Министерство образования Республики Беларусь
Учреждение образования
«Полоцкий государственный университет»

ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ: ДОСТИЖЕНИЯ, ПРОБЛЕМЫ, ИННОВАЦИИ (ИКТ-2018)

Электронный сборник статей
I Международной научно-практической конференции,
посвященной 50-летию Полоцкого государственного университета

(Новополоцк, 14–15 июня 2018 г.)

Новополоцк
Полоцкий государственный университет
2018

Информационно-коммуникационные технологии: достижения, проблемы, инновации (ИКТ-2018) [Электронный ресурс] : электронный сборник статей I международной научно-практической конференции, посвященной 50-летию Полоцкого государственного университета, Новополоцк, 14–15 июня 2018 г. / Полоцкий государственный университет. – Новополоцк, 2018. – 1 электрон. опт. диск (CD-ROM).

Представлены результаты новейших научных исследований, в области информационно-коммуникационных и интернет-технологий, а именно: методы и технологии математического и имитационного моделирования систем; автоматизация и управление производственными процессами; программная инженерия; тестирование и верификация программ; обработка сигналов, изображений и видео; защита информации и технологии информационной безопасности; электронный маркетинг; проблемы и инновационные технологии подготовки специалистов в данной области.

Сборник включен в Государственный регистр информационного ресурса. Регистрационное свидетельство № 3201815009 от 28.03.2018.

Компьютерный дизайн М. Э. Дистанова.

Технические редакторы: Т. А. Дарьянова, О. П. Михайлова.

Компьютерная верстка Д. М. Севастьяновой.

211440, ул. Блохина, 29, г. Новополоцк, Беларусь
тел. 8 (0214) 53-21-23, e-mail: irina.psu@gmail.com

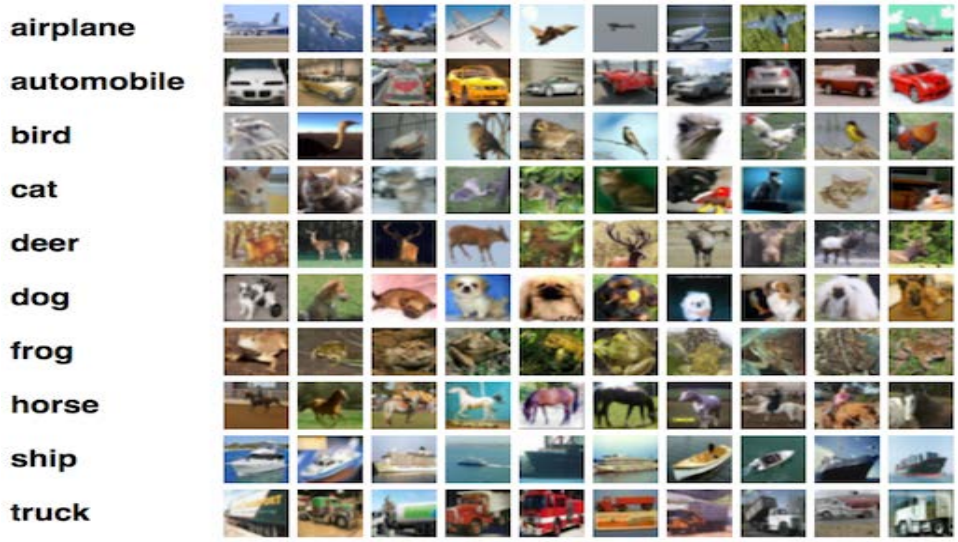
ISBN 978-985-531-602-3

УДК 004

РАСПОЗНАНИЕ ИЗОБРАЖЕНИЯ С ПОМОЩЬЮ СВЕРТОЧНЫХ СЕТЕЙ

канд. техн. наук А.В. ХИЖНЯК, лейтенант М.Д. КУЗЬМЕНОК
(Военная академия Республики Беларусь, Минск)

Изучая технологии *глубокого обучения нейронных сетей*, оценим производительность на наборе CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).



Рассматриваемый нами многослойный перцептрон представляет собой самую мощную и возможных нейронных сетей прямого распространения. Он состоит из нескольких слоев, где каждый слой организован таким образом, что каждый нейрон в одном слое получает свою копию *всех* выходных данных предыдущего слоя. Эта модель идеально подходит для определенных типов задач, например, обучение на ограниченном количество более или менее неструктурированных параметров.

Тем не менее, посмотрим, что происходит с количеством параметров (весов) в такой модели, когда ей на вход поступают необработанные данные. Например, CIFAR-10 содержит 32 x 32 x 3 цветных изображений, и если мы будем считать каждый канал каждого пикселя независимым входным параметром для MLP, каждый нейрон в первом скрытом слое добавляет к модели около 3000 новых параметров! И с ростом размера изображений ситуация быстро выходит из-под контроля, причем происходит это намного раньше, чем изображения достигают того размера, с которыми обычно работают пользователи реальных приложений.

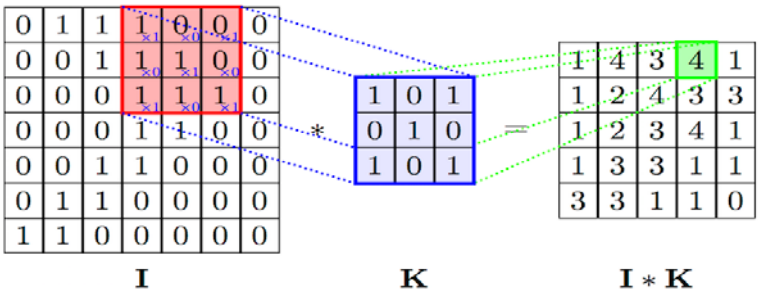
Одно из популярных решений – *понижать разрешение* изображений до той степени, когда MLP становится применим. Тем не менее, когда мы просто понижаем разрешение, мы рискуем потерять большое количество информации, и было бы здорово, если бы можно было осуществлять полезную первичную обработку информации еще до применения понижения качества, не вызывая при этом взрывного роста количества параметров модели.

Оказывается, существует весьма эффективный способ решения этой задачи, который обращает в нашу пользу саму структуру изображения: предполагается, что пиксели, находящиеся *близко* друг к другу, теснее “взаимодействуют” при формировании интересующего нас признака, чем пиксели, расположенные в противоположных углах. Кроме того, если в процессе классификации изображения небольшая черта считается очень важной, не будет иметь значения, на каком участке изображения эта черта обнаружена.

Введем понятие **оператора свертки**. Имея двумерное изображение **I** и небольшую матрицу **K** размерности (так называемое ядро свертки), построенная таким образом, что графически кодирует какой-либо признак, мы вычисляем свернутое изображение **I * K**, накладывая ядро на изображение всеми возможными способами и записывая сумму произведений элементов исходного изображения и ядра:

На самом деле, точное определение предполагает, что матрица ядра будет транспонирована, но для задач машинного обучения не важно, выполнялась эта операция или нет.

На рисунках ниже схематически изображена вышеуказанная формула.



Оператор свертки составляет основу сверточного слоя (convolutional layer). Слой состоит из определенного количества ядер(с аддитивными составляющими смещения для каждого ядра) и вычисляет свертку выходного изображения предыдущего слоя с помощью каждого из ядер, каждый раз прибавляя составляющую смещения. В конце концов ко всему выходному изображению может быть применена функция активации. Обычно входной поток для сверточного слоя состоит из d каналов, например, red/green/blue для входного слоя, и в этом случае ядра тоже расширяют таким образом, чтобы они также состояли из *d каналов*; получается следующая формула для одного канала выходного изображения сверточного слоя, где **K –** ядро, а *b* — составляющая смещения:

Обратите внимание, что так как все, что мы здесь делаем — это сложение и масштабирование входных пикселей, ядра можно получить из имеющейся обучающей выборки методом *градиентного спуска*, аналогично вычислению весов в многослойном перцептроне (MLP). На самом деле MLP мог бы в совершенстве справиться с функциями сверточного слоя, но времени на обучение (как и обучающих данных) потребовалось бы намного больше.

Заметим также, что оператор свертки вовсе не ограничен двухмерными данными: большинство фреймворков глубокого обучения предоставляют слои для одномерной или трехмерной свертки прямо “из коробки”.

Стоит также отметить, что хотя сверточный слой сокращает количество параметров по сравнению с полносвязным слоем, он использует больше **гиперпараметров** – параметров, выбираемых *до* начала обучения. В частности, выбираются следующие гиперпараметры:

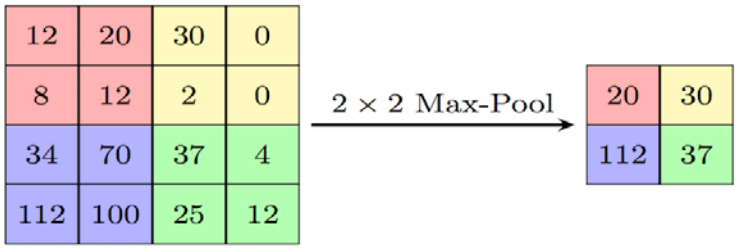
– *Глубина* (*depth*) – сколько ядер и коэффициентов смещения будет задействовано в одном слое;

– *Высота* (*height*) и *ширина* (*width*) каждого ядра;

– *Шаг* (*stride*) – на сколько смещается ядро на каждом шаге при вычислении следующего пикселя результирующего изображения. Обычно его принимают равным 1, и чем больше его значение, тем меньше размер выходного изображения;

– *Отступ* (*padding*): заметим, что свертка любым ядром размерности более, чем 1x1 *уменьшит* размер выходного изображения. Так как в общем случае желательно сохранять размер исходного изображения, рисунок дополняется нулями по краям.

Популярный способ субдискретизации изображения – слой *подвыборки* (также называемый слоем *субдискретизации*, по-английски *downsampling* или *pooling* layer), который получает на вход маленькие отдельные фрагменты изображения (обычно 2x2) и объединяет каждый фрагмент в одно значение. Существует несколько возможных способов агрегации, наиболее часто из четырех пикселей выбирается максимальный. Этот способ схематически изображен ниже.



Обычную архитектуру CNN для распределения изображений по *k* классам можно разделить на две части: цепочка чередующихся слоев свертки/подвыборки(иногда с несколькими слоями свертки подряд) и несколько полносвязных слоев (принимающих каждый пиксель как независимое значение) с слоем softmax в качестве завершающего. Я не говорю здесь о функциях активации, чтобы наша схема стала проще, но не забывайте, что обычно после каждого сверточного или полносвязного слоя ко всем выходным значениям применяется функция активации, например. Один проход влияет на изображение следующим образом: он сокращает длину и ширину определенного канала, но увеличивает его значение (глубину).

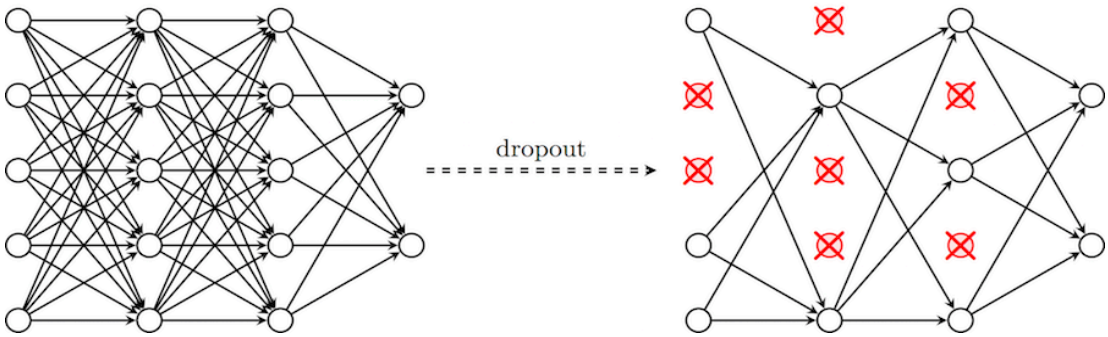
Напомним, что функция softmax превращает вектор действительных чисел в вектор *вероятностей* (неотрицательные действительные числа, не превышающие 1). В нашем контексте выходные значения являются вероятностями попадания изображения в определённый класс. Минимизация потерь перекрестной энтропии обеспечивает уверенность в определении принадлежности изображения определенному классу, не принимая во внимание вероятность остальных классов, таким образом, для вероятностных задач softmax предпочтительней, чем, например, метод квадратичной ошибки.

Обращу ваше внимание на глубокое **переобучение** (overfitting). Отрицательный эффект переобучения заметно проявляется на сетях, подобных той, что мы собираемся построить, а значит, необходимо найти способ защититься от этого явления прежде, чем мы пойдем дальше. К счастью, существует очень простой метод, который мы и применим.

Переобучение – это излишне точное соответствие нейронной сети конкретному набору обучающих примеров, при котором сеть теряет способность к обобщению. Другими словами, наша модель могла выучить обучающее множество (вместе с шумом, который в нем присутствует), но она не смогла распознать скрытые процессы, которые это множество породили.

У глубоких сверточных нейронных сетей масса разнообразных параметров, особенно это касается полносвязных слоев. Переобучение может проявить себя в следующей форме: если у нас недостаточно обучающих примеров, маленькая группа нейронов может стать ответственной за большинство вычислений, а остальные нейроны станут избыточны; или наоборот, некоторые нейроны могут нанести ущерб производительности, при этом другие нейроны из их слоя не будут заниматься ничем, кроме исправления их ошибок.

Чтобы помочь нашей сети не утратить способности к обобщению в этих обстоятельствах, мы вводим приемы *регуляризации*: вместо сокращения количества параметров, мы накладываем ограничения на параметры модели во время обучения, не позволяя нейронам изучать шум обучающих данных. Здесь я опишу прием **dropout**", помогает исключить ситуации, описанные выше. В частности, dropout с параметром p за одну итерацию обучения проходит по всем нейронам определенного слоя и с вероятностью p *полностью исключает их из сети на время итерации*. Это заставит сеть обрабатывать *ошибки* и не полагаться на существование определенного нейрона (или группы нейронов), а полагаться на "*единое мнение*" (consensus) нейронов внутри одного слоя. Это довольно простой метод, который эффективно борется с проблемой переобучения сам, без необходимости вводить другие регуляризаторы. Схема ниже иллюстрирует данный метод.



Наша модель достигает точности около 76.6% на тестовом множестве; для такой сложной задачи, где даже человеческий взгляд показывает точность всего около 90%, а также учитывая относительную простоту модели, это вполне достойный результат. Тем не менее, более сложные модели в последних исследованиях достигали точности 96.53%.