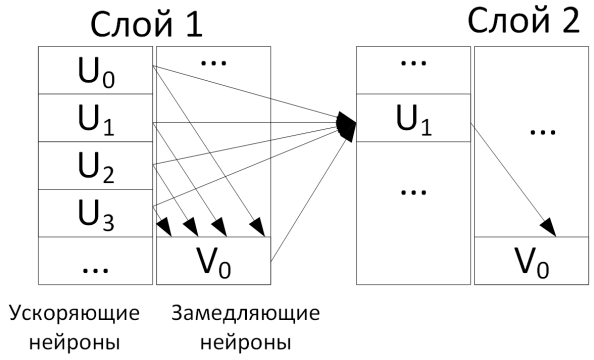

Рисунок 10 – иллюстрации работы двух слоев когнитрона.

Отметим, что согласно оригинальной идеи когнитрон имел 2 слоя. Обучение проходило по принципу самоорганизации - таким образом, что выход второго слоя должен был описывать входное изображение.

В 1980 году и С. Мийяке (Fukushimo, Miyake) была предложена модифицированная архитектура нейронной сети - неокогнитрон [35]. Основными отличиями новой архитектуры от когнитрона, описанной выше были:

- использование многослойной архитектуры;
- организация нейронов каждом слое в виде двух-мерных структур;

- использование в каждом скрытом слое нескольких т.н. планов;
- использование в каждом слое простых (С-клетка) и сложных (S-клетка) нейроннов;
- С-клетки выполняют функции клеток возбуждения в когнетроне;
- S-клетки выполняют функцию выделения максимальной информации (остальная информация локальной области обнуляется) - то есть выполняют функцию нелинейности (функцию активации);
- снижение размеров планов с увеличением глубины слоя таким образом, что последний слой представлял собой одно число на каждый план.

Использование нескольких планов в каждом слое нейронной сети позволяет по разному выучивать сетью одни и те же признаки во входных данных (особенности входного изображения). Использование S-клеток позволяет гарантировать что выделение максимальной информации о каждом признаке в независимости от его расположения. Таким образом целью С клеток является увеличение воздействия на каждый следующий нейрон, а целью S клеток является снижение такого воздействия при минимизации выходной информации нейрона. Архитектура С и S клеток в неокогнитроне называется латентным торможением [36]. Иллюстрация архитектуры неокогнитрона приведена на рисунке 11. Понятие "план" в архитектуре неокогнитрона близки по понятию **канал или карта признаков** в современных сверточных нейронных сетях. Понятие С-клетки близко к понятию **свертка**, а понятие S-клетки близко к понятию **субдесктретизация (pooling)** в современных сверточных нейронных сетях. Кроме того использование S-клетки в качестве функций активации близко к понятию т.н. maxout функций активации. Частным случаем maxout является наиболее популярная в настоящее время функция активации ReLU (полулинейная функция активации).

В основе представления данных в архитектурах когнитрон и неокогнитрон, по существу, был положены:

- принцип образования локальных связей;
- принцип конкуренции локальных связей;
- принцип образования **рецептивного поля.**

Описанные принципы заложены в основе современных глубоких сверточных нейронных сетей. Иллюстрации локальной связанности, конкуренции локальных связей и образования рецептивного поля приведены на рисунках 12 А), Б) и В) соответственно.

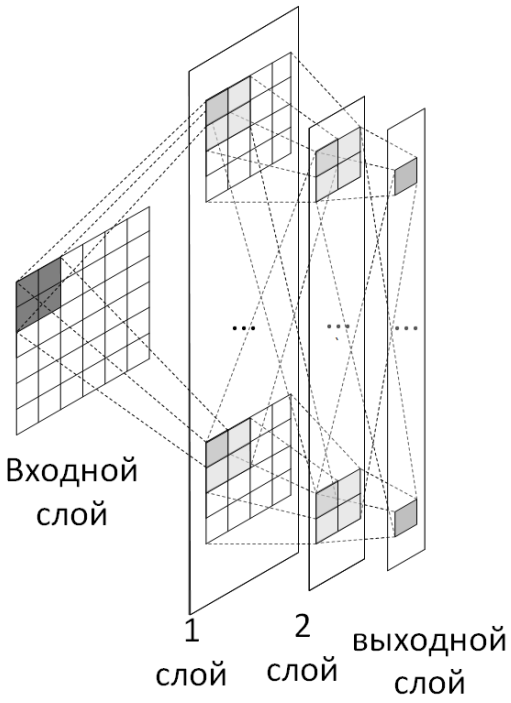

Рисунок 11 – иллюстрации архитектуры неокогнитрона. Разделение слоя на C- и S-клетки опущено.

Локальная связанность состоит в том, что каждый нейрон следующего слоя имеет связи только с близлежащими нейронами предыдущих слоев. Таким образом, каждый нейрон отвечает за информацию содержащуюся в определенной пространственно-локализованной области входного изображения (упорядоченного двух-мерного массива данных) (рис. 12 (А)). **Конкуренция локальных связей** заключается в том, что каждый участок входных данных может быть по разному учтен в разных нейронах следующего слоя (с разным весовым коэффициентом) (рис. 12 (Б)). Принцип локальной связанности позволяет сжать информацию с предыдущих слоев взвешенным суммированием. Конкуренция локальных связей позволяет избежать потери важной информации с предыдущего слоя за счет ее дублирования в разных нейронах.

Принцип образования рецептивного поля состоит в том, что в многослойной структуре нейроны каждого следующего слоя включают информацию сразу об локальной области нейронов предыдущего слоя. Таким образом, с увеличением количества слоев, каждый нейрон последующего слоя содержит в себе информацию о все большей области входного изображения (рис. 12 (В)). В силу локальности связей нейроны начальных слоев содержат в себе сравнительно простые составные части входного изображения, называемые **низкоуровневые признаки**. Каждый низкоуровневый признак сам по себе несет сравнительно мало информации - то есть "почти наверное" не позволяет сделать однозначный вывод о содержании

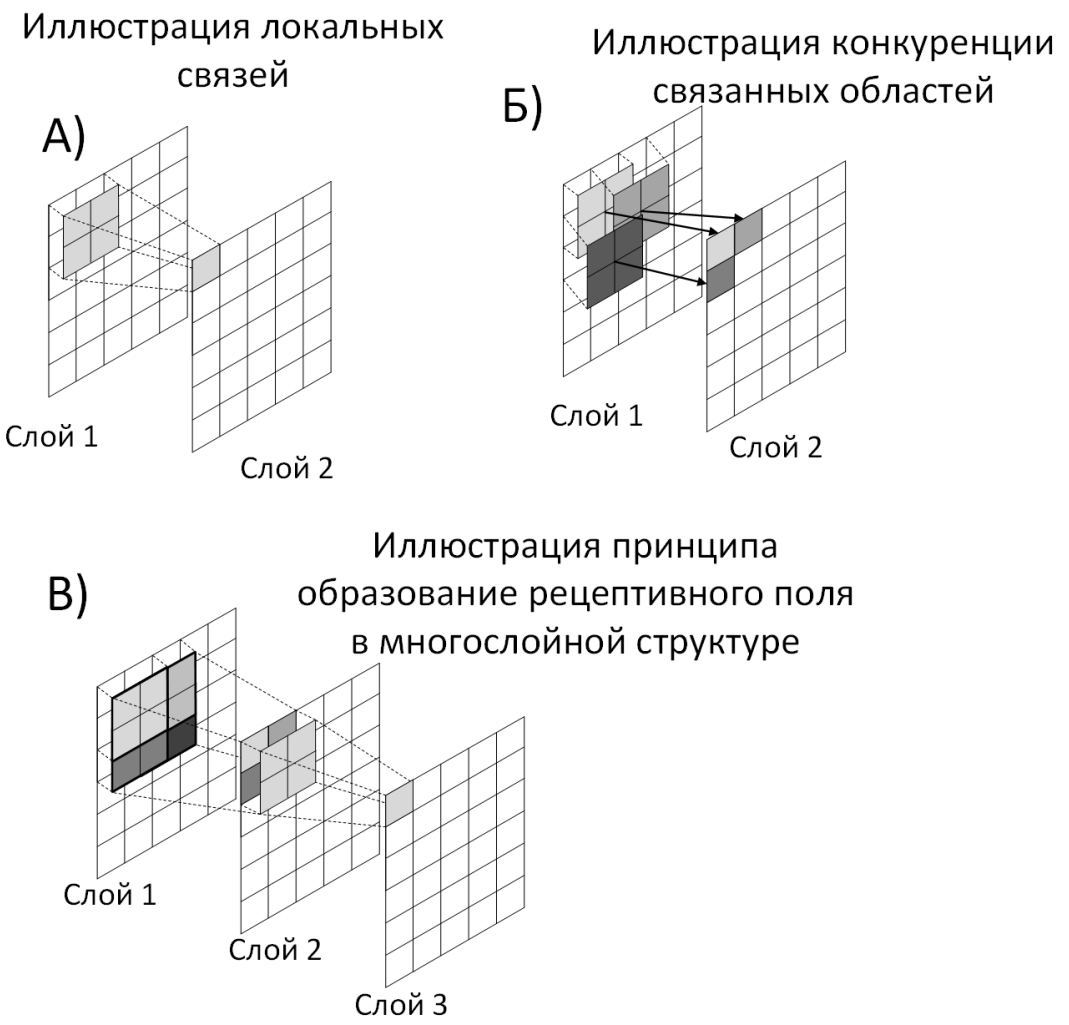

Рисунок 12 – иллюстрации локальной области связанности когнитрона (А), конкуренции областей связанности (Б) и образования рецептивного поля (В).

входных данных. Однако, с увеличением числа слоев в нейронной сети возрастает и сложность признаков - каждый признак несет в себе все больше информации о входных данных - то есть повышается вероятность однозначного сопоставления входных данных целевому результату. Описанные признаки называются **высокоуровневые признаки**. Иллюстрация возможности выделения низкоуровневых и высокоуровневых признаков при помощи рецептивного поля многослойной нейронной сети с локальной связанностью приведена на рисунке 13. Отметим, что принцип рецептивного поля был предложен в 1962 г в работе D. Hubel [37] в ходе анализа зрительной системы животных. Также следует отметить работу [38], в которой исследована иерархическая структуры глаза и многослойный принцип образования в нем рецептивного поля.

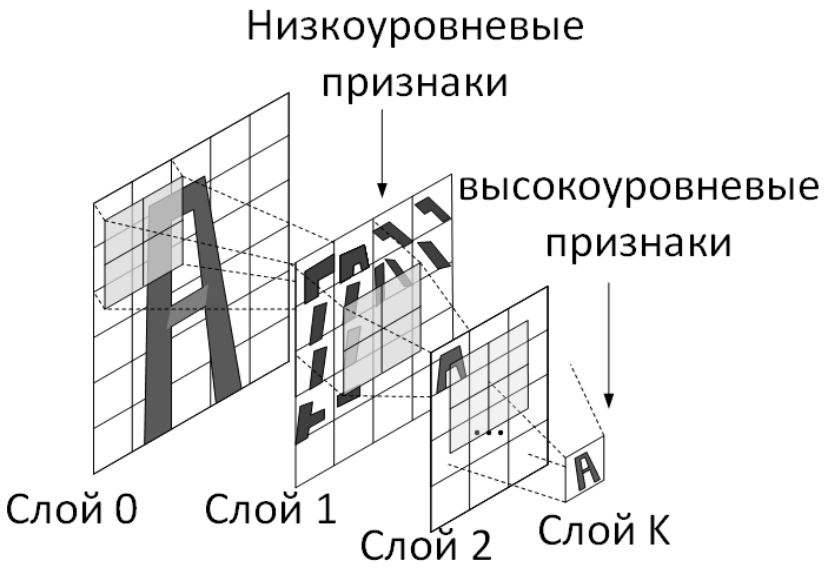

Рисунок 13 – Иллюстрация возможности выделения низкоуровневых и высокоуровневых признаков при помощи рецептивного поля многослойной нейронной сети с локальной связанностью.

В 1988 была предложена инвариантная к сдвигу искусственная нейронная сеть (SIANN, shift-invariant artificial neural network) - фактически ставшая прообразом сверточных нейронных сетей [39]. Сеть содержала два скрытых слоя, каждый из которых был организован подобно С-клеткам нейрокогнитрона + функция активации. В качестве функций активации использовались т.н. двойные сигмоиды ($f(z) = 2/(1 + \exp(-z)) - 1$). Сеть обучалась методом обратного распространения ошибки. Варианты модификаций данной архитектуры активно изучались в 1990-х [40]. Однако сегодня данная архитектура может быть рассмотрена только как подкласс сверточных нейронных сетей [41]. Также в 1989 году была описана архитектура одномерной сверточной нейронной сети для решения задач распознавания звуковых фонем [42].

1.2.2 Архитектура ConvNet

В 1989 г. Яном Лекуном была предложена первая сверточная нейронная сеть ConvNet [43]. Нейронная сеть была для решения задачи распознавания рукописных цифр по их фотографиям [44]. Архитектура ConvNet была первой сверточной нейронной сетью (convolutional neural network ,CNN) в современном понимании данного типа архитектур . В частности, в работе [44] термин сверточные нейронные сети (convolutional neural networ, CNN) был впервые популяризован [41]. Нейронная сеть ConvNet была успешно применена для распознавания рукописных цифр на почтовых индексах [43]. Иллюстрация архитектуры ConvNet, описанная в работе

[43] приведена на рисунке 14.

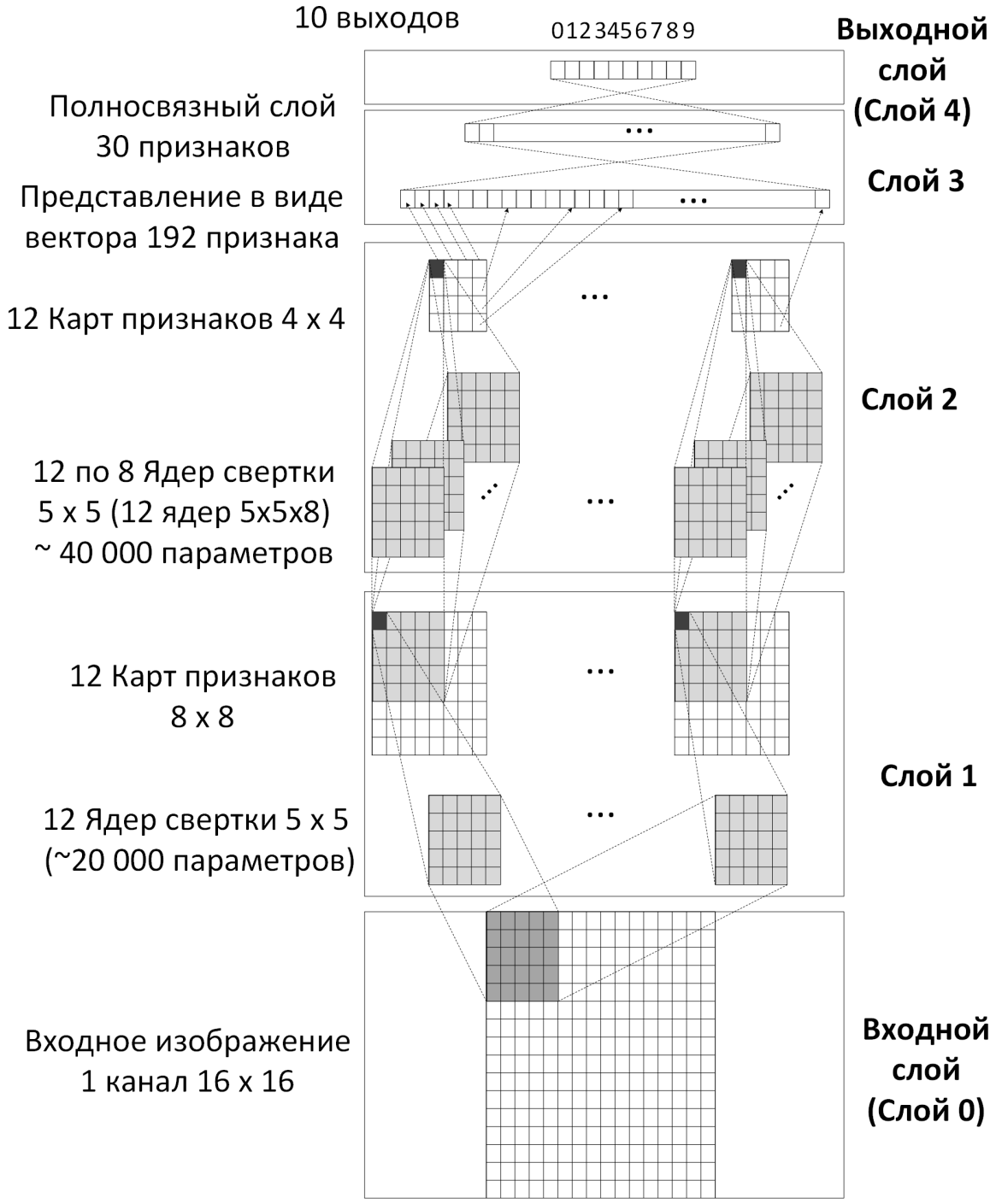

Рисунок 14 – Иллюстрация архитектуры сверточной нейронной сети ConvNet 1989 г.

Архитектура ConvNet обучалась методом обратного распространения ошибки, в качестве функции потерь использовался средний квадрат ошибки. В качестве функций активации - логистические функции. Нейронная сеть имела 3 скрытых слоя, два их которых сверточные и один полносвязный. Выходной слой также полносвязный и имеет 10 выходов по одному для каждой цифры. В основе архитектуры ConvNet (рис. 14) лежит понятие двухмерной свертки. Данное понятие является основополагающим для сверточных нейронных сетей. Основная идея использования свертки в нейронных сетях заключается в замене полносвязного слоя на одно или несколько сравнительно небольших сверточных ядер (не больших по общему числу параметров).

В случае одного двух-мерного ядра и одного канала изображения запишем операцию свертки в следующем виде:

$$r[i,j] = (k * x)[ij] = \sum_{a=0}^{h_k-1} \sum_{b=0}^{w_k-1} k[a,b]x[i+a, j+b], \tag{13}$$

где:

- $k[a,b]$ - значение ядра свертки с координатами $[a,b]$ (k имеет размерность $h_k \times w_k$);
- $x[i,j]$ - значение входного двухмерного массива с координатами $[i,j]$ (x имеет размерность $h_x \times w_x$);
- $r[i,j]$ - значение результата свертки с координатами $[i,j]$ (r имеет размерность $h_x - h_k + 1 \times w_x - w_k + 1$);
- $*$ - операция свертки.

Каждый сверточный слой нейронной сети ConvNet (рис. 14) образован несколькими наборами двухмерных сверток. Каждое двухмерное ядро свертки имеет свои значения весовых параметров. Результатом воздействия каждого сверточного ядра является карта признаков. Каждая карта выделяет свои признаки (но признаки одно уровня для одного слоя).

Каждый набор ядер свертки может быть описан многомерной структурой, которая в широком смысле может быть названа **тензор**. Например, слой 1 (рис. 14) можно описать при помощи трехмерных тензоров сверточных ядер и карт признаков на выходе слоя. В случае слоя 2 (рис. 14) имеется набор трехмерных тензоров сверточных ядер (четырех мерный тензор) и трехмерная карат признаков на выходе. При этом каждый элемент каждой карты образован суммой результатов соответствующих ей двухмерных сверточных ядер со смещением (**параметр смещения -bias** - дополнительный параметр каждого ядра).

Таким образом для выхода слоя 2:

$$r[c_{out}, i, j] = (k * x)[c_{out}, i, j] = \sum_{c=0}^{C-1} \sum_{a=0}^{h_k-1} \sum_{b=0}^{w_k-1} k[c_{out}, c, a, b] x[c, i + a, j + b] + \theta[c_{out}], \quad (14)$$

где

- C - число каналов свертки для входного массива;
- $x[c, i, j]$ - значение канала c входного массива с координатами $[i, j]$ (x имеет размерность $C \times h_x \times w_x$);
- $c_{out} = 0, ..., C_{out} - 1$ - число выходных карт признаков;
- $\theta[c_{out}]$ - смещение результата свертки ядра $k[c_{out}, :, :, :]$ (каждого ядра);
- $r[c_{out}, i, j]$ - значение выходной карты признаков c_{out};
- $k[c_{out}, c, a, b]$ - 4-х мерный массив ядер свертки;
- $\theta[c_{out}]$ - смещение ядра свертки.

1.2.3 Функция активации softmax

В 1989-1990 годах в работах [45, 46] было предложено использование в выходном слое нейронных сетей функцию активации softmax вместо набора выходов, каждый из которых имел свою логистическую функцию активации. Так, например, в работе [44] архитектура ConvNet1989 использовалась для решениях задача распознавания рукописных цифр. При этом сеть имела 10 выходных нейронов - каждый для одной цифры от 0 до 9 (см. рис. 14). Каждый нейронов имел свою логистическую функцию (2). Результатом работы сети выбирался ответ с максимальным значением (**принцип победитель получает все**). Однако, такой подход затрудняет интерпретацию результатов, особенно если результатов будет не 10, а скажем 100. *Например, пусть для одной эпохи обучения мы получили для двух выходов сети значения 0,95 и 0,92, а для следующей эпохи 0,96 и 0,955 - как понять на какой эпохе обучение было оптимальным?* Функция softmax позволяет устранить описанную проблему. В случае с функцией softmax результаты всех выходов нейронной сети нормируются так, чтобы в сумме всегда была 1. Тогда каждый результат можно интерпретировать как вероятность того, что, один экземпляр входного набора данных соответствует одному значению выходных данных (например, одному номеру метки класса). Функция softmax и ее производная $\text{softmax}'$ может быть записана следующим образом:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{c=0}^{C-1} \exp(z_c)}, \quad \text{softmax}'(z_i) = \text{softmax}(z_i)(1 - \text{softmax}(z_i)) \quad (15)$$

где C - число классов; z_i - входное значение функции - т.н. **логит (logit)**, (например для многоклассовой логистической регрессии $z_i = \sum_{j=0}^{N-1} w_{ij}x_j$, где w_{ij} матрица весов размером $C \times N$, x_j - набор входных значений).

Для реализации функции softmax нейронная сеть должна иметь C выходных нейронов без функций активации для каждого.Вектор результирующих значений выходных нейронов должен быть пересчитан для каждого значения по формуле 15. Результатом работы нейронной сети считается номер позиции максимального значения вектора:

$$\hat{y} = \arg\max_{\mathrm{i}}(\mathrm{softmax}(z_i)), \tag{16}$$

где $\arg\max_{\mathrm{i}}$ - функция номера позиции с максимальным значением; $\hat{y}$ - результат работы алгоритма. Иллюстрация работы слоя, реализующего выражения (15) и (16) приведена на рисунке

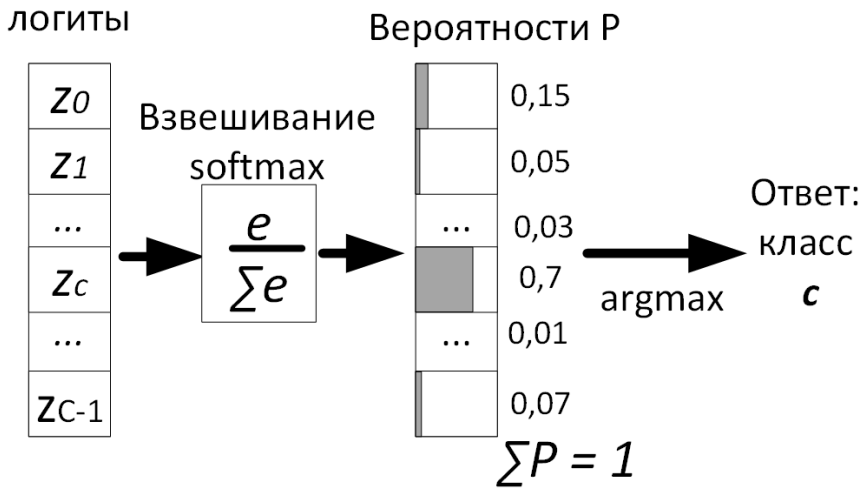

Рисунок 15 – Иллюстрация работы выходного слоя нейронной сети с функцией активации softmax

При обучении нейронной сети с функцией активации выходного слоя softmax схема работы будет несколько отличаться от проиллюстрированной на рис. 15. Так как во время обучения номер целевого класса известен вместо выражения 16 вероятность класса будет оцениваться и максимизироваться именно для целевых классов.

В работе [47] 1997 года преимущества использования функции softmax были показаны для задачи распознавания лиц по фотографиям с использованием сверточной нейронной сети и ряда других алгоритмов.

Отметим, что в современных нейронных сетях задачи классификации можно разделить на три вида:

- **задачи бинарной классификации** - необходимо оценить только наличие или отсутствие одного класса (например, есть или нет кошка на изображении); один выходной нейрон, выходная функция активации - логистическая функция;

- **задачи много-классовой классификации** - необходимо оценить наличие одного из нескольких классов (например, оценить это слон, собака или кошка); выходных нейронов, выходная функция активации softmax;

- textbfзадачи много-меточной классификации- необходимо оценить вероятность наличия по каждому классу одновременно (например одновременно оценить есть ли на изображении и кошки и собаки и слоны) - выходных нейронов,, для каждого выходная функция активации - логистическая функция.

Также следует отметить, что **если решается задача регрессии** (например определить позицию или длину кошки), то используется число нейронов, равное числу оцениваемых параметров, каждый выходной нейрон не имеет функции активации.

1.2.4 Архитектура LeNet 5

В 1998 Яном Лекуном была предложена модифицированная относительно ConvNet (рис. 14) нейронная сеть **LeNet5** [18]. Данная нейронная сеть была применена для распознавания рукописных цифр без предварительной обработки изображений. Также в работе [18] был впервые использован набор данных - рукописных MNIST (Modifyed NIST), который в настоящее время является одним из наиболее популярных в задачах компьютерного зрения (набор доступен по ссылке [48]. Иллюстрация архитектуры нейронной сети LeNet5 приведена на рисунке 16. По существу, большинство современных сверточных нейронных сетей (на момент написания книги) являются наследниками архитектуры LeNet5. Более того LeNet5 часто является базовой архитектурой, с которой проводятся сравнения более совершенных архитектур. Также архитектура LeNet стала первой, доказавшей, что системы распознавания, основанные на автоматическом обучение (нейронные сети) могут работать лучше чем системы, основанные на вручную описанных эврестических правилах (классические методы машинного обучения) [18]. Отметим, что помимо архитектуры LeNet5 в работе [18] были рассмотрены и другие варианты систем компьютерного зрения, не показавшие высокой точности в рассмотренной задаче. В частности графовые нейронные сети, сети пространственного смещения, классификаторы не основе метода ближайших соседей и метод опорных векторов, а также некоторые другие.

Архитектура СНС LeNet5 включает в себя 5 слоев (2 сверточных слоя, 2 полносвязных слоя и 1 выходной слой).

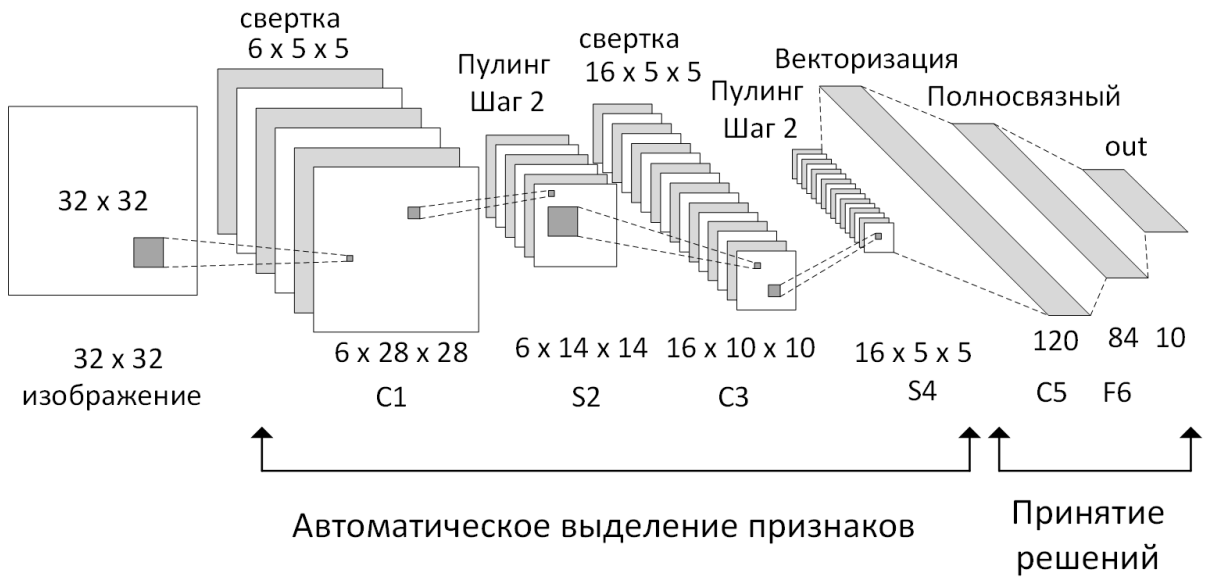

Рисунок 16 – Иллюстрация архитектуры сверточной нейронной сети LeNet5

В отличии от ConvNet 1989 г в архитектуре LeNet5 каждый сверточный слой дополнен операцией **субдискретизация (пулинг, pooling)** [18]. Идея слоя пулинга уже упоминалась ранее касательно неокогнитрона. Однако, в LeNet она имеет несколько иной смысл. Слой субдескретизации позволяет снизить размерность карт признаков. Предполагается, что при этом не произойдет потери информации. Для проведения субдескретизации по карте признаков устанавливается окно (например размером 2×2 скользящее по карте с шагом 2 (возможны и другие варианты). В каждом окне формируется одно значение (в LeNet5 среднее значение - **average pooling**):

$$r\left[\frac{i}{s}, \frac{j}{s}\right] = \sum_{a=i}^{i+w_p} \sum_{b=j}^{j+h_p} x[a,b], \tag{17}$$

где:

- $r\left[\frac{i}{s}, \frac{j}{s}\right]$ - результат операции пулинг (размерность $w_r \times h_r$);
- $x[a,b]$ - входной двух-мерный массив (размерность $w_x \times h_x$);
- $w_p \times h_p$ - размеры окна пулинга;
- s - шаг перемещения окна пулинга;
- $w_r = \frac{w_x - w_p}{s} + 1; h_r = \frac{h_x - h_p}{s} + 1.$

Иллюстрация работы пулинга приведена на рисунке 17. Следует отметить, что использование субдискретизации потенциально может привести к потери информации. По этому каждый слой, содержащий субдескретизацию должен иметь увеличенное число карт признаков по сравнению с предыдущим слоем (как правило удвоенное) - для компенсации возможных потерь. Автором LeNet также постулируется общее положение: **снижение пространственного разрешение в нейронной сети должно сопровождаться пропорциональным увеличением числа карт признаков.**

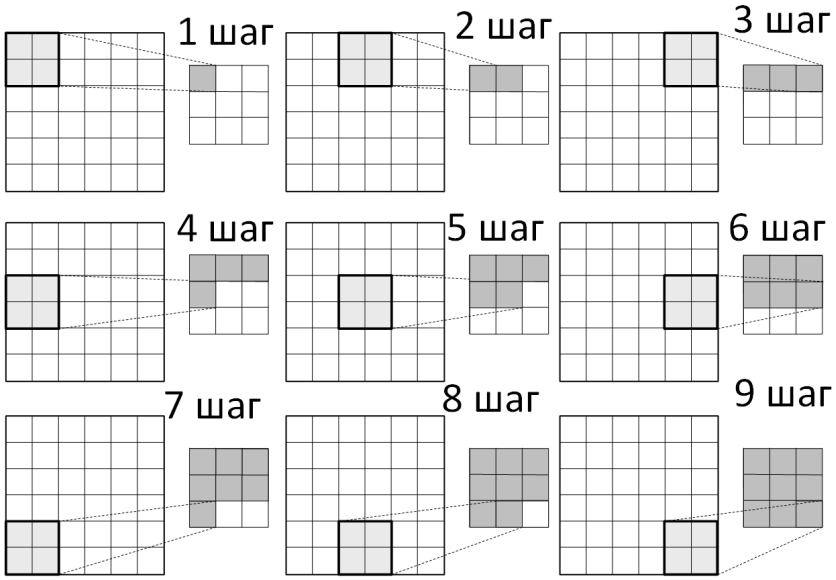

Рисунок 17 – Иллюстрация принципа работы пулинга

Также следует отметить, что в настоящее время операция субдескретизации, в виде, описанном выше (17)- локальный усредняющий пулинг используется редко. Это связано с неустойчивостью усреднения к шумам и выбросам в данных. Наиболее популярный на сегодняшний день вид субдискретизации - максимальный пулинг (**макспулинга, maxpooling**). Результатом работы макспулинга является число с наибольшей интенсивностью среди всех выделенным окном субдискретизации - то есть с наибольшей информацией.

$$r\left[\frac{i}{s},\frac{j}{s}\right]=\max_{a,b}x[a,b],\ a=i,...,i+w_p,b=j,...,j+h_p. \tag{18}$$

Идея макспулинга уже упоминалась в приложении к неокогнитрону, где она называлась S-клетка. Однако, окончательно данная идея была сформулирована в работе [49] в 1992 году [12].

По существу, нейронная сеть LeNet и все архитектуры на ее основе состоит из двух основных частей:

- 1 часть: **энкодер признаков - система автоматического выделения признаков (feature extractor)** - каскад сверточных слоев.
- 2 часть: **головная часть - решения задачи - в данном случае многослойный персептрон**. Данная часть архитектуры может быть выбрана различным способом в зависимости от задачи.

В классических методах машинного обучения предполагается наличие лишь части 2 из описанных выше. При этом признаки - входные данные части 2 в классических методах выделяются вручную (например основываясь на некоторых математических моделях решаемой задачи или эвристически - на основе наблюдений). В архитектуре LeNet не требуется решение проблем подбора или выделения признаков из данных, их описания и т.д. эта задача решается в ходе обучения нейронной сети. Такой подход особенно полезен в системах компьютерного зрения - где, как уже было сказано выше, формальное описание задачи вручную крайне сложно, если вообще возможно.

Также следует отметить, что часть 1 описанного выше разделения архитектуры НС, выделяет признаки аналогично концепции рецептивного поля, описанной выше (см. неокогнитрон). Таким образом, чем глубже сеть (чем больше слоев в энкодере признаков) тем большая область входного изображения может быть обработана. То есть чем больше слоев - тем более высокоуровневый признак и признак большего размера может быть выделен во входном изображении.

Однако, подход LeNet имеет ряд ограничений по сравнению с классическими методами машинного обучения. Во-первых, подход требует большего времени обучения и большего размеры выборки данных - так как помимо головной части системы необходимо обучить энкодер признаков. Во-вторых, НС требует больше ресурсов для работы (память, вычислительные ресурсы) - так как НС имеет больше параметров, чем классические системы. В-третьих, НС не могут обучаться он-лайн - в ходе своей работы. То есть можно обучить НС, использовать ее, и копить новую информацию и данные, затем, если нужно, то необходимо полностью переобучать НС и использовать уже новый результат на практике. Отметим, что на практике данные ограничения не снижают применимости НС в задачах компьютерного зрения и многих других.

По задумке автора архитектура нейронной сети LeNet обладала следующими полезными свойствами инвариантности. Данные свойства полезно указать, так как они являются частью общих достоинств сверточных нейронных сетей перед другими алгоритмами обучения [18, 50].

- **Инвариантность к масштабу** целевой сцены (или объекта) на входном изображении (в некоторых пределах). Инвариантность достигается за счет реализации концепции локального рецептивного поля по средствам регулирования глубины энкодера признаков и размерах ядер свертки.

- **Инвариантность к положению** целевой сцены (или объекта) на входном изображении. Инвариантность достигается за счет пере-использования весовых параметров (использование свертки) в приманенная к разным частям изображения. В частности в работе [18] отмечается, что если объект на входном изображении сдвинуть, то и объекты на каждой карте признаков сдвинуться, но не изменится - сам признак останется выделанным. Однако, если на изображении есть несколько сцен и их взаимные позиции изменились, то инвариантность к этой ситуации может не сохраниться [18].

- **Инвариантность к искажению** целевой сцены (или объекта) на входном изображении (в некоторых пределах) - то есть расширение обобщающей способности. Инвариантность достигается за счет использования слоев субдискретизации. Таким образом, из каждой области каждой карты признаков будет выделена лишь наиболее информативная составляющая. При этом предполагается, что небольшие искажения тестовых объектов по отношению к тренировочным не дадут высокой интенсивности - так как под них НС не обучена. Другими словами ядра свертки не коррелируют с искажениями на которых они не обучены и дадут что то около нуля.

- Переиспользование весовых параметров (использование свертки, **пространственная инвариантность весовых параметров**). При этом при использовании сверток, в отличии от нейронов неокогнитрона, в СНС число параметров меньше чем в полносвязных НС. Так для LeNet5 число тренируемых параметров ~ 60000, а аналогичная полносвязная НС имела бы ~ 340000.

Отметим также деталь архитектуры LeNet5, полезную с точки зрения дальнейшего изложения материала: В слои 3 фактически ***использовалась групповая свертка*** *(4 группы) - то есть каждая выходная карта признаков образовывалась только частью входных карт признаков[18]. Это во-первых позволяет экономить вычислительные ресурсы, а во-вторых позволяет избежать т.н. симметрии слоя - ситуации, когда все карты признаков выделяют один и тот-же признак.*

В 1998 году Лекуном также была опубликована работа, посвященная анализу различных приемов улучшения точности LeNet [51]. В частности в работе предложены и описаны достоинства следующих приемов.

- Предлагалось использовать использовать **в качестве функции активации гиперболический тангенс** вместо сигмодиа $f(z) = tanh(z)$ или его модификацию $f(z) = Atanh(Sz) + Bz$, где A, S, B - константы.

- Предлагалось проводить **нормализацию входных изображений** таким образом, чтобы математическое ожидание (Среднее по всем экземплярам) было равно 0 а дисперсия 1. Это необходимо для понижения вероятности переобучения слоев с сигмоидом, а также стандартизации условий работы сети (например, все весовые параметры можно инициализировать значениями от 0 до 1.

- Предлагалось проводить **инициализацию весовых коэффициентов** каждого слоя нейронной сети случайными величинами с нормальным распределением и дисперсией обратно-пропорциональной числу параметров в слое - **инициализация Лекуна**.

$$W_i \sim N\left(0, \frac{1}{\sqrt{n_i}}\right), \tag{19}$$

  где n_i - число параметров слоя с номером i, W_i - набор весовых коэффициентов слоя с номером i; $N(0, 1/\sqrt(n_i))$ - нормальное распределение с 0 математическим ожиданием и дисперсией $1/n_i$ При использовании инициализации Лекуна дисперсия весовых коэффициентов в каждом слое равна 1, что является оптимальной ситуацией с точки зрения снижения вероятности вымывания градиента.

- Предлагалось использовать **стохастический пакетный градиентный спуск (minibatch stochastic gradient descent, SGD)**. В каждую эпоху обучения предлагалось случайным образом **разделять набор тренировочных данных на пакеты (батчи, batch)** фиксированного размера. Для каждого пакета (или некоторой выборки пактов) предлагалось проводить процедуру прямого прохождения и обратного распространения ошибки.

- Предлагалось использовать обновления весовых значений методом SGD с т.н. моментом (экспоненциальным сглаживанием, **SGD with moment**). Использование момента позволяет снизить влияние случайности в стохастическом пакетном градиентном спуске. При этом выражение 7 может быть переписано в виде:

$$W^{\{t+1\}} - W^{\{t\}} = -\eta \nabla L\left(\hat{y}^{\{t+1\}}, y^{\{t+1\}}\right) + \mu\left(W^{\{t\}} - W^{\{t-1\}}\right), \tag{20}$$

  где μ коэффициент сглаживания. Как правило $\mu \approx 0.9$.

- Предлагалось использовать **адаптивные стратегии изменения скорости обучения**. Подход позволяет снизить виляние неверного выбора скорости обучения не результат, и ускорить обучение (сначала высокая скорость, потом низкая). Позже в книге данные методы будут рассмотрены более подробно.

- Предлагалось использовать **процедуру перекрестной проверки (кросс валидация, cross-validation)** в качестве процедуры остановки обучения (т.н. **ранняя остановка, early stop**). Для проведения данной процедуры тренировочная выборка данных должна была быть предварительно поделена на две части - тренировочную (порядка 70%) и валидационную (порядка 30%). Обновление значений весовых параметров происходит только по результатам средней ошибки, вычисленной по тренировочной выборке. Затем результат работы сети (ошибка работы) вычисляется для валидационной выборки. Если ошибки на тренировочной выборке падает, а на валидационное нет (в насыщении или растет), то такая ситуация называется **переобучение (overfitting)**. Если ошибка на валидационной выборке близка и снижается вместе с ошибкой на тренировочной выборке, то такая ситуация называется **недообучение (underfitting)**. В момент, когда ошибка на тренировочной и валидационной выборке равны друг другу (зависимости пересекаются) обучение считается оптимальным. Таким образом, валидационная выборка позволяет определить эпоху обучения НС в которую значения весовых коэффициентов имеет значения, позволяющие получить максимальную обобщающую способность. Иллюстрация процесса обучения (график зависимости ошибки обучения $L(\hat{y}, y)$ от эпохи (ep) приведен на рисунке 18. Также на рисунке отмечены ситуации недообучения, переобучения и эпохи ранней остановки ep^* - оптимального обучения. Отметим, что при обучении нейронной сети весовые параметры учатся выделять признаки из входных данных. Ситуация переобучения нейронной сети соответствует тому, что сеть начинает воспринимать искажения или другие помехи/шумы в тренировочном наборе данных как признаки. При этом сеть старается подстроиться под них, что позволяет добиться точности на тренировочном наборе вплоть до 100%. Однако, НС запоминает на наиболее общие признаки, а частные искажения данных тренировочной выборки, то на валидационных данных ошибка не будет снижаться. Ситуация недообучения соответствует тому, что НС не научилась выделять все необходимые признаки, необходимые для правильного принятия решений.

Работа [51] стало обобщением опыта по обучению нейронных сетей на момент ее написания. Многие из приемов, описанных в [51] используются в НС компьютерного зрения по настоящее время.

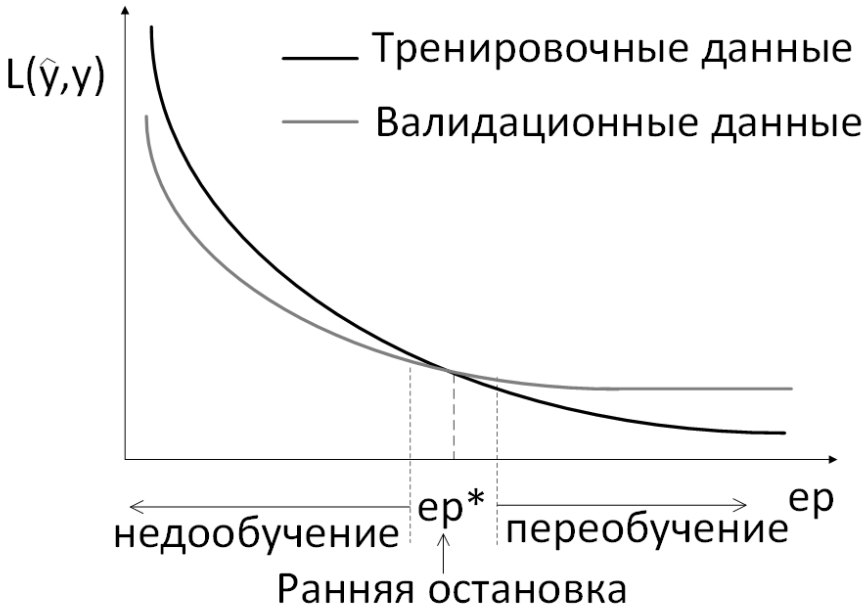

Рисунок 18 – Иллюстрация зависимости ошибки обучения от эпохи для тренировочной и валидационной выборок, ep^* - эпоха ранней остановки

1.2.5 Регуляризация обучения нейронных сетей

В работах 1989-1990 годов Погио и Гироси было показано, что точность работы нейронных сетей может быть повышена регуляризацией функции потерь при их обучении методом обратного распространения ошибки [52, 53]. Типичные техники регуляризации были предложены задолго до работа данных авторов. В том числе, наиболее популярная техника - гребневая регуляризация была предложена советским математиком А.Н. Тихоновым [54, **?**]. Данный тип регуляризации в наиболее простом виде часто называется **L2 регуляризация или регуляризация Тихонова**. По существу авторы работ [52, 53] L2 регуляризацию к методу обратного распространения ошибки. По существу, задача обучения с регуляризацией соответствует введению дополнительного ограничения для оптимизируемых параметров (весовых коэффициентов):

$$\begin{cases} L\left(y, \hat{y}(W, X)\right) \to \min \\ \sum_j w_j^2 \leq \text{const} \end{cases} \Rightarrow L\left(y, \hat{y}(W, X)\right) + \frac{\lambda}{2} \sum_j w_j^2 \to \min, \tag{21}$$

где $L\left(y, \hat{y}(W, X)\right)$ - функция потерь нейронной сети; $W = \{w_j\}$ - набор обучаемых весовых коэффициентов нейронной сети; λ функция регуляризации. Использование регуляризации запрещает переобучаться нейронной сети за счет введения т.н. штрафа λ в функцию потерь за слишком большую норму значений весовых параметров ($\|W\|_2^2 = \sum_j w_j^2$). Другими словами задача обучения из формулировки

"функция потерь должна быть минимальной" превращается в формулировку "должен быть найден баланс между минимум функции потерь и максимум нормы значений весовых параметров". Заметим, что интуитивно понятно (и доказывается математически), что чем меньше значения весовых параметров тем меньше дисперсия (разброс) результатов - то есть выше обобщающая способность нейронной сети. Однако, можно также заметить, что введение регуляризации вводит смещение в функцию потерь, то есть выражение 21 ни когда не достигнет нулевых значений кроме как в тривиальном случае (когда все веса $= 0$, но этот случай нас не интересует). Таким образом, введение регуляризации вносит дополнительную ошибку в результаты обучения нейронной сети. Другими словами - принцип регуляризации - это компромисс между минимизацией дисперсии и смущением ошибки обучения нейронной сети. На практике значения λ выбираются достаточно небольшими (порядка $10^-5 - 10^-6$), чего однако, часто хватает для необходимого повышения обобщающей способности без существенной потери точности работы нейронной сети. Отметим, что помимо L2 регуляризации в ряде случаев в нейронных сетях используется L1 регуляризация (LASSO) [55] и комбинация L1 и L2 операций регуляризации [56]. Однако, именно L2 регуляризация является наиболее популярным среди аналогов [57]. Также в ряде источников семейство описанных методов регуляризации может называться weight decay. Особенностях данных и некоторых других типов регуляризации будут еще раз рассмотрены в книге в соответствующих разделах.

1.2.6 Аугментация данных

Существенно улучить результаты Лекуна удалось авторам работы [58] 2003 года. Для этого авторами был предложен ряд техник расширения тренировочного набора данных путем искажений оригинальных экземпляров изображений - данная техника в настоящее время называется **Аугментация (augmentation - дополнение)**. Техники аугментации изображений были предложены в работе [59] 1992 года, однако популярность обрели только после работы [58]. Аугментация как правило выполняется путем аффинных (обратимых) искажений входных изображений или добавления к ним шумов или помех (например блики) [60]. Примеры результатов аугментации для изображения рукописной цифры приведены на рисунке 19. Отметим, что операцию аугментации следует выполнять с осторожностью. Мы рекомендуем начинать с обучения нейронной сети без аугментации, а затем постепенно увеличивая долю аугментированных данных следить за ошибкой на тестовых данных. Дело в том, что слишком интенсивной аугментацией можно изменить частоту встречи тех или иных признаков в данных или создать новые признаки. В современных нейронных сетях аугментация проводиться в процессе обучения нейронной сети случайным образом, что позволяет частично снизить

вероятность появления описанных выше проблем. Отметим, что аугментация может быть рассмотрена как техника регуляризации нейронных сетей.

Рисунок 19 – Иллюстрация примеров аугментации изображений для расширения входной выборки

1.3 Глубокие нейронные сети 2000-2012

1.3.1 Сети глубокого доверия, автокодирующие сети и метод жадного предобучения

В середине 90-х и до 2005 (порядка 10 лет, не считая работ некоторых авторов) интерес к изучению нейронных сетей упал в силу технических ограничений вычислительной техники и проблемы вымывания градиента. Однако, например, по оценкам Лекуна [50] в конце 90-х около 10% чеков в банках США обрабатывались с использованием сверточных нейронных сетей.

В 2006 год Джефри Хинтоном была предложена техника предобучения нейронных сетей при помощи жадного послойного обучения [61, 62]. Данный подход частично позволял избежать проблем выявленных ранее [9]. В основе данной идее лежала архитектура нейронной сети - ограниченная машина Больцмана [63, 64], - архитектура предложена в 1986 г, полная машина Больцмана предложена в 1985 г. на основе т.н. сети Хопфилда (1982 г) ([65]) - подробное изложение данных архитектур выходит за рамки данной книги. В 2006 году ограниченная машина Больцмана была применена Хинтоном для решения задачи кодирования изображений (сжатие изображений) [66]. Данная архитектура нейронной сети в настоящее время известна как **автоэнкодер**. Отметим, что технически идея автокодерования была описана в 1987 в работе [67] где была названа автоассоциативной сетью. Идея автокодирования в данной архитектуре основывалась на гипотезе о том, что многослойные нейронные сети позволяют выделять внутренние (закодированные) представления о регулярностях в данных (признаках) в окружающей среде (в обучающем наборе данных). Отметим, что также в работе [67] была указана гипотеза о преимуществах иерархического (многослойного) кодирования входных данных [67].

Типичная архитектура автоэенкодера представлена на рисунке 20. Сеть стоит из двух частей:

- часть 1: **энкодер** - сжатие входных данных - автоматическое выделение из них признаков. Принцип работы данной части аналогичен энкодеру признаков LeNet 5. Выход данного слоя часто называется **скрытым или латентным пространством (latent space).**
- часть 2: **декодер** - автоматическое восстановление данных из входных признаков (скрытого пространства).

Сеть автоэнкодер обучается без учителя таким образом, что выходное изображение должно повторить входное. Ошибка воспроизведения минимизируется в ходе итерационного обучения. Выход обученного энкодера используется как кодирующая сеть. В случае необходимости закодированное изображение может быть восстановлено при помощи декодера.

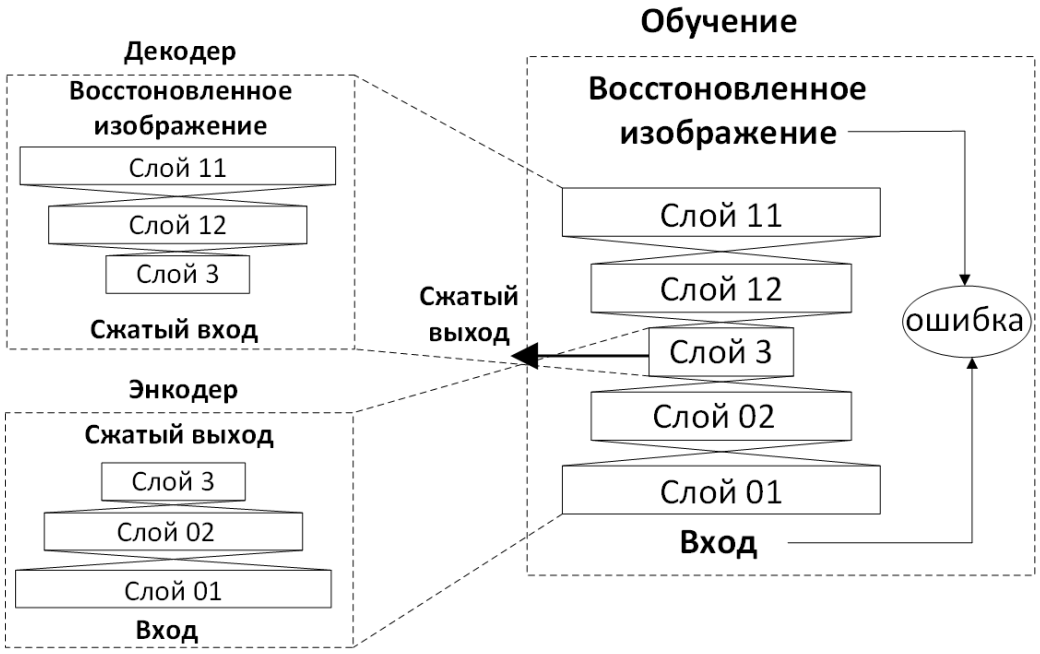

Рисунок 20 – Иллюстрация архитектуры автоэенкодера

Техника жадного послойного обучения представляет собой использование автоэнкодера для каждого слоя нейронной сети. Иллюстрация данного подхода приведена на рисунке 21. Для первого слоя входными данными являются целевые входные данные НС. Результат предобучения первого слоя используется в качестве входных данных для предобучения второго слоя и т.д. Нейронные сети, предобученные методом жадного послойного предобучения часто называются **нейронные сети глубокого доверия.** [9].

После предобучения (**pretrained**) НС должна быть дообучена (**fine tuning**) для решения целевой задачи. Например, в методом обратного распространения ошибки - в случае задачи обучения с учителем.

Идея предобучения нейронных сетей заключается в ожидании того, что весовые параметры обученных сетей будут настроены таким образом чтобы максимально эффективно выделять характерные признаки из входных данных. При этом ожидается, что приблизительно те же признаки должна научиться выделять НС в ходе обучения так как признаками должны быть все наиболее регулярные особенности тренировочной выборки. Другими словами весовые параметры предобученной НС почти наверное позволяют использовать НС для решения целевой задачи. В работе 2010 года [68] на основании большого числа экспериментов авторами было показано, что использование предобучения нейронных сетей является методом регуляризации обучения нейронных сетей для целевой задачи. Также было показано, что использование предобучения НС повышает обобщающую способность и снижает дисперсию ошибок для тестовых данных. Другими словами снижается

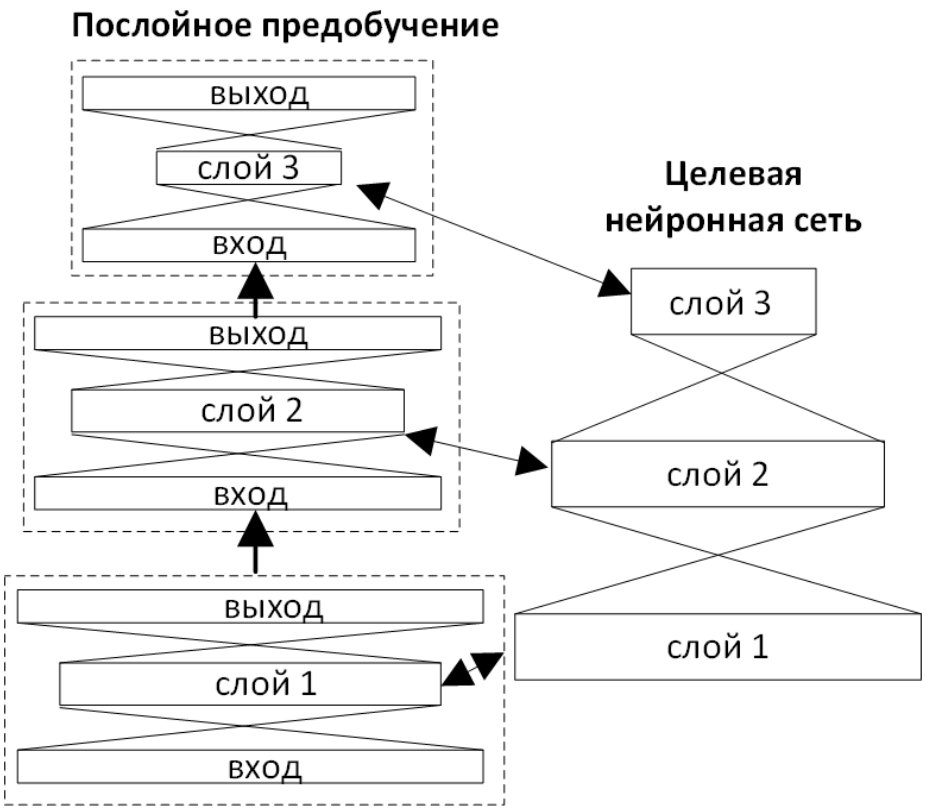

Рисунок 21 – Иллюстрация принципа жадного послойного предобучения (сети глубокого доверия)

как ошибка на тренировочных данных, так и ее разность с ошибкой на тестовых данных, так и дисперсия ошибки на тестовых данных [68].

В 2007 году Й. Бенджо обобщил результаты Д. Хинтона и показал, что использование предобучения нейронных сетей позволят тренировать нейронные сети гораздо большей глубины [69]. Также отметим, что Д. Хинтона, Й. Бенджио и Я. Лекуна на 2006 год работали совместно в рамках программы CIFAR NCAP [9]. В 2004-2006 годах в работах [70, 71, 72] были показаны преимущества использования графических ускорителей (Graphic Process Unit, GPU) при обучении сверточных нейронных сетей. Фактически с описанных работ начинается **эпоха глубокого обучения нейронных сетей**.

1.3.2 Термин "Глубокое Обучение"

Идея **глубокого обучения (Deep Learning) нейронных сетей** была высказана в работе Бенджо и Лекуном в работе [73]. В данной работе авторы авторы призвали сообщество исследователей методов машинного обучения к разработке алгоритмов, позволяющих максимально автоматизировать процесс выбора признаков, необходимых для решения задач. Также авторы показали, что необходимым условием выделения наиболее абстрактных признаков (высокоуровнеаых признаков) является увеличение глубины нейронных сетей. Для глубоких нейронных сетей было показано, что с увеличением глубины нейронных сетей возрастает их обобщающая способность. При этом обобщающая способность обусловлена нелинейностями в НС [73]. Также отмечается, что для сверхточных нейронных сетей все признаки являются пространственно локализованными. Признаки различных уровней образуют иерархию, в которой локальные высокоуровнеые признаки формируются нелинейными комбинациями более низкоуровных локальных признаков. [50].

Отметим, что авторы [73] в противоположность глубокому обучению авторы понятие мелкого обучения (**shallow learning**) - нейронные сети с одним или небольшим количеством слоев. При этом, обещающая способность мелкой нейронной сети можно назвать локальной. Преимущества глубокой нейронной сети в этом случае можно обосновать обобщением большого количества локальных областей обобщения. Также авторами [73] была отмечена одна из основных проблем глубокого обучения - т.н. **"проклятие размерности"** как основного фактора, ограничивающего углубление нейронных сетей. Проклятие размерности заключается в экспоненциальном росте числа параметров слоя НС при линейном увеличении числа входов. Решение данной проблемы является одним из трендов развития НС по настоящее время. При этом авторами [73] была выдвинута гипотеза, что переход от однослойной архитектуры к многослойной позволяет сократить число параметров, необходимое для решения задачи.

ГЛАВНОЕ ПРЕИМУЩЕСТВО DL; выделение, преобразование и выбор признаков, релевантные поставленной задачи. Однако процедура обучения требует достаточно большого набора данных для правильной обработки таких признаков

1.3.3 Функция активации ReLU

В 2009-10х годах в работах [74, 75] было предложено использование ректифицированных функций активации вместо логистической функции активации и других вариантов функций с насыщением (ректифицированных - исправляющих, например операция модуль в качестве функции активации). Среди различных вариантов ректификационных функций наилучших результатов удалось добиться для функции **ректификационный линейный модуль (ReLU - rectified linear unit)**. График функции ReLU и ее производной ReLU'(z) представлены на рисунке 22 Функция ReLU(z) и ее производная ReLU'(z) имеют следующий вид:

$$\mathrm{ReLU}(x) = \max(0, x), \quad \mathrm{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{22}$$

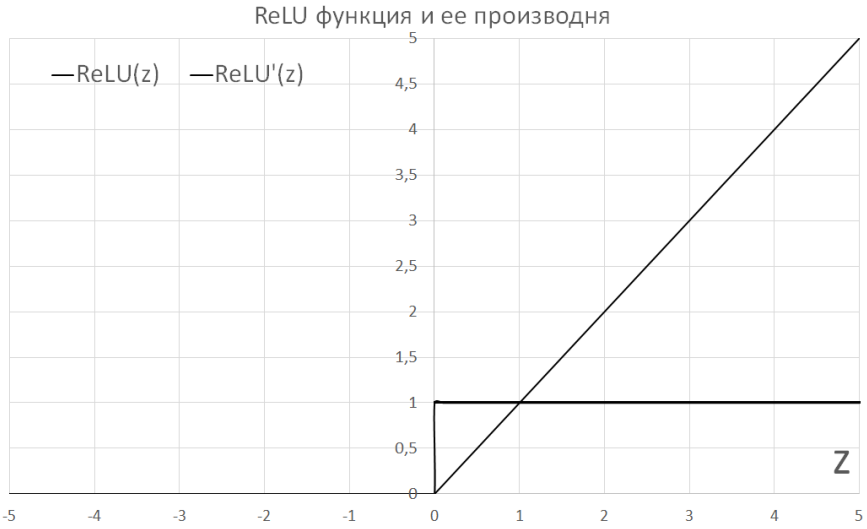

Рисунок 22 – график функции ReLU и ее производной

Использование функции ReLU позволяет частично избежать проблем, связанных с вымыванием градиента. Так как функция не имеет насыщения в области $z > 0$. При этом функция имеет низкую вычислительную сложность и позволяет обнулить часть весовых коэффициентов, что также может исключить часть вычислений (если один из параметров 0, то умножение на него можно не проводить). Функция ReLU является одной из наиболее популярных в настоящее время. Как правило, нейронные сети с использованием ReLU показывают наилучшие результат среди других функций активации. Однако, потенциально у функции ReLU есть ряд недостатков, о которых будет сказано в свое время. Также следует отметить, что в настоящее время по мимо описанной реализации ReLU в литературе предложено семейство линейных и нелинейных модификаций этой функции активации, которые могут быть использованы если стандартной реализации недостаточно [76].

1.3.4 Инициализация Весовых параметров

В 2009 году Хавьером Глори и Джефри Хинтоном было предложено в качестве альтернативы предобучению нейронных сетей проводить инициализацию их весовых параметров следующим способом, называемым **метод инициализации Хавьера (иногда называют инициализация Глори)** [77]:

$$W_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}\right] \tag{23}$$

где $U[a, b]$ - равномерное распределение со значениями от a до b; w_i is the weights of i-th layer, n_i, n_{i+1} число входных параметров слоя с номером i (n_i) и число выходных параметров (n_{i+1}).

Инициализация Хавьера основана на схеме инициализации Лекуна (19). Однако, авторы [77] утверждали, что в оригинальной идеи Лекуна был проанализирован случай только прямого распространения ошибки. При получении выражения 23 было также учтено и обратное распространение ошибки. Авторами [77] утверждалось, что при инициализации весовых параметров вида 23 вероятность переобучения достаточно сильно снижается. При этом авторами было обнаружено, что нейронные сети с логистическими функциями активации могут выходить из области насыщения в ходе обучения. Эксперименты авторов показали, что НС, с инициализацией весов 23 имеет точность на тестовых данных выше, чем без инициализации, но ниже чем с пред обучением.

В 2015 году Каймингом Хе и соавторами из компании Microsoft был предложен метод инициализации функции ReLU, позволявший обходить результаты, получаемые при помощи инициализации Хавьера Глори [78]. В основе предложенного метода инициализации была положена идея учета несимметричного характера поведения функции ReLU. **Метод инициализации Хе (иногда называют инициализация Кайминга)**:

$$W_i \sim w_i \sim N\left[0, \frac{2}{n_i}\right] \tag{24}$$

В настоящее время существует несколько вариантов выражений для инициализацией Хе и Глори, а также некоторых других, рекомендуемых для различных вариантов функций активации [79]. Использование инициализации весов при отсутствии возможности или необходимости предобучения нейронных сетей позволяет повысить обобщающую способность нейронных сетей, ускоряет процесс обучения и стабилизирует результаты их обучения (при повторениях процедуры).

1.3.5 Техники регуляризации методом Дропаут(DropOut)

В 2012 году Джефри Хинтоном и соавторами, в том числе Алексом Крижевски, была опубликована работа, в которой авторы указали на одну из возможных интерпретаций переобучения и предложили решение данной проблемы - **т.н. метод дропаутов (dropping out, dropout)**[80]. Авторами замечено, что в переобученной нейронной сети имеет место **проблема содаоптации (co-apadtation** - это состояние обучения нейронной сети, когда нейроны каждого следующего слоя обучаются корректировать результаты работы нейронов предыдущего слоя. В Нормально обученной нейронной сети (не переобученной) предполагается, что каждый следующий слой нейронной сети работает независимо от предыдущего. То есть обучается выделять регулярные признаки из входных для него данных. В переобученной нейронной нейрон перестает выделять регулярности и начинает подстраивать ответ под конкретные экземпляры входных данных, включая все их случайные и регулярные особенности [80]. Другими словами нейронная сеть перестает работать как система с гибким статистическим выводом и превращается в аналог конечного автомата. В работе [80] было предположено, что решениями проблемы со-адаптации могу быть использование нескольких нейронных сетей, каждая из которых обучена на части входных данных или реализация такой структуры внутри одной сети. Последний вариант предполагает, что в каждой эпохе обучения нейронная сеть не учитывает ряд случайных признаков (выходов нейронов для каждого слоя) в результатах работы слоя - этот метод называется - метод Дропаутов. Число случайно выключаемых признаков как правило задается т.н. вероятностью дропаута p, который является гиперпараметром при обучении нейронной сети. Отметим, что метод дропаутов используется только при обучении нейронной сети. При ее работе все нейроны включены, а результат работы каждого нейрона должен быть домножен на $1 - p$. Иллюстрация работы метода дропаутов во время двух разных эпох обучения для $p = 0,33$ приведена на рисунке 23. Отметим, что использование метода дропуатов требует увеличения общего числа параметров нейронной сети. Также метод увеличивает дисперсию результатов, поэтому как правило рекомендуется снизить скорость обучения и использовать варианты метода обратного распространения ошибки с моментом. Также метод дропаутов увеличивает общее время обучения нейронной сети в силу описанных особенностей. Следует отметить, что метод дропаутов используется после активационных функций. Несмотря на свои недостатки метод дропаутов в своих различных вариантах реализации остается одним из наиболее популярных методов регуляризации в нейронных сетях [57, 81]. В 2013 году была опубликована работа [82], посвященная математическому обоснованию работоспособности метода дропаутов. В работе [82] было показано, что метод дропаутов увеличивает

обобщающую способность нейронной сети (регуляризирует обучение) и выполняет роль нормализации весовых параметров - то есть приводит их распределение к нормальному закону. Также в работе [83] было экспериментально показано, что метод дропаутов может быть использован совместно с L2 и некоторыми другими типами классической регуляризации.

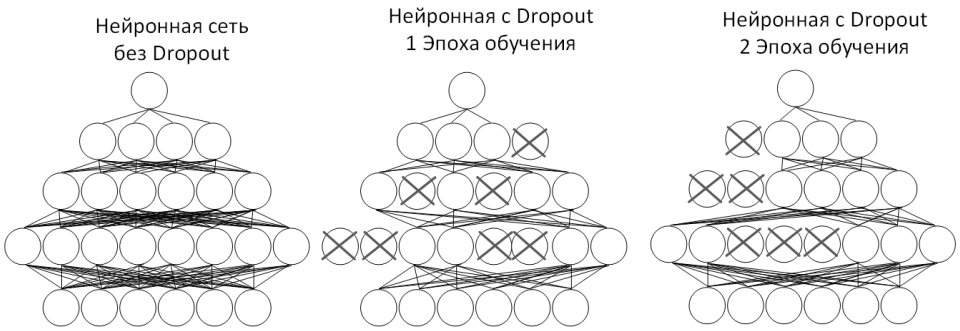

Рисунок 23 – Иллюстрация работы метода дропаутов во время двух разных эпох обучения для $p = 0,33$.

1.3.6 Графические ускорители как основной инструмент обучения нейронных сетей

Как уже было отмечено выше впервые преимущества использования графических ускорителей при обучении нейронных сетей были показаны в работах [70, 71, 72]. Так в работе [70] было заявлено 12-кратное ускорение обучения нейронной сети с использованием GPU по сравнению с обучением нейронной сети на центральном процессоре (ЦПУ) компьютера. Преимущества GPU заключаются в их возможности максимально быстро выполнять перемножения векторов и матриц - то есть выполнять операции умножения с накоплением над одномерными и двухмерными массивами числе с плавающей запятой с высокой степенью параллельности. Отметим, что операция умножения с накоплением для чисел с плавающей запятой часто называется flop (float-point operation) большинство линейных операций (например $W^T X = \sum wx$) сводятся к операциями типа flop.

Обучение на GPU в 2004-2006 годах (в работах [70, 71, 72]) представляло высокую сложность с точки зрения программного обеспечения. В силу технических ограничений существовавших на тот момент GPU разработка методов прямого прохождения через нейронную сеть и обратного распространения ошибки приходилось выполнять при помощи инструкций, предназначенных для расчета графических операций. В 2007 году компанией NVIDIA была представлен среда разработки под GPU CUDA (фреймворк CUDA), а также семейство графических ускорителей общего назначения (general purpose gpu, GPGPU). Использование CUDA и GPGPU позволило значительно упростить процесс разработки программного

обеспечения для ускорения обучения нейронных сетей. Фреймворк CUDA имеет Си-подобный синтаксис и позволяет вести достаточно гибкую и быструю разработку программного обеспечения для графических ускорителей компании NVIDIA. Тенденция использования CUDA и GPU NVIDIA для обучения нейронных сетей сохраняется по настоящее время. Однако, сегодня разработчику не нужно осваивать непосредственно работу с CUDA - так как есть достаточно большое количество высокоуровневых фреймворков позволяющих компилировать код программного обеспечения как для CUDA (под GPU NVIDIA), так и для центральных процессоров (CPU). Большинство современных высокоуровневых фреймворков обучения глубоких нейронных сетей, в том числе для задач компьютерного зрения разработаны для языка программирования Python [84].

В 2010 и 2011 годах Киерсаном и авторами работ [85, 86] были описаны реализации глубоких полносвязной [85], и сверточной [86] нейронных сетей, обученных на графических ускорителях (GPU) компании NVIDIA с использованием на тот момент, недавно представленного фреймворка разработки CUDA. Разработанные архитектуры нейронных сетей [85, 86] показали рекордные, на тот момент, точности для задач распознавания рукописных цифр (MNIST) и для набора цветных изображений 10 классов фотографий животных и техники (CIFAR-10 [87]). Важно отметить, что в работах [85, 86] нейронные сети обучались с использованием аугментации и без предобучения. Сверточная нейронной сеть основывалась на архитектуре LeNet 5, однако была значительно модифицирована, в частности, в ней слои усредняющей субдискретизации (average pooling) были заменены на слои max-pooling [86].

Результаты Киерсана стали отправной точкой в разработке современных фреймворков глубокого обучения с использованием GPU. Одним из первых успешных фреймвороков разработки глубоких нейронных сетей с использованием GPU NVIDIA стало Tehano, представленное для языка программирования Python в 2010 году [88]. Фрейемровки типа Tehano значительно ускорили процесс развития глубокого обучения нейронных сетей. Во-первых такие фреймворки позволяют конечному пользователю не задумываться о тонкостях реализации низкоуровневых операций в нейронных сетях (например, свертка, варианты градиентного спуска или метод обратного распространения ошибки) - эти операции уже реализованы в фреймворке и запускаются "из коробки". Во-вторых конечному пользователю не нужно заботиться об особенностях запуска своего кода для конкретных аппаратных конфигураций, как правило фреймворки имеют свои реализации рассчитанные как для любых GPU компании NVIDIA, так и для произвольных CPU. В современных фреймворках также допускается использование графических ускорителей компании AMD - но в качестве эксперимента. В-третьих, современные фреймворки предоставляют API (Application Programming Interface - интерфейс программного

приложения) для языка Python [89]. Данный язык программирования является одним из наиболее простых и высокоуровневых языков программирования, что позволяет освоить разработку нейронных сетей на Python практически любому, начиная со школьной скамьи.

Следует отметить, что набор CIFAR-10, а также его расширенная версия CIFAR-100 являются классическими наборами для тестирования и сравнения нейронных сетей (т.н. бентчмарков - benchmark) наборы разработаны Алексом Крижевски и представлены в работе Крижевски и Хинтона в 2010 году [90].

В 2011 году Киерсан и его коллектив соавторов показали наилучшие результаты в соревновании по распознаванию автомобильных знаков [91] при помощи вышеописанной архитектуры вида [86]. В том числе, авторы [91] показали превосходство своей нейронной сети по сравнению с экспертами, также прежде не видевшими тестовый набор данных. При этом точность работы нейронной сети составила 1,02%, эксперты смогли достичь точности в 1,19% [12].

В 2012году Киерсаном и его коллективом было показано, что ансамбль нейронных сетей позволяет достичь точности эквивалентной человеку для рукописного набора цифр MNIST (погрешность 0,2%) [92]. Отметим, что понятие **ансамбль нейронных сетей** означает, что используется структура из нескольких параллельно работающих нейронных сетей, результаты работы которых объединяются некоторым алгоритмом принятия конечного решения. Входные данные для такого алгоритма называются метаданными, а сам алгоритм мета-алгоритм. В качестве такого алгоритма может быть простое усреднение, выбор по большинству (голосование) или еще одна нейронная сеть - мета-сеть. Иллюстрация архитектуры ансамбля нейронных сетей приведена на рисунке 24. в работе [92] в качестве алгоритма использовалось простое усреднение, а каждая пара нейронных сетей имела свою пред обработку. Отметим, что ансамблевый подход к решению задач требующих презеционной точности характерен и в настоящее время. Как правило соревнования, подобные описанным выигрываются именно с использованием ансамблей нейронных сетей.

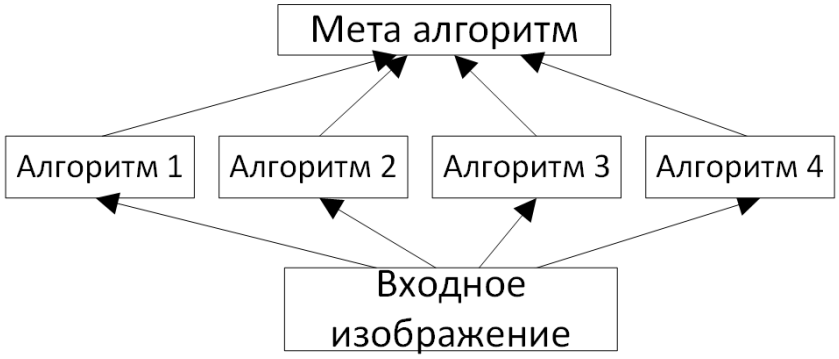

Рисунок 24 – Иллюстрация архитектуры ансамбля нейронных сетей

1.4 Экстенсивное развитие глубоких нейронных сетей 2012-2016 гг

1.4.1 ImageNet, GPU, Internet - катализаторы прогресса

В начале 21 века (в начале эпохи глубокого обучения) ключевыми тенденциями в глубоком обучении были: возможность использования обучения без учителя и самообучения нейронных сетей, а также ключевая проблема статистического оценивая – высокое качество обобщения на новые данные после обучения на небольшом количестве примеров [9]. Данные тенденции были смещены на второй план в 2010х. В это время в результате сильных прорывов в цифровизации технологий стали позволять проводить обучения НС на достаточно больших объемах данных. По крайне мере проводить предобучение на достаточно больших объемах данных. Этому способствовали как успехи в развитии GPU - вычислительной базы и развитии инструментов работы с GPU, так и успехи в развитии широкополосного доступа к сети интернет - то есть возможности сбор больших выборок данных для обучения. Тенденции 2010х привели к возможности углубления нейронных сетей и простоте работы с ними. Это в свою очередь привлекло внимание как научного так и инженерного сообществ и привело к гонке экстенсивного развития нейронных сетей.

Одним из наиболее важных катализаторов интереса к развитию нейронных сетей стали соревнования алгоритмов компьютерного зрения ImageNet Large Scale Visual Recognition Challenge (ImageNet LSVRC, ILSVRC), запущенные в 2010 году. Соревнования ILSVRC проводятся регулярно с 2010 года и основаны на тестовой части набора данных **ImageNet** [93, 94]. Набор данных ImageNet включал в 2012 году порядка 10 миллионов изображений в высоком разрешении, размеченных для задач классификации на 10 тысячи категорий. Для соревнований ILSVRC по классификации изображений использовалась тестовая выборки из набора ImageNet, включающие порядка 150000 изображений для 1000 категорий [95, 94]. Отметим, что ILSVRC проводятся по нескольким номинациям, которые от года к году могут меняться [94]. В 2012 году авторам работы [96] удалось достичь точности классификации изображений ILSVRC 84%, тогда как в 2011 году точность была 75% [94]. В 2020 для набора данных ImageNet достигается точность 1.2% [97]. Важно отметить, что описанные выше цифры точности соответствуют т.н. **"top-5 accuracy"** - это метод оценивая при котором результат классификации считается правильным если он соответствует любому из 5 выходов нейронной сети, которые имеют наиболее высокую вероятность. Оценка точности в классическом понимании (ответ с максимальным значением вероятности - правильный) соответствует термину "top-1 accuracy". В 2012 году точность по top-1 была 64%, в 2011 % 51 %, в 2020 91% [97].

Следует отметить, что ImageNet LSVRC являются наиболее престижными соревнованиями между алгоритмами компьютерного зрения по настоящее время. Хотя, набор ImageNet и не является уже наиболее крупным набором изображений [98].

1.4.2 Глубокие сверточные нейронные сети AlexNet и ZFNet

В 2012 году Алексом Крижевски и коллективом со-авторов был выигран ILSVRC 2010 в категории классификация изображений. [96]. Архитектура предложенной нейронной сети авторами работы [96] принято называть AlexNet (2012 год). Архитектура состоит из 7 скрытых слоев (плюс один выходной), 5 из которых сверточные и два полоносвязных скрытых слоя и один полносвязный выходной слой.Иллюстрация архитектуры AlexNet приведена на рисунке 25.

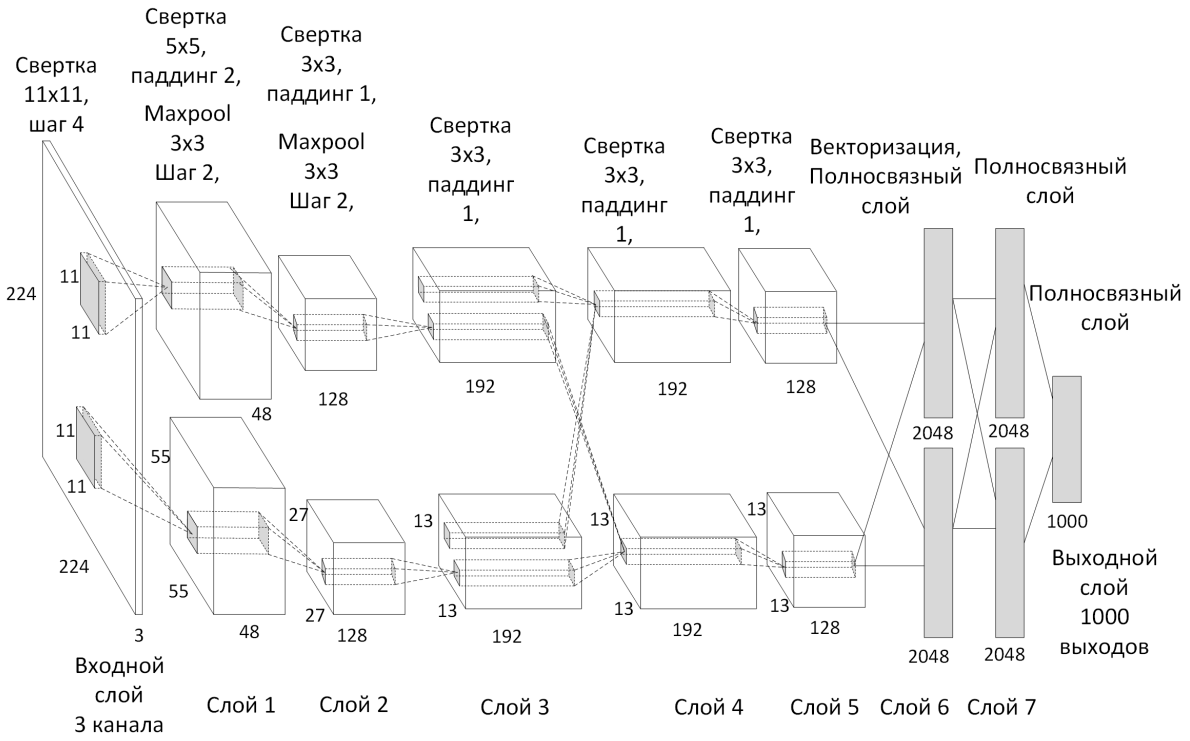

Рисунок 25 – Иллюстрация архитектуры глубокой сверточной нейронной сети AlexNet

Особенности архитектуры AlexNet, позволившие перейти к 7 слоям (от 5 в LeNet) были [96]:

- использование функций активации ReLU в скрытых слоях;

- использование аугментации данных, в том числе все изображения сжимались до размеров 256×256 из которых затем случайным образом выбирались патчи размером 224×224 - т.н. случайный кроп (random crop)- один из наиболее популярных видов аугментации по настоящее время; а также проводилось изменение интенсивности цветовых каналов входных изображений (входные изображения были в формате RGB - красный, зеленый и синий каналы).

- использование обучения стохастическим пакетным градиентным спуском с моментом и L2 регуляризацией;

- регуляризация методом дропаутов с вероятностью 50%;

- обучение на нескольких GPU при помощи параллельной обработкой групп каналов свертки - т.н. групповая свертка, когда все каналы C (см. рис 14) разделяются на группы и затем объединяются.

Все перечисленные выше техники будут более подробно рассмотрены в соответствующих разделах книги.Отметим, что по мимо выше приведенных операций инновационными в архитектуре стали: использование операции субдискретизации с перекрытием и использование локальной нормализация отклика фильтров (local response normalization, LNR) для предотвращения взрывного роста весов [96] - однако, данные типы операций не стали классикой глубоких архитектур нейронных сетей.

Архитектура AlexNet стала одним из наиболее значимых прорывов в области развития глубоких нейронных сетей. Архитектура AlexNet была первой одержавшей победу на соревнованиях ImageNet, более того, архитектура показала погрешность почти в 2 раза ниже, чем классические алгоритмы. Начиная с архитектуры AlexNet у научного и инженерного сообщества отпали сомнения в том, каким должен быть подход к решению задач компьютерного зрения. Начиная с 2012 года и по настоящее время подавляющее большинство задач компьютерного зрения решаются только методами глубокого обучения нейронных сетей [99]. Также работа [96] задала направления развития глубокого обучения сверточных нейронных сетей.

В 2013 победу на соревнованиях ILSVRC одержала модифицированная архитектура AlexNet - т.н. **ZFNet** [100]. Авторы работы [100] увеличили долю аугментации данных; провели предобучение на нескольких наборах данных, схожих с ImageNet; а также предложили метод визуализации работы слоев нейронной сети. Благодаря визуализации авторам работы [100] удалось выбрать гиперпараметры архитектуры улучшающие точность работы нейронной сети. Также метод LNR было предложено заменить на метод локальной нормализации контраста (local contrast normalization, LCR) [100].

1.4.3 Расширение концепции сверточного слоя в 2012-2014 годах

До 2014 года концепция сверточного слоя глубокой нейронной сети оставалась почти без изменений с работ Лекуна 1989 года. Хотя за этот период были измены функция активации, к слою были добавлены слой пулинг, были предложены некоторые методы регуляризации.

В 2014 авторами коллектива Visual Geometry Group (VGG) был предложен метод сокращения числа параметров свреточного слоя - **каскадная свертка**. ulineИдея каскадной свертки, заключается в замене одной свертки со сравнительно большим размером ядра (например 7×7) на последовательное соединение нескольких небольших сверточных ядер, описывающих тот же размер рецептивного поля (например 3 свертки 3×3) [101]. Это позволило авторами работы [101] предложить ряд значительно более глубоких архитектур сверточных нейронных сетей, имеющих от 11 до 19 слоев (VGG-11, VGG-13, VGG-16, VGG-19). Архитектуры VGG-19 и VGG-16 показали точности 92% и 91,4% соответственно по "top-5 accuracy" на ILSVRC-14. Архитектура VGG-16 используются по настоящее время в силу простоты. Однако, по соотношению числа параметров к точности она являются одной из наименее эффективных среди современных архитектур [97].

В 2013 Лин и соавторы опубликовали работу [102], в которой работе были предложены следующие приемы оптимизации сверточных нейронных сетей, ставшие впоследствии классическими.

- **Концепция расширения понятия сверточный слой до понятия "сеть в сети" (Network In Network, NIN)**. Лином предложена архитектура слоя как последовательного соединения свертки и полносвязной сети, входы которой канальные выход операции свертки (см. выражение 14). Основная идея "сеть в сети" заключается в увеличении числа нелинейностей (активационных функций) в каждом слое - соответственно увеличении возможностей для выделения нелинейных признаков каждым слоем нейронной сети. Сами авторы работы предлагали интерпретацию своей концепции как микросети, скользящей по входному изображению.

- **Использование глобального усредняющего пулинга (global average pooling, GAP)** вместо слоев векторизации набора полносвязных слоев в головной части нейронной сети. Основная идея глобального пулинга заключается в устранении набора полносвязных слоев, необходимых в случае векторизации, соответственно число параметров сети значительно снижается без существенных потерь в точности. Это делает обучение нейронных сетей более простым и снижает вероятность переобучения.

- **Использование точечной свертки** ($1 \times 1 \times C$ **- pointwise convolution) для выделения межканальных признаков**. Точечная свертка одна из наиболее часто используемых операций. Она позволяет сжимать или увеличивать число каналов в каждом сверточном слое - что позволяет варьировать число каналов и число выделяемых ими признаков.

Сам по себе подход Network In Network, предложенный авторами работы [102] не получил широкой популярности. Однако, идея использования "сети в сети" легла в основу архитектуры **GoogleInception V1 (GoogLeNet V1)**, предложенной компанией Google (коллектив авторов во главе с Кристианом Седжеди) в 2014 году [103].

Авторы работы [103](GoogLeNet V1) переработали концепцию NIN, расширили каждый слой до нескольких параллельных сверток и использовали для каждой свертки дополнительно свертку 1×1 для регулирования общего числа параметров сети. Иллюстрация типичной архитектуры слоя сети GoogLeNet V1 приведена на рисунке 26. Отметим, что в работе [103] были также предложены использование промежуточных слоев принятия решений и метод аугментации для предобучения нейронных сетей наложением разных экземпляров данных и их меток. Архитектура GoogLeNet V1 включала 22 слоя и показала точность 93,3 % на ILSVRC 2014. В последующие несколько лет Кристианом Седжеди с соавторами были предложены несколько модифицированных вариантов архитектуры Inception [104, 105]. Данное семейство архитектур будет рассмотрено в книге более подробно.

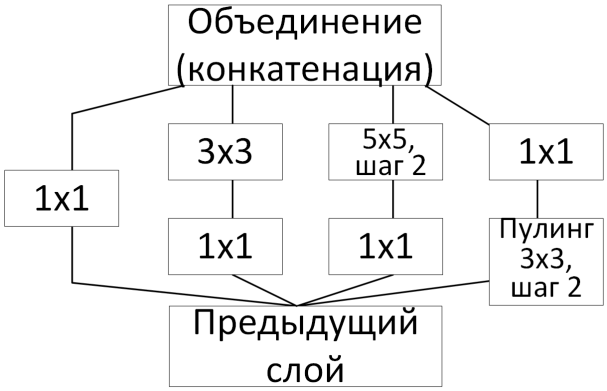

Рисунок 26 – Иллюстрация типичной архитектуры слоя сети GoogLeNet V1

Отметим, что расширение концепции сверточного слоя не ограничивается отмеченными выше работами. Скорее они являются лишь началом на этом пути [106]. В последующих разделах будет показано, как к данной концепции будут добавлены пакетная нормализация, тождественные связи, механизм внимания и другие приемы, позволяющие значительно повысить обобщающую способность

и точность нейронных сетей. Подробное исследование сверточных слоев будет проведено в последующих главах книги.

1.4.4 Пакетная нормализация (BatchNorm)

В 2015 году Сергеем Иоффе и Кристианом Седжеди была предложена **техника пакетной нормализации - батчнормализации (batch normalization, batch-norm, технически будет правильней назвать межпакетная нормализация)** [107].

Идея метода батчнормализации заключалась в следующем. Как правило обучение нейронной сети проводится методом стохастического пакетного градиентного спуска, то есть не по всей тренировочной выборке а по случайным подвыборкам - мини пакетам. Различия статистических характеристик между такими мини пакетами замедляет обучение нейронной сети и повышает требования к точности подбора гиперпараметров и требования к инициализации весовых параметров - то есть повышает вероятность переобучения и снижает обобщающую способность. Под статистическими характеристиками авторы понимают в первую очередь среднее и дисперсию по каждому признаку для каждой подвыборки (мини-пакета) входных данных. Описанный эффект авторы назвали **внутренний ковариационный сдвиг, internal covariate shift**. Для решения проблемы ковариационного сдвига в работе [107] было предложено использование нормализации результатов работы каждого слоя нейронной сети. При этом статистические характеристики должны усредняться по всем мини-пакетам и использоваться при работе обученной нейронной сети [107]. Математически батч-нормализацию можно выразить следующим образом:

$$\begin{aligned} \hat{y}_i = f(\tilde{z}_i); \quad \tilde{z}_i = \gamma \frac{z_i - \mu}{\sqrt{\epsilon + \sigma^2}} + \beta, \\ \mu_{\text{TEST}} = \alpha \mu_{\text{TEST}} + (1 - \alpha)\mu \\ \sigma_{\text{TEST}} = \alpha \sigma_{\text{TEST}} + (1 - \alpha)\sigma \end{aligned} \tag{25}$$

где

- $\hat{y}_i$ - результат работы нейрона;
- $f(\cdot)$ - функция активации;
- $\tilde{z}_i$ - нормализованный линейный выход;
- z_i - линейный выход (например $z_i = W^T X_i$);
- γ, β - коэффициенты масштабирования (обучаются во время тренировки нейронной сети);

- $\mu = \frac{1}{N}\sum_{i=0}^{N-1} z_i$ - среднее по минипакету,
  N - размер минипакета (используется при обучении);
- $\sigma = \frac{1}{N}\sqrt{\sum_{i=0}^{N-1}(z_i^2 - \mu)^2}$ - среднеквадратическое отклонение по минипакету (используется при обучении);
- ϵ - константа для предотвралщения деления на 0;
- $\mu_{\text{TEST}}, \sigma_{\text{TEST}}$ - значения среднего и среднеквадратического отклонения, которые будут использоваться вне обучения нейронной сети;
- α - коэффициент сглаживания экспоненциальным средним при обновлении $\mu_{\text{TEST}}, \sigma_{\text{TEST}}$.

Из выражения (25) и пояснений к нему могут быть сделаны следующие заключения касательно батчнормализации.

- Операция батчнормализации по разному работает при обучении и в основном режиме работы нейронных сетей.
- При обучении нейронных сетей в выражении (25) используются значения $\mu\sigma$, вычисленные для текущего минипакета.
- Значения коэффициентов γ и β обучаются во время тренировки и фиксируются в рабочем режиме нейронной сети.
- В рабочем режиме нейронной сети в выражении (25) используются значения μ_{TEST} и σ_{TEST}, вычисленные входе обучения.
- Операция батчнормализация должна быть использована перед активационной функцией.
- Операция батчнормализации сводится к тому, что каждый пакет приводится по каждому признаку: сначала к среднему 0 (μ) и дисперсии 1 (σ^2), а затем масштабируется (γ) и смещается (β). По задумке авторов смещение и масштаб должны быть таковыми, чтобы результат нормализации ($\tilde{z}_i$) всегда оказывался в "правильном" диапазоне значений для активационной функции. Например, для логистической функции желателен диапазон $\tilde{z}_i \sim 0,6 - 0,8$, для функции ReLU $\tilde{z}_i > 0$ и т.д. При этом так как параметры γ и β обучаются, сеть должна сама выбрать нужный диапазон значений $\tilde{z}_i$.
- Поскольку набор параметров $\mu, \sigma, \gamma, \beta$ одинаков для всех минибачей, следует ожидать одинакового поведения нейронной для каждого из них. Это снижает вероятность изменения ошибки работы нейронной сети в силу того, что установленные гиперпараметры больше подходят для одного минипакета и меньше подходят для дорого. Данное обстоятельство можно считать регуляризацией работы нейронной сети.

Благодаря описанным выше особенностям метод батч нормализации стал одним из наиболее популярных методов регуляризации обучения нейронных сетей. После выхода работы [107] многими авторами были предложены различные вариации данного метода, а также ряд альтернативных вариантов нормализации (например нормализация весовых параметров) [57, 108, 109]. **Недостатками классической версии (25)** являются: чувствительность метода к изменению размера пакета; снижение регуляризационных свойств при небольшом размере пакета. Для корректной работы батчнормализации рекомендуется использовать пакеты от $50 - 100$ экземпляров данных [110]. Использование метода батчнормализации будет более подробно рассмотрена в соответствующем разделе книги.

Отметим, что в оригинальной работа [107] были представлены лишь интуитивные пояснения касательно того почему метод батчнормализации работает. Позже рядом авторов были предприняты попытки формального математического обоснования метода. Однако, общего результата нет по настоящее время [111, 112]. Также следует отметить, что батчнормализация на практике редко используется совместно с методом дропаутов. Дискуссия касательно обобщающих свойств обоих методов и рекомендаций к их использованию также остается открытой [113, 110].

1.4.5 Слой остаточного обучения (residual layer)

В 2015 году Каймингом Хе и соавторами (компания Microsoft) было предложено использовать **принцип идентичных связей** для обучения нейронных сетей увеличенной глубины. По задумке автора идентичные связи должны быть параллельны основному слою нейронной сети [114]. Слой (или бок слоев) нейронной сети с идентичными связми принято называть **skip-connection, identity-connection, residual-connection или residual-layer, остаточный слой, тождественный слой**. Иллюстрации остаточного блока нейронной сети без остаточного слоя, аналогичного блока с остаточным слоем [114] и блока с модифицированным остаточным слоем [115] (2016 г., Каймингом Хе и соавторы) приведены на рисунках 27 А), Б) и В) соответственно.

Благодаря использованию остаточной связи (см. рис 27 Б) авторами работы [114] удалось увеличить глубину нейронных сетей до 152 слоев. Архитектуры сверточных нейронных сетей с остаточными связями, разработанных авторами [114] принято называть ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, где цифра обозначает число слоев. В соревнованиях ILSVRC 2015 модель ResNet-152 достигла погрешности 4,6% для одной сети и 3,6% для ансамбля сетей [114]. При этом значения ошибки ниже, чем для эксперта (у человека ошибка для тех же условий порядка 5 % [116]).

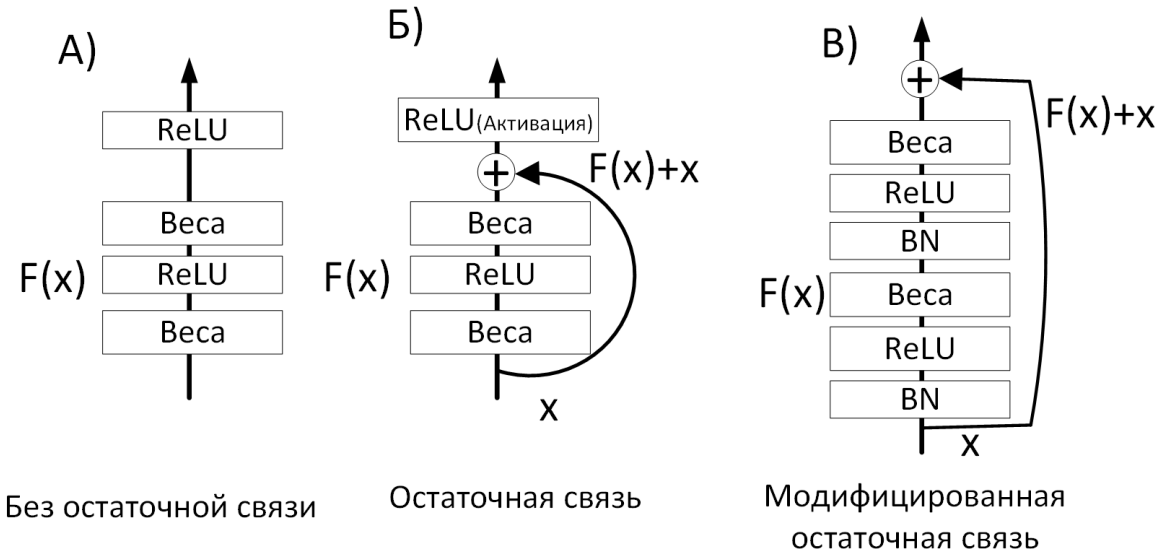

Рисунок 27 – Иллюстрация слоя с остаточной связью: А) блок слоев без остаточной связи; Б) блок с остаточной связью; В) блок с модифицированной остаточной связью

В основе работы остаточных связей лежат следующие идеи. Если во время тренировки нейронной сети в основном слое или блоке слоев возникает вымывание или взрыв градиента, то остаточная связь действует как регуляризация, компенсируя проблему. Компенсация происходит за счет суммирования информации (входных данных) перед блоком и результата работы самого блока. В случае, если блок не нужен нейронной сети (например слишком глубокий слой), то по задумке авторов сеть должна обучиться так, чтобы влияние остаточной связи преобладало в блоке (говоря на языке электронной схемотехнике - шунтировало блок). Таким образом остаточная связь является приемом регуляризации обучения нейронных сетей. Благодаря остаточным связям стало возможным увеличивать глубину нейронных сетей практически до бесконечности.

В 2016 году Каймингом Хе и соавторами была опубликована работа [115] в которой авторы смогли увеличить глубину сети ResNet с 152 до 1001 слоя, благодаря разработке модифицированной версия блока с остаточными связями (рис. 27 В). При этом оказалось, что точность на наборе данных CIFAR-10 увеличилась лишь с 5,46 % до 4,62 % [115]. Данный результат можно объяснить тем, что в результате обучения сети большинство слоев выше 100-200 полностью или почти полностью представляют собой остаточную связь - то есть не нужны. Также отметим, что в 2016 году в работе [117] была опубликована архитектура рекордно глубокой сети, имеющей 1202 слоя. Сеть был обучена используя прием дропаута слоев (т.н. стохастическая глубина), что, позволило несколько увеличить точность но оставило вопрос об эффективности такой архитектуры (в отношении точность/время/вычислительные

ресурсы) открытым.

Эксперименты проведенные Каймингом Хе и соавторами в работах [114, 115] показали сообществу исследователей нейронных сетей, что увеличение возможностей нейронных сетей за счет их углубления не является бесконечным и заставило задуматься о других путях совершенствования нейронных сетей. В 2016 году сообщество исследователей нейронных сетей начало работать на увеличением ширины слоя нейронных сетей при фиксированном числе слоев, а затем перешло к оптимизации каждого слоя. Так, в соревнованиях ILSVRC 2016 второе место с отрывом в 0,04% одержала архитектура ResNeXt - вариант архитектуры ResNet с оптимизированной структурой блока [118]. При этом первое место занял ансамбль из набора известных архитектур, не внесший существенного вклада в развитие методов глубокого обучения[119]. Подробно архитектуры на основе ResNet будут рассмотрены в соответствующем разделе книги.

Среди различных вариантов реализации идеи остаточного слоя в данном разделе хочется отметить работу 2017 года [120]. В данной работе было предложено расширить понятие блока с остаточными связями до вида, который принято называть **DenseNet**. Иллюстрация блока DenseNet приведена на рисунке 28. По существу, архитектура DenseNet может быть классифицирована как отдельный класс сверточных нейронных сетей. Идея блока DenseNet заключается в организации набора остаточных связей таким образом, что информация с каждого слоя блока DenseNet добавляется к результату последующего блока. При этом такое добавление информации происходит без соответствующего увеличения числа параметров, что позволяет обойти т.н. проблему проклятия размерности о которой говорилось выше. Таким образом, не смотря на большое число признаков в каждом из слоев, общее число параметров в блоке DenseNet остается небольшим, что дополнительно снижает вероятность переобучения нейронной сети и ускоряет ее обучение. Благодаря использованию блока DenseNet авторам работы [120] удалось создать архитектуру с числом слоев от 121 до 264. При этом данные архитектуры имели точности сопоставимые с классическими моделями ResNet с тем же числом слоев, однако при этом в архитектурах DenseNet число параметров было в 3-10 раза ниже [120].

Идея использование остаточной связи стала революционной в сверточных нейронных сетях, практически все последующие работы, предлагающие новые архитектуры сверточных и полносвязных нейронных сетей содержали остаточные связи в различных вариантах в рамках кодировщика признаков (feature encoder) [106].

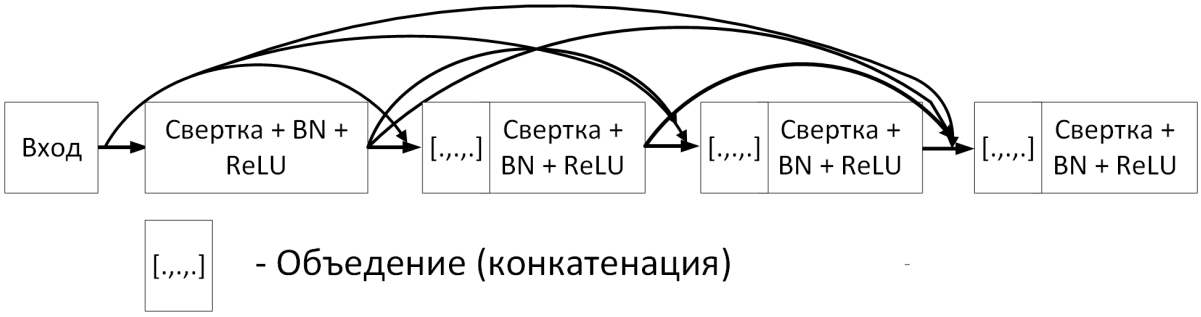

Рисунок 28 – Иллюстрация блока DenseNet с остаточными связями

1.5 Современное состояние глубоко обучения в задачах компьютерного зрения

1.5.1 Мобильные сверточные архитектуры 2016-2017

Как уже было отмечено выше с 1989 до 2016г основным направление развития сверточных нейронных сетей было поиск путей их углубления. В том числе, эта тенденция была связана с ростом возможностей серверной вычислительной техники которая подразумевалась в основных приложениях компьютерного зрения. Однако, к 2016 году успех глубокого обучения сверточных нейронных сетей все с большей интенсивностью открывали вопрос об их использовании в низко-производительных (в т.ч. мобильных) вычислительных устройствах. К этому моменту уже были предложены ряд техник по сжатию (компрессии) обученных нейронных сетей для их реализации в конечных устройствах [121]. Были предложены и специализированные архитектуры аппаратных ускорителей для нейронных сетей [122]. Однако, процесс компрессии и ускорения не позволял изначально обучить модель с требуемым размером. То есть для экспериментов с обучением нейронных сетей все равно требовались высокопроизводительные серные вычислительные устройства и высокие временные затраты.

В 2016 году для решения проблемы мобильных архитектур авторами работы [123] была предложена архитектура нейронной сети, обладающей производительностью на уровне AlexNet (см. выше) при числе параметров в 50 раз ниже (5 МБайт против 240 у AlexNet). Архитектура сети была названа **SqeezeNet**.Основные идеи SqeezeNet заключались в следующем.

- Замена части традиционных сверток на свертки 1x1 (точечные свертки) - что снижает число параметров, например в случае свертки 3×3 снижает в 9 раз.

- Использования слоя сокращения числа карт признаков перед сверткой (сжимающий слой, sqeeze layer). Предполагается, что не все карты признаков необходимы для конечного результата. В ходе обучения нейронная сеть должна научиться оставлять только не обходимые карты. Интуитивно данный принцип следует из предположения, что в правильно обученной нейронной сети каждый слой работает независимо от последующих слоев.

- Реализация субдискретизации (пулинга) ближе к оконечным слоям нейронной сети. По мнению авторов и результатам работы Кайменга Хе такой подход позволяет иметь карты признаков большего размера в начальных слоях, что должно способствовать увеличению обобщающей способности при классификации [124]. Такой подход принято называть **задержанная субдискретизация (delayed downsampling).**

- Основной блок нейронной сети sqeezenet - т.н. fire block, представляющий собой сначала слой сокращение числа карт признаков (squeeze), затем слой расширение (expand). Таким образом общее число карт признаков и соответственно параметров нейронной сети остается сравнительно не большим при реализации прочих идей AlexNet. Также следует отметить, что все свертки слоя расширения делились на два блока - точечные свертки и свертки 3×3, после проведения операции сворачивания и активации результаты объединялись (конкатенировали). Доля сверток каждого вида регулировалось при сохранении общего числа карт признаков.

Идеи, описанные в рамках реализации SqueezeNet положили стали отправной точкой в исследовании архитектур СНС для низко-производительных устройств [125, 41].

Одним из основных этапов развития мобильных нейронных сетей в период 2016-2018 гг. стали работы Андрю Ховрада и команды соавторов из компании Google по разработки семейства архитектур **MobileNet V1/V2**[126] (MobileNets V1 2017г) [127] (MobileNet V2 2018г). Основные идеи архитектур MobileNet V1/V2 заключаются в следующем.

- Замена блока классической свертки 3×3 на т.н. **глубокую разделяемую свертку (deep wise convolution)**. Такая свертка, по существу, является комбинацией предельного случая групповой свертки - когда число групп равно числу карт признаков и последующей точечной свертки регулирующей число карт признаков. Данная идея была предложено в архитектуре Xception [128] - вариации семейств архитектур ResNet и Google Inception. Главным достоинством такого подхода является сокращение общего числа параметров свертки. Позже эта свертка будет рассмотрна в соответствующей главе.

- Реализация субдискретизации (пулинга) путем использования сверток с увеличенным шагом. То есть сверточное ядро движется по входной карте признаков не с шагом в 1 пиксель (в одну позицию), а с увеличенным шагом. Отметим, что в работе [129] было показано, что такая замена не снижает точности по сравнению с макспулингом, однако снижает общее число вычислительных операций в нейронной сети.

- основной блок в архитектуре сети - MobileNet V1/V2 блок состоит из слоя расширения числа карт признаков (expantion); слоя глубокой разделяемой свертки и слоя сокращения числ карт признаков (слой проекции - projection или bottleneck layer). Также блок MobileNet V2 содержит остаточную связь. Основная идея блока MobileNet заключается в том, чтобы увеличить число карт признаков, провести над ними операцию фильтрации (глубокой разделяемой свертки) и затем оставить только полезные признаки (слой проекции).

- Для оптимизации архитектуры нейронной сети выбиралась общая структура сети - базовая сеть (baseline - определяет число слоев их взаимосвязи); при этом варьировались два параметра: параметр ширины (width multiplier) и параметр входного разрешения (resolution multiplier). **Параметр ширины** - это какое количество карт признаков будет использоваться в блоке (слой expantion), при этом число карт признаков после сокращения фиксировано (слой projection). **Параметр разрешение** - это то какой размер входного изображения ожидается - чем меньше размер, тем меньше вероятность выделения признаков небольших размеров, но в квадрат раз ниже вычислительная сложность.

Иллюстрация блока MobileNet V2 приведена на рисунке 29. Отметим, что в каждый сверточный слой блока содержал помимо свертки также батч-нормализацию и функцию активации ReLU6 - вариант функции ReLU с насыщением (будет рассмотрен позже). На рисунке символ α - параметр ширины; символ ρ - параметр разрешения.

1.5.2 Попытки использования идеи "слой внимания"

Идея "подсвечивания" для нейронной сети наиболее информативных участков входных данных (информативных с точки зрения задачи) присутствовала в сообществе исследователей нейронных сетей начиная с 1990-х [130]. В 2014 Бахданау (Bahdanau) предложил использование данной идеи в задачах машинного перевода, предложенная архитектура была названа **механизм внимания (attention)** [131]. Идея механизма внимания получила широкое развитие и распространение в различных приложениях обработки естественного языка и других схожих задачах [132]. В 2016-2017 годах в работах [133, 134, 135] была развита идея **механизма**

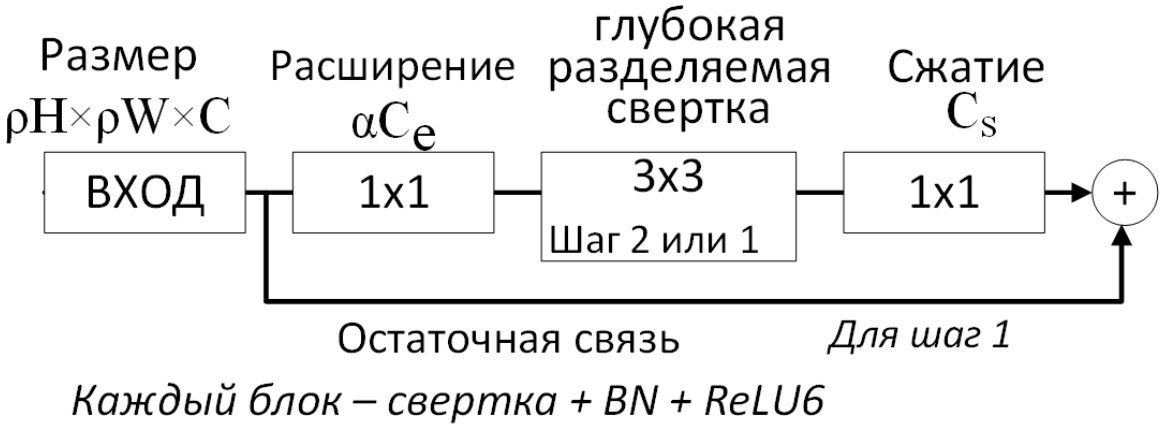

Рисунок 29 – Иллюстрация блока MobileNet V2

само-внимания (self-attention) для задач обработки естественного языка. Также в работе [135] были предложены т.н. **много-головое внимание (multi-head attention) и блок трансформер (transformer)**. Начиная с 2017 года идеи механизма внимания активно исследуются в приложениях к сверточным нейронным сетям в различных задачах компьютерного зрения [136, 137]. Отметим, что в противовес к само-вниманию, классический механизм внимания называют **кросс-внимание (cross-attention)**. Основной идеей механизма внимания является обучение слоя (блока) выделять наиболее важные участки входных данных при помощи нормализации функцией softmax (15) [136]. В случае само-внимания слой обучается только на основе входных данных. Если речь идет о кросс-внимание то слой обучается на входных и уже предсказанных выходных данных. Иллюстрация механизма само-внимания для сверточных сетей показана на рисунке 30.

Иллюстрация на рисунке 30 соответствует типу **меж-канальное само-внимания**, так как взвешивание функцией softmax проводится по каждой позиции пикселя, но для всех каналов. Отметим, что иллюстрация 30 это упрощенный пример авторов и носит только иллюстративный характер. Однако, архитектура такого слоя была парализована кем либо ранее. Некоторые практически используемые варианты слоя внимания будут рассмотрены в книге далее.

В работе [138] показано, что слой самовнимания позволяет провести перегруппировку пикселей входного слоя - то есть делает результат работ нейронной сети независимым от нерегулярных особенностей каждого экземпляра входных данных. Другими словами слой внимания дополняет одно из наиболее важных свойств сверки - инвариантность к положению объекта (как по координатам, так и по наклону, повороту и т.д.). В работе [138] 2019 года авторы привели математические доказательства, что много-голове само внимание эквивалентно свертке. Однако, отметили, что вероятно, предпочтительно использовать оба типа слоев вместе.

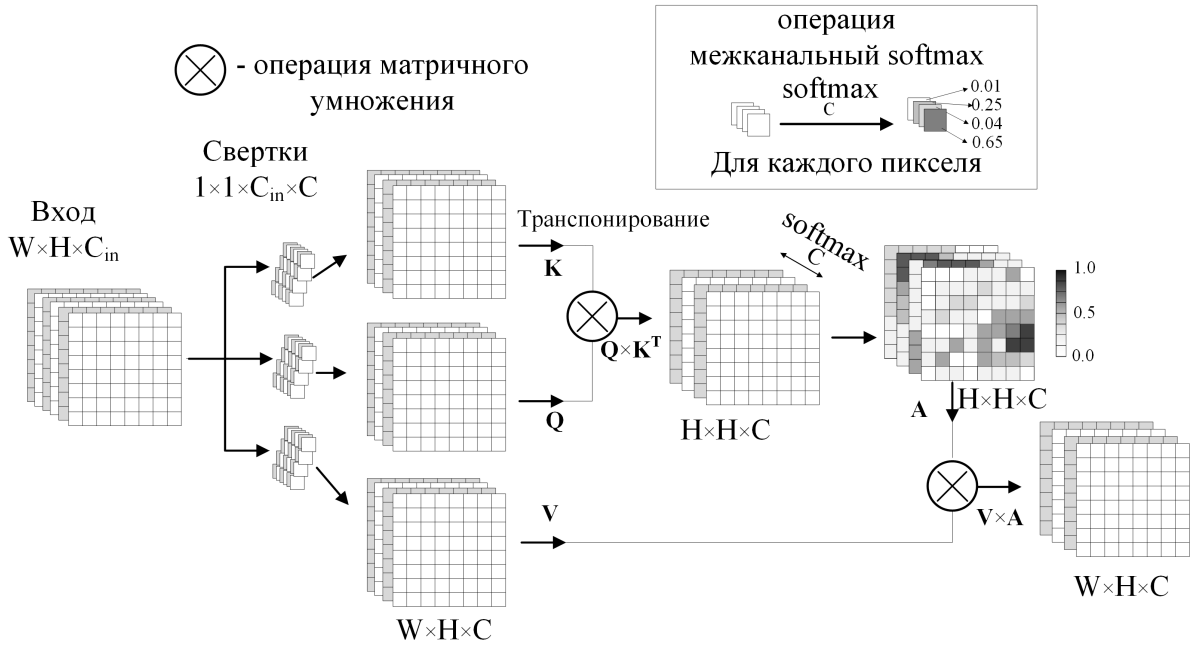

Рисунок 30 – Иллюстрация слоя само-внимания для сверточных сетей

В 2017 году в работе [139] концепция внимания для сверточных сетей был значительно переработана. В результате Jie Hu и коллективом соавторов был предложено **слой сжатия-возбуждения (Squeeze-and-Excitation layer, SE layer)**. Структура блока сжатия-возбуждения приведена на рисунке 31. В основе работы блока лежит идея выделения наиболее важных карт признаков (каналов) путем через автоэнкодер (верхняя часть рисунка) и затем через фикцию активации сигмоид (логистическая функция для каждого элемента). В результате такой операции формируются коэффициенты α_C, характеризующие "важность" каждой карты признаков. Таким образом, SE Layer призван помочь нейронной сети принимать во внимание только наиболее важные (регулярные, релевантные задаче) признаки. Отметим, что строго говоря слой SE Layer не является блоком внимания в классическом смысле так как использует логистическую функцию активации для каждого выхода вместо единой нормализации функцией softmax. Однако, часто слой SE Layer рассматривают в качестве варианта межканального внимания.

В 2017 году команда авторов работы [139] одержали победу в соревнования по классификации ILSVRC 2017 с ошибкой 2,25 (4,5)% по методике top-5-accuray [139]. Слой SE позже был использован во многих успешных архитектурах сверточных нейронных сетей. В частности, с использование слоя SE были синтезированы блоки MobleNet V3 в 2019 году [140].

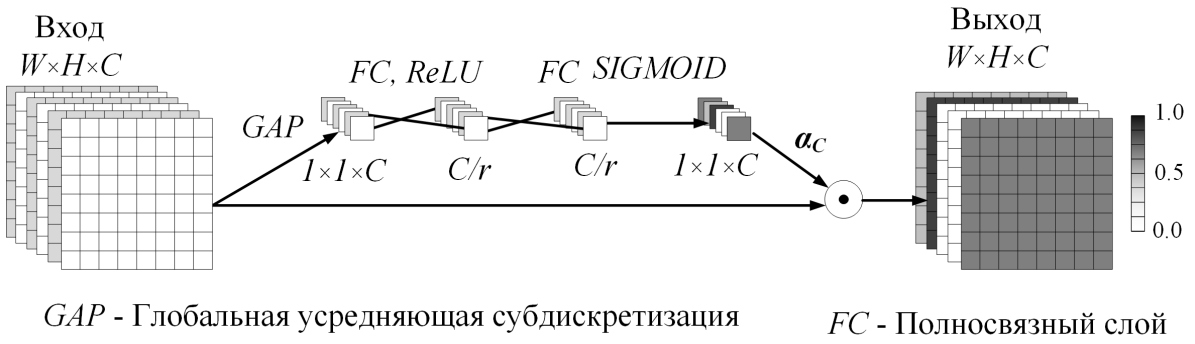

Рисунок 31 – Иллюстрация слоя Сжатия-возбуждения (SELayer)

1.5.3 Автоматический поиск архитектур

Идея автоматизации поиска архитектур нейронных сетей развивалась в сообществе исследователей с периода возникновения самой идеи глубокого обучения, по крайней мере с 2002 года [141]. Однако, существенных успехов удалось достичь только в 2017 году, когда был предложен метод, известный как **Neural Architecture Search (NAS)** [142]. Также отметим, что начиная с 2015 года проводятся соревнования по автоматическому синтезу алгоритмов машинного обучения, в том числе, но не только по автоматичесеому поиску архитектур глубокого обучения (AutoML Challenge). Также отметим, что термин AutoML - автомтаический поиск архитектур, включает, но не ограничен методами NAS [143].

По существу, **методы NAS подразумевают три составляющие.**

- **Пространство поиска (Search Space)**. Набор блоков, слоев или архитектур которые могут быть использованы. Выбор таких архитектур исходит из опыта исследователя. Отметим, что данный факт является наиболее фундаментальным ограничением методов NAS.

  В современных системах общая структура блоков и их максимальное число часто фиксированы, но параметры блоков (наличие пулинга, число карт признаков, размер входного массива, число остаточных связей и т.д.) могут меняться.

- **Стратегия поиска (Search Strategy)**. Набор правил для выбора тех или иных блоков. От выбора стратегии поиска может зависть число возможны комбинаций в пространстве поиска, как правило число комбинаций должно быть максимально ограничено.

  В первых работах по NAS рассматривались такие стратегии как [142]:

- генетические эволюционные алгоритмы (начало 2000-ных);
- методы Байесовской оптимизации (2010-е);
- обучение с подкреплением (2017);
- рекуррентное обучение с кодированием всех вариантов в категориальные вектора (2018).

- **Оценка качества модели (Performance Estimation Strategy)**. Данный вопрос также является не тривиальным. Так как к архитектуре могут быть предъявлены совместно требования высокой точности, оптимальной производительности для конкретной конфигурации оборудования или времени работы, требования к занимаемому месту в памяти и другие требования.
  Также к данному пункту относится выбор метода снижения вычислительной нагрузке при подборе архитектуры. К таким методом можно отнести следующие [142]:

  - сокращение тренировочной выборки (Lower fidelity estimates);
  - испытания всех моделей с пониженным числом параметров (например можно снизить число карт в 2 для каждой исследуемой модели) (Lower fidelity estimates);
  - ранняя остановка обучения моделей с экстраполяцией результатов (Learning Curve Extrapolation);
  - инициализация весовых параметров новых вариантов архитектур весовыми параметрами предыдущих вариантов (сокращает число эпох тренировки сети) (Network Morphisms);
  - обучение единой модели, включающей все варианты и использование комбинаций ее составляющих в качестве новых предобученных моделей (One-Shot Architecture Search).

  Отметим, что все описанные подходы к упрощения процесса выбора архитектуры приводят к смещению конечного результата, однако могут быть использованы для ранжирования вариантов. После выбора итоговой архитектуры - она должна быть дообучена [142].

В 2018 были опубликованы результаты оказавшие наибольшее влияние на развитие NAS: Зопф Б. и соавторы (команда Google Brain) предложили архитектуру **NASNet** [144], а Лью Ч. в соавторстве с коллективом, включающим Зопф Б. предолжили **метод прогрессивного поиска архитектур (Progressive NAS, PNAS)**, а также архитектура **PNASNet** [145].

Для синтеза NasNet использовался рекуррентный подход, предложенный Зопфом в работе [146] и метод регуляризации путем выключения частей блоков при обучении каждого варианта архитектуры. В работе [145] авторы предложили стратегию последовательно усложнения структуры блоков с прореживанием пространства поиска (PNAS). Прореживание выполнялось предварительной оценкой точности для каждого из кандидатов-блоков. При этом общая структура сети фиксировалась перед проведением экспериментов.

Архитектуры PNASNet и NASNet показали практически эквивалентные результаты для набора данных ImageNet, однако метод PNASNet потребовал в 8 раз меньше вычислительных ресурсов и при этом архитектура PNASNet требовала несколько меньшего объема памяти. Архитектура PNASNet стала победителем на ILSVRC 2018 и достигла точности 96,2 % (на 1,2% выше чем вручную оптимизированные архитектуры)[145].

В 2019-2020 годах исследователи из Google дополнили метод NAS и предложили семейства архитектур **EfficientNet V1** (2019) [147] и **EfficientNet V2** (2020) [148]. В основе предложенного подхода лежала идея поиска оптимальной базовой архитектуры методом PNAS (baseline) и ее масштабирование по трем параметром - т.н. compound scaling method. Общая структура сети определялась методом PNAS, масштабированию подлежали входное разрешение, число слоев каждого блока сети (глубина) и число карт признаков в каждом блоке (ширина). При этом масштаб изменялся с одинаковым коэффициентом для всех трех параметров [147]. Семейства архитектур EfficientNet V1/V2 стали одними из основных архитектур для решения практических задач компьютерного зрения в 2020-2021 годах. Помимо официальных версий EfficientNet в литературе предложены варианты архитектур для специальных задач, например для использования в мобильных телефонах [125].

1.5.4 Архитектуры типа трансформер в задачах компьютерного зрения

Как уже было сказано ранее в 2017 году различными исследователями было предложено использование идеи слоев внимания в задачах компьютерного зрения. Также в 2017 году А. Васвани был предложен блок, названный трансформер (transformer) для задач обработки естественного языка [135]. В составе данного блока использовались только слои внимания и полносвязные слои. В 2018 году было предложено использовать блока трансформер в задачах генерации изображений и обработки видео. В 2020 году были предложены первые успешные архитектуры для задачи классификации изображений [149]. Многими авторами предлагалось замена части слоев сверточных сетей на блоки трансформеры (полносвязные блоки с вниманием (такие же как Васвани)) [150]. В 2020 году была предложена архитектура ViT, в которой авторы использовали только блоки трансформеры и смогли достичь точности на наборе ImageNet сопоставимой со сверточными сетями. По существу,

архитектура ViT являлась переосмыслением архитектуры BERT для обработки естественных языков (2018 г. [151]). Иллюстрация архитектуры ViT показана на рисунке 32.

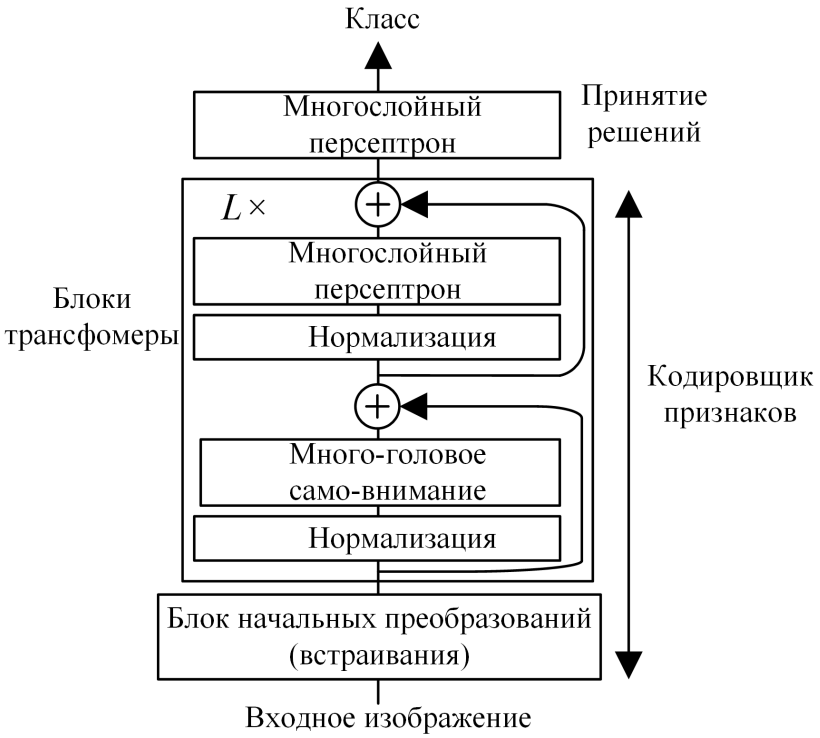

Рисунок 32 – Иллюстрация архитектуры ViT

Архитектура ViT состоит из блока начальных преобразований; набора слоев трансформеров и блок принятия решений на основе многослойного персептрона. Блоки трансформеры и блок начальных преобразований образуют кодировщик признаков. Каждый блок трансформер состоит из слоев нормализации, слоя много-голового само-внимания и многоголового персептрона. Таким обзорам архитектура ViT не имеет сверточных слоев. Более подробная иллюстрация ViT приведена на рисунке 33. Иллюстрация показана для архитектуры ViT-L с 16 каналами многоголового само-внимания. Архитектура ViT стала первой доказавшей, что решение задач компьютерного зрения возможно по средствам полносвязных нейронных сетей. Данная модель показала точность 88,5% по методу top-1 на наборе данных ImageNet. Однако, для достижения высокой точности модель ViT была предобучена на наборе данных JFT-300M (набор с 300 млн. изображений [152]) [153]. Отмечается, что в случае предобучения на небольших наборах данных (менее 14 млн изображений) точность модели ViT значительно снижается (примерно до 75%). Также число параметров ViT в 10 раз выше чем для сверточной сети (600 тыс. параметров). Архитектура ViT стала одним из основных этапов в принятии трансформеров сообществом исследователей систем компьютерного зрения

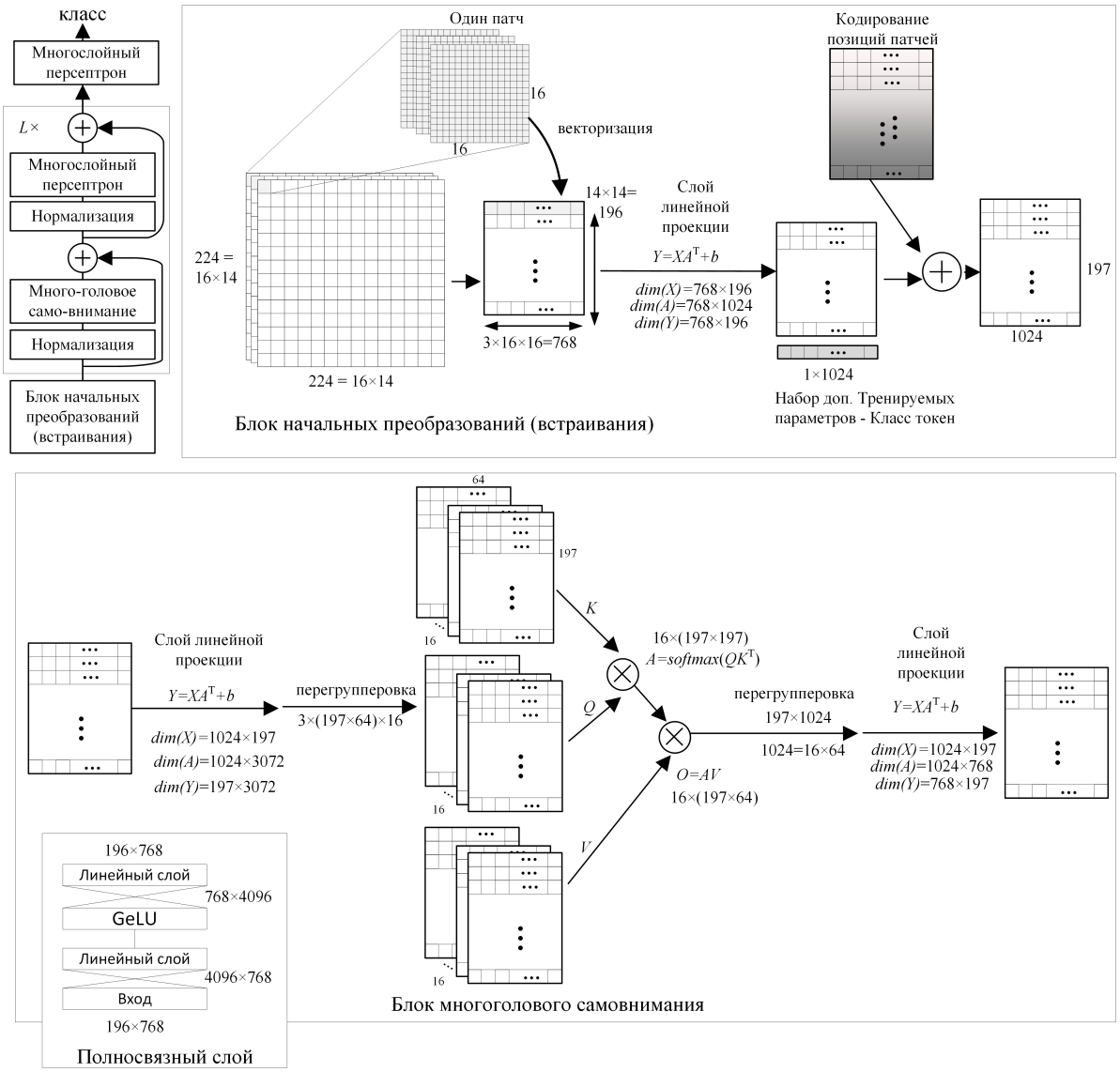

Рисунок 33 – Подробная иллюстрация архитектуры ViT-L

[154]. Позже в работе [155] была предложена архитектура DeiT, в которой число параметров было снижена до 80 тыс. и при этом модель показала точность 83,5% по методу top-1 на наборе данных ImageNet с использованием только данных ImageNet (без предобучения на JFT-300M). В конце июня 2021 года была также представлена архитектура Vision Outlooker (VOLO) показавшая точность 85,5% в условиях аналогичных DeiT, однако имеющая больше параметров [156]. Также июне была представлена модель ViT-G, показавшая точность 90,5% по методу top-1 на наборе данных ImageNet и имеющая порядка 2-х миллиардов параметров [157].

Главными достоинствами архитектур трансформеров являются следующие [154, 149].

- Возможность построения сложных зависимостей между признаками локальными участков изображения независимо от их взаимного расположения.
- Трансформеры позволят обучать модели с значительно увеличенным числом параметров. Что также дает преимущества при предобучении на больших наборах данных, таких как JFT-300M. В предыдущие эпохи (до 2015 гг) данное обстоятельство скорее было недостатком. В настоящее время в силу развития вычислительной техники и возможностей сети Интернет имеется возможность сравнительно быстрой работы с большими объемами данных. При этом путь повышения точности и обобщающей способности за счет предобучения является предпочтительным.
- Возможность для некоторых архитектур предобучения на не размеченных данных. Ряд архитектур трансформеров имеют возможность предобучаться подобно жадному предобучению о котором говорилось выше. Такая возможность особенно важна в силу сказанного в предыдущем пункте. Также отмечается [158, 159], что предобучения баз разметки потенциально способствует повышению выразительной силы (сложности признаков) и обобщающей способности нейронной сети. Метод предобучения на не размеченных данных принято называть **self-supervised learning (самообучение).**

1.6 Обзор других задач компьютерного зрения

В данном разделе преимущественно говорилось о задаче классификации изображений. Данная задача в определенном смысле была первой задачей компьютерного зрения, в которой появилась крайняя необходимость в использовании методов машинного обучения и в которой глубокие нейронные сети показали преимущества перед остальными методами. Как уже сообщалось в книге в первую очередь это было связано с автоматизацией выделения признаков изображений (в отличии от остальных методов, где требовалась формализация признаков и их описание "вручную"). Однако, задачи компьютерного зрения не ограничиваются задачей классификации изображений. В данном разделе кратко будут названы и описаны другие типичные задачи компьютерного зрения. Отметим, что архитектуры нейронных сетей для задачи классификации, которые описывалась ранее являются основой для решения других задач компьютерного зрения. Правильней даже будет сказать так, что кодировщики признаков (см. энкодер рисунок 16), являющиеся основной частью архитектур для классификации, используются при решении других задач компьютерного зрения.

Среди основных задач компьютерного зрения могут быть выделены следующие задачи, которые как правило справедливы как для отдельных изображений, так и для видео-сцен [160].

1. Классификация (рассмотрена выше).
2. Обнаружение и локализации объектов (Object Detection, Object Localization).
3. Сегментации классов объектов и сцен (Semantic Segmentation).
4. Объектная сегментация (Instance Segmentation).
5. Сегментации объектов и сцен (Panoptic Segmentation).
6. Поиск и выделения ключевых точек (Key-Point Detection).
7. Генерации изображений (Image Generation).
8. повышение качества изображений и видео-сцен (Super-Resolution).

Среди описанных выше задач в книге будут рассмотрены 1-3 и 7, а также сопутствующие им задачи.

1.6.1 Кратко о задаче семантической сегментации

Задача семантической сегментации известна в приложениях компьютерного зрения достаточно давно [161, 162]. Однако, первые успешные попытки ее решения при помощи нейронных сетей были опубликованы в 2012 году в работе C.Farabet и соавторов [163]. В частности, в данной работе был предложен пиксельной классификации - то есть присвоение класса каждому пикселю входного изображения.

То есть результатом работы нейронной сети должен стать набор карт признаков - по одной для каждого класса. При этом каждая карта признаков должна выделить всю площадь (или контур) объектов данного класса. Для повышения надежности сегментации объектов разных масштабов в работе предлагалось использовать несколько сверточных энкодеров признаков и затем объединить полученные карты. Однако, архитектура, предложенная в [163] не получила широкого распространения. Таким образом, в рамках этой и последующих архитектур глубокого обучения нейронных сетей задача семантической классификации может быть рассмотрена как расширение идеи классификации, но классификации по каждому пикселю выходной карты признаков.

В 2014 году в работе J. Long и соавторов [164] была предложена **полностью сверточная архитектура (fully convolutional network (FCN))** сети семантической сегментации. В данной архитектуре авторы модифицировали архитектуру AlexNet (см. рисунок 25). Было предложено заменить последние слои (головную часть, см. рисунок 16) на сверточные слои и после них поставить слой увеличения размерности при помощи передескретизации (upsampling, unpooling) **методом билинейной интерполяции (bilinear interpolation)**. Суть данного метода будут рассмотрена позже [164]. В 2015 году в работе [165] была предложена архитектура архитектура сверточной нейронной сети с использованием слоев т.н. обратная (транспонированная) свертка (deconvolution, transposed convolution) и повышающей передискретизации. Архитектуру было предложено назвать **DeconvNet**. Иллюстрация архитектуры DeconvNet приведена на рисунке 34. В данной архитектуре для выполнения операции unpooling выполнялась путем сохранения индексов пискселей, прошедших процедуру прямую дискретизации в соответствующих слоях. Позже в 2015 году архитектура была оптимизирована и опубликована в работе [166] под названием **SegNet**. По существу, архитектура DeconvNet состоит из двух частей: кодировщик признаков (энкодер) и декодировщик признаков (декодер), а также слоя пописксельной классификации. Цель использования энкодера заключается в выделение релевантных признаков во входных данных. Цель использования декодера заключается в восстановлении из выделенных признаков т.н. маски целевого класса (силуэт или контур соответствующего объекта или сцены). Число выходных карт признаков (выходных каналов) сети должно соответствовать числу классов на которые должно быть поделено (сегментировано) изображение. Решение о том, к кому классу принимается путем использования операции softmax между всеми каналами по каждому пикселю. Таким образом, каждый выходной канал содержит силуэт соответствующий одному целевому классу.

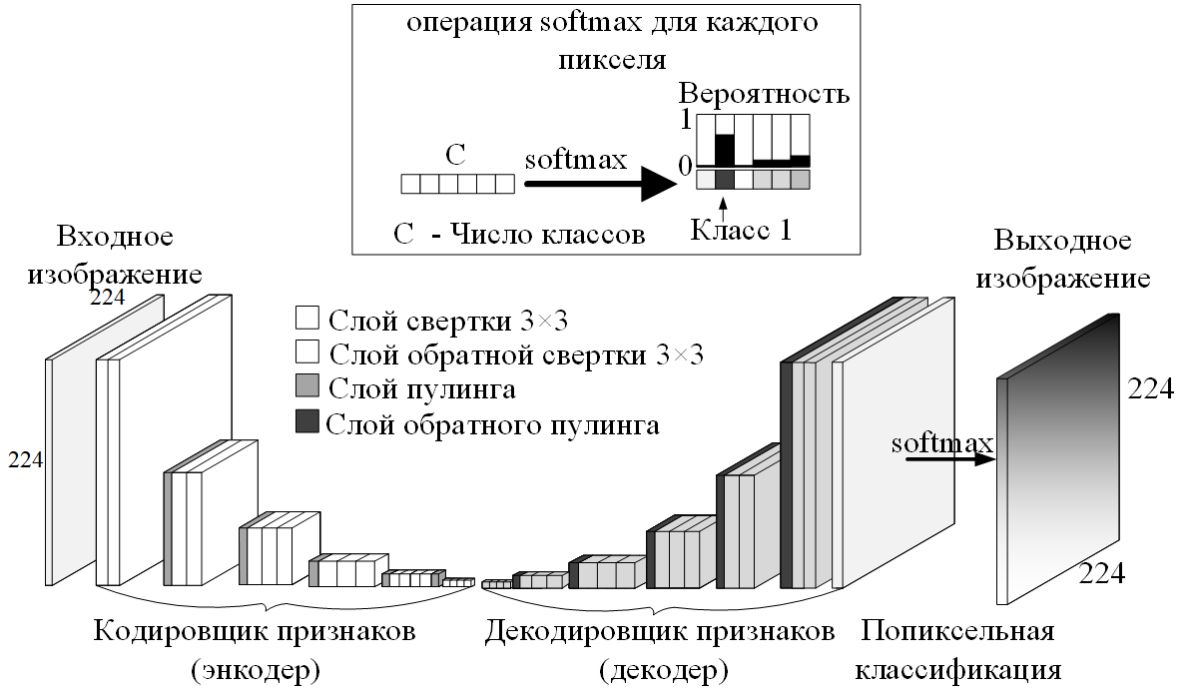

Рисунок 34 – Иллюстрация архитектуры DeconvNet

В 2015 году О. Роненбергом и соавторами была предложена архитектура U-Net [167]. Особенностью данной архитектуры было использование в декодирующей части архитектуры совмещенных (конкатенация) карт признаков энкодера и декодера. Такой подход позволил существенно увеличить точность особенно в случае малоразмерных деталей изображения. Изначально архитектура U-Net была предложена для решения медицинских задач [168]. В настоящее время архитектура является одной из наиболее популярных среди любых задач, сводящихся к семантической сегментации (по данным портала paperswithcode.com) [169]. Однако, U-Net и не является наиболее точной архитектурой, например для классического набора данных PASCAL VOC 2012 [170] по данным портала paperswithcode.com [171]. Иллюстрация архитектуры U-Net приведена на рисунке 35.

После 2015 года различными авторами были предложены большое количество архитектур на основе U-Net и DeconvNet [172]. В том числе, наиболее успешные архитектуры, которые, однако переосмыслены с использованеием расширенных понятий свертки (речь о котором пойдет в свое время) и с использованием метода NAS. Отметим, что, в описанных архитектурах семантической сегментации предполагается подход обучения с учителем. Однако, в литературе также обсуждаются архитектуры семантической сегментации с т.н. слабым обучением (weak supervised) - то есть с использованием только классов объектов на изображении или частичной разметки и меток классов [173].

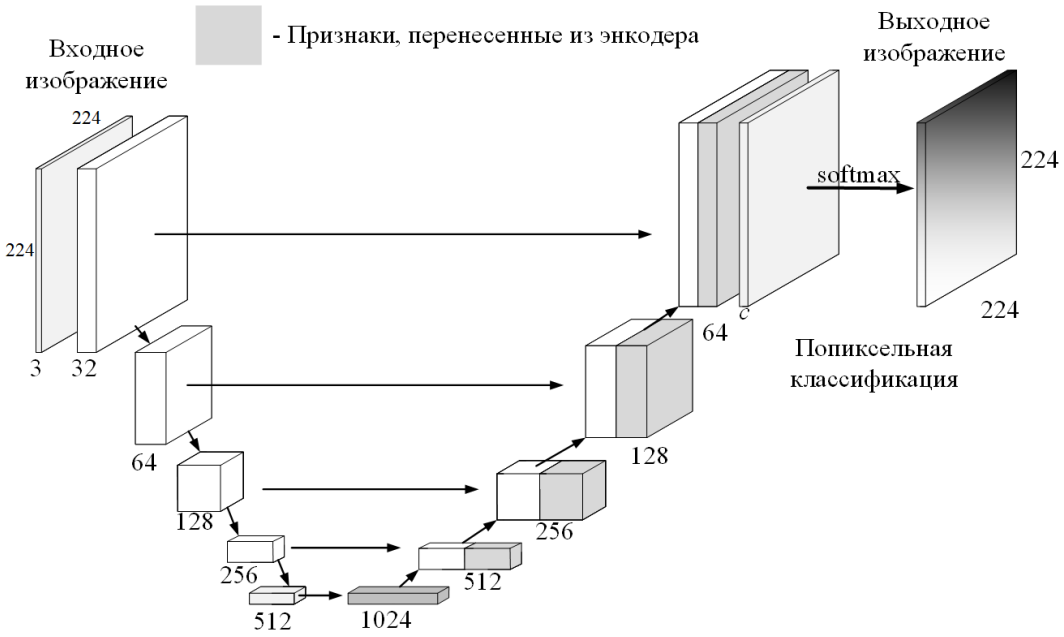

Рисунок 35 – Иллюстрация архитектуры U-Net

1.6.2 Кратко о задачах Object Detection и Instance Segmentaiton

Идеи использования нейронных сетей для обнаружения и локализации объектов возникают по крайней мере начиная с 1990-х годов [174, 175]. Одной из первых удачных попыток создания полностью сверточной нейронной сети для обнаружения объектов можно считать архитектуру Overfeat (2013 год) [176]. Наиболее важной работой в области обнаружения и локализации объектов является работа 2014 года [177] Girshick R. и соавторов из университета Беркли.Авторы [177] предложили архитектуру **регионная сверточная нейронная сеть (Regions with CNN features, R-CNN)**. Данная архитектура объединила в себе наиболее продуктивные на тот момент практики в области обнаружения и локализации объектов.

Иллюстрация принципа работы архитектуры R-CNN приведена на рисунке 37. Сеть R-CNN осуществляет предсказание области нахождения объекта и его класса в два этапа - поэтому такой подход называется **много-этапное обнаружение (multi-stage object detection)** [177].

Архитектура R-CNN работает следующим образом [177].

- 0 Этап - подготовка перед началом обучения.
  - Производится выбор 2000 регионов изображения, в которых предположительно есть объекты (**Regeons of Interest, ROI**). Авторы отмечают, что поиск может быть выполнен любым алгоритмом.

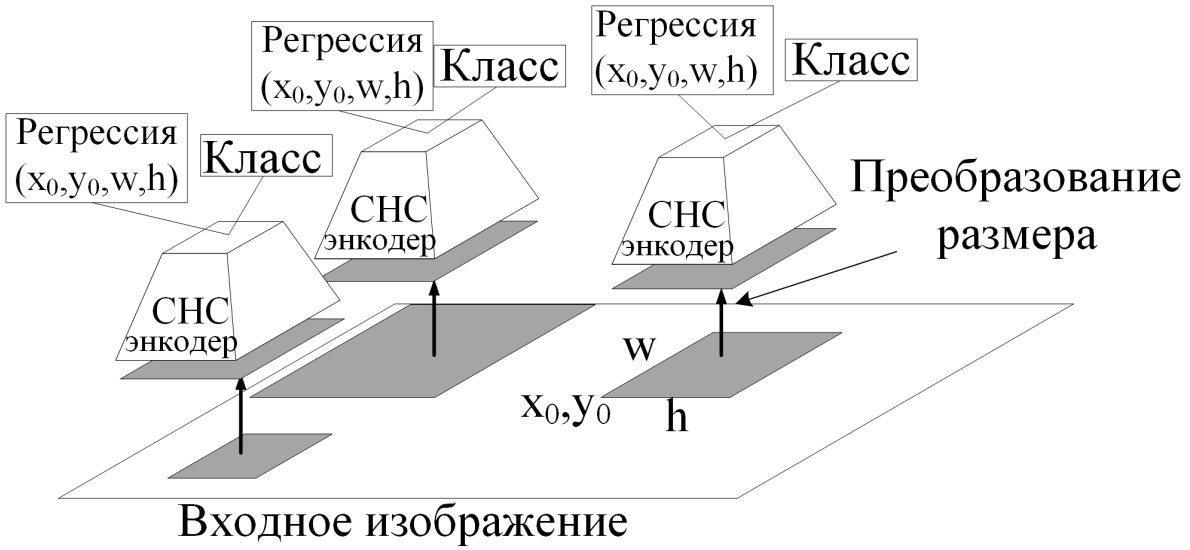

Рисунок 36 – Иллюстрация принципа работы архитектуры R-CNN

- В оригинальной статье выбор регионов проводится методом селективный поиск (selective search). Это метод в котором выделяются контуры на изображении и проводится объектов в рамках контуров путем оценки схожести соответствующих участков изображения [178].
- Все выделенные регионы могут иметь разный размер. Выделенные регионы могут как содержать целевой объект, так и нет.
- Отметим, что данный этап используется только как подготовка изображения к обработке нейронной сетью, то есть этап не включен в работу нейронной сети.

- 1 Этап - Автоматическое выделение признаков.
  - Производится сжатие всех выделенных регионов до одного размера.
  - Производится автоматическое выделение признаков (вектор 4096 значений). Выделение признаков производится при помощи сверточной нейронной сети AlexNet, в которой удален слой классификации - то есть при помощи кодировщика (энкодера) признаков AlexNet (см. 25).

- 2 Этап - классификация регионов.
  - Много-меточная (multilabel) классификация всех регионов ROI. Авторы используют для этого метод опорных векторов (SVM) отдельно для каждого класса.
  - Удаление лишних регионов методом **не максимальное сжатие (non-maximum supression, NMS)**. В наиболее простом виде метод NMS заключается в следующем:

    * для каждого класса производится поиск регионов с относительной площадью пересечения (**Intersectino over Uninon, IoU**) больше заданного порога;
    * среди найденных регионов остается регион с наибольшей вероятность классификации (с наибольшим результатом выхода классифицирующего слоя).

- 3 Этап - предсказание области новых регионов.
  - область нахождения каждого объекта описывается **граничным прямоугольником (bounding box)**. Такой прямоугольник задается начальными координатами x_0, y_0, а также шириной и длинной (w, h).
  - Для предсказания новых значений x_0, y_0, w, h для каждого ROI используется регрессионный оконечный слой с 4 выходами (по одному на каждый параметр). Как к и в случае классификации слой используется поверх вектора признаков, полученного на 1-м этапе.

- В режиме тренировки нейронной сети для новых значений x_0, y_0, w, h производится следующая эпоха обучение нейронной сети (начиная с этапа 1).
- В режиме работы нейронной сети выходами являются объекты, прошедшие процедуру NMS, для которых присвоена метка класса и координаты bounding box: x_0, y_0, w, h.

Отметим, что в оригинальной статье [177] кодировщик признаков был предобучен на наборе данных ImageNet. Сеть была протестирована на наборе данных для решения задач обнаружения объектов PASCAL VOC [170]. Также отметим, что в литературе рассматриваются и исследуются различные модификации таких операций, как NMS и других, использованных в головной части R-CNN [179].

В 2015 Girshick R. и соавторы предложили усовершенствованный вариант RCNN, называемый **Fast-RCNN** [180]. Главным отличием новой архитектуры стал перенос операции от предварительного селективного поиска с этапа подготовки изображения на этап после энкодера. Таким образом авторам удалось увеличить скорость работы сети в порядка 10 раз. Отметим, что, по существу, селективный поиск по карте признаков представляет собой определенный пулинг называемые ROI Pooling, идея такого пулинга была вдохновлена работай Кайменга Хе посвященной сети с пространственным пирамидальным пулингом (SppNet) [181]. Позже в 2015 году в работе [182] коллективом авторов, включающим как Кайменга Хе, так и Girshick R. было предложено заменить селективный поиск на небольшую дополнительную нейронную сеть прогнозирующую регионы ROI (Regeon Proporsal Network, RPN). Подход был назвал **Faster-R-CNN**. В 2017 году Кайменгом Хе и соавторами, в том числе Girshick R. было предложено улучшение архитектуры Faster-R-CNN, в

том числе дополнительно к операциям классификации и определения координат объекта было предложено добавить небольшую сверточную сеть семантической сегментации результатов ROI. Архитектуру было предложено назвать **Mask-R-CNN** [183]. Архитектура Mask-R-CNN позволяет повысить точность решения задачи определения и локализации объектов, а также позволяет решить задачу объектной сегментации (Instance segmentaion).

Архитектуры Faster-R-CNN и Mask-R-CNN одни из наиболее популярных для решения задач обнаружения и локализации объектов [179]. А также Mask-R-CNN остается одной из наиболее популярных архитектур для решения задачи объектной сегментации [184]. Однако, данные сети не являются наиболее точными для классических тестовых наборов данных, таких как COCO [185] по данным ресурса paperwithcode.com [186, 187] .

В 2016 году в работе [188] Редмондом Й. и соавторами была предложена "быстрая" - одно-этапная архитектура для решения задач обнаружения объектов. Архитектуру было предложено называть **YOLO** (you look only onece). Архитектура YOLO состоит из энкодера признаков (модифицированного Inception GoogLeNet см. рисунок 26 [103]) и набора выходных слоев. Последний выходной слой содержит K каналов по $S \times S$ ячеек. По задумке авторов, каждая из $N = S \times S$ ячеек должна соответствовать определенной части входного изображения. Число выходных каналов определяется как $B \times 5 + C$, где B - число возможных боксов для каждой ячейки; C - число классов; цифра 5 - это значения предсказанных координат и размеров бокса (x, y, w, h) и уверенности в том, что в боксе есть объект. В оригинальной работе $S = 7, B = 2, C = 20$. Таким образом YOLO может предсказать до $N = S \times S$ объектов, принадлежащих C классам, если каждый объект достаточно мал, чтобы быть спровоцированным на одну ячейку. Если один объект попадает на несколько ячеек, то для определения его координат используется метод не максимального сжатия по всем смежным ячейкам одного класса. Также архитектура может предсказать лишь ограниченное число боксов для каждой ячейки. Таким образом, одним из основных недостатков подхода является невозможность разделения объектов с высокой степенью пересечения. Однако, скорость работы YOLO значительно выше, чем для Faster-RCNN [189]. В последствии были предложены различны модификации архитектуры YOLO[189], а также альтернативный вариант одноэтапных архитектур SSD (single short detection) [190] и RetinaNet [191]. На настоящее время архитектуры на основе YOLO являются наиболее точными для решения задач обнаружения объектов в реальном масштабе времени [192], а также одними из наиболее среди всех архитектур для набора данных COCO по данным портала paperswithcode.com [186].

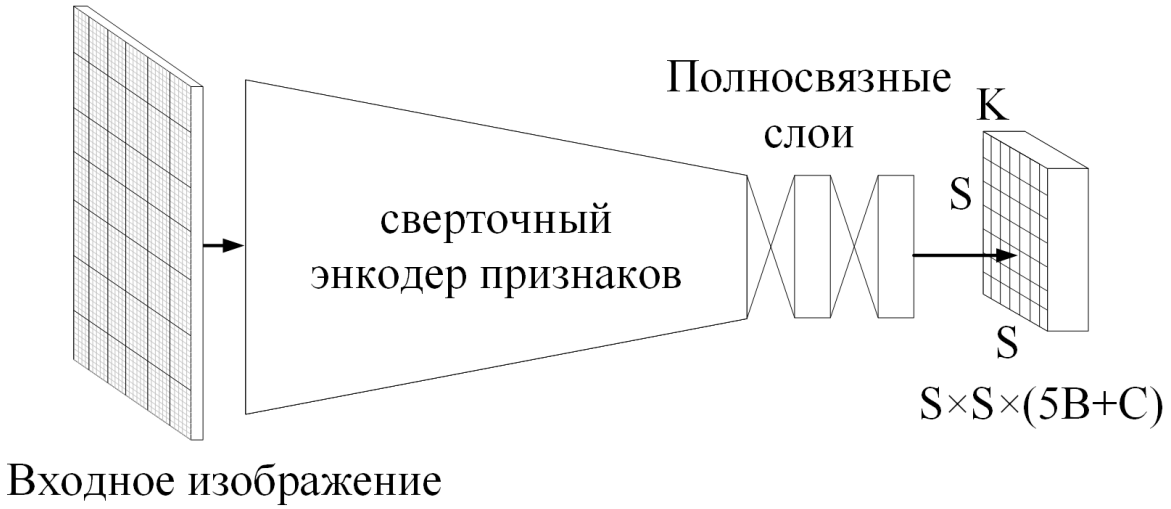

Рисунок 37 – Иллюстрация принципа работы архитектуры YOLO

1.6.3 Кратко о задачах генерации в компьютерном зрении

Как правило в компьютерном зрении и других приложениях глубокого обучения нейронных сетей рассматриваются задачи в которых из входных данных выделяются некоторые признаки, на основании которых принимаются целевые решения. Такие задачи можно назвать **дискриминативные задачи** [17]. Для разных экземпляров данных признаки могут быть регулярными (характерными), однако при этом они могу быть немного разными. Так например характерным признаком лица являются глаза, однако, расположение, форма и т.д. глаз у на разных фотографиях будут разными. Тогда можно сказать что нейронная сеть выучивает именно признак ни как конкретное число или набор чисел, а как конкретное распределение. Другими словами, признаки о которых идет речь можно рассматривать как не детерминированные - то есть стохастические (по существу, случайные) величины, которые обладают некоторым характерным распределением, имеющим характерное среднее значение и другие параметры. При таком рассмотрении сети помимо дискременативной задачи можно также поставить задачу генерации не только ответа сети, но и самих признаков, формирующих целевой ответ. Другими словами задачу генерации случайных значений признаков, полученных из соответствующих признаку распределений. Например, если сеть выучила распределение признаков лица (глаза, нос, уши и т.д.), то мы можем генерировать случайное лицо из некоторого посевного случайного набора чисел. Тут посевным набором называется число, исходя из которого выбирается некоторый результат распределения (если представить распределение как $p(x)$, то x - посевное число. Описанные задачи называются **генеративные задачи (или порождающие задачи)** [17].

Цель обучения порождающей нейронной сети в наиболее общем виде - это минимизация различий между распределениям некоторых признаков входных данных и выходных данных генерируемых нейронной сетью. Например, если есть набор лиц людей, то порождающая нейронная сеть должна давать на выходе изображения лиц, в которых например распределение размеров глаз, их положений или цвета будет таким же как для входных данных. Однако, так как на входе у нас есть только некоторые дискретные примеры лиц - то есть частные случае распределений, которые должна выучить сеть, то за счет обобщающею способности сеть может давать и другие частные случае распределений - случаи, которых нет во входных данных. Отметим, что на этом примере хорошо видно, что при обучении сеть аппроксимирует распределения признаков по их частным значениям. При этом эти аппроксимации формируются именно за счет связанных нелинейных преобразований входных данных. Отметим, что в зависимости от конкретных случаев обучение порождающей нейронной сети может производится как с учителем, так и на неразменных данных (unsupervised) или на данных с частичной разметкой (semi-supervised) [17].

Современные глубокие генерирующие модели могут быть разделены на два класса - модели с явным представлением распределения признаков и модели со скрытым представлением [193]. Наиболее популярен второй класс [17]. Среди подходов первого вида следует выделить модели, основанные на автоэнкодерах, в частности **вариационный автоэнкодер (Variational Auto Encoder, VAE)** [194]. Модель VAE представляет собой модификацию обычной автокодирующей сети (автоэнкодера, см. рисунок 20). Однако, в VAE кодирующая часть (энкодер) создает не непосредственно скрытое пространство, а пытается его аппроксимировать гауссовыми распределениями, для этого модель энкодера пытается оценить среднее и дисперсию для каждого параметра. Таким образом, создается как-бы распределение параметров. Затем из распределений создается входной вектор для декодирующей части из которого должен получиться ответ, в соответствующий распределению входных данных. Другими словами VAE должен генерировать новый результат на выходе - результат не похожий на входной экземпляр данных (как для обычного автоэенкодера), но из того же распределение что и экземпляр входных данных. По существу, такой подход соответствует идеи вариационного вывода, известной в других областях машинного обучения [195]. В литературе предложено достаточно большое количество модификаций идеи VAE для решения задач компьютерного зрения [196, 197]. Также идеи вариационного вывода, и VAE в частности, используются в генерации текстов [198] и как решение задачи анализа временных рядов [199] и потока видеоданных [200].

Отмечается, что основной недостаток подхода VAE - характерное гауссово размытие результатов работы [197]. Однако, влияние данного эффекта может быть снижено, например, путем использования дополнительных нейронных сетей, создающих дополнительные условия контрастности при обучении [201].

В 2014 году в работе [202] Яном Гудфеллоу и соавторами, в том числе Йошуа Бенджо была предложена архитектура **генеративно-состязательная нейронная сеть (ГАН, Generative-Adversel Neural Network, GAN).** Архитектура представляет собой класс моделей со скрытым представлением и является наиболее популярным подходом к задачам генерации.

Основная идея архитектуры заключается в использовании двух сетей - генерирующей (generative) и дискриментаивной (discriminative), которые обучаются совместно. Цель генеративной сети заключается в создании изображения из входных - посевных данных, которыми могут быть как случайный шум, так и некоторая разметка желаемого результата. На вход дискреминативной сети попеременно передаются данные, сгенерированные сетью (поддельные - fake данные) и примеры реальных данных, соответствующих, требуемой задачи. Например, изображения лиц - реальные данные (real). Цель дискреминативной сети определить поддельные или реальные данные поданы на ее вход. Таким образом, цель обучения архитектуры ГАН - одновременно понизить погрешность генеративной сети и повысить погрешность дискреминативной сети. Другими словами нужно сделать так, чтобы дискреминативная сеть не смогла определить какие данные поданы на ее вход - реальные или поддельные. При достижении этой цели можно предположить, что генеративная сеть смогла достоверно аппроксимировать распределения признаков, соответствующих задаче. Сеть называется генеративно-состязательной, так как, при обучении ГАН происходит одновременно попытка понижения ошибки генерирующей сети и попытка повышения ошибки дискреминативной сети. Отметим, что в связи с вышесказанным дискреминативная сеть может рассматривается как метод регуляризации работы генеративной сети. Иллюстрация работы архитектур ГАН приведена на рисунке 38.

В оригинальной статье Гудфеллоу предлагалось обучать ГАНы без разметки данных (то есть без учителя), при этом данные должны были генерироваться из шума. Однако, позже были предложены различные варианты реализации подхода как для задач работы с изображениями, так и для задач генерации текстовых данных, музыки [203] и в других, более экзотических, приложениях [204, 205]. В 2016 году Гудфеллоу была опубликована работа [193], в которой было подробно проанализировано место подхода ГАН среди других подходов к генеративному обучению и были даны подробные разъяснения касательно аспектов функционирования ГАН.

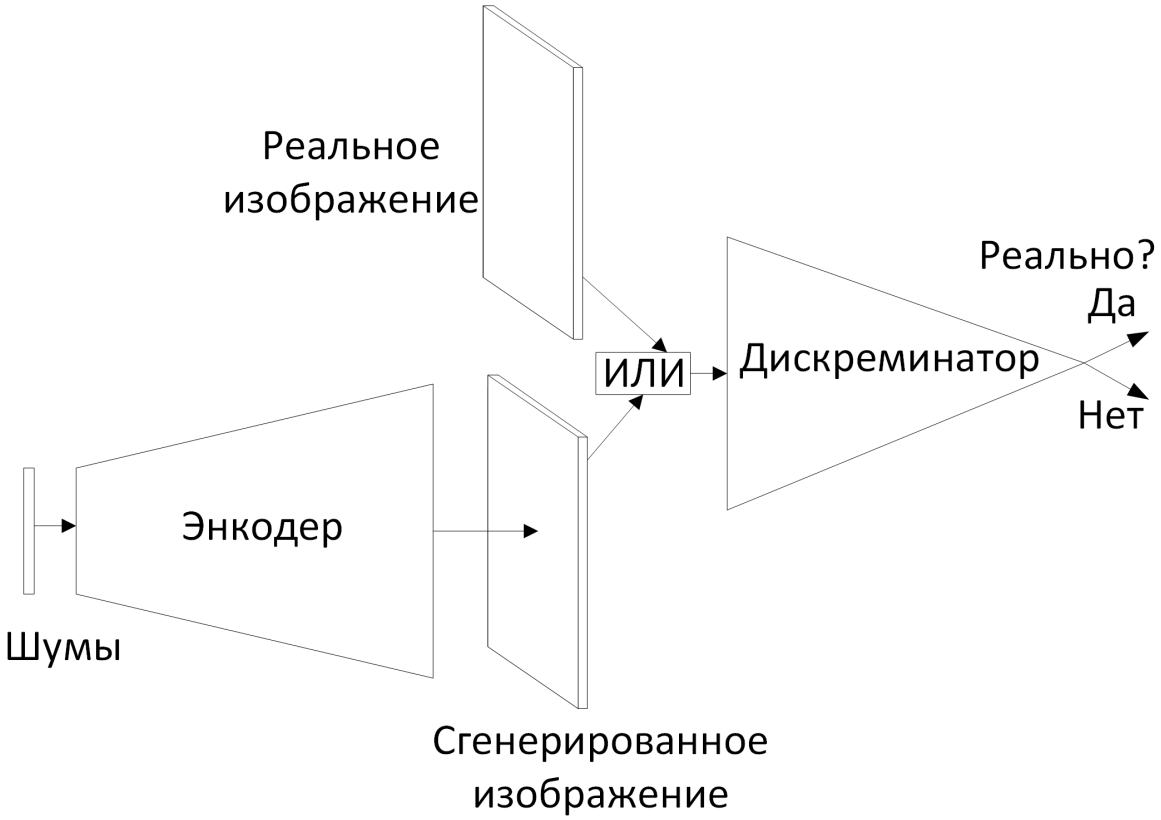

Рисунок 38 – Иллюстрация принципа работы архитектуры GAN

В работе [193] Гудфеллоу были обозначены следующие возможные цели создания и развития архитектур глубоких нейронных сетей генеративного обучения.

- Обучение и выборка из генеративных моделей является перспективным методом проверки способности представлять и управлять многомерными распределениями вероятностей. Распределения вероятностей многомерных данных являются важными объектами в широком спектре прикладных математических и инженерных областей.
- Использование генеративного подхода в обучении с подкреплением в качестве генератора входной среды. При этом также может быть получена статистика и по самой модели - путем проверки ее поведения в различных вариантах ситуаций одного типа.
- Использование генеративного подхода в качестве метода предсказания нескольких вариантов по текущим данным (например для временного ряда или видео ряда) - то есть для задач с мультимодальным выходом.

- Использование генеративного подхода в качестве метода обучения данных с пропусками. Например, для обучения с полу-учителем, для поиска аномалий или для решения проблем несбалансированных данных - когда данных одного класса в несколько раз больше чем другого. Например, в случае двух классов, когда размечен только один класс или часть данных - для известного класса обучается ГАН, а затем обученный дискреминатор используется для разметки остальных данных.
- Использование генеративного подхода для повышения разрешения входных данных - т.н. задача super-resolution.
- Решение задач типа перевод одного изображения в другое (Image-to-image translation). Например преобразование некоторой зарисовки или шаблона в конечное изображение.

Отметим, следующие перспективные задачи, решаемые генеративными нейронными сетями и не вошедшие в список выше. В 2017 году был описан метод аугментации данных при помощи ГАН [206]. В 2016 году была предложена идея генерации из текст в изображение (изображающее описанное в тексте) при помощи ГАН [207] и наоборот (изображение в текст) [208]. В ряде работ предлагаются архитектуры для генерации изображений на основании семантически набросков [209] или при помощи переноса признакового пространства из одной задачи в другую. Подход ГАН может быть использован для повышения устойчивости к изменениям данных при обучении нейронных сетей - то есть к т.н. соревновательным атакам [210]. Также, отметим, что одним из первых успешных применений архитектур - трансформеров в компьютерном зрении было их использование в генеративных задачах [211].

Как правило, на практике подход ГАН подход оказывается более точным, чем VAE и другие генеративные подходы. Однако, в некоторых случаях, например [212], арихтекутры на основе VAE оказываются более точными. Отметим, что в литературе рассматриваются архитектуры включающие совместно ГАН и VAE подходы [213]. Также в ряде современных исследований переделаются VAE-подобные подходы, которые, как указывается, разрешают основные проблемы ГАН архитектур и потенциально позволяют достичь больших точностей, чем способны ГАН [214].

В последние несколько лет также встречается большое количество работ, где генеративные нейронные используется для генерации текстовых данных [215]. Однако, указывается что на данный момент это направление, особенно в случае подхода ГАН, имеет ряд недостатков, связанных с, по существу, дискретной природой текста, а классическое генеративное обучение строит непрерывные аппроксимации признакового пространства. Другими словами, например, для генерации изображений лиц, распределение ширины глаз должно быть непрерывным. Для

текста непрерывное распределение не подходит, так как осмысленные грамматически верные слова - представляют собой дискретные значения признаков. В литературе рассматриваются несколько вариантов решения данной проблемы, однако не один из них не является зарекомендовавшим себя по настоящее время [216, 215]. Также генеративное обучение, в частности ГАН, могут быть использованы при синтезе музыкальных композиций, [217] и при анализе временных рядов [218].

1.7 Современные тренды - итоги исторического обзора

Глубокое обучение нейронных сетей, в том числе сверточных нейронных сетей получило широкое развитие в различных приложениях компьютерного зрения. Большинство фундаментальных результатов в этой области были получены в период конца 1990-х - 2015 годах. Однако, большая часть решений в их современном виде (архитектуры слоев и их блоков, методы регуляризации, типы функций активации скрытых слоев и т.д.) были разработаны после 2010-2012 годов. Так, например, общий подход к построению архитектуры сверточных нейронных сетей начинается с AlexNet. При этом иронично то, что и современные реализации LeNet 5 по сути представляют собой модификацию AlexNet. В обозначенный период времени утвердились такие решения, как:

- построение архитектур на основании блоков, включающих сверточные слои, слои пулнга, слои внимания и остаточные связи;
- регуляризация обучения путем батч-нормализации или методом дропаутов (а также адаптивные методы обучения о которых речь пойдет в свое время);
- предпочтительность использования автоматизированных методов подбора архитектур для конечных конфигураций аппаратного обеспечения и задачи;
- предпочтительность инициализации параметров путем предобучения нейронных сетей на больших наборах данных или на нескольких объединённых наборах данных.

При этом по настоящее время открытыми остаются следующие вопросы.

Выбор структуры блока сверточной сети и стратегии автоматической оптимизации

В научной литературе предложены достаточно большое количество вариантов построения блоков сверточных слоев. В некоторых случаях такие блоки ориентированы на их использование в специализированном оборудовании (например, для процессоров мобильных телефонов). В других случаях, блок должен быть универсальным. В зависимости от предназначения оптимальная структура блока может меняться. При этом вопрос о том, какой структура должна быть в тех или иных случаях остается не решенным. Отметим, что часто можно встретить мнение о том, что оптимальная архитектура нейронной сети (или ее блока) может быть выбрана методами автоматического поиска (например, NAS). Однако, следует помнить, что NAS позволяет провести оптимизацию только по выбранному пространству. Другими словами даже такие методы ограничены опытом исследователи и знаниями об эффективных структурах сверточных и других блоков. Кроме того в отношении NAS всегда следует помнить о корректности выбранной

стратегии поиска и корректности выбранных критериев оценки качества модели.

Выбор методов регуляризации обучения и работы нейронной сети Как было указано выше метод батч-нормализации является одним из основных методов регуляризации обучения нейронных сетей. Метод имеет ряд достоинств, в том числе [219]:

- снижение эффекта ковариационного сдвига и увеличение обобщающей способности;
- ускорение обучения за счет правильного диапазона значений функции активации - устранение т.н. "mean-shif" эффекта для ранних эпох обучения;
- возможность увеличения размеров батча и более грубого выбора скорости обучения.

Однако, условия для работоспособности метода батч-нормализации не всегда удается выполнить, о чем отмечают многие авторы [57, 108, 219]. В качестве основных мотивов для использования альтернативных методов нормализации приводятся следующие недостатки классической батч-нормализации:

- несоответствие результатов работы во время обучения и работы нейронной сети, а также в случаях предобучения и дообучения на пакетах разного размера или наборах данных с разным размером входных изображений;
- батч-нормализация не оказывает регуляризирующего эффекта для небольших размеров мини-пакетов (порядка нескольких экземпляров данных);
- при длительном обучении сетей с остаточными связями батч-нормализация приводит к преобладанию остаточных связей в слоях (снижает точность обучения).

В случаях, когда батч-нормализация не применима могут быть использованы альтернативные подходы[108]. Также открытым остается вопрос о том, что лучше: метод дроп-аутов или батч-нормализация. Отметим, что и в дополнение к данным методам нейронные сети всегда могут быть обучены с использованием классической L1/L2 регуляризации и с использованием аугментации данных. Однако, в работе [220] показано, что L2 регуляризация почти не дает результата в случае батч-нормализации. А в работе [83] показано, что L2 улучшает работы метода дропаутов. Отметим, что в работе 2020 года [221] показано, что аугментация данных методом случайного удаления частей изображений соответствует принципу работы метода дропаутов. Также данный подход вероятно может быть применен совместно с батч-нормализацией.

В 2020 году командой Google Deep Mind была предложена модификация архитектур ResNet и EfficientNet с оригинальным пересмотром идеи слоя батч-нормализации [222, 219]. Архитектуры разработанной нейронной сети было предложено называть NFNet [219]. Авторы NFNet смогли достичь точности 97.9% по "top-5" и 86.5% по методу "top-1" для набора данных ImageNet без использования дополнительных данных для предобучения. Для нейронных сетей без предобучения на дополнительных данных - это наивысший результат на начало августа 2021 года. [97].

Оптимизация стратегий обучения глубоких нейронных сетей

В 2019 наилучший результат для набора данных ImageNet показала архитектура BiT - вариант архитектуры ResNet с расширенными слоями (увеличено число карт признаков) и предобученный на наборе данных JFT-300M (300 миллионов изображений, [152]) при помощи специальных правил BiT-HyperRule [223]. Архитектура показала точность 98,5% по "top-5" и 88 % по методу "top-1" [97]. Пример архитектуры BiT показал, что при условии достаточного предобучения возможно расширение слоев нейронных и увеличение обобщающей способности. Однако, оставил открытым вопрос о том, каким образом лучше проводить предобучение. Отметим, что помимо классического предобучения с использование больших наборов данных в современной литературе исследуются следующие методы оптимизации обучения:

- обучение со смешиванием экземпляров данных и их меток [60];
- обучение с добавлением шумов к меткам (lable smoothing [224]);
- обучения с использованием дополнительной нейронной сети - методы "учитель-ученик" (относят к классу т.н. методов полу-контролируемого обучения, Semi-supervised Learning) [225];
- методы само-предобучения (Self-Supervised Learning) [226].

Среди методов смешивания данных и меток следует выделить работы 2017 и 2019 годов, в которых исследованы методы аугментации MixUp - наложение изображений и сложение меток в соответствующих прозрачности пропорциях [227] и метод Cut-Mix - вставка частей изображений и и сложение меток в соответствующих площади пропорциях. Также различными авторами предлагаются модификации данных и других схожих методов [60].

Методы обучения "Учитель-Ученик" предполагают наличие двух нейронных сетей, которые в зависимости от техники обучения могут иметь разное соотношение размеров. Так, например, для популярной техники дисциляции знаний (Knowlege Distilation) предполагается, что сеть-учитель (или ансамбль сетей) имеют больший размер, по сравнению с целевой сетью - учеником. Целью такого обучения является

подстраивание результатов работы сети-ученика под результаты сети учителя. При этом предполагается, что сеть ученик также научится копированию поведения учителя в отношении обобщающей способности. [228]

В 2020 наилучшие результаты для набора данных ImageNet показали архитектуры EfficientNet-B7 [229], и EfficientNet-L2 [230], предобученные и тренированные разработанными вариантами методов учитель-ученик. В случае EfficientNet-L2 проводилось обучение с полу-учителем, методом названным Meta-Pseudo-Labeling [230]. При этом авторы работы [230] достигли точности 98,2 % по методу "top-5" и 90,2 % по методу "top-1" [97].

Подходы к решению задач: трансформеры, полносвязные или сверточные сети

Как следует из обзора основной подход у решению задач компьютерного зрения - это использование глубоких сверточных нейронных сетей. Однако, работы 2020-2021, посвященные архитектурам-трансформерам показали перспективность подхода для больших наборов данных (на этапе предобучения) и при больших размерах экземпляров данных. Часть предложенных в литературе архитектур-трансформеров полностью полносвязные, часть имеют и сверточные и полносвязрые слои. Например в августе 2021 года была представлена архитектура BotNet. Авторы BotNet предложили заменить часть слоев архитектуры ResNet на слои многоголового само-внимания и при этом получили небольшой прирост точности (0,5%) по сравнению с оригинальной архитектурой, однако, они увеличили число параметров[231]. Таким образом вопрос о том, какой подход (или комбинация подходов) - лучше остается открытым. В работе [232] посвященной сверточных сетей и трансформеров приходят к выводу, что трансформеры быстрее и более точно выучивают крупно-размерные признаки, чем сверточные сети. Однако, архитектурам-трансформерам требуется больше примеров для мелко-размерных признаков.

В 2021 году был представлен ряд трансформеров-подобных полносвязных архитектур для решения задач компьютерного зрения. В таких архитектурах соли многоголового само-внимания был заменены на наборы операций с полносвязными слоями [233]. Наиболее известная из таких архитектур MLP-Mixer, предложенная командой Google Brain. Архитектура показала точность (87,9% по методу top-1 на наборе данных ImageNet) с предобучением на наборе данных JFT-300M, показав при этом производительность в 2,5 раза выше, чем архитектура ViT [234]. На данный момент полностью полносвязные архитектуры не дают высокой точности, сопоставимой с трансформерами, однако потенциально способных их заменить. Основным преимуществом таких архитектур является пониженное число параметров (из за отсутствия многоголового само-внимания) и высокая скорость

работы (операции с полносвязными слоями - самые быстрые).

Также отметим подход CLIP переставленный командой OpenAI в работе [235]. Авторы предложили проводить предобучение на набора изображений с описанием. В основе подхода лежит идея, что для набора обучаются два кодировщика признаков (один для изображения, другой для описания) так, чтобы они давали наиболее схожие представления на выходе. Таким образом, по задумке авторов, модель должна выучить текстовое описание для изображений. Авторы рекомендуют использовать в своем подходе архитектуры трансформеры. Подход интересен своей оригинальностью и возможно является перспективным. Главным достоинством подхода является высокая обобщающая способность. Однако, в настоящее время подход показывает достойные результаты лишь на некоторых задачах.

2 EOC

ГЛАВНОЕ ПРИЕМУЩЕСТВО DL; выделение, преобразование и выбор признаков, релевантные поставленной задачи. Однако процедура обучения требует достаточно большого набора данных для правильной обработки таких признаков

СПИСОК ЛИТЕРАТУРЫ

[1] McCulloch Warren S, Pitts Walter. A logical calculus of the ideas immanent in nervous activity // The bulletin of mathematical biophysics. 1943. Т. 5, № 4. С. 115–133.

[2] Hebb Donald Olding. The organization of behavior; a neuropsycholocigal theory // A Wiley Book in Clinical Psychology. 1949. Т. 62. с. 78.

[3] Novikoff Albert B. On convergence proofs for perceptrons: Tech. Rep.: : Stanford Research Inst, Menlo park CA, 1963.

[4] Rosenblatt Frank. The perceptron: a probabilistic model for information storage and organization in the brain. // Psychological review. 1958. Т. 65, № 6. с. 386.

[5] Rosenblatt Frank. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957.

[6] Hay John C, Lynch Ben E, Smith David R. Mark I Perceptron Operators' Manual: Tech. Rep.: : Cornell Aeronautical Lab Inc, Buffalo NY, 1960.

[7] Kelleher J.D. Deep Learning. The MIT Press Essential Knowledge series. MIT Press, 2019. URL: https://books.google.ru/books?id=ZU6qDwAAQBAJ.

[8] Widrow Bernard, Hoff Marcian E. Adaptive switching circuits: Tech. Rep.: : Stanford Univ Ca Stanford Electronics Labs, 1960.

[9] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT press, 2016.

[10] Minsky Marvin, Papert Seymour. An introduction to computational geometry // Cambridge tiass., HIT. 1969.

[11] Ivakhnenko Alexey Grigorevich. Polynomial theory of complex systems // IEEE transactions on Systems, Man, and Cybernetics. 1971. № 4. С. 364–378.

[12] Schmidhuber Jürgen. Deep learning in neural networks: An overview // Neural networks. 2015. Т. 61. С. 85–117.

[13] Rummelhart David E, McClelland James L, Group PDP Research [и др.]. Parallel distributed processing. 1986.

[14] Rumelhart David E, Hinton Geoffrey E, Williams Ronald J. Learning internal representations by error propagation: Tech. Rep.: : California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[15] Галушкин Александр Иванович. Синтез многослойных систем распознавания образов. Энергия, 1974.

[16] Барцев СИ, Охонин ВА. Адаптивные сети обработки информации // Красноярск: Ин-т физики СО АН СССР. 1986. № 59Б. с. 20.

[17] Николенко Сергей, Кадурин Артур, Архангельская Екатерина. Глубокое обучение. " Издательский дом"" Питер""", 2017.

[18] Gradient-based learning applied to document recognition / Yann LeCun, Léon Bottou, Yoshua Bengio [и др.] // Proceedings of the IEEE. 1998. Т. 86, № 11. С. 2278–2324.

[19] Cybenko George. Approximation by superpositions of a sigmoidal function // Mathematics of control, signals and systems. 1989. Т. 2, № 4. С. 303–314.

[20] Hornik Kurt. Approximation capabilities of multilayer feedforward networks // Neural networks. 1991. Т. 4, № 2. С. 251–257.

[21] Kidger Patrick, Lyons Terry. Universal approximation with deep narrow networks // Conference on learning theory / PMLR. 2020. С. 2306–2327.

[22] Tabuada Paulo, Gharesifard Bahman. Universal approximation power of deep residual neural networks via nonlinear control theory // arXiv preprint arXiv:2007.06007. 2020.

[23] Арнольд Владимир Игоревич. О функциях трех переменных // Доклады Академии наук / Российская академия наук. Т. 114. 1957. С. 679–681.

[24] Колмогоров Андрей Николаевич. О представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения // Доклады Академии наук / Российская академия наук. Т. 114. 1957. С. 953–956.

[25] Вапник Владимир Наумович, Червоненкис Алексей Яковлевич. О равномерной сходимости частот появления событий к их вероятностям // Теория вероятностей и ее применения. 1971. Т. 16, № 2. С. 264–279.

[26] Vapnik V. Estimation of dependences based on empirical data berlin. 1982.

[27] Vapnik Vladimir, Levin Esther, Le Cun Yann. Measuring the VC-dimension of a learning machine // Neural computation. 1994. Т. 6, № 5. С. 851–876.

[28] Вапник Владимир Наумович, Червоненкис Алексей Яковлевич. Теория распознавания образов. М.: Наука, 1974.

[29] Wolpert David H, Macready William G. No free lunch theorems for optimization // IEEE transactions on evolutionary computation. 1997. Т. 1, № 1. С. 67–82.

[30] Hochreiter Sepp. Untersuchungen zu dynamischen neuronalen Netzen // Diploma, Technische Universität München. 1991. Т. 91, № 1.

[31] Fukushima Kunihiko. Cognitron: A self-organizing multilayered neural network // Biological cybernetics. 1975. Т. 20, № 3. С. 121–136.

[32] Hubel David H, Wiesel Torsten N. Receptive fields and functional architecture of monkey striate cortex // The Journal of physiology. 1968. Т. 195, № 1. С. 215–243.

[33] Hubel David H, Wiesel Torsten N. Receptive fields of single neurones in the cat's striate cortex // The Journal of physiology. 1959. Т. 148, № 3. С. 574–591.

[34] Roberts Lawrence G. Machine perception of three-dimensional solids. Ph.D. thesis: Massachusetts Institute of Technology. 1963.

[35] Fukushima Kunihiko, Miyake Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition // Competition and cooperation in neural nets. Springer, 1982. С. 267–285.

[36] Аксёнов С.В., Новосельцев В.Б. Повышение качества распознавания сцен нейронной сетью "Неокогнитрон". 2006. № 7.

[37] Hubel David H, Wiesel Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex // The Journal of physiology. 1962. Т. 160, № 1. С. 106–154.

[38] Marr David. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co // Inc., New York, NY. 1982. Т. 2, № 4.2.

[39] Zhang Wei [и др.]. Shift-invariant pattern recognition neural network and its optical architecture // Proceedings of annual conference of the Japan Society of Applied Physics. 1988.

[40] A shift-invariant neural network for the lung field segmentation in chest radiography / Akira Hasegawa, Shih-Chung B Lo, Jyh-Shyan Lin [и др.] // Journal of VLSI signal processing systems for signal, image and video technology. 1998. Т. 18, № 3. С. 241–250.

[41] A survey of convolutional neural networks: analysis, applications, and prospects / Zewen Li, Fan Liu, Wenjie Yang [и др.] // IEEE Transactions on Neural Networks and Learning Systems. 2021.

[42] Phoneme recognition using time-delay neural networks / Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton [и др.] // IEEE transactions on acoustics, speech, and signal processing. 1989. Т. 37, № 3. С. 328–339.

[43] Backpropagation applied to handwritten zip code recognition / Yann LeCun, Bernhard Boser, John S Denker [и др.] // Neural computation. 1989. Т. 1, № 4. С. 541–551.

[44] LeCun Yann [и др.]. Generalization and network design strategies // Connectionism in perspective. 1989. Т. 19. С. 143–155.

[45] Bridle John S. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters // Advances in neural information processing systems. 1990. С. 211–217.

[46] Bridle John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition // Neurocomputing. Springer, 1990. С. 227–236.

[47] Face recognition: A convolutional neural-network approach / Steve Lawrence, C Lee Giles, Ah Chung Tsoi [и др.] // IEEE transactions on neural networks. 1997. Т. 8, № 1. С. 98–113.

[48] The MNIST database of handwritten digits. URL: http://yann.lecun.com/exdb/mnist/.

[49] Weng Juyang, Ahuja Narendra, Huang Thomas S. Cresceptron: a self-organizing neural network which grows adaptively // [Proceedings 1992] IJCNN International Joint Conference on Neural Networks / IEEE. Т. 1. 1992. С. 576–581.

[50] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning // nature. 2015. Т. 521, № 7553. С. 436–444.

[51] Efficient backprop / Yann LeCun, Leon Bottou, G Orr [и др.] // Neural Networks: Tricks of the Trade. New York: Springer. 1998.

[52] Poggio Tomaso, Girosi Federico. A theory of networks for approximation and learning: Tech. Rep.: : Massachusetts INST of TECH Cambridge Artificial Intelligence LAB, 1989.

[53] Girosi Federico, Jones Michael, Poggio Tomaso. Regularization theory and neural networks architectures // Neural computation. 1995. Т. 7, № 2. С. 219–269.

[54] Tihonov Andrei Nikolajevits. Solution of incorrectly formulated problems and the regularization method // Soviet Math. 1963. Т. 4. С. 1035–1038.

[55] Tibshirani Robert. Regression shrinkage and selection via the lasso // Journal of the Royal Statistical Society: Series B (Methodological). 1996. Т. 58, № 1. С. 267–288.

[56] Zou Hui, Hastie Trevor. Regularization and variable selection via the elastic net // Journal of the royal statistical society: series B (statistical methodology). 2005. Т. 67, № 2. С. 301–320.

[57] Moradi Reza, Berangi Reza, Minaei Behrouz. A survey of regularization strategies for deep models // Artificial Intelligence Review. 2020. Т. 53, № 6. С. 3947–3986.

[58] Best practices for convolutional neural networks applied to visual document analysis. / Patrice Y Simard, David Steinkraus, John C Platt [и др.] // Icdar. Т. 3. 2003.

[59] Baird Henry S. Document image defect models // Structured Document Image Analysis. Springer, 1992. С. 546–556.

[60] Shorten Connor, Khoshgoftaar Taghi M. A survey on image data augmentation for deep learning // Journal of Big Data. 2019. Т. 6, № 1. С. 1–48.

[61] Hinton Geoffrey E, Osindero Simon, Teh Yee-Whye. A fast learning algorithm for deep belief nets // Neural computation. 2006. Т. 18, № 7. С. 1527–1554.

[62] Hinton Geoffrey E. To recognize shapes, first learn to generate images // Progress in brain research. 2007. Т. 165. С. 535–547.

[63] Ackley David H, Hinton Geoffrey E, Sejnowski Terrence J. A learning algorithm for Boltzmann machines // Cognitive science. 1985. Т. 9, № 1. С. 147–169.

[64] Parallel distributed processing / James L McClelland, David E Rumelhart, PDP Research Group [и др.]. MIT press Cambridge, MA, 1986. Т. 2.

[65] Hopfield John J. Neural networks and physical systems with emergent collective computational abilities // Proceedings of the national academy of sciences. 1982. Т. 79, № 8. С. 2554–2558.

[66] Hinton Geoffrey E, Salakhutdinov Ruslan R. Reducing the dimensionality of data with neural networks // science. 2006. Т. 313, № 5786. С. 504–507.

[67] Ballard Dana H. Modular learning in neural networks. // AAAI. Т. 647. 1987. С. 279–284.

[68] Why does unsupervised pre-training help deep learning? / Dumitru Erhan, Aaron Courville, Yoshua Bengio [и др.] // Proceedings of the thirteenth international conference on artificial intelligence and statistics / JMLR Workshop and Conference Proceedings. 2010. С. 201–208.

[69] Greedy layer-wise training of deep networks / Yoshua Bengio, Pascal Lamblin, Dan Popovici [и др.] // Advances in neural information processing systems. 2007. С. 153–160.

[70] Oh Kyoung-Su, Jung Keechul. GPU implementation of neural networks // Pattern Recognition. 2004. Т. 37, № 6. С. 1311–1314.

[71] Steinkraus Dave, Buck Ian, Simard PY. Using GPUs for machine learning algorithms // Eighth International Conference on Document Analysis and Recognition (ICDAR'05) / IEEE. 2005. С. 1115–1120.

[72] Chellapilla Kumar, Puri Sidd, Simard Patrice. High performance convolutional neural networks for document processing // Tenth international workshop on frontiers in handwriting recognition / Suvisoft. 2006.

[73] Bengio Yoshua, LeCun Yann [и др.]. Scaling learning algorithms towards AI // Large-scale kernel machines. 2007. Т. 34, № 5. С. 1–41.

[74] What is the best multi-stage architecture for object recognition? / Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato [и др.] // 2009 IEEE 12th international conference on computer vision / IEEE. 2009. С. 2146–2153.

[75] Nair Vinod, Hinton Geoffrey E. Rectified linear units improve restricted boltzmann machines // Icml. 2010.

[76] Activation functions / Mohit Goyal, Rajan Goyal, P Venkatappa Reddy [и др.] // Deep learning: Algorithms and applications. Springer, 2020. С. 1–30.

[77] Glorot Xavier, Bengio Yoshua. Understanding the difficulty of training deep feedforward neural networks // Proceedings of the thirteenth international conference on artificial intelligence and statistics / JMLR Workshop and Conference Proceedings. 2010. С. 249–256.

[78] Delving deep into rectifiers: Surpassing human-level performance on imagenet classification / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // Proceedings of the IEEE international conference on computer vision. 2015. С. 1026–1034.

[79] Narkhede Meenal V, Bartakke Prashant P, Sutaone Mukul S. A review on weight initialization strategies for neural networks // Artificial Intelligence Review. 2021. С. 1–32.

[80] Improving neural networks by preventing co-adaptation of feature detectors / Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky [и др.] // arXiv preprint arXiv:1207.0580. 2012.

[81] Labach Alex, Salehinejad Hojjat, Valaee Shahrokh. Survey of dropout methods for deep neural networks // arXiv preprint arXiv:1904.13310. 2019.

[82] Baldi Pierre, Sadowski Peter J. Understanding dropout // Advances in neural information processing systems. 2013. Т. 26. С. 2814–2822.

[83] Dropout: a simple way to prevent neural networks from overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky [и др.] // The journal of machine learning research. 2014. Т. 15, № 1. С. 1929–1958.

[84] Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey / Giang Nguyen, Stefan Dlugolinsky, Martin Bobák [и др.] // Artificial Intelligence Review. 2019. Т. 52, № 1. С. 77–124.

[85] Deep, big, simple neural nets for handwritten digit recognition / Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella [и др.] // Neural computation. 2010. Т. 22, № 12. С. 3207–3220.

[86] Flexible, high performance convolutional neural networks for image classification / Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci [и др.] // Twenty-second international joint conference on artificial intelligence. 2011.

[87] The CIFAR-10 dataset. URL: https://www.cs.toronto.edu/ kriz/cifar.html.

[88] Theano: A CPU and GPU math compiler in Python / James Bergstra, Olivier Breuleux, Frédéric Bastien [и др.] // Proc. 9th python in science conf. Т. 1. 2010. С. 3–10.

[89] The deep learning compiler: A comprehensive survey / Mingzhen Li, Yi Liu, Xiaoyan Liu [и др.] // IEEE Transactions on Parallel and Distributed Systems. 2020. Т. 32, № 3. С. 708–727.

[90] Krizhevsky Alex, Hinton Geoff. Convolutional deep belief networks on cifar-10 // Unpublished manuscript. 2010. Т. 40, № 7. С. 1–9.

[91] A committee of neural networks for traffic sign classification / Dan Cireşan, Ueli Meier, Jonathan Masci [и др.] // The 2011 international joint conference on neural networks / IEEE. 2011. С. 1918–1921.

[92] Ciregan Dan, Meier Ueli, Schmidhuber Jürgen. Multi-column deep neural networks for image classification // 2012 IEEE conference on computer vision and pattern recognition / IEEE. 2012. С. 3642–3649.

[93] ImageNet: A Large-Scale Hierarchical Image Database / J. Deng, W. Dong, R. Socher [и др.] // CVPR09. 2009.

[94] ImageNet Large Scale Visual Recognition Challenge / Olga Russakovsky, Jia Deng, Hao Su [и др.] // International Journal of Computer Vision (IJCV). 2015. Т. 115, № 3. С. 211–252.

[95] ImageNet web site. URL: https://image-net.org/challenges/LSVRC/2012/index.php.

[96] Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. 2012. Т. 25. С. 1097–1105.

[97] Image Classification on ImageNet. PapersWithCode.com. URL: https://paperswithcode.com/sota/image-classification-on-imagenet.

[98] Prabhu Vinay Uday, Birhane Abeba. Large image datasets: A pyrrhic win for computer vision? // arXiv preprint arXiv:2006.16923. 2020.

[99] Sik-Ho Tsang. Summary: My Paper Reading Lists, Tutorials & Sharings // sh-tsang blog, https://sh-tsang.medium.com/. 2020. URL: https://sh-tsang.medium.com/overview-my-reviewed-paper-lists-tutorials-946ce59fbf9e.

[100] Zeiler Matthew D, Fergus Rob. Visualizing and understanding convolutional networks // European conference on computer vision / Springer. 2014. С. 818–833.

[101] Simonyan Karen, Zisserman Andrew. Very deep convolutional networks for large-scale image recognition // arXiv preprint arXiv:1409.1556. 2014.

[102] Lin Min, Chen Qiang, Yan Shuicheng. Network in network // arXiv preprint arXiv:1312.4400. 2013.

[103] Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. С. 1–9.

[104] Rethinking the inception architecture for computer vision / Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. С. 2818–2826.

[105] Inception-v4, inception-resnet and the impact of residual connections on learning / Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke [и др.] // Thirty-first AAAI conference on artificial intelligence. 2017.

[106] A survey of the recent architectures of deep convolutional neural networks / Asifullah Khan, Anabia Sohail, Umme Zahoora [и др.] // Artificial Intelligence Review. 2020. Т. 53, № 8. С. 5455–5516.

[107] Ioffe Sergey, Szegedy Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift // International conference on machine learning / PMLR. 2015. С. 448–456.

[108] Normalization techniques in training dnns: Methodology, analysis and application / Lei Huang, Jie Qin, Yi Zhou [и др.] // arXiv preprint arXiv:2009.12836. 2020.

[109] Summers Cecilia, Dinneen Michael J. Four things everyone should know to improve batch normalization // arXiv preprint arXiv:1906.03548. 2019.

[110] Dive into Deep Learning / Aston Zhang, Zachary C. Lipton, Mu Li [и др.] // arXiv preprint arXiv:2106.11342. 2021.

[111] Lubana Ekdeep Singh, Dick Robert P, Tanaka Hidenori. Beyond BatchNorm: Towards a General Understanding of Normalization in Deep Learning // arXiv preprint arXiv:2106.05956. 2021.

[112] Towards understanding regularization in batch normalization / Ping Luo, Xinjiang Wang, Wenqi Shao [и др.] // arXiv preprint arXiv:1809.00846. 2018.

[113] Understanding the disharmony between dropout and batch normalization by variance shift / Xiang Li, Shuo Chen, Xiaolin Hu [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. С. 2682–2690.

[114] Deep residual learning for image recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. С. 770–778.

[115] Identity mappings in deep residual networks / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // European conference on computer vision / Springer. 2016. С. 630–645.

[116] A state-of-the-art survey on deep learning theory and architectures / Md Zahangir Alom, Tarek M Taha, Chris Yakopcic [и др.] // Electronics. 2019. Т. 8, № 3. с. 292.

[117] Deep networks with stochastic depth / Gao Huang, Yu Sun, Zhuang Liu [и др.] // European conference on computer vision / Springer. 2016. С. 646–661.

[118] Aggregated residual transformations for deep neural networks / Saining Xie, Ross Girshick, Piotr Dollár [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. С. 1492–1500.

[119] Sik-Ho Tsang. Review: Trimps-Soushen — Winner in ILSVRC 2016 (Image Classification) // sh-tsang blog, https://sh-tsang.medium.com/. 2018. URL: https://towardsdatascience.com/review-trimps-soushen-winner-in-ilsvrc-2016-image-classification-dfbc423111dd.

[120] Densely connected convolutional networks / Gao Huang, Zhuang Liu, Laurens Van Der Maaten [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. С. 4700–4708.

[121] Mishra Rahul, Gupta Hari Prabhat, Dutta Tanima. A survey on deep neural network compression: Challenges, overview, and solutions // arXiv preprint arXiv:2010.03954. 2020.

[122] Minerva: Enabling low-power, highly-accurate deep neural network accelerators / Brandon Reagen, Paul Whatmough, Robert Adolf [и др.] // 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA) / IEEE. 2016. С. 267–278.

[123] SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size / Forrest N Iandola, Song Han, Matthew W Moskewicz [и др.] // arXiv preprint arXiv:1602.07360. 2016.

[124] He Kaiming, Sun Jian. Convolutional neural networks at constrained time cost // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. С. 5353–5360.

[125] Hollemans Matthijs. New mobile neural network architectures // Matthijs Hollemans blog, https://machinethink.net/blog/. 2020.

[126] Mobilenets: Efficient convolutional neural networks for mobile vision applications / Andrew G Howard, Menglong Zhu, Bo Chen [и др.] // arXiv preprint arXiv:1704.04861. 2017.

[127] Mobilenetv2: Inverted residuals and linear bottlenecks / Mark Sandler, Andrew Howard, Menglong Zhu [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. С. 4510–4520.

[128] Chollet François. Xception: Deep learning with depthwise separable convolutions // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. С. 1251–1258.

[129] Striving for simplicity: The all convolutional net / Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox [и др.] // arXiv preprint arXiv:1412.6806. 2014.

[130] Itti Laurent, Koch Christof. Computational modelling of visual attention // Nature reviews neuroscience. 2001. Т. 2, № 3. С. 194–203.

[131] Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate // arXiv preprint arXiv:1409.0473. 2014.

[132] An attentive survey of attention models / Sneha Chaudhari, Varun Mithal, Gungor Polatkan [и др.] // arXiv preprint arXiv:1904.02874. 2019.

[133] Cheng Jianpeng, Dong Li, Lapata Mirella. Long short-term memory-networks for machine reading // arXiv preprint arXiv:1601.06733. 2016.

[134] A decomposable attention model for natural language inference / Ankur P Parikh, Oscar Täckström, Dipanjan Das [и др.] // arXiv preprint arXiv:1606.01933. 2016.

[135] Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Parmar [и др.] // Advances in neural information processing systems. 2017. С. 5998–6008.

[136] Weng Lilian. Attention? Attention! // Lil'Log blog, lilianweng.github.io/lil-log. 2021. URL: http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html.

[137] Yang Xiao. An Overview of the Attention Mechanisms in Computer Vision // Journal of Physics: Conference Series / IOP Publishing. Т. 1693. 2020. с. 012173.

[138] Cordonnier Jean-Baptiste, Loukas Andreas, Jaggi Martin. On the relationship between self-attention and convolutional layers // arXiv preprint arXiv:1911.03584. 2019.

[139] Hu Jie, Shen Li, Sun Gang. Squeeze-and-excitation networks // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. С. 7132–7141.

[140] Searching for mobilenetv3 / Andrew Howard, Mark Sandler, Grace Chu [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. С. 1314–1324.

[141] Weng Lilian. Neural Architecture Search // Lil'Log blog, lilianweng.github.io/lil-log. 2020. URL: http://lilianweng.github.io/lil-log/2020/08/06/neural-architecture-search.html.

[142] Elsken Thomas, Metzen Jan Hendrik, Hutter Frank. Neural architecture search: A survey // The Journal of Machine Learning Research. 2019. Т. 20, № 1. С. 1997–2017.

[143] Hutter Frank, Kotthoff Lars, Vanschoren Joaquin. Automated machine learning: methods, systems, challenges. Springer Nature, 2019.

[144] Learning transferable architectures for scalable image recognition / Barret Zoph, Vijay Vasudevan, Jonathon Shlens [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. С. 8697–8710.

[145] Progressive neural architecture search / Chenxi Liu, Barret Zoph, Maxim Neumann [и др.] // Proceedings of the European conference on computer vision (ECCV). 2018. С. 19–34.

[146] Zoph Barret, Le Quoc V. Neural architecture search with reinforcement learning // arXiv preprint arXiv:1611.01578. 2016.

[147] Tan Mingxing, Le Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks // International Conference on Machine Learning / PMLR. 2019. С. 6105–6114.

[148] Tan Mingxing, Le Quoc V. Efficientnetv2: Smaller models and faster training // arXiv preprint arXiv:2104.00298. 2021.

[149] A survey on visual transformer / Kai Han, Yunhe Wang, Hanting Chen [и др.] // arXiv preprint arXiv:2012.12556. 2020.

[150] A Survey of Transformers / Tianyang Lin, Yuxin Wang, Xiangyang Liu [и др.] // arXiv preprint arXiv:2106.04554. 2021.

[151] Bert: Pre-training of deep bidirectional transformers for language understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee [и др.] // arXiv preprint arXiv:1810.04805. 2018.

[152] Sun Chen, Shrivastava Abhinav, Singh Saurabh [и др.]. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. 2017.

[153] An image is worth 16x16 words: Transformers for image recognition at scale / Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov [и др.] // arXiv preprint arXiv:2010.11929. 2020.

[154] Transformers in vision: A survey / Salman Khan, Muzammal Naseer, Munawar Hayat [и др.] // arXiv preprint arXiv:2101.01169. 2021.

[155] Training data-efficient image transformers & distillation through attention / Hugo Touvron, Matthieu Cord, Matthijs Douze [и др.] // International Conference on Machine Learning / PMLR. 2021. С. 10347–10357.

[156] Volo: Vision outlooker for visual recognition / Li Yuan, Qibin Hou, Zihang Jiang [и др.] // arXiv preprint arXiv:2106.13112. 2021.

[157] Scaling vision transformers / Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby [и др.] // arXiv preprint arXiv:2106.04560. 2021.

[158] Gidaris Spyros, Singh Praveer, Komodakis Nikos. Unsupervised representation learning by predicting image rotations // arXiv preprint arXiv:1803.07728. 2018.

[159] Jing Longlong, Tian Yingli. Self-supervised visual feature learning with deep neural networks: A survey // IEEE transactions on pattern analysis and machine intelligence. 2020.

[160] A taxonomy of deep convolutional neural nets for computer vision / Suraj Srinivas, Ravi Kiran Sarvadevabhatla, Konda Reddy Mopuri [и др.] // Frontiers in Robotics and AI. 2016. Т. 2. с. 36.

[161] Sultana Farhana, Sufian Abu, Dutta Paramartha. Evolution of image segmentation using deep convolutional neural network: a survey // Knowledge-Based Systems. 2020. Т. 201. с. 106062.

[162] Thoma Martin. A survey of semantic segmentation // arXiv preprint arXiv:1602.06541. 2016.

[163] Learning hierarchical features for scene labeling / Clement Farabet, Camille Couprie, Laurent Najman [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2012. Т. 35, № 8. С. 1915–1929.

[164] Long Jonathan, Shelhamer Evan, Darrell Trevor. Fully convolutional networks for semantic segmentation // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. С. 3431–3440.

[165] Noh Hyeonwoo, Hong Seunghoon, Han Bohyung. Learning deconvolution network for semantic segmentation // Proceedings of the IEEE international conference on computer vision. 2015. С. 1520–1528.

[166] Badrinarayanan Vijay, Handa Ankur, Cipolla Roberto. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling // arXiv preprint arXiv:1505.07293. 2015.

[167] Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-net: Convolutional networks for biomedical image segmentation // International Conference on Medical image computing and computer-assisted intervention / Springer. 2015. С. 234–241.

[168] Punn Narinder Singh, Agarwal Sonali. Modality specific U-Net variants for biomedical image segmentation: A survey // arXiv preprint arXiv:2107.04537. 2021.

[169] Semantic Segmentation Models. PapersWithCode.com. URL: https://paperswithcode.com/methods/category/segmentation-models.

[170] PASCAL VOC dataset and challenge. URL: http://host.robots.ox.ac.uk/pascal/VOC/.

[171] Semantic Segmentation on PASCAL VOC 2012 test. URL: https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012.

[172] Deep semantic segmentation of natural and medical images: a review / Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen [и др.] // Artificial Intelligence Review. 2021. Т. 54, № 1. С. 137–178.

[173] A survey of semi-and weakly supervised semantic segmentation of images / Man Zhang, Yong Zhou, Jiaqi Zhao [и др.] // Artificial Intelligence Review. 2020. Т. 53, № 6. С. 4259–4288.

[174] Vaillant Régis, Monrocq Christophe, Le Cun Yann. Original approach for the localisation of objects in images // IEE Proceedings-Vision, Image and Signal Processing. 1994. Т. 141, № 4. С. 245–250.

[175] Rowley Henry A, Baluja Shumeet, Kanade Takeo. Neural network-based face detection // IEEE Transactions on pattern analysis and machine intelligence. 1998. Т. 20, № 1. С. 23–38.

[176] Overfeat: Integrated recognition, localization and detection using convolutional networks / Pierre Sermanet, David Eigen, Xiang Zhang [и др.] // arXiv preprint arXiv:1312.6229. 2013.

[177] Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick, Jeff Donahue, Trevor Darrell [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. С. 580–587.

[178] Selective search for object recognition / Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers [и др.] // International journal of computer vision. 2013. Т. 104, № 2. С. 154–171.

[179] Object detection in 20 years: A survey / Zhengxia Zou, Zhenwei Shi, Yuhong Guo [и др.] // arXiv preprint arXiv:1905.05055. 2019.

[180] Girshick Ross. Fast r-cnn // Proceedings of the IEEE international conference on computer vision. 2015. С. 1440–1448.

[181] Spatial pyramid pooling in deep convolutional networks for visual recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2015. Т. 37, № 9. С. 1904–1916.

[182] Faster r-cnn: Towards real-time object detection with region proposal networks / Shaoqing Ren, Kaiming He, Ross Girshick [и др.] // Advances in neural information processing systems. 2015. Т. 28. С. 91–99.

[183] Mask r-cnn / Kaiming He, Georgia Gkioxari, Piotr Dollár [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. С. 2961–2969.

[184] Hafiz Abdul Mueed, Bhat Ghulam Mohiuddin. A survey on instance segmentation: state of the art // International journal of multimedia information retrieval. 2020. С. 1–19.

[185] Microsoft coco: Common objects in context / Tsung-Yi Lin, Michael Maire, Serge Belongie [и др.] // European conference on computer vision / Springer. 2014. С. 740–755.

[186] Object Detection on COCO test-dev. URL: https://paperswithcode.com/sota/object-detection-on-coco.

[187] Instance Segmentation on COCO test-dev. URL: https://paperswithcode.com/sota/instance-segmentation-on-coco.

[188] You only look once: Unified, real-time object detection / Joseph Redmon, Santosh Divvala, Ross Girshick [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. С. 779–788.

[189] Hui Jonathan. Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3 // Jonathan Hui blog, https://jonathan-hui.medium.com/. 2018. URL: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088.

[190] Ssd: Single shot multibox detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan [и др.] // European conference on computer vision / Springer. 2016. С. 21–37.

[191] Focal loss for dense object detection / Tsung-Yi Lin, Priya Goyal, Ross Girshick [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. С. 2980–2988.

[192] Real-Time Object Detection on COCO. URL: https://paperswithcode.com/sota/real-time-object-detection-on-coco.

[193] Goodfellow Ian. Nips 2016 tutorial: Generative adversarial networks // arXiv preprint arXiv:1701.00160. 2016.

[194] Kingma Diederik P, Welling Max. Auto-encoding variational bayes // arXiv preprint arXiv:1312.6114. 2013.

[195] Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey / Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray [и др.] // arXiv preprint arXiv:2101.00734. 2021.

[196] Asperti Andrea, Evangelista Davide, Piccolomini Elena Loli. A Survey on Variational Autoencoders from a Green AI Perspective // SN Computer Science. 2021. Т. 2, № 4. С. 1–23.

[197] A comprehensive survey and analysis of generative models in machine learning / GM Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey [и др.] // Computer Science Review. 2020. Т. 38. с. 100285.

[198] Wang Tianming, Wan Xiaojun. T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion. // IJCAI. 2019. С. 5233–5239.

[199] Gp-vae: Deep probabilistic time series imputation / Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch [и др.] // International conference on artificial intelligence and statistics / PMLR. 2020. С. 1651–1661.

[200] An uncertain future: Forecasting from static images using variational autoencoders / Jacob Walker, Carl Doersch, Abhinav Gupta [и др.] // European Conference on Computer Vision / Springer. 2016. С. 835–851.

[201] Mescheder Lars, Nowozin Sebastian, Geiger Andreas. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks // International Conference on Machine Learning / PMLR. 2017. С. 2391–2400.

[202] Generative adversarial nets / Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza [и др.] // Advances in neural information processing systems. 2014. Т. 27.

[203] A review on generative adversarial networks: Algorithms, theory, and applications / Jie Gui, Zhenan Sun, Yonggang Wen [и др.] // arXiv preprint arXiv:2001.06937. 2020.

[204] Manaswi Navin K. Generative Adversarial Networks with Industrial Use Cases: Learning How to Build GAN Applications for Retail, Healthcare, Telecom, Media, Education, and HRTech. BPB Publications, 2020.

[205] Jabbar Abdul, Li Xi, Omar Bourahla. A survey on generative adversarial networks: Variants, applications, and training // arXiv preprint arXiv:2006.05132. 2020.

[206] Antoniou Antreas, Storkey Amos, Edwards Harrison. Data augmentation generative adversarial networks // arXiv preprint arXiv:1711.04340. 2017.

[207] Generative adversarial text to image synthesis / Scott Reed, Zeynep Akata, Xinchen Yan [и др.] // International Conference on Machine Learning / PMLR. 2016. С. 1060–1069.

[208] He Xiaodong, Deng Li. Deep learning for image-to-text generation: A technical overview // IEEE Signal Processing Magazine. 2017. Т. 34, № 6. С. 109–116.

[209] Semantic image synthesis with spatially-adaptive normalization / Taesung Park, Ming-Yu Liu, Ting-Chun Wang [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. С. 2337–2346.

[210] AI-GAN: Attack-inspired generation of adversarial examples / Tao Bai, Jun Zhao, Jinlin Zhu [и др.] // arXiv preprint arXiv:2002.02196. 2020.

[211] Image transformer / Niki Parmar, Ashish Vaswani, Jakob Uszkoreit [и др.] // International Conference on Machine Learning / PMLR. 2018. С. 4055–4064.

[212] Vahdat Arash, Kreis Karsten, Kautz Jan. Score-based Generative Modeling in Latent Space // arXiv preprint arXiv:2106.05931. 2021.

[213] Autoencoding beyond pixels using a learned similarity metric / Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle [и др.] // International conference on machine learning / PMLR. 2016. С. 1558–1566.

[214] Dhariwal Prafulla, Nichol Alex. Diffusion models beat gans on image synthesis // arXiv preprint arXiv:2105.05233. 2021.

[215] Iqbal Touseef, Qureshi Shaima. The survey: Text generation models in deep learning // Journal of King Saud University-Computer and Information Sciences. 2020.

[216] Text Generation with Deep Variational GAN / Mahmoud Hossam, Trung Le, Michael Papasimeon [и др.] // arXiv preprint arXiv:2104.13488. 2021.

[217] Survey on deep learning in music using GAN / Rajat Kulkarni, Rutik Gaikwad, Rudraksh Sugandhi [и др.] // International Journal of Engineering Research & Technology. 2019. Т. 8, № 9. С. 646–648.

[218] Hartmann Kay Gregor, Schirrmeister Robin Tibor, Ball Tonio. EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals // arXiv preprint arXiv:1806.01875. 2018.

[219] High-performance large-scale image recognition without normalization / Andrew Brock, Soham De, Samuel L Smith [и др.] // arXiv preprint arXiv:2102.06171. 2021.

[220] Van Laarhoven Twan. L2 regularization versus batch and weight normalization // arXiv preprint arXiv:1706.05350. 2017.

[221] Random erasing data augmentation / Zhun Zhong, Liang Zheng, Guoliang Kang [и др.] // Proceedings of the AAAI Conference on Artificial Intelligence. Т. 34. 2020. С. 13001–13008.

[222] Brock Andrew, De Soham, Smith Samuel L. Characterizing signal propagation to close the performance gap in unnormalized ResNets // arXiv preprint arXiv:2101.08692. 2021.

[223] Big transfer (bit): General visual representation learning / Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai [и др.] // Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16 / Springer. 2020. С. 491–507.

[224] Müller Rafael, Kornblith Simon, Hinton Geoffrey. When does label smoothing help? // arXiv preprint arXiv:1906.02629. 2019.

[225] A Survey on Deep Semi-supervised Learning / Xiangli Yang, Zixing Song, Irwin King [и др.] // arXiv preprint arXiv:2103.00550. 2021.

[226] Self-supervised learning: Generative or contrastive / Xiao Liu, Fanjin Zhang, Zhenyu Hou [и др.] // IEEE Transactions on Knowledge and Data Engineering. 2021.

[227] mixup: Beyond empirical risk minimization / Hongyi Zhang, Moustapha Cisse, Yann N Dauphin [и др.] // arXiv preprint arXiv:1710.09412. 2017.

[228] Knowledge distillation: A survey / Jianping Gou, Baosheng Yu, Stephen J Maybank [и др.] // International Journal of Computer Vision. 2021. Т. 129, № 6. С. 1789–1819.

[229] Self-training with noisy student improves imagenet classification / Qizhe Xie, Minh-Thang Luong, Eduard Hovy [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. С. 10687–10698.

[230] Meta pseudo labels / Hieu Pham, Zihang Dai, Qizhe Xie [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. С. 11557–11568.

[231] Bottleneck transformers for visual recognition / Aravind Srinivas, Tsung-Yi Lin, Niki Parmar [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. С. 16519–16529.

[232] Do Vision Transformers See Like Convolutional Neural Networks? / Maithra Raghu, Thomas Unterthiner, Simon Kornblith [и др.] // arXiv preprint arXiv:2108.08810. 2021.

[233] Can Attention Enable MLPs To Catch Up With CNNs? / Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu [и др.] // arXiv preprint arXiv:2105.15078. 2021.

[234] Mlp-mixer: An all-mlp architecture for vision / Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov [и др.] // arXiv preprint arXiv:2105.01601. 2021.

[235] Learning transferable visual models from natural language supervision / Alec Radford, Jong Wook Kim, Chris Hallacy [и др.] // arXiv preprint arXiv:2103.00020. 2021.