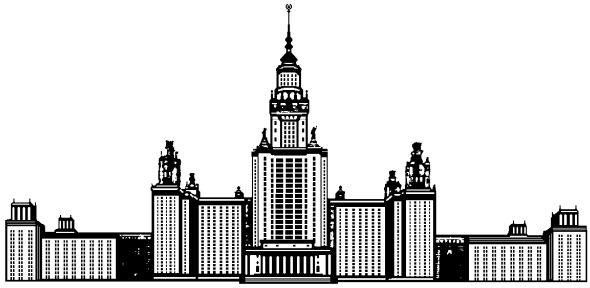

Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

ЭССЕ СТУДЕНТА 317 ГРУППЫ

«Капсульные сети»
«Capsule neural networks»

Работу выполнил
студент 317 группы
Махин Артем Александрович

Москва
2021

Содержание

Аннотация

Данный обзор написан по курсу лекций Александра Геннадьевича Дьяконова Deep Learning. Он будет полезен для читателей, знакомых с сверточными нейронными сетями, а так же интересующихся различными архитектурами нейронных сетей.

1 Введение

В нашем мире существует множество задач, связанных с изображениями: классификация, сегментация, детекция и так далее. Со многими из них научились справляться с помощью сверточных нейронных сетей, однако, несмотря на высокое качество в большинстве задач, сверточные нейронные сети имеют свои недостатки. Появились капсульные сети, похожие на сверточные нейронные сети и лишенные этих недостатков. Им и посвящена данная работа.

2 Сверточные нейронные сети

2.1 Описание сверточных нейронных сетей

Сверточные нейронные сети – это подкласс нейронных сетей, в основе которых присутствуют слои свертки (convolution layers). В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя с различными весами. В сверточном слое используется небольшая матрица весов, которая двигается по изображению. Для каждого канала имеется свой фильтр, ядро свёртки которого обрабатывает предыдущий слой по фрагментам, суммируя результаты поэлементного произведения для каждого фрагмента. Веса в каждом ядре являются обучаемыми параметрами.

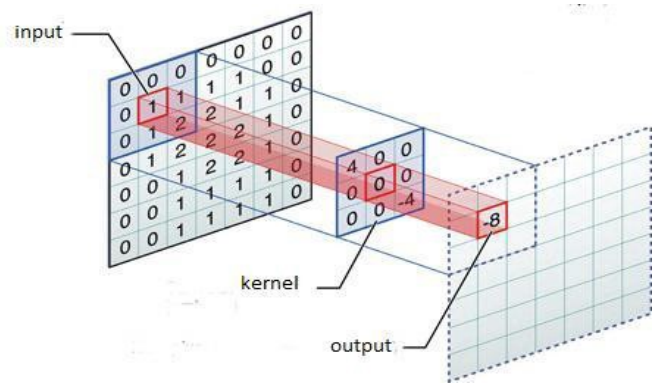

Рис. 1: Пример работы сверточного слоя. [1]

Интерпретация сверточных слоев заключается в поиске каких-либо паттернов на изображении, на каждом последующем слое паттерны все сложнее и сложнее. К примеру, на первых слоях нейрон будет иметь большое значение, если встретит что-то похожее на вертикальную линию, на последних – если встретит что-то похожее на образ машины.

Далее результат сверточного слоя проходит через нелинейность (к примеру, сигмоида), после чего обычно присутствует слой субдискретизации (пулинга). Он представляет собой слой нелинейного уплотнения карты признаков, наиболее часто используема при этом функция максимума (max–pooling). Берется множество непересекающихся квадратов, каждый из которых ужимается в одно значение: максимум из всех чисел в квадрате, что позволяет уменьшить признаковое пространство. Интерпретация слоя пулинга состоит в оставлении самого сильного сигнала признака из полученных из слоя свертки признаков.

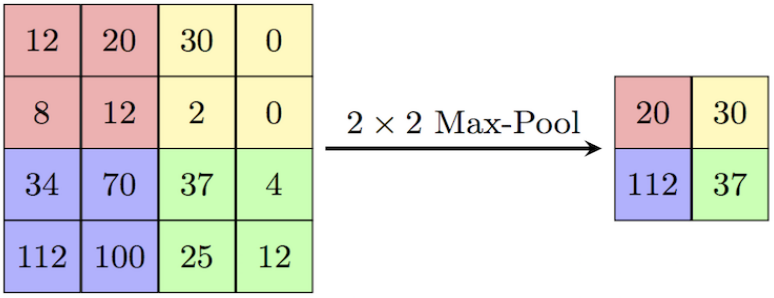

Рис. 2: Пример работы Max–Pooling размера 2x2. [2]

На данное время для многих задач сверточные нейронные сети показывают лучшие результаты. К примеру, классификация на датасете ImageNet [3]. В этом датасете содержится около 10 миллионов изображений с разными объектами, необходимо предсказать, какой именно объект присутствует на катринке. С помощью сверточных нейронных сетей удалось достичь более 0.9 показателя точности ответов [4].

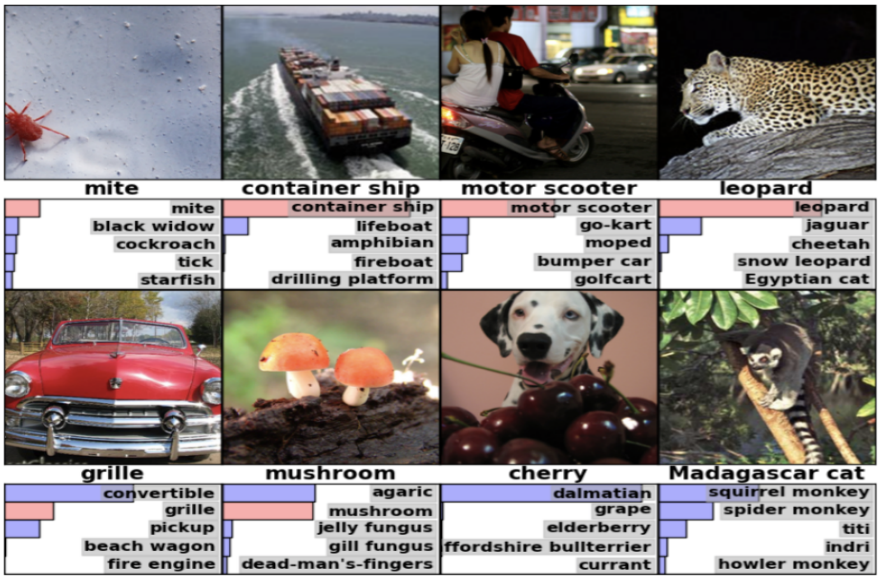

Рис. 3: Пример изображений из ImageNet. [5]

2.2 Проблемы сверточных сетей

Несмотря на хорошие результаты сверточных нейронных сетей во многих задачах, они имеют недостатки. Внутреннее представление данных сверточной нейронной сети не учитывает пространственной иерархии между простыми и сложными объектами. Также при применении max–pooling слоя теряется информация о расположении найденного паттерна (так как при выборе максимального значения не сохраняется положение, из которого был выбран этот максимум). Так, если на изображении в случайном порядке изображены глаза, нос и губы для свёрточной нейронной сети это явный признак наличия лица. А поворот объекта ухудшает качество распознавания, тогда, как человеческий мозг легко решает эту задачу (данную проблему можно решить, увеличив выборку путем аугментации данных (поворотов, изменения яркости, зеркальных отражений и так далее), однако количество данных для обучения очень сильно увеличится).

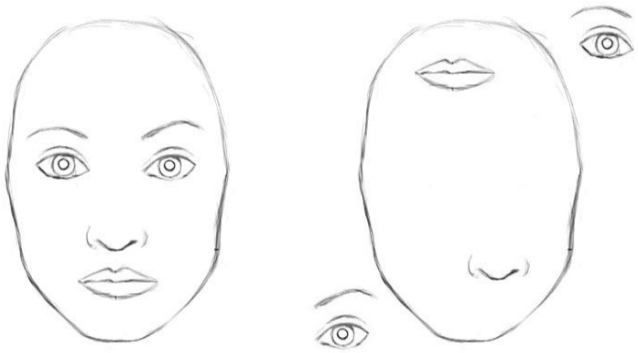

Рис. 4: Для сверточных нейронных сетей на обеих картинках изображено лицо. [6]

Автор статьи про капсульные сети [8] Джеффри Хинтон утверждает, что операция пулинга в сверточных нейронных сетях является большой ошибкой, факт того, что это хорошо работает, является катастрофой [7].

3 Капсульные нейронные сети

3.1 Идея

Для устранения вышеперечисленных недостатков сверточных нейронных сетей в 2017 году Джеффри Хинтоном в статье "Dynamic Routing Between Capsules" была предложена идея капсул и архитектура сети CapsNet [8].

Капсульные сети используют капсулы вместо обычных нейронов, которые на выходе дают скалярные значения. Капсулы же инкапсулируют всю важную информацию в векторе. Они кодируют вероятность обнаружения объекта как длину выходного вектора, а состояние обнаруженной функции кодируется как направление вектора (состояние может включать в себя точную позу, освещение, деформацию визуального объекта относительно неявно определенной канонической версии этого объекта и так далее). Поэтому, когда обнаруженная функция перемещается по изображению или состояние изображения изменяется, вероятность остается неизменной (длина вектора не изменяется), но ориентация вектора меняется. К примеру, на картинке изображена машина, а так же присутствует капсула, отвечающая за машины. Тогда при перемещении машины по картинке длина выходного вектора этой капсулы будет большой и одинаковой для всех перемещений, однако направление будет изменяться.

3.2 Разница нейрона и капсулы

Покажем разницу нейрона и капсулы [9]. Нейрон описывается 3 стадиями:

1. Скалярное взвешивание входных скаляров
2. Сумма взвешенных входных скаляров
3. Нелинейное скалярное преобразование

Капсула имеет векторные формы вышеуказанных 3 шагов, в дополнение к новому этапу аффинного преобразования ввода:

1. Матричное умножение входных векторов
2. Скалярное взвешивание входных векторов
3. Сумма взвешенных входных векторов
4. Векторная нелинейность

Опишем подробнее векторную нелинейность. В отличие от обычных нелинейностей, применяемых в сверточных нейронных сетях, векторная обрабатывает не числа отдельно и независимо друг от друга, а вектора, полученные от капсулы, при этом сохраняет их направление и приводит к шкале от 0 до 1:

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$$

В данной записи входной вектор обозначается s_j, а результат применения векторной нелинейности v_j. Правая часть уравнения (синий прямоугольник) масштабирует входной вектор так, что вектор будет иметь длину 1, а левая сторона (красный прямоугольник) выполняет дополнительное масштабирование в шкалу от 0 до 1.

3.3 Описание работы капсулы

Формально различия между нейроном и капсулой можно представить в таблице:

Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\widehat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\widehat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

Рис. 5: Разница между капсулой и нейроном.[6]

Рассмотрим подробнее все обозначения на примере детекции лица. На вход капсуле j, детектирующей лицо, поступают вектора u_i – выходы с предыдущих капсул, каждая из которых отвечает за свой конкретный признак. Пусть имеется 3 таких вектора от капсул, отвечающих за глаза, нос и рот. Эти вектора умножаются на соответствующие матрицы W_{ij}, которые кодируют важные отношения между низкоуровневыми признаками (в данном случае, глаза, нос и рот) и высокоуровневыми признаками (в данном случае, лицо). К примеру, матрица W_{2j} может кодировать отношения между носом и лицом: лицо сосредоточено вокруг носа, его размер в 10 раз больше размера носа, а его ориентация в пространстве соответствует ориентации носа, потому что все они лежат в одной плоскости. После перемножения мы получаем предсказанную позицию высокоуровневого признака ($\hat{u}_{j|2}$ выражает предсказанное расположение лица в зависимости от расположения носа). Интуитивно понятно, что если все предсказанные положения лиц примерно совпадают, то это лицо.

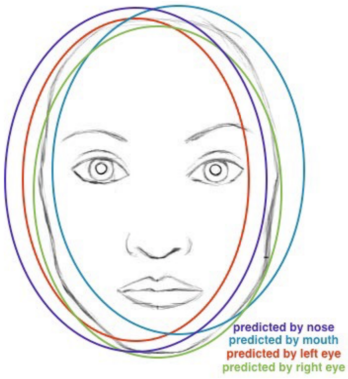

Рис. 6: Предсказанные положения лица. [6]

Если в обычном нейроне веса (w_i) – обучаемые параметры, то в капсуле они (c_{ij}) находятся путем динамической маршрутизации (dynamic routing) и удовлетворяют следующим свойствам:

1. Все веса являются неотрицательными скалярными величинами
2. Для каждой капсулы нижних уровней i: $\sum_j c_{ij} = 1$
3. Для каждой капсулы нижних уровней i количество весов c_{ij} совпадает с количеством капсул на следующем уровне
4. Веса определяются динамической маршрутизацией

Из свойств следует, что веса c_{ij} для каждой капсулы определяют распределение вероятностей принадлежности ее выхода к капсулам следующего уровня.

3.4 Динамическая маршрутизация

Главная интуиция динамической маршрутизации заключается в следующем: капсулы нижних уровней передают результат своей работы той капсуле следующего уровня, которая "согласуется"с выходом.

Псевдокод алгоритма приведен ниже.

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\mathbf{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

Рис. 7: Алгоритм динамической маршрутизации. [8]

На вход подаются все капсулы и их выходы на уровне l, а также число итераций r. Результатом работы является вектор v_j - выход капсулы $l+1$ уровня.

Во второй строке появляются коэффициенты b_{ij}. Это временные коэффициенты, которые будут итеративно обновляться, а в конце работы алгоритма значения будут подставлены в c_{ij}.

В третьей строке говорится, что содержимое 4-7 строк будет повторяться r раз. В четвертой строке вычисляются значения векторов c_i. Благодаря софтмаксу, все свойства весов будут выполнены. На первой итерации капсулы нижнего уровня не знают, какой капсуле следующего уровня передавать свой выход, поэтому все коэффициенты будут одинаковыми (равномерное распределение).

В пятой и шестой строках строятся выходы капсул $l+1$ уровня, как линейная комбинация с найденными коэффициентами с применением векторной нелинейности (squash), описанной выше.

В седьмой строке идет пересчет весовых коэффициентов. Новый вес b_{ij} являются суммой старого веса и скалярного произведения вектора выхода i-ой капсулы предыдущего уровня и подсчитанный выход нынешней капсулы v_j. Иными словами, веса изменяются на тем большее число, чем более похожи векторы выходов капсул.

После r итераций веса будут установлены, мы получим итоговый выход капсулы v_j. В статье [8] авторы заключили, что:

- Большее число итераций ведет к переобучению.
- Оптимальное количество итераций – 3.

3.5 Обучение

Заметим, что все этапы работы капсулы дифференцируемы. Это означает, что можно применить обычный градиентный спуск для обучения капсульной нейронной сети.

4 CapsNet

В статье [8] предложена архитектура капсульной нейронной сети для решения задачи классификации на наборе данных MNIST [10]. Набор представляет собой черно-белые изображения рукописных цифр, размер изображений 28x28.

Рис. 8: Пример изображений из набора MNIST.

CapsNet имеет 2 части: кодировщик и декодировщик.

4.1 Кодировщик

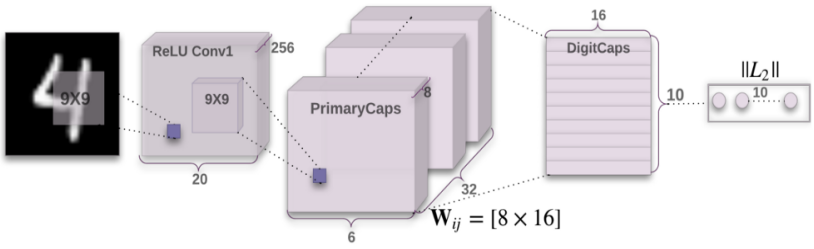

Рис. 9: Архитектура кодировщика CapsNet. [8]

На вход кодировщик получает изображения 28x28 из MNIST, на выходе десять векторов размерности 16, каждый из которых кодирует одну цифру. Выход всей сети в процессе предсказания является вектор размерности 10, составленный из длин выходных векторов кодировщика.

Первым слоем является слой обычной свертки, 256 ядер размера 9x9x1, затем применяется ReLU.

Второй слой (PrimaryCaps layer) состоит из 32 капсул, задача которых состоит в том, чтобы брать основные признаки, обнаруженные сверточным

слоем, и создавать комбинации этих признаков. Слой состоит из 32 «первичных капсул», которые очень похожи на сверточный слой. Каждая капсула применяет восемь сверточных ядер 9x9x256 (с шагом 2) к входному объему 20x20x256 и, следовательно, создает выходной тензор 6x6x8. Так как таких капсул 32, выходной объем имеет форму 6x6x8x32.

Третий слой (DigitCaps layer) имеет 10 капсул (по капсуле на каждое число), каждая капсула работает ровно так, как описано в разделе "Капсульные нейронные сети". После создается вектор, состоящий из длин выходных векторов капсул.

Функция потерь для данной архитектуры выбрана нестандартной.

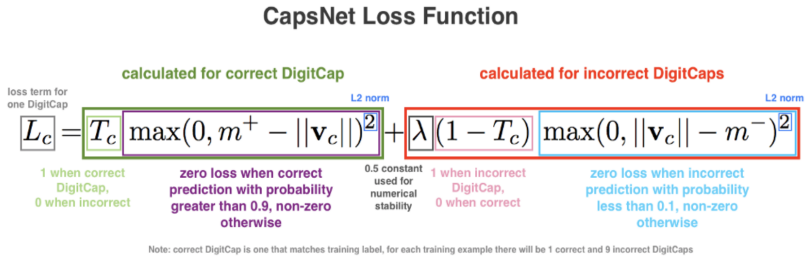

Рис. 10: Функция потерь CapsNet. [6]

На рисунке выше представлена формула потерь для 1 цифры, общие потери получаются как сумма всех L_c. В формуле T_c является 1, если рассматриваемая цифра является правильной, 0 иначе. Константа λ принимается равной 0.5; $m^+ = 0.9$, $m^- = 0.1$.

4.2 Декодировщик

До сих пор остается непонятным, зачем мы учимся предсказывать вектора, когда в качестве ответа берем лишь их длины. Для большего понимания изображений нейросетью авторы придумали дополнительную задачу, реализованную в декодировщике: "разворачивание" вектора в исходное изображение. Берется вектор из DigitCap капсулы, отвечающей за верную цифру, и учимся предсказывать по нему исходное изображение. Архитектура декодировщика приведена ниже.

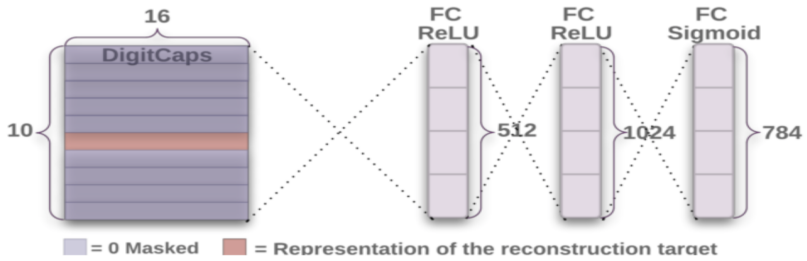

Рис. 11: Архитектура декодировщика CapsNet. [8]

Данная часть состоит из 3 полносвязных слоев и учится предсказывать 784 числа. В качестве функции потерь используется Евклидово расстояние между исходным и восстановленным изображением. Используется декодировщик для лучшего понимания изображений и регуляризации.

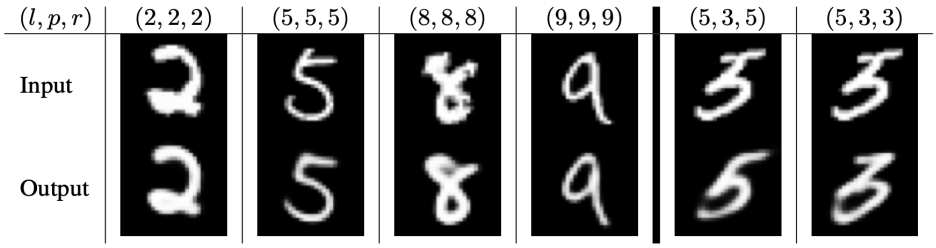

Рис. 12: Пример работы декодировщика. (l, r, p) – истинное число, предсказание и цель восстановления. [8]

5 Эксперименты

Проводились эксперименты над наборами данных MNIST [10] и MultiMNIST [11] (сгенерирован из MNIST, несколько цифр на одном изображении). Выбиралось различное количество итераций в динамической маршрутизации (Routing), а так же без декодировщика и с декодировщиком для регуляризации (Reconstruction). Baseline – сверточная нейронная сеть с 3 сверточными слоями (256, 256, 128 каналов), размер ядра 5x5, после 3 полносвязных слоя (размерностей 328, 192, 10). Такую сеть выбрали в качестве базового решения в связи с сравнимыми затратами на вычисления с предложенной капсульной сетью.

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34_{\pm 0.032}$	-
CapsNet	1	yes	$0.29_{\pm 0.011}$	7.5
CapsNet	3	no	$0.35_{\pm 0.036}$	-
CapsNet	3	yes	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

Рис. 13: Результаты работы. [8]

Можно видеть, что декодировщик значительно увеличивает качество, а так же 3 итерации динамической маршрутизации лучше, чем 1. На обоих наборах капсульная сеть показала себя значительно лучше обычной свертночной сети. Заметим, что на данных задачах достигается гораздо лучшее качество при помощи нескольких нейронных сетей, в данных экспериментах рассматривается только применение одной сети.

6 Выводы

В данной работе рассмотрена идея капсульных нейронных сетей, а так же конкретная архитектура CapsNet для задачи классификации на датасете MNIST. Показано, что данная сеть работает на этой задаче лучше сверточной нейронной сети с сравнимыми вычислительными затратами. Чтобы выяснить, будут ли капсульные сети работать лучше сверточных на более сложных задачах (к примеру, на наборе данных ImageNet), необходимо проводить эксперименты, которых на данный момент нет.

Список литературы

[1] Théodore Bluche, *Deep Neural Networks Applications in Handwriting Recognition.* — **http://technodocbox.com/3D_Graphics/70716176-Deep-neural-networks-applications-in-handwriting-recognition.html**

[2] Max–Pooling. — **https://computersciencewiki.org/index.php/Max-pooling_/_Pooling**

[3] ImageNet. — **https://www.image-net.org/update-mar-11-2021.php**

[4] Image Classification on ImageNet. — **https://paperswithcode.com/sota/image-classification-on-imagenet**

[5] Adrian Colyer, *ImageNet Classification with Deep Convolutional Neural Networks.* — **https://blog.acolyer.org/2016/04/20/imagenet-classification-with-deep-convolutional-neural-networks/**

[6] Max Pechyonkin, *Understanding Hinton's Capsule Networks.* — **https://medium.com/ai-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b**

[7] Высказывание Джеффри Хинтона. — **https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/**

[8] Geoffrey E. Hinton, *"Dynamic Routing Between Capsules"*, arXiv preprint arXiv:1710.09829, 2017 — **https://arxiv.org/pdf/1710.09829.pdf**

[9] Капсульные нейронные сети. — **http://data4.ru/kapsulnn**

[10] Набор MNIST. — **http://yann.lecun.com/exdb/mnist/**

[11] Набор MultiMNIST. — **https://paperswithcode.com/dataset/multimnist**