Приложение № 2
к распоряжению РТУ МИРЭА
от__ _____№_____

Титульный лист материалов по дисциплине
(заполняется по каждому виду учебного материала)

ДИСЦИЛИНА Проектирование и обучение нейронных сетей
(полное наименование дисциплины без сокращений)

ИНСТИТУТ Информационые технологии

КАФЕДРА Вычислительная техника
полное наименование кафедры

ВИД УЧЕБНОГО МАТЕРИАЛА Лекция
(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ Сорокин Алексей Борисович
(фамилия, имя, отчество)

СЕМЕСТР 7 семестр, 2023/2024
(указать семестр обучения, учебный год)

1. ЛЕКЦИЯ. КАПСУЛЬНАЯ НЕЙРОННАЯ СЕТЬ

В 2017 году Джеффри Хинтон (один из основоположников подхода обратного распространения ошибки) опубликовал статью, в которой описал капсульные нейронные сети и предложил алгоритм динамической маршрутизации между капсулами для обучения предложенной архитектуры. По словам автора, особенности данной архитектуры направлены на решение проблем одного из самых применяющихся к данной задачи методов – сверточных нейронных сетей. Как Вы уже знаете, данная сеть обычно состоит из множества чередующихся слоев трех основных видов: сверточного, субдискретизирующего (пулинга) и полносвязного

Хинтон указывает на три основных недостатка архитектуры сверточных нейронных сетей:

- Сверточные сети в своей структуре имеют слишком малое количество уровней: из нейронов складываются слои, из слоев – сеть. Необходимо, некоторым образом, объединить нейроны в группы на каждом слое так, чтобы появилась возможность производить внутренние вычисления и выдавать на выходе компактный результат;
- По мере продвижения от первых слоев сверток, где выделяются локальные признаки исходного изображения, к более глубоким слоям признаки превращаются в так называемые домены признаков, что является одним из главных достоинств сверточных нейронных сетей. При этом критике подвергается только слой пулинга: на каждом таком слое забывается информация о локации выделенного признака, что плохо вписывается в механизм восприятия формы человеческого мозга. Исчезают пространственные связи между объектами или их частями;
- Наличие слоев пулинга также приводит к небольшой трасляционной инвариантности, которая по мнению Хинтона является недостатком сети, а не достоинством. Проблема заключается в неспособности сети определять положение объекта в пространстве, а также реагировать на его изменения (такие как поворот или смещение). Для решения данного вопроса часто применяется аугментация данных – расширение обучающей выборки, заключающееся в создании дополнительных данных из уже имеющихся путем различных преобразований: поворота, отражения, масштабирования, изменения цвета. Это приводит к увеличению входного набора, а значит и длительности процесса обучения. Вместо инвариантности автор советует прийти к эквивариантности, т.е.

пониманию свойств объекта таким образом, что при их изменении меняется соответственно и результат представления сети о данном объекте.

Инвариант является свойством объекта, которое не изменяется в результате некоторого преобразования. Например, площадь круга не изменится, если круг сдвинуть влево.

Неформально эквивариантность - это свойство, которое предсказуемо изменяется при преобразовании. Например, при смещении центр круга перемещается на ту же величину, что и круг. Неэквивариантность - это свойство, значение которого не изменяется предсказуемо при преобразовании. Например, преобразование круга в эллипс означает, что его периметр больше не может быть вычислен как π, умноженное на диаметр. В компьютерном зрении ожидается, что класс объекта будет инвариантом для многих преобразований. То есть кошка остается кошкой, если ее сдвинуть, перевернуть вверх ногами или уменьшить в размерах. Однако многие другие свойства вместо этого эквивалентны. Объем кота меняется при масштабировании.

Эквивариантные свойства, такие как пространственная взаимосвязь фиксируются в позе, данные, описывающий объект перевод, вращение , масштаб и отражение. Перевод - это изменение местоположения в одном или нескольких измерениях. Вращение - это изменение ориентации. Масштаб - это изменение размера. Отражение - это зеркальное отображение. Неконтролируемые сети изучают глобальное линейное многообразие между объектом и его позицией в виде матрицы весов. Другими словами, капсульные нейронные сети могут идентифицировать объект независимо от его позы, вместо того, чтобы учиться распознавать объект, включая его пространственные отношения как часть объекта. В сетке поза может включать свойства, отличные от пространственных отношений, например, цвет (кошки могут быть разных цветов).

Умножение объекта на многообразие создает объект (для объекта в пространстве).

Предположим, что на изображении присутствуют два одинаковых признака. В результате операции свертки, на выходе в соответствующих местах следует ожидать схожие значения. Однако, при использовании слоя глобальной агрегации средних значений (англ. global average pooling), в пределах окна фиксированного размера или полносвязных слоев в конце архитектуры, теряется эквивариантность в пользу инвариантности признаков. Ко всему прочему, после прохождения через слои субдискретизации, теряется доля информации о нахождении этих признаков на изображении. Причина данной проблемы заключается в механизме слоев субдискретизации (рис. 1).

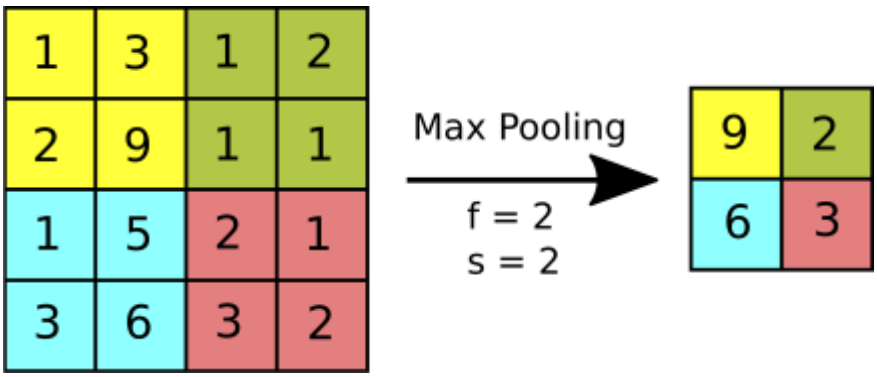

Рис. 1. Принцип работы слоя субдискретизации максимальных значений.

Если у объектов на двух изображениях признаки будут отличаться поворотом или размером, то, в результате, два объекта могут быть не распознаны как объекты одного класса (рис. 2), если не применить аугментацию данных.

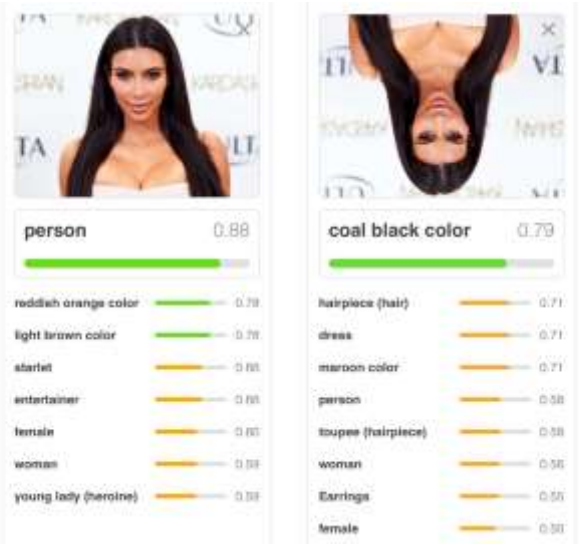

Рис. 2. Визуализация проблемы с классификацией двух изображений

И, напротив, сверточная сеть может воспринимать изображение, содержащее разбросанные признаки объекта, как сам объект. Визуализация данной проблемы представлена на рис. 3.

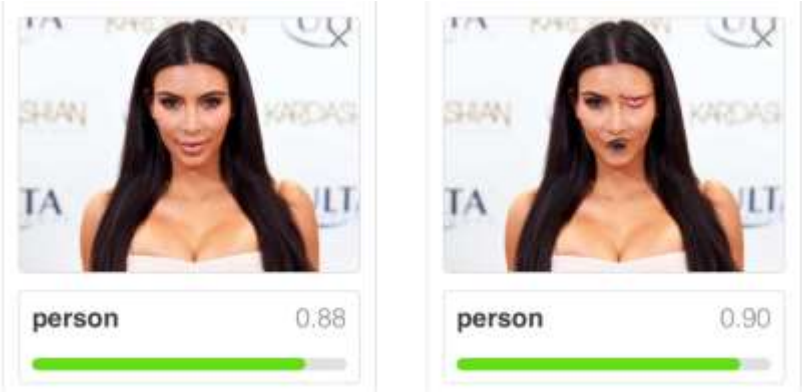

Рис. 3. Визуализация проблемы с классификацией набора признаков как объект

Таким образом, capsule Neural Network (CapsNet) представляет собой систему машинного обучения, которая является одним из видов искусственной

нейронной сети (ИНС), которые могут быть использованы для более модели иерархических отношений. Подход представляет собой попытку более точно имитировать биологическую нейронную организацию. Идея состоит в том, чтобы добавить структуры, называемые «капсулами», в сверточную нейронную сеть (CNN) и повторно использовать выходные данные нескольких из этих капсул для формирования более стабильных (по отношению к различным возмущениям) представлений для более высоких капсул. Результатом является вектор, состоящий из вероятности наблюдения и позы для этого наблюдения. Этот вектор аналогичен тому, что делается, например, при классификации с локализацией в CNN.

Капсула - это набор нейронов, которые индивидуально активируются для различных свойств типа объекта, таких как положение, размер и оттенок. Формально капсула - это набор нейронов, которые вместе создают вектор активности с одним элементом для каждого нейрона, чтобы удерживать значение экземпляра этого нейрона (например, оттенок). Графические программы используют значение создания экземпляра для рисования объекта. CapsNets пытается извлечь их из своих входных данных. Вероятность присутствия объекта на конкретном входе - это длина вектора, а ориентация вектора количественно определяет свойства капсулы.

Искусственные нейроны традиционно выводят скалярную активацию с действительным знаком, которая в общих чертах представляет вероятность наблюдения. Capsnets заменяют детекторы функций скалярного вывода капсулами с векторным выводом, а max-pooling - маршрутизацией по соглашению.

Поскольку капсулы независимы, когда несколько капсул совпадают, вероятность правильного обнаружения намного выше. Минимальный кластер из двух капсул с учетом шестимерного объекта согласуется с точностью до 10% случайно только один раз из миллиона испытаний. По мере увеличения количества измерений вероятность случайного совпадения в более крупном кластере с более высокими измерениями экспоненциально уменьшается.

Капсулы на более высоких уровнях принимают выходные данные от капсул на более низких уровнях и принимают те, выходы которых группируются. Кластер заставляет высшую капсулу выводить высокую вероятность наблюдения присутствия сущности.

Таким образом, предложенный Хинтоном новый вид сети старается исправить эти недостатки следующим образом:

1. Замена нейронов капсулами. Конструкция капсулы строится на устройстве искусственного нейрона, но расширяет его до векторной формы, чтобы обеспечить более мощные репрезентативные возможности. Также вводятся весовые коэффициенты матрицы для кодирования иерархических связей между особенностями разных слоев. Достигается эквивариантность нейронной активности в отношении изменений входных данных и инвариантности в вероятностях обнаружения признаков. «Выход нейрона – вектор, способный передать позу объекта».

2. Замена слоя пулинга на алгоритм маршрутизации по соглашению. В результате происходит не простой выбор максимальных значений, а инкапсуляция ценной информации в векторе выхода.

Базовая капсульная архитектура

Базовая капсульная архитектура для решения задачи классификации CapsNet была предложена в 2017 состояла из двух сверточных и одного полносвязного капсульного слоя (рис. 4).

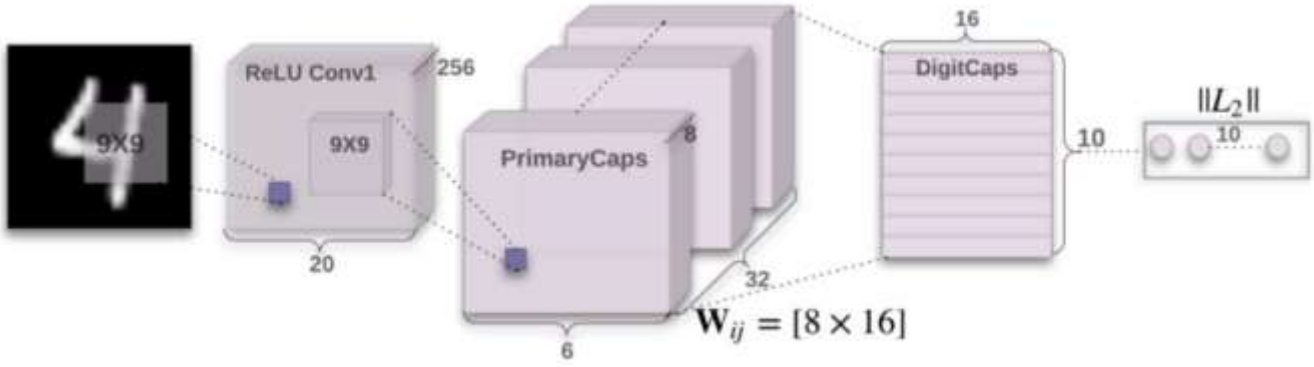

Рис. 4. Архитектура CapsNet

Для понимания работы сети CapsNet введем пример входного изображения (рис. 5).

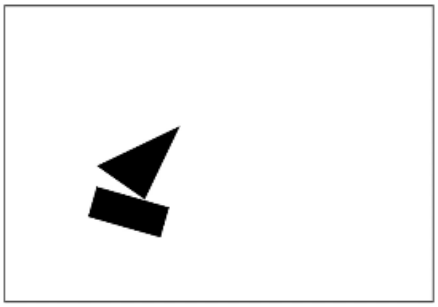

Рис. 5. Пример входного изображения

В первом слое (ReLU Conv1) на входном изображении применялась операция свертки с размером ядра 9 × 9 и 256 каналами. Шаг окна равнялся 1 пикселю. После чего применялась нелинейная функция активации ReLU

Второй слой (PrimaryCaps) обладает теми же параметрами, что и первый, за исключением смещения окна, которое сдвигалось на два пикселя. Группы капсул образовывались путем деления 256 каналов на 32 части по 8 значений в каждой. Таким образом, для изображения с одним цветовым каналом и размером 28 пикселей количество капсул равняется 1152. Для каждой капсулы применяется нелинейная функция активации, которую авторы назвали функцией сжатия (англ. squash function). В ней все векторы, которые имеют короткую длину, «сжимают» к длине, близкой к нулю. А векторы с большой длиной до длины, близкой к единице:

$$\bar{v}_j = \frac{||\bar{s}_j||^2}{1 + ||\bar{s}_j||^2} \frac{\bar{s}_j}{||\bar{s}_j||} \quad (1)$$

где s_j – входное векторное значение капсулы j.

Правая часть уравнения (синий прямоугольник) масштабирует входной вектор так, что вектор будет иметь длину блока, а левая сторона (красный прямоугольник) выполняет дополнительное масштабирование. Суть функции в том, что длина вектора – это уверенность нейронной сети в существовании признака. Это позволяет акцентировать внимание только на существующих признаках объекта. Единица – это максимальное значение длины вектора и полная уверенность в существовании признака. В примере получено две капсулы с наибольшей величиной: признака треугольника и прямоугольника (рис. 6).

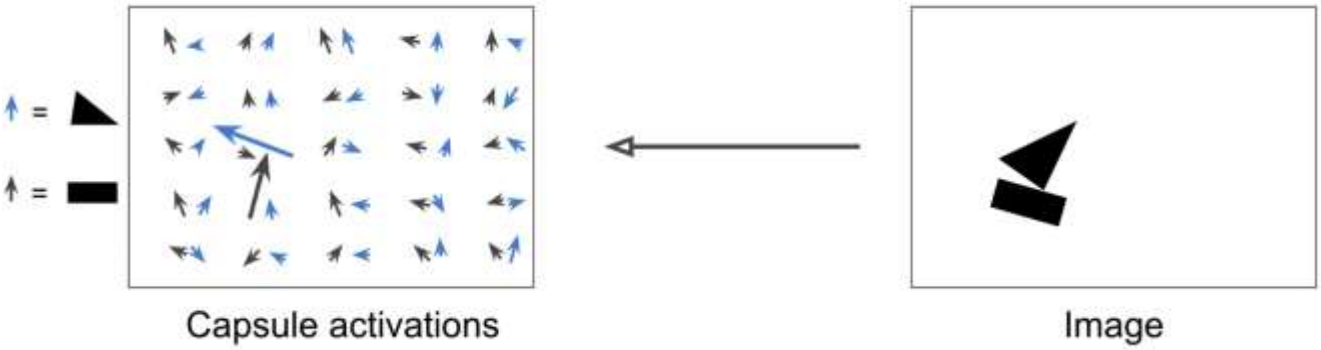

Рис. 6. Длина капсул отражает уверенность в существовании признака объекта

В последнем, полносвязном слое (DigitCaps) происходят процедуры согласованной и динамической маршрутизации (англ. routing by agreement and dynamic routing). Выходное значение $\hat{u}_j$ слоя DigitCaps – это произведение всех векторов, полученных на предыдущем слое i, путем умножения значений на матрицу весов W_{ij}:

$$\hat{u}_j = \sum_i W_{ij} \bar{u}_i \quad (2)$$

Предположим, в тестовом примере сеть определяет два класса: дома и лодки. В таком случае слой DigitCaps будет состоять из двух капсул, каждая из которых примет выход всех значений капсул: прямоугольник и треугольник из слоя PrimaryCaps. После применения формул (2) и (1), длина векторного выхода капсулы класса «лодка» будет ближе к единице, а капсулы класса «дом» – к нулю (рис. 7). Данную процедуру авторы назвали процедурой согласованной маршрутизации.

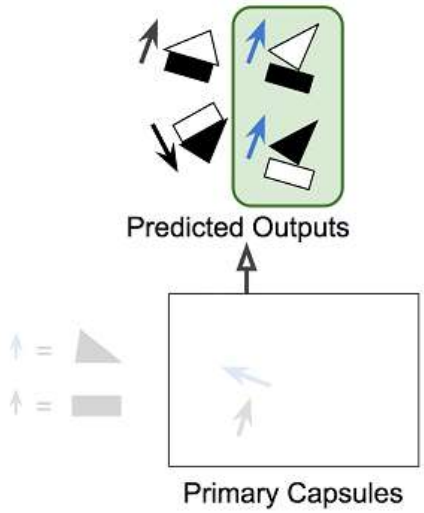

Рис. 7. Принцип работы согласованной маршрутизации

Далее начинается процедура динамической маршрутизации. В начале алгоритма входные векторные значения капсул представляются в качестве элементов векторного пространства с равным весом и центром масс (рис. 8).

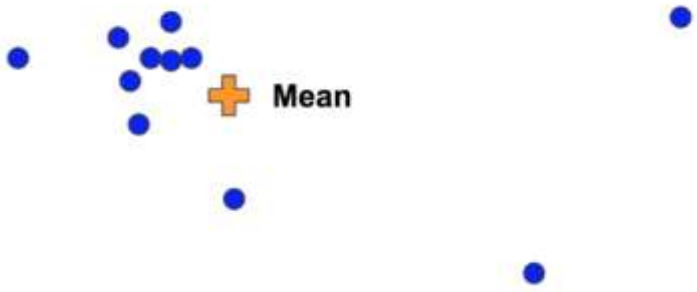

Рис. 8. Пример в двумерном пространстве

На основе c_{ij} – коэффициентов, полученных при помощи функции маршрутизированного мягкого максимума (англ. routing softmax function) и вычисляемых с помощью (3), возможно получить $\bar{s}_j$, который является взвешенной суммой всех векторов $\hat{u}_j$ (4).

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{3}$$

где b_{ij} – нулевой тензор, но в процессе итераций динамической маршрутизации принимающий значение $b_{ij} = b_{ij} + \hat{u}_j v_j$.

$$\bar{s}_j = \sum_i c_{ij} \hat{u}_j \tag{4}$$

Из-за различия коэффициентов веса, элементы векторного пространства будут обладать другой точкой центра масс (рис. 9). Данную процедуру можно повторить множество раз. Однако, стоит учесть, что это вычислительно сложная операция. Выходом слоя DigitCaps являются вектора $\bar{v}_j$ после r итераций.

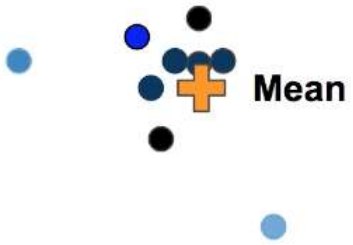

Рис. 9. Пример в двумерном пространстве

Принятие решения о классе объекта происходит путем сравнения размеров выходов капсул по L_2 норме. Предсказанный класс обладает наибольшим значением.

Чтобы избежать переобучения сети, которое возникает сравнительно быстро, необходим метод регуляризации для капсульных сетей – декодер (рис. 10). Он состоит из двух полносвязных слоев с функцией активации ReLU и одного полносвязного слоя с сигмоидальной функцией активацией. На вход подается матрица, полученная из последнего слоя архитектуры. Входные значения матрицы капсул приравниваются к нулю, кроме одной строки, которая имеет наибольшую L_2 норму. Выходом декодера является вектор изображения, эквивалентный подаваемому на вход в архитектуре CapsNet. Полученный вектор подается в функцию потерь декодера, которая может найти разницу между полученным и исходным изображением. Например, это может быть функция среднеквадратической ошибки.

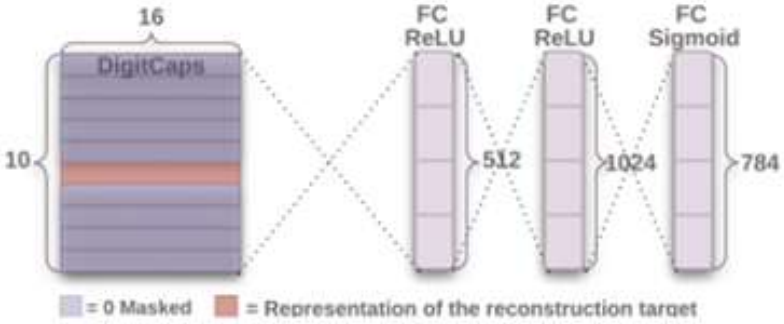

Рис. 10. Декодер CapsNet

Для обучения сети требуется функция ошибки, которая будет максимизировать входные векторные значения капсулы верного класса, минимизируя остальные:

$$L_k = T_k max(0, m^+ - ||v_k||)^2 + \lambda(1 - T_k)\, max(0, ||v_k|| - m^-)^2 \qquad (5)$$

где:

– T_k – компонента унитарного кода, которая равна единице, если k–верный номер класса;

– m^+ – константа длины для входного значения капсулы верного класса. Рекомендуемое для неё значение $= 0.9$;

– $||v_k||$ – L_2 норма выходной капсулы k;

– m^- – константа длины для входных значений капсул других классов. Рекомендуемое для неё значение $= 0.1$;

– λ – коэффициент для предотвращения сокращения длин векторных значений всех капсул на начальных этапах обучения. Рекомендуемое значение $= 0.5$.

Итоговое значение функции потерь равна сумме ошибок для всех входных значений капсул и функции потерь декодера.

Глубокие капсульные нейронные сети (DeepCaps)

Из-за неглубокой архитектуры CapsNet нельзя получить хорошие показания точности на наборе данных сложнее чем MNIST (объёмная база данных образцов рукописного написания цифр), но добавить больше слоев в изначальную сеть или увеличить количество итераций в динамическую маршрутизацию не представляется возможным по двум причинам:

1) из-за полносвязной архитектуры с увеличением количества капсул возникает ситуация затухания градиента;

2) вычислительная сложность операции динамической маршрутизации, которая существенно увеличит время и необходимые мощности для обучения и использования сети.

Чтобы обойти данные ограничения, была предложена глубокая капсульная архитектура с независящим от класса декодером.

Для описания работы данной нейросети потребуется ввести обозначения:

– w^l – ширина и высота карты признаков слоя l,

– n^l – размер капсулы слоя l,

– c^l – количество капсул слоя l,

– Φ^l – тензор слоя l

В DeepCaps содержится 15 сверточных слоев – ConvCaps, которые были изменены для работы с капсулами. Данные слои принимают на вход четырхмерный тензор вида $\Phi^l \in \mathrm{R}^{(w^l, w^l, c^l, n^l)}$ и применяют операцию свертки на преобразованном тензоре $\Psi^l \in \mathrm{R}^{(w^l, w^l, c^l \times n^l)}$. Далее полученный тензор разбивается на c^{l+1} частей размера n^{l+1} и применяется измененная капсульная функция (формула 1) активации сквош 3D:

$$\hat{s}_{pqr} = \frac{\left\|\bar{s}_{pqr}\right\|^2}{1+\left\|\bar{s}_{pqr}\right\|^2} \times \frac{\bar{s}_{pqr}}{\left\|\bar{s}_{pqr}\right\|}$$

В слое, который авторы назвали ConvCaps3D, используется функция 3D свертки и 3 итерации пространственной динамической маршрутизации (рис. 11).

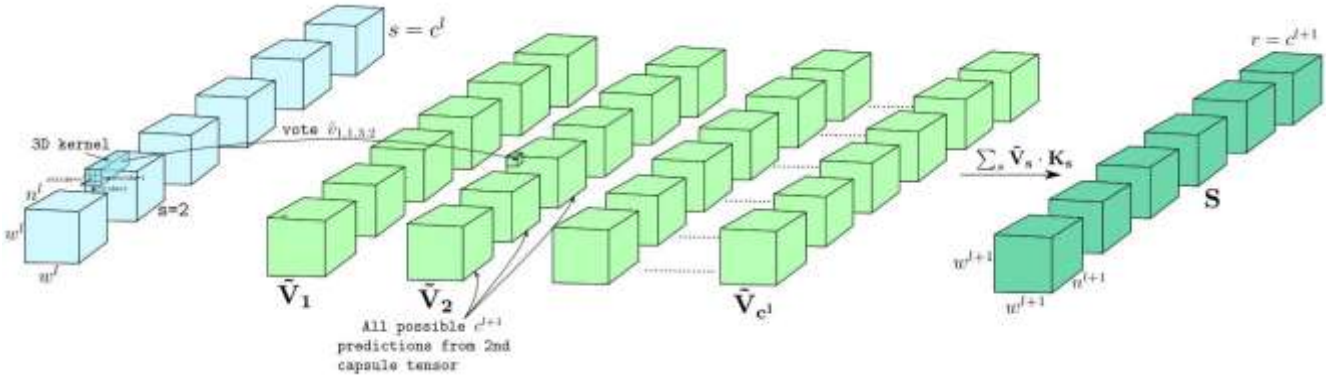

Рис. 11. Схема работы слоя ConvCaps3D

Для его реализации, требуется улучшить формулу (3) для работы с трехмерным пространством:

$$k_{pqrs} = \frac{exp(b_{pqrs})}{\sum_x \sum_y \sum_z exp(b_{xyzs})} \quad (7)$$

где $p, q \in w^{l+1}, r \in c^{l+1}, s \in c^l$.

Алгоритм слоя ConvCaps3D получает на вход тензор $\Phi^l \in R(w^l, w^l, c^l, n^l)$, количество итераций i, c^{l+1} – количество капсул слоя $l+1$ и n^{l+1} – размер выходного вектора капсулы слоя $l+1$. При помощи операции изменения размерности тензора, $\Phi^l \in R(w^l, w^l, c^l, n^l)$ преобразуется в тензор $\widehat{\Phi}^l \in R(w^l, w^l, c^l, n^l)$ к которому применяется трехмерная операция свертки, в результате применения которой получаем тензор $V \in R(w^{l+1}, w^{l+1} c^l c^{l+1} \times n^{l+1})$. Далее полученный тензор проходит через операцию изменения размерности тензора, выходом которой является тензор $\hat{V} \in R(w^{l+1}, w^{l+1}, n^{l+1}, c^{l+1}, c^l)$. Следующим шагом является инициализация нулевого тензора $B \in R(w^{l+1}, w^{l+1}, c^{l+1}, c^l)$. Пусть $p, q \in w^{l+1}, r \in c^{l+1}, s \in c^l$, тогда для i итераций:

1) для всех p, q, r в тензоре B применяется формула (7) для вычисления коэффициентов k_{pqrs};

2) получение взвешенного значения капсулы: $S_{\mathrm{pqr}} = \sum_s k_{\mathrm{kqrs}} \hat{V}_{\mathrm{pqrs}}$;

3) применение трехмерной функции сжатия (6) на взвешенные значения капсул S_{pqr} для получения нормированных значений капсул $\hat{S}_{\mathrm{pqr}}$;

4) обновление коэффициентов: $b_{pqrs} = b_{pqrs} + \hat{S}_{pqr} \cdot \hat{V}_{pqrs}$.

Выходом слоя ConvCaps3D является тензор $\hat{S}_{pqr}$ после i итераций.

Слой FlatCaps изменяет размерность тензора на двумерную матрицу размера $(w_l \times w_l \times c_l, n_l)$.

Работа слоя FCCaps похожа на работу обычного полносвязного слоя. Каждый выход капсул умножается на матрицу весов $W_{ij} \in R^{(n^l, n^{l+1})}$.

1.3.2. Архитектура сети

Архитектура сети представлена на рисунке 12.

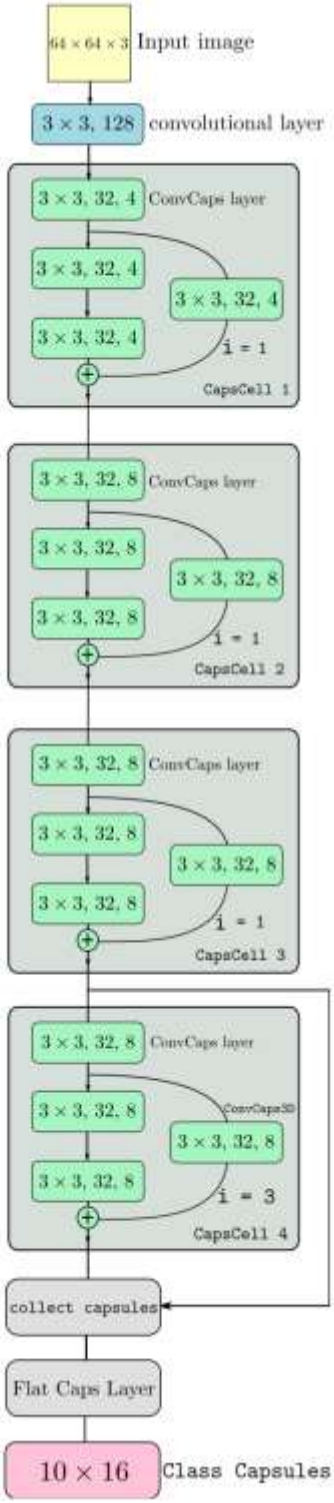

Рис. 12. Модель DeepCaps

Первые три блока сети состоят из четырех слоев ConvCaps. Один из слоев в каждом блоке реализован в остаточном соединении. Третий блок соединен непосредственно со слоем сбора капсул и конкатенируется с последним блоком в слое сбора капсул. Четвертый блок использует слой ConvCaps3D в остаточном соединении с тремя итерациями пространственной динамической маршрутизации. Следующий шаг – конкатенация выхода капсул с третьего и последнего блока, предварительно преобразованного в “плоский” вид (слой ”collect capsules”). Далее, после сбора капсул слоем ”collect capsules”, они попадают в полносвязный капсульный слой Flat Caps Layer. Полученные значения капсул можно задействовать в декодере для регуляризации сети. Функция потерь совпадает с функцией ошибки CapsNet (формула 5)

CapsNets отвергают стратегию уровня объединения обычных CNN, которая уменьшает количество деталей, которые должны обрабатываться на следующем более высоком уровне. Объединение в пул обеспечивает определенную степень трансляционной инвариантности (он может распознавать один и тот же объект в несколько другом месте) и позволяет представлять большее количество типов функций. Сторонники Capsnet утверждают, что объединение:

- нарушает восприятие биологической формы, поскольку не имеет собственной системы координат;
- обеспечивает инвариантность (отбрасывание позиционной информации) вместо эквивалентности (распутывание этой информации);
- игнорирует линейное многообразие, лежащее в основе многих вариаций изображений;
- направляет статически вместо того, чтобы сообщать потенциальную «находку» функции, которая может ее оценить;
- повреждает расположенные поблизости детекторы объектов, удаляя информацию, на которую они полагаются.