МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В.ЛОМОНОСОВА
Механико-математический факультет

# ОПТИМИЗАЦИЯ КОЛИЧЕСТВА ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ, РЕШАЮЩЕЙ ЗАДАЧУ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

*К.С. Кондаурова, Н.А. Семенов*

**Аннотация**

В настоящей статье были рассмотрены методы классификации изображений, основанные на нейронных сетях, и представлено несколько модификаций классических архитектур, ощутимо сокращающих число параметров. Сначала нами было предложено преобразование свёрточного слоя, которое заключается в использовании Convolution 3d вместо обычной двумерной свёртки. Желание ещё больше сократить вычислительные затраты без видимой потери точности натолкнуло на мысль о создании собственной архитектуры, которая является по сути обобщением полносвязной конструкции, но с некоторым приравниваем коэффициентов. Несмотря на то что нам не удалось добиться высоких показателей в выбранных метриках, исследование этой иновации (она упоминается в работе как Meta слой) привело к идее замены полносвязной головы нейронной сети с относительно большим числом параметров на комбинацию нескольких полносвязных слоёв с меньшим числом нейронов и последующего соединения их выходов посредством Meta слоя. В результате этой манипуляции не только сократилось количество обучаемых параметров, но и увеличилась точность алгоритма. Таким образом, в произвольно взятой нейронной сети, в которой присутствует большой полносвязный слой, имеет смысл заменять его на комбинацию нескольких маленьких и Meta.

**Ключевые слова:** машинное обучение, нейронные сети, классификация изображений, свёртка, полносвязный слой, Meta слой, точность, выразимость, вычислительные затраты
**Key words:** machine learning (ML), neural networks, classification, convolution, full connection, Meta layer, accuracy, expressivity, computational costs

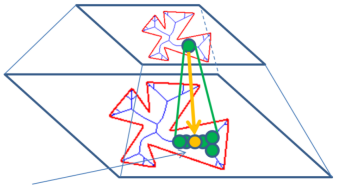
## 1. Введение

Проблема работы с изображениями методами классического ML и многослойными персептронами состоит в том, что его надо растянуть в одномерный вектор перед началом работы алгоритма. В результате такого преобразования теряется связь между соседними пикселями. Если говорить точнее - связь остается, но она становится неявной. Модели нужно самой понять, какие пиксели являются соседними и порождающими какой-то объект, а какие находятся в разных частях изображения. Из-за этого решение становится неустойчивым к поворотам, отражениям, сдвигам и другим преобразованиям, которые на самом деле сохраняют изображённые объекты. Свёрточные нейронные сети (CNN) с уверенностью решают эту проблему и не меняют начальной топологии, поэтапно векторизуя изображение и подготавливая его на вход полносвязному слою.

Вышеупомянутое достоинство CNN наталкивает на мысль об их более широком применении, ведь свёрточные сети по сути способны классифицировать любые данные, на которых возможно ввести топологию, а именно можно сказать, какие элементы близки к друг другу, а какие нет.

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

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



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

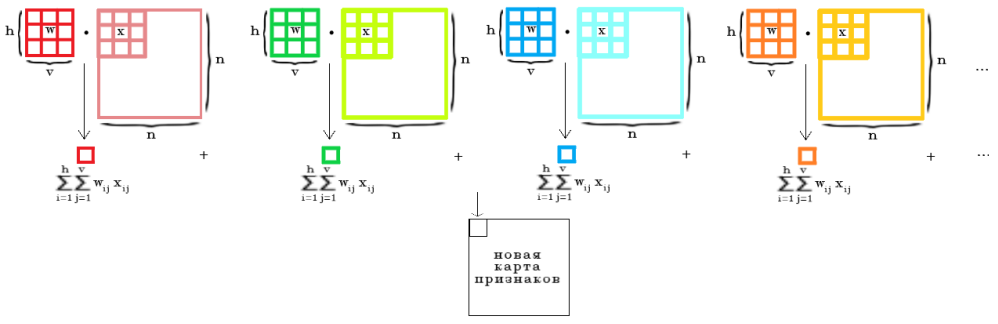
## 2. Оптимизация свёрточного слоя

### 2.1. Мотивация для использования Convolution 3d

Вспомним, как устроен свёрточный слой. Предположим, что ему на вход подаётся $C$ карт признаков размера $n \times n$. Определим фильтр размера $h \times v$ , $h, v$ - нечётные, $h < n$, $v < n$. Операция свёртки устроена таким образом, что для каждой карты признаков будет задан свой фильтр с $hv$ обучаемыми параметрами, а пиксель выходного слоя получится путём суммирования результатов применения соответствующего фильтра к карте. Значит, для формирования одной новой признаковой карты используется столько двумерных фильтров, сколько было прошлых признаковых карт, и новый "пиксель" содержит информацию из всех карт признаков предыдущего уровня, так как вычислен по формуле:

$$\sum_{k=1}^{C} \sum_{i=1}^{h} \sum_{j=1}^{v} w_{kij} x_{kij} \,,$$

где $w_{kij}$ - вес в $k$-ом фильтре, $x_{kij}$ - признак в $k$-ой карте. Этот процесс можно изобразить так:



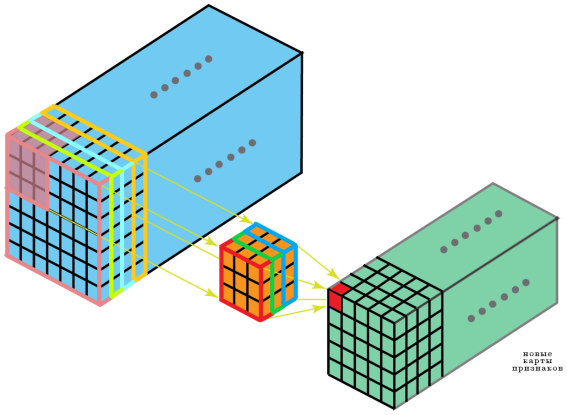
Чтобы получить на выходе $D$ карт признаков, понадобится $(h \cdot v + 1) \cdot D$ обучаемых параметров.

Это число можно сократить, если **воспользоваться трёхмерной свёрткой.** Она устроена похожим образом. Её отличие состоит лишь в том, что на вход подаются трёхмерные данные, которые можно себе представить как параллелепипед. Фильтр, в свою очередь, тоже приобретает дополнительную размерность и скользит по параллелепипеду, вычисляя скалярное произведение весов и признаков.

Итак, перед применением трёхмерной свёртки необходимо преобразовать данные. У каждой из $C$ карт признаков введём дополнительную размерность и склеим по ней. Применим Convolution 3d с фильтром размера $h \times v \times u$ , $u < C$ и единичным сдвигом по третьей оси. Таким образом, новый признак объединяет информацию из $u$ карт признаков предыдущего уровня, так как вычислен по формуле:

$$\sum_{k=a}^{a+u} \sum_{i=1}^{h} \sum_{j=1}^{v} w_{kij} x_{kij},$$

где $w_{kij}$ - вес в $k$-ом фильтре, $x_{kij}$ - признак в $k$-ой карте, $a \in \{1, \dots C - u\}$. Изобразим схематично этот процесс:




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

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

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

Тогда количество обучаемых весов равно:

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

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

- Трёхмерная свёртка позволяет экономить параметры, а, следовательно, снижает вычислительную нагрузку.

- Трёхмерная свёртка, вообще говоря, выявляет другие признаки нежели привычная нам двумерная.
- Двумерная свёртка является частным случаем трёхмерной свёртки.
- Трёхмерная свёртка олицетворяет собой свёртку карт признаков как целиковых объектов.

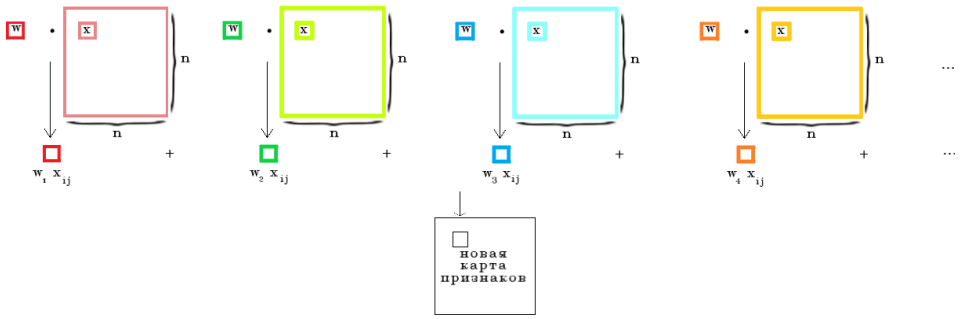
### 2.2. Мотивация для создания Meta слоя

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

Как и ранее, допустим, что на вход подаётся $C$ карт признаков размера $n \times n$. На выходе хотим получить карту, пиксель которой объединяет информацию со всех карт предыдущего слоя, но в предположении, что все относящиеся к одной карте веса равны. Тогда понадобится всего $C$ обучаемых параметров, и новые признаки будут вычисляться по формуле:

$$\sum_{k=1}^{C} w_k x_{kij},$$

где $w_k$ - вес, присвоенный $k$-ой карте, $x_{kij}$ - признак в $k$-ой карте. Этот процесс можно изобразить так:




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

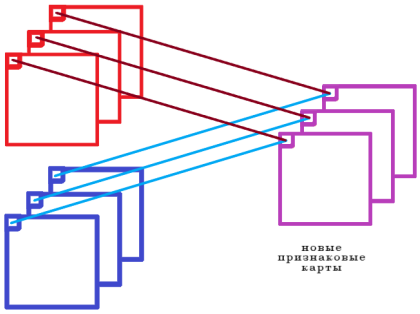
Далее введём некоторое обобщение. Допустим, что входные карты признаков разделены на $N$ групп, в каждой из которых по $C : N$ карт. Тогда на вход Meta слою будем подавать эти $N$ групп, а на выходе требовать $M$ групп по $C : N$ новых признаковых карт. На алгоритмическом языке слой Meta(N, M) можно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

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

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

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

Output:
new_output
```

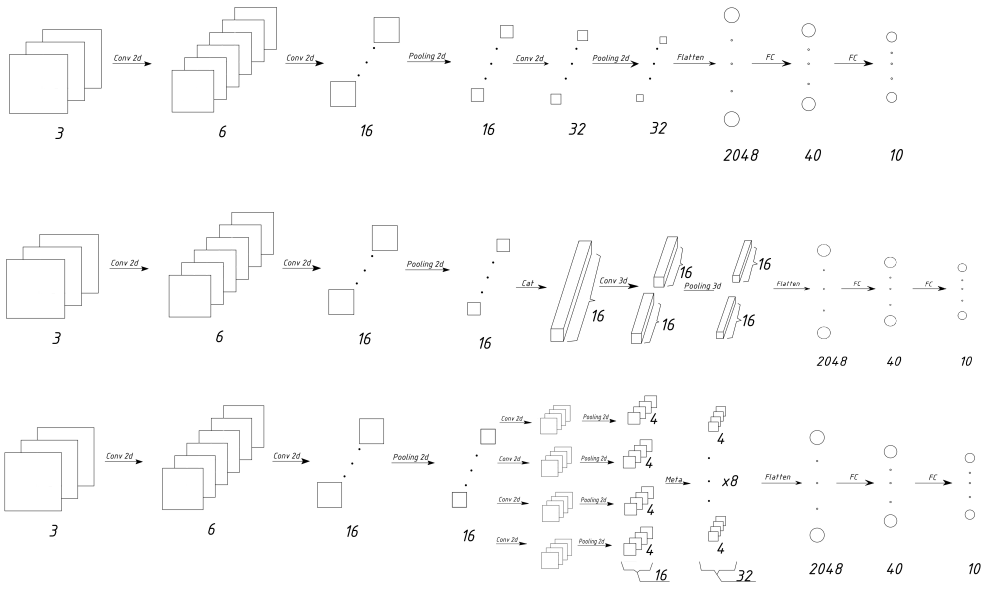
Несмотря на то что описанная конструкция даёт большой вычислительный граф, количество требуемых параметров всего лишь $N \cdot M$.

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




### 2.3. Тестирование

Итак, необходимо сравнить результаты работы трёх слоёв: двумерного свёрточного, трёхмерного свёрточного и слоя Meta. Предъявим наглядную иллюстрацию одного из возможных вариантов согласованных архитектур:



Для тестирования нами была выбрана база данных CIFAR-10, включающая 60 000 цветных изображений размера 32x32. Эти изображения разделены на 10 непересекающихся классов: самолеты, автомобили, птицы, кошки, олени, собаки, лягушки, лошади, корабли и грузовики. Чтобы сравнение методов было корректным, осуществим классификацию изображений с помощью нескольких архитектур с разным количеством параметров, однако число признаковых карт оставим согласованным. Иллюстрация к первому тестированию предъявлена выше.

Отметим, что каждая нейронная сеть обучалась в одинаковых условиях (batch size, learning rate, количество эпох и так далее). В качестве функции активации была выбрана недавно зарекомндовавшая себя функция PReLU. В сравнительной таблице приведены данные об архитектуре, числе обучаемых параметров и средней точности по десяти запускам:

| **Architecture** | | | **Total parameters** | **Accuracy** |
|---|---|---|---|---|
| Conv2d→<br>BatchNorm2d→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>Dropout2d→ | *Conv2d→*<br>*BatchNorm2d→*<br>*MaxPool2d→*<br>*Dropout2d→* | FC→<br>Dropout2d→<br>FC | 88 170 | **72,80%** |
| | *Conv3d→*<br>*BatchNorm3d→*<br>*MaxPool3d→* | | 83 526 | 68,82% |
| | *4×(Conv2d→*<br>*BatchNorm2d→*<br>*MaxPool2d)→*<br>*Meta→* | | 84 126 | 70,39% |
| Conv2d→<br>BatchNorm2d→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>Dropout2d→<br>Conv2d→<br>BatchNorm2d→ | *Conv2d→*<br>*BatchNorm2d→*<br>*MaxPoo2dl→*<br>*Dropout2d→*<br>Conv2d→<br>BatchNorm2d→<br>*Conv2d→*<br>*BatchNorm2d→*<br>*AdaptiveAvgPool2d→*<br>*Dropout2d→* | FC→<br>Dropout2d→<br>FC | 357 265 | **83,40%** |
| | *Conv3d→*<br>*BatchNorm3d→*<br>*MaxPool3d→*<br>Conv2d→<br>BatchNorm2d→<br>*Conv3d→*<br>*BatchNorm3d→*<br>*AdaptiveAvgPool3d→* | | 176 761 | 80,93% |
| | *8×(Conv2d→*<br>*BatchNorm2d)→*<br>*Meta→*<br>*MaxPool3d→*<br>Conv2d→<br>BatchNorm2d→<br>*16×(Conv2d→*<br>*BatchNorm2d)→*<br>*Meta→*<br>*MaxPool3d→* | | 187 433 | 78,34% |

| Architecture | | | Total parameters | Accuracy |
|---|---|---|---|---|
| Conv2d→<br>BatchNorm2d→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>Dropout2d→ | *ResBlock:*<br>*Conv2d*→<br>*BatchNorm2d*→<br>*Conv2d*→<br>*BatchNorm2d*→<br>*Dropout2d*→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>*ResBlock*→ | AdaptiveAvgPool2d→<br>Flatten→<br>FC→<br>Dropout2d→<br>FC | 1 573 902 | **84,27%** |
| | *ResBlock:*<br>*Conv3d*→<br>*BatchNorm3d*→<br>*Conv2d*→<br>*BatchNorm2d*→<br>*Dropout2d*→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>*ResBlock*→ | | 846 222 | 82,99% |
| | *ResBlock:*<br>*8×(Conv2d*→<br>*BatchNorm2d)*→<br>*Conv2d*→<br>*BatchNorm2d*→<br>*Dropout2d*→<br>Conv2d→<br>BatchNorm2d→<br>MaxPool2d→<br>*ResBlock*→ | | 931 086 | 82,96% |

### 2.4. Вывод

Проведённые нами эксперименты говорят о том, что замена двумерной свёртки на Convolution 3d или комбинацию нескольких независимых свёрток с меньшим числом признаковых карт и Meta слоя привела к потере точности на 1-5 % при сокращении числа параметров на 5-50 %. Наши преобразования показали лучшие результаты в известной архитектуре ResidualNetwork: точность алгоритма изменилась незначительно, однако вычислительные затраты уменьшилисть почти вдвое. Из этого факта можно сделать сделать вывод о том, что для формирования признаков более высокого уровня с помощью Convolution 3d или Meta желательно иметь информацию из нескольких предыдущих карт.

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

Ещё раз отметим: если в практических задачах на классификацию изображений допустима погрешность точности $\pm 1\%$, ресурсы ограничены, а лучшими

показателями обладает ResidualNetwork по каким-либо причинам, то имеет смысл использовать вместо обычных свёрток Convolution 3d или Convolution 2d и Meta.

## 3. Оптимизация полносвязного слоя

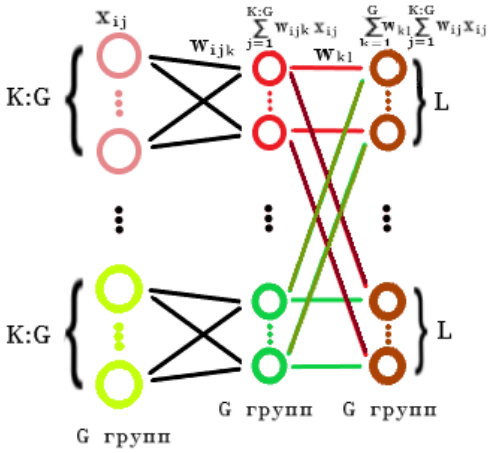
### 3.1. Мотивация для использования Meta слоя

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

Описанный в предыдущей главе Meta слой позволяет обрабатывать группы признаков, приравнивая некоторые весовые коэффициенты. Предположим, что $K$ признаков теперь разбито на $G$ (при условии $K \vdots G$) групп по $K : G$ в каждой. Для всех групп зададим отдельно полносвязный слой с $L = H : G$ выходами. А затем свяжем их друг с другом с помощью слоя Meta$(G,\ G)$. Тогда новый $k$-ый признак в $l$-ой группе будет вычислен по формуле:

$$\sum_{k=1}^{G} w_{kl} \sum_{j=1}^{K:G} w_{ijk} x_{ij}\,,$$

где $w_{ijk}$ - $j$-ый вес в $i$-ой группе, соединяющий $k$-ый промежуточный признак, $x_{ij}$ - $j$-ый признак в $i$-ой входной группе, $w_{kl}$ - вес, соединяющий $k$-ую промежуточную группу и $l$-ую выходную. Проиллюстрируем конструкцию:




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

### 3.2. Тестирование

Как и ранее, было выбрано несколько свёрточных архитектур для сравнения. Каждая нейронная сеть обучалась в условиях, освещённых в главе 2. В таблице ниже приведены данные об архитектуре, числе обучаемых параметров и точности:

| Architecture | | | Total parameters | Accuracy |
|---|---|---|---|---|
| Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ MaxPool2d→ Dropout2d→ Conv2d→ BatchNorm2d→ MaxPool2d→ Dropout2d→ | *FC→ Dropout2d→* | FC | 88 170 | **72,80%** |
| | *4×(FC→ Dropout2d)→ Meta→* | | 26 750 | 71,98% |
| Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ MaxPool2d→ Dropout2d→ Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ MaxPoo2dl→ Dropout2d→ Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ AdaptiveAvgPool2d→ Dropout2d→ | *FC→ Dropout2d→* | FC | 357 265 | 83,40% |
| | *16×(FC→ Dropout2d)→ Meta→* | | 292 961 | **84,25%** |
| Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ MaxPool2d→ Dropout2d→ ResBlock: Conv2d→ BatchNorm2d→ Conv2d→ BatchNorm2d→ Dropout2d→ Conv2d→ BatchNorm2d→ MaxPool2d→ ResBlock→ AdaptiveAvgPool2d→ Flatten→ | *FC→ Dropout2d→* | FC | 1 573 902 | 84,23% |
| | *16×(FC→ Dropout2d)→ Meta→* | | 1 516 766 | **84,27%** |

### 3.3. Вывод

Таким образом, нами было протестировано три различные комбинации двумерных свёрточных слоёв. Для каждой из них была совершена попытка заменить последующий полносвязный слой на 4 или 16 полносвязных слоёв с кратно меньшим числом нейронов, выходы которых считались независимо и объединялись с помощью Meta слоя. Как видно из таблицы, это действие позволило сократить количество обучаемых параметров и в некоторых случаях даже получить выигрыш в точности около 1%.

Такой результат можно объяснить идеологически: полносвязный слой параметризуется большим количеством значений, которые позволяют приближать любую функцию от $n$-мерного вектора, однако на практике целые группы параметров отвечают за схожие признаки объекта, передающиеся и

преобразующиеся слой за слоем в архитектуре. Они по сути являются одним и тем же ребром вычислительного графа, но используются несколько раз, а значит, их веса можно приравнять. Именно этот механизм и реализует Meta слой, позволяющий, с одной стороны, снизить вычислительную нагрузку при обучении сети, а с другой - повысить ее обобщающую способность путем использования более удачной параметризации искомой функции.

Стоит отметить, что одинаковые показатели точности были достигнуты в архитектурах, где количество параметров отличается в 5 раз, но нейроны соединены разными способами. Таким образом, крайне важным аспектом нейронной сети является вычислительный граф соединения рёбер. Задача его оптимального построения нуждается в дальнейших исследованиях, некоторые из которых были проделаны нами и описаны в данной статье. В частности, Meta слой может быть успешно применён для оптимизации полносвязных слоев в большом наборе архитектур, который не стоит ограничивать свёртыми.

**Примечание**

Все вычисления, описанные в статье, воспроизводимы через репозиторий GitHub, расположенный по данной ссылке: `https://github.com/konks/meta.git`

**Литература**


1. *George Cybenko.* Approximation by Superpositions of a Sigmoidal function. Mathematics of Control, Signals, and Systems. 1989.
2. *Lu Z. et al.* The expressive power of neural networks: A view from the width. Advances in neural information processing systems. – 2017. – С. 6231-6239.
3. *Ian Goodfellow, Yoshua Bengio, Aaron Courville* - The MIT Press. -2016.-Nov. -С. 800.
4. *Визильтер Ю.В., Горбацевич В.С..* Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей. ММРО-2017.
5. *Воронцов К. В.* Математические методы обучения по прецедентам (теория обучения машин).
6. *Razvan Pascanu, Guido Montufar, and Yoshua Bengio.* On the number of response regions of deep feed forward networks with piece-wise linear activations. arXiv preprint arXiv:1312.6098, 2013
7. *Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio.* On the number of linear regions of deep neural networks. In Advances in neural information processing systems, pages 2924–2932, 2014.
8. *Ronen Eldan and Ohad Shamir.* The power of depth for feedforward neural networks. arXiv preprint arXiv:1512.03965, 2015
9. *Matus Telgarsky.* Representation benefits of deep feedforward networks. arXiv preprint arXiv:1509.08101, 2015.
10. *James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel.* On the representational efficiency of restricted boltzmann machines. In Advances in Neural Information Processing Systems, pages 2877–2885, 2013.
11. *Monica Bianchini and Franco Scarselli.* On the complexity of neural network classifiers: A comparison between shallow and deep architectures. Neural Networks and Learning Systems, IEEE Transactions on, 25(8):1553– 1565, 2014.
12. *Maithra Raghu et al.* On the Expressive Power of Deep Neural Networks. 2016.

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

Сведения о каждом из авторов статьи:

**Кондаурова Ксения Сергеевна** – МГУ имени М.В.Ломоносова, механико-математический факультет, студент специалитета, 2019-2025

E-mail: *kseniya0681@mail.ru*

**Семенов Никита Антонович** – МГУ имени М.В.Ломоносова, механико-математический факультет, студент специалитета, 2019-2025

E-mail: *nikita2014semenov@gmail.com*