

Московский государственный университет имени М. В. Ломоносова  
Факультет Вычислительной Математики и Кибернетики  
Кафедра Математических Методов Прогнозирования



Доклад на тему

## Архитектуры свёрточных нейронных сетей ч.2

Выполнил:  
студент 3 курса 317 группы  
*Самбурукский Александр Ильич*

Москва, 2021

## **Аннотация**

Данная работа является конспектом лекции А.Г. Дьяконова, посвящённой существующим свёрточным нейронным сетям, и подробно описывает модифицированные архитектуры, повышающие эффективность обучения по вычислительным затратам, памяти и точности предсказания. Свёрточные нейросетевые подходы демонстрируют высокое качество решения в задачах обработки изображений и являются перспективными для дальнейшего развития и исследования. Доклад позволяет углубиться в тематику этого направления глубокого обучения и будет полезным подспорьем для читателей, заинтересованных в изучении эффективного применения нейронных сетей и уже знакомых с базовыми понятиями этого направления.

## **Содержание**

<b>1 Введение</b>	<b>2</b>
<b>2 Основные понятия</b>	<b>2</b>
<b>3 Архитектуры</b>	<b>4</b>
3.1 Внутреннее усложнение архитектуры . . . . .	4
3.1.1 Network in Network . . . . .	4
3.1.2 HyperNets . . . . .	6
3.2 Альтернативное развитие прокидывания связей. Борьба с проблемами глубоких нейросетей. . . . .	7
3.2.1 Deep Networks with Stochastic Depth . . . . .	7
3.2.2 FractalNet . . . . .	10
3.2.3 Fractal of FractalNet . . . . .	14
3.2.4 Densely Connected Convolutional Networks (DenseNet) . .	15
3.2.5 WideResNet . . . . .	18
3.2.6 EfficientNet . . . . .	20
3.3 Снижение вычислительных ресурсов . . . . .	23
3.3.1 MobileNet . . . . .	23
3.3.2 SqueezeNet . . . . .	27
3.3.3 ShuffleNet . . . . .	29
3.3.4 FBNet . . . . .	31
3.3.5 RevNet и iRevNet . . . . .	33
3.4 Некоторые новые архитектуры . . . . .	36
3.4.1 SpineNet . . . . .	36
3.4.2 ThunderNet . . . . .	38
3.4.3 CSPNet . . . . .	39
3.4.4 SAUNet . . . . .	41
<b>4 Выводы</b>	<b>42</b>

## 1 Введение

Одним из серьёзных направлений в анализе данных на данный момент является обработка больших потоков информации: звука, текста, изображений. В этой сфере на компьютерную технику ставятся непростые задачи, требующие сложных вычислений и быстрого отклика. В случае обработки изображений существует много прикладных и нетривиальных вопросов: классификация объектов, сегментация, описание изображённого действия, генерация, детектирование - и многие другие. Самым популярным инструментом для решения подобных задач являются нейросетевые алгоритмы. Как правило, они показывают наилучшие результаты среди других подходов распознавания. Это достигается благодаря возможности настройки большого числа модельных параметров и установленным связям между ними. В совокупности данные факторы приводят к высокой точности результата, отчего нейросети пользуются большой популярностью.

Подкласс свёрточных нейронных сетей благодаря основным принципам архитектуры позволяет обрабатывать огромные массивы данных быстро и достаточно точно. Эти алгоритмы обходят другие методы во многих задачах обработки картинок и фотографий. Направление открывает большое поле для возможностей улучшения качества и повышения скорости работы программных реализаций. В данной работе будут рассмотрены известные архитектуры, позволяющие добиться этих положительных показателей и повлиявших на дальнейшее развитие подобных методов

## 2 Основные понятия

Для описания различных архитектур потребуется предписание некоторых понятий. Приведём их в этом разделе. Напомним, что нейронные сети, как правило, решают задачи supervised learning. Их архитектура представляет последовательность вычислительных слоёв, в которых происходит признакомое преобразование векторов входных объектов. Путём обратного распространения ошибки модель настраивает так называемые веса, регулирующие характер этих преобразований. В конце сети происходит передача полученных новых признаков для прогнозирования.

Среди основных использующихся операций/слоёв сети выделяются линейный (полносвязный) слой, слой нелинейности (или, что то же, активации), пулинг, батч-нормализация, softmax-преобразование. Полносвязный слой производит линейное преобразование признаков и во время обучения сети настраивает его коэффициенты. Нелинейность (например, ReLU, сигмоида, гиперболический тангенс) – поэлементная операция над признаковыми тензорами. Пулинг является сжимающим преобразованием, заменяющим подгруппу значений признаков на одно. Зачастую применяются операции max- и average-пулинга. Пулинг допускает преобразование с различными размерами ядра и его шагами. Батч-нормализация (BN) позволяет предобработать входящие

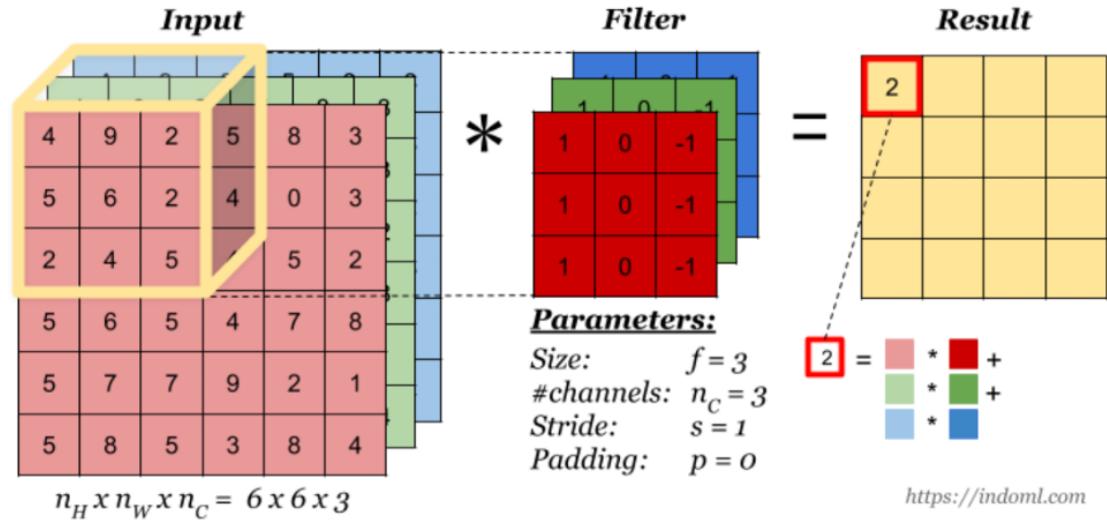


Рис. 1: Схема работы свёрточного слоя [19]

в слой данные батча нормированием, что повышает производительность и стабильность работы сетей. Softmax-преобразование расположено в конце сети и составляет распределение достоверности категорий для решения задачи классификации.

Как полезный приём, в сетях часто используют подход Drop-out. Во время обучения часть конструкции некоторых слоёв деактивируется, что позволяет лучше обучить оставшиеся части сети. Это повышает точность и эффективность обучения. В архитектурах, описанных в данной работе, данный приём также используется в различных проявлениях.

Для обработки изображений, как написано ранее, подходит семейство свёрточных нейронных сетей. Их особенностью является свёрточный слой, преобразующий очередной набор признаков посредством применения ядрового фильтра к нему – в качестве нового признака берётся линейная комбинация некоторых признаков исходного тензора. Признаки, участвующие в линейной комбинации и коэффициенты при них определяются ядром свёртки. Именно свёрточные слои позволяют обрабатывать изображения, опираясь на локальность образов. Следующий рисунок демонстрирует работу свёртки (Рис. 1).

Обычно в свёрточных нейронных сетях используются 3x3 свёртки вместе с батч-нормализацией и функцией активации. Может также применяться метод stride, который заключается в том, чтобы пропустить некоторые области, над которыми скользит ядро свёртки. За время изучения свёрточных нейронных сетей было реализовано несколько классических архитектур, с которыми, как правило, производят сравнение при создании новых моделей. Среди таких знаменитых архитектур выделяются LeNet-5 – простой отправной точкой для остальных сетей, AlexNet, после которой произошёл взрыв популярности свёрточных сетей, VGG16 / VGG19, GoogLeNet, Inception. Далее идёт ResNet, добавившее так называемое прокидывание связей – добавление новых

вычислительных путей, перекинутых через некоторые слои, для упрощения распространения градиента ошибки и облегчения обучения. Это позволило создавать практически неограниченные по глубине сети, делая их очень точными. В то же время увеличение глубины в некоторый момент становится причиной очень трудоёмких вычислений и необходимых для этого ресурсов, с чем борются архитектуры, представленные в данной работе. Для оценки качества работы сетей используются доля правильно классифицированных примеров, доли ошибок top-1 и top-5. Как правило, используются датасеты ImageNet, CIFAR-10, CIFAR-100.

## 3 Архитектуры

В этой секции рассмотрим разработанные архитектуры, для которых удалось добиться прироста эффективности при обучении, хранимой памяти или необходимой вычислительной мощности. Для большей наглядности и структурированности доклада архитектуры были условно поделены на 3 группы, отличающиеся по центральным целям, поставленных перед нейронными сетями, а также по принципам их построения. Разделение на группы весьма условно: рассматриваемые архитектуры благодаря своим выделяющимся особенностям демонстрируют хорошее качество и при этом упрощают вычислительный процесс на обучении. Каждая нейронная сеть делает это по-своему.

### 3.1 Внутреннее усложнение архитектуры

В данном подразделе будут рассмотрены две архитектурные концепции, позволяющие сделать обработку данных более гибкой путём надстройки дополнительных блоков в основной сети. Данные приёмы предоставляют возможность настраивать сеть на высокий уровень внутреннего анализа информации, что позволяет получать более хорошее качество.

#### 3.1.1 Network in Network

Как одно из качественных усложнений сети авторы работы [1] предлагают усложнить обычное свёрточное преобразование, добавив внутрь него небольшую нейронную сеть на выход непосредственного применения ядра свёртки к картам признаков. Основной особенностью данного подхода является расширение возможностей сети для глубокого распознавания свёрточным преобразованием исходных образов и дополнительным учётом этих локальных структур для более качественного отображения набора признаков на следующие этапы. Схематичный пример этой архитектуры, называемой Network in Network, представлен на рисунке 2.

Преобразования тензоров становятся более комплексными. Внутренние сети позволяют извлекать дополнительные полезные признаки внутри внешнего блока с более продуманной дальнейшей обработкой. В качестве добавленной внутренней структуры выступает многослойный персепtron, допускающий заметное увеличение сложности и гибкости схемы. В эту подсеть, как

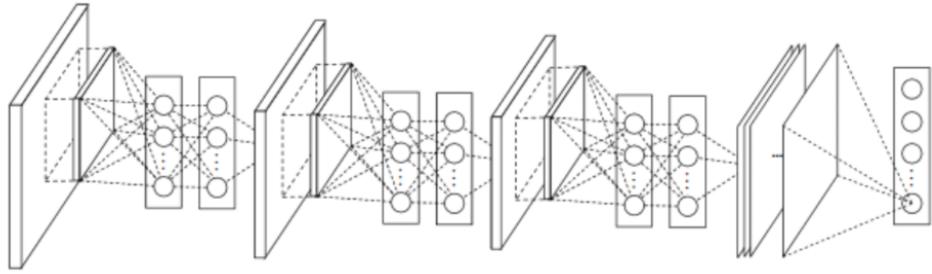


Рис. 2: Общий вид NIN [1]

можно видеть, подаются уже небольшие наборы наборы признаков, однако благодаря новым введённым параметрам подсетей архитектура усложняется на качественном уровне. Слои, включающие внутренние подсети, называются в статье *mlpconv*-слоями.

Дополнительно в статье отмечается особенность использования усредняющего пулинга. Идея заключается в том, что вместо традиционного полно связного слоя, применяющегося в конце сети, можно на последнем слое формировать карты признаков для каждой отдельной категории задачи классификации и подать в конец (на SoftMax слой) усреднения по каждой из такой карт признаков. Таким образом, архитектура может сохранять более строгое соответствие между классами и картами признаков. Плюс ко всему, подход снижает риск переобучения, так как применяющийся глобальный пулинг не перенастраивается (у него нет параметров) и более устойчив к пространственным преобразованиям входных данных (так как он усредняет эти признаки).

Используемые слои *mlpconv* лучше аппроксимируют карты признаков отдельных категорий, формируемыми сетью. Поэтому усредняющий глобальный пулинг гармонично вписывается в архитектуру, выступая в качестве регуляризатора, требующего соблюдения указанного соответствия карт и классов. Его роль, как регуляризатора, видна по следующей таблице:

Method	Testing Error
mlpconv + Fully Connected	11.59%
mlpconv + Fully Connected + Dropout	10.88%
mlpconv + Global Average Pooling	10.41%

Рис. 3: Эффект использования глобального пулинга [1]

Дополнительное повышение качества модели достигается применением дропаута между слоями *mlpconv*. Это происходит благодаря повышению способности модели генерализировать данные посредством сохранения части работающих вычислительных путей. Улучшение качества данной архитектуры относительно других методов видно по результатам экспериментов на датасете CIFAR-10 (Рис. 4).

Method	Test Error
Stochastic Pooling	15.13%
CNN + Spearmint	14.98%
Conv. maxout + Dropout	11.68%
<b>NIN + Dropout</b>	<b>10.41 %</b>
CNN + Spearmint + Data Augmentation	9.50%
Conv. maxout + Dropout + Data Augmentation	9.38%
DropConnect + 12 networks + Data Augmentation	9.32%
<b>NIN + Dropout + Data Augmentation</b>	<b>8.81 %</b>

Рис. 4: Сравнение NIN с другими методами [1]

Итак, Network in Network является архитектурой, увеличивающей гибкость нейронной сети благодаря внедрению дополнительных внутренних многослойных персепtronов и использовании метода глобального усредняющего пулинга в качестве регуляризатора, инвариантного относительно пространственных преобразований данных. Работа с признаками производится при соблюдении соответствия между формируемыми сетью картами признаков и категориями задачи классификации.

### 3.1.2 HyperNets

Истоки другого качественного усложнения свёрточных сетей описаны в работе [2]. Основной выдвигаемый в ней принцип сводится к переводу операций сети к динамическому выполнению и регулировке свёрток как результата работы другой нейронной сети, называемой гиперсетью.

Кратко обозначим два описанных в работе метода: статических и динамических гиперсетей. В первом методе внешняя двухслойная сеть генерирует фильтр свёртки из принятого эмбеддинга очередного слоя внутренней сети. В процессе обучения гиперсеть будет настраивать свои собственные параметры для снижения значения функционала ошибки, полученной основной сетью с использованием генерируемых свёрток. Сами эмбеддинги слоёв так же настраиваются, и в процессе работы уже обученной сети берутся как аргументы для генерации свёрток на тестировании.

Для проделывания такого трюка в случае рекуррентных нейросетей можно воспользоваться подходом динамических гиперсетей. В случае надстроек для RNN такие гиперсети называются HyperRNN. Она на каждом шаге  $t$  принимает на вход конкатенацию входного вектора  $x_t$  и скрытое состояние с предыдущего временного слоя рекуррентной сети  $h_{t-1}$  и генерирует следующее скрытое состояние, с помощью которого формируются веса модели на текущем временном шаге. Обе сети обучаются методом обратного распространения градиента. На рисунке 5 изображена общая структура их слоёв.

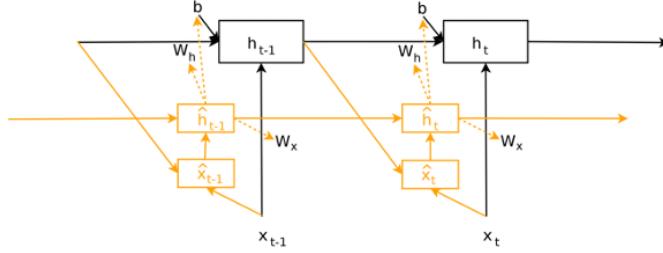


Рис. 5: Устройство блоков реккурентных гиперсетей [2]

Данные решения позволяют снизить число настраиваемых модельных параметров с небольшой потерей точности. Это демонстрируется в следующей таблице:

Model	Test Error	Param Count
Network in Network (Lin et al., 2014)	8.81%	
FitNet (Romero et al., 2014)	8.39%	
Deeply Supervised Nets (Lee et al., 2015)	8.22%	
Highway Networks (Srivastava et al., 2015)	7.72%	
ELU (Clevert et al., 2015)	6.55%	
Original Resnet-110 (He et al., 2016a)	6.43%	1.7 M
Stochastic Depth Resnet-110 (Huang et al., 2016b)	5.23%	1.7 M
Wide Residual Network 40-1 (Zagoruyko & Komodakis, 2016)	6.85%	0.6 M
Wide Residual Network 40-2 (Zagoruyko & Komodakis, 2016)	5.33%	2.2 M
Wide Residual Network 28-10 (Zagoruyko & Komodakis, 2016)	4.17%	36.5 M
ResNet of ResNet 58-4 (Zhang et al., 2016)	3.77%	13.3 M
DenseNet (Huang et al., 2016a)	3.74%	27.2 M
Wide Residual Network 40-1 <sup>2</sup>	6.73%	0.563 M
Hyper Residual Network 40-1 (ours)	8.02%	0.097 M
Wide Residual Network 40-2 <sup>2</sup>	5.66%	2.236 M
Hyper Residual Network 40-2 (ours)	7.23%	0.148 M

Рис. 6: Сравнение подхода HyperNets с другими архитектурами [2]

Так гиперсети могут производить эффективный перенос требуемых ресурсов для обучения одной сети в настройку сниженного числа параметров другой. Это решение нашло развитие в других архитектурах, таких как SENet и Attention.

### 3.2 Альтернативное развитие прокидывания связей. Борьба с проблемами глубоких нейросетей.

#### 3.2.1 Deep Networks with Stochastic Depth

В статье [3] рассматривается задача эффективного обучения глубоких нейронных сетей. Комплексное обучение всей модели, как правило, сопряжено с серьёзными трудностями: огромного времени обучения, проблемы затухания градиента или снижение роли полезных признаков в процессе прямого распространения. В то же время обученная глубокая сеть более гибкая и способна хорошо настраиваться на задачи распознавания. Соответственно, глубокие нейронные сети необходимы для сохранения высокого качества работы. Для того, чтобы решить эти проблемы, предлагается использовать следую-

шую стратегию: на этапе обучения для каждого батча объектов необходимо выкидывать большую часть слоёв сети и производить настройку параметров оставшейся неглубокой части модели, что сильно сокращает время и вычислительные ресурсы. Можно считать, что выкинутые слои заменяются тождественными преобразованиями набора признаков. На этапе теста используются все обученные таким образом слои и по итогу сеть получается глубокой и более точной.

В качестве решающего правила выкидывания блока исходной архитектуры анализируется значение случайной величины, обладающей распределением Бернулли с регулируемым параметром  $p_k$ . Для реализации данной идеи можно использовать глубокую сеть ResNet и исключать по такому распределению её блоки, следующие за первым блоком Conv-BN-ReLU в сети. Выход  $k$ -го слоя в таком случае можно выразить следующим образом:

$$H_k = \text{ReLU}(b_k f_k(H_{k1}) + id(H_{k1})) \quad (1)$$

На следующем рисунке визуализирована работа используемого ResNet-блока. При выкидывании слоя остаётся часть с тождественным преобразованием.

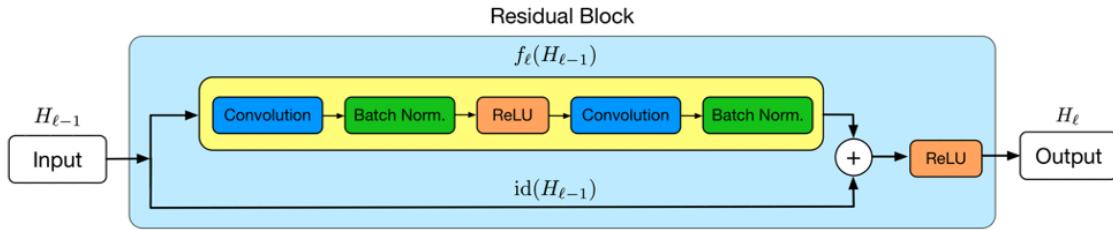


Рис. 7: Структура стандартного блока сетей ResNet [3]

На этапе теста модели преобразования блоков должны производится по следующим правилам:

$$H_k^{Test} = \text{ReLU}(p_k f_k(H_{k-1}^{Test}; W_{k-1}) + H_{k-1}^{Test}). \quad (2)$$

Это требуется для корректной перекалибровки весов модели после восстановления всех выкинутых блоков. Таким образом, блок архитектуры ResNet по данному правилу на этапе обучения может быть заменён на тождественное преобразование, сокращая глубину сети и сложности при её обучении. Вероятности выкидывания  $p_k$  являются гиперпараметрами модели. Их можно инициировать равномерно по слоям, либо по линейному убывающему закону от первого до последнего слоя (в предположении, что первые слои играют более важную роль в предсказании и требуют более точной настройки при обучении). Выпишем формулу для данного распределения. Пусть  $L$  – общая глубина исходной сети. Тогда  $p_l = 1 - l/(1 - p_L)$  Регулируя данные гиперпараметры, можно настроить как глубину, так и время обучения нейросе-

ти, избегая вычислительных проблем. Приведём пример. В случае линейного убывания вероятностей с  $p_L = 0.5$  среднеожидалася глубина сети на обучении может составлять 35-40% от исходного значения для больших значений  $L$ . При этом удается сохранить 25% времени на обучение модели:

	CIFAR10+	CIFAR100+	SVHN
Constant Depth	20h 42m	20h 51m	33h 43m
Stochastic Depth	15h 7m	15h 20m	25h 33m

Рис. 8: Ускорение обучения модели с использованием метода [3]

Помимо вычислительных и временных преимуществ такой подход позволяет снижать долю ошибок на различных тестовых данных относительно стандартной архитектуры ResNet с постоянной глубиной (см. таблицу):

	CIFAR10+	CIFAR100+	SVHN	ImageNet
Maxout	9.38	-	2.47	-
DropConnect	9.32	-	1.94	-
Net in Net	8.81	-	2.35	-
Deeply Supervised	7.97	-	1.92	33.70
Frac. Pool	-	27.62	-	-
All-CNN	7.25	-	-	41.20
Learning Activation	7.51	30.83	-	-
R-CNN	7.09	-	1.77	-
Scalable BO	6.37	27.40	1.77	-
Highway Network	7.60	32.24	-	-
Gen. Pool	6.05	-	1.69	28.02
ResNet with constant depth	6.41	27.76	1.80	21.78
ResNet with stochastic depth	5.25	24.98	1.75	21.98

Рис. 9: Ошибка классификации в сравнении с другими моделями [3]

Это может объясняться тем, что обучаемые неглубокие сети на этапе валидации входят в композицию, что можно интерпретировать, как реализацию модели ансамбля. Такой класс моделей способен качественно решать сложные задачи, поэтому в данном случае принципы его работы могут положительно сказаться на результатах.

На рисунке представлены результаты сравнения данного подхода с неизменяемой глубиной ResNet в случае большого числа слоёв в сети.

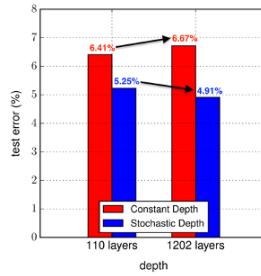


Рис. 10: Улучшение обучаемости глубоких сетей [3]

Как видно, с существенным увеличением числа слоёв стратегия стохастического распределения глубины сети показывает улучшение, в то время как, обычный ResNet деградирует из-за вычислительных проблем. На следующем графике представлены результаты обучения на различных значениях гиперпараметров  $p_L$ . При обеих стратегиях их инициализации можно получить улучшение качества относительно базовой архитектуры с постоянной глубиной. Кроме того, в линейном подходе инициализации этих параметров, как указывается в статье [3], допускается широкий интервал варьирования параметров, сохраняющий высокий уровень точности предсказания. Так же на графике обозначены некоторые оптимальные конфигурации этих параметров с указанием множителя ускорения времени обучения модели.

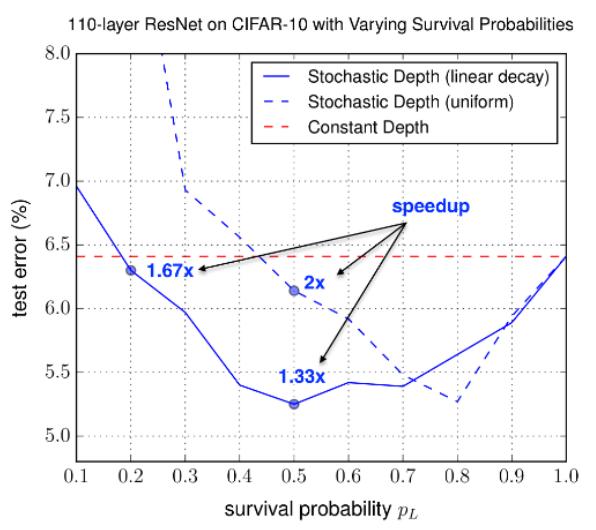


Рис. 11: Показатели качества и ускорения обучения [3]

Таким образом, подход с выборочным выкидыванием слоёв является эффективной модификацией архитектур свёрточных нейронных сетей и позволяет добиться существенного ускорения обучения моделей, а также сохранить и даже превзойти точность исходных моделей.

### 3.2.2 FractalNet

Как альтернативу классическим свёрточным нейронным сетям с прокидыванием связей авторы работы [4] предлагают использовать фрактальную архитектуру для построения нейронных сетей. Данный подход открывает возможности для регулирования сложности процесса обучения в балансе с точностью итоговой модели. Основная идея метода заключается в том, что заданная фрактальная структура нейросети может достаточно хорошо связывать признаки разного уровня преобразований. В итоге это приводит к тому, что сеть можно отдельно настраивать на совместную обработку важных признаков и возможностью усложнять в процессе обучения модельную архитектуру – эта

идея соответствует подходу Fractal of FractalNet. Ниже на рисунке представлено основное правило рекуррентного усложнения архитектуры для достижения моделью большей гибкости:

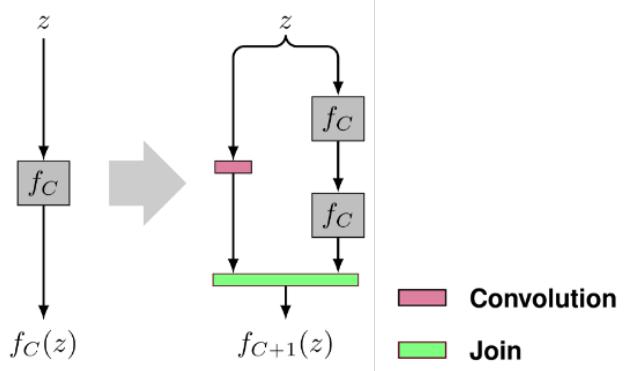


Рис. 12: Рекуррентное усложнение архитектуры [4]

Формально его можно записать в следующем виде:

$$f_1(z) = conv(z) \quad (3)$$

$$f_{C+1}(z) = [(f_C * f_C)(z)] + [conv(z)] \quad (4)$$

Некоторое преобразование во фрактальном блоке (например, обычная свёртка, либо другой фрактальный блок со своими внутренними преобразованиями) может быть заменено на два независимых разных по глубине и сложности отображения признаков, которые впоследствии конкатенируются и обрабатываются следующими блоками одновременно. Как правило, от одного внешнего блока к другому происходит снижение признакового пространства посредством операции пулинга. Из набора подобных блоков состоят слои сети FractalNet. В итоге получается глубокая нейронная сеть с аналогичными результатами, как у ResNet. На рисунке 13 представлен пример устройства такой сети.

Как видно, каждый блок данного примера допускает рекуррентное усложнение, задаваемое исходным фрактальным преобразованием. Данная архитектура довольно проста и не требует дополнительного прокидывания связей, какие используются в ResNet. Можно также отметить, что получаемая архитектура включает в себя разные по сложности сети, отличающиеся глубиной, а, следовательно, и сложностью обучения, и точностью работы. В дальнейшем такие “параллельные” структуры можно обучать по-отдельности и регулировать баланс между вычислительным объёмом для обучения и качеством на teste. Противопоставляя данную модель архитектуре ResNet отмечаются следующие свойства:

- Блоки FractalNet не разделяют пришедшие признаки по значимости. То есть внутри одного блока будут одинаково обрабатываться признаки,

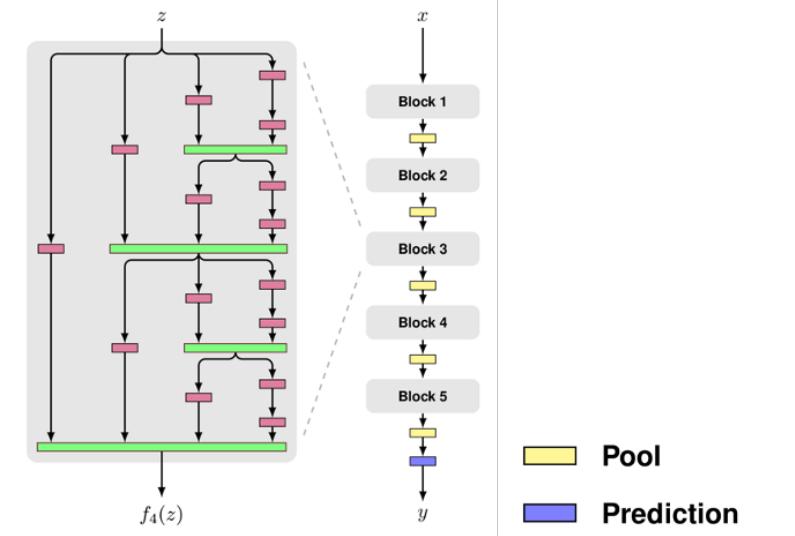


Рис. 13: Структура FractalNet [4]

прошедшие через разные по глубине подсети, что позволяет органично анализировать как полученные, так и исходные признаки.

- На практике оказывается, что из обученной фрактальной сети для обработки информации можно выделить часть самых продуктивных подсетей и делать предсказания на них.

После указанных базовых принципов архитектуры рассматривается вопрос об укреплении сильных качеств модели благодаря аналогу дропаута, называемого авторами как drop-path. Метод заключается в исключении путей некоторых блоков, что позволяет отдельно настроить их сложные и простые подсети. И на тесте при совместной работе данные модели покажут хорошие результаты. Как было написано выше, из этих подструктур можно выделить наиболее полезные и производить обработку данных на них, что ускорит процесс тестирования. В качестве реализаций drop-path можно рассмотреть следующие:

- Локальная реализация

Из блока сети каждая связь выбрасывается в соответствии с фиксированной вероятностью, при этом необходимо гарантировать, что по крайней мере один полноценный путь до выхода сети сохранится.

- Глобальная реализация

Из всей сети случайно выбирается единственный полноценный путь от входа до выхода, при этом выполняется условие, что он охватывает одинаковые пути по уровню фрактальной структуры.

На рисунке представлены возможные реализации глобального и локального drop-path:

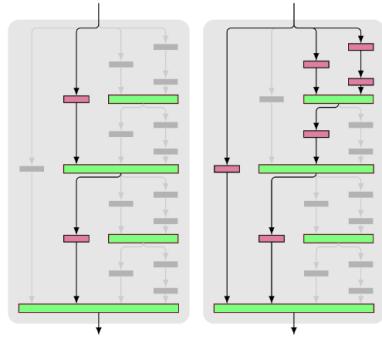


Рис. 14: Примеры drop-path [4]

Каждая из подобных подсетей формируется для каждого отдельного батча. Это позволяет сделать отдельные ветви сети более точными. В качестве эксперимента реализация с drop-path сравнивалась с архитектурами VGG-16 и ResNet-34 C на датасете ImageNet. Реализация FractalNet включала первый и последний слои ResNet-34, и между ними 4 блока по 8 слоёв каждый. Как значение параметра выкидывания для drop-path была взята величина в 15%. Результаты сравнения приведены в таблице ниже.

Method	Top-1 (%)	Top-5 (%)
VGG-16	28.07	9.33
ResNet-34 C	24.19	7.40
FractalNet-34	24.12	7.39

Рис. 15: Сравнение ошибок относительно других архитектур [4]

Как видно, при корректной настройке усложнения фрактальной схемы можно добиться повышения качества. Помимо этого, FractalNet допускает увеличение глубины сети без заметной потери качества. Данное свойство демонстрируется на рисунке 16 (эксперимент проводился на датасете CIFAR-100++).

Cols.	Depth	Params.	Error (%)
1	5	0.3M	37.32
2	10	0.8M	30.71
3	20	2.1M	27.69
4	40	4.8M	27.38
5	80	10.2M	26.46
6	160	21.1M	27.38

Рис. 16: Зависимость доли ошибок от глубины сети [4]

Таким образом, данная архитектура допускает реализации с большими показателями глубины сетей. Дополнительно можно отметить следующее свойство: в обученной FractalNet самая глубокая подсеть показывает примерно такое же качество, как и вся сеть. То есть после обучения допускается использовать небольшую часть модели, что может заметно снизить вычислительные расходы на тестовой выборке при предсказании.

Как отмечается в [4], эффективность такой архитектуры, как и ResNet, можно объяснить так: модель содержит вычислительные блоки большой глубины для высокой гибкости и вместе с ними наращивает блоки с короткими вычислительными ветвями, позволяющие эффективно разрешать проблемы обратного распространения. В совокупности эти факторы позволяют получить глубокие нейронные сети с высоким качеством обработки.

### 3.2.3 Fractal of FractalNet

Рассмотренная в прошлом разделе архитектура FractalNet демонстрирует неплохие результаты в случае очень глубоких сетей. Это связано с совмещением глубоких и неглубоких подсетей в одной структуре, что позволяет совмещать высокое качество и преодолевать вычислительные проблемы на этапе обучения. Авторы статьи [5] предлагают взять блок сети FractalNet за основу и с его помощью построить другую сеть. Допускается увеличить ширину и снизить глубину сети. Общий вид сети изображен на следующем рисунке:

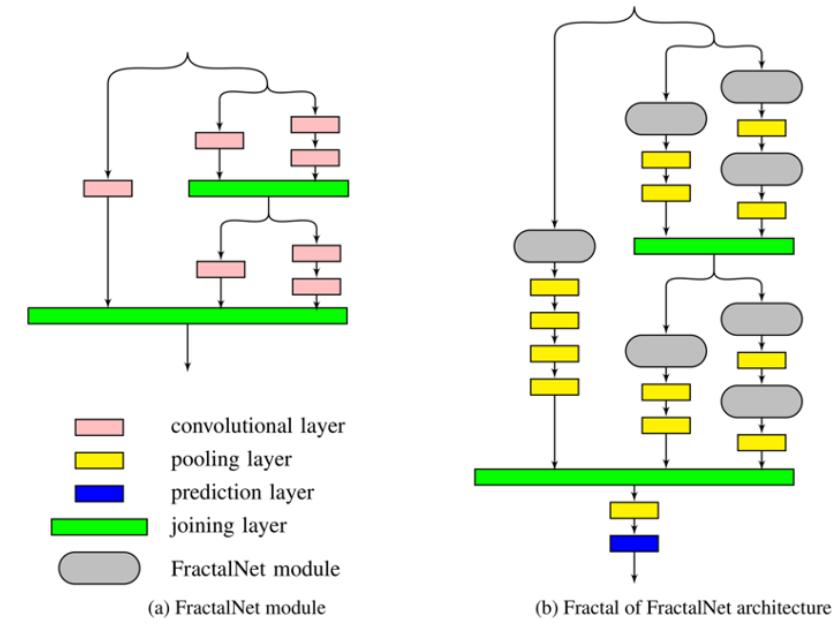


Рис. 17: Расширение фрактальной сети [5]

Как видно, в отличие от FractalNet данная реализация “переносит” архитектурную сложность из глубины сети по путям вычислительных блоков. Вычислительные пути обрабатываются на разных вычислительных уровнях (по числу операций свёртки) и при каждом объединении благодаря балансу слоёв пулинга совмещаются по размерности. Такое решение приводит к увеличению количества вычислительных путей сети, что позволит расширить эффективность модификации подхода drop-path, называемого авторами freeze-path. Подход подразумевает фиксацию определённых весов сети – это достигается обнулением параметра learning rate соответствующих слоёв – по итогу обратное распространение градиента их не изменит. Данный принцип можно применить в методе freeze-drop-path. При его использовании сеть обучается поэтапно. На первом этапе выкидываются все пути, за исключением самого вычислительно простого. Сеть обучается, а далее все найденные веса фиксируются. На следующем этапе производится обучение нового пути, более глубокого, с найденными весами. Так итерационно фрагменты сети корректируют неточности более простых. При этом первые итерации обучения отрабатывают быстрее. Ввиду того, что первые обученные подсети составляют определённые предсказания, последующим будет проще аппроксимировать целевую функцию для распознавания, основываясь на результатах простых подсетей. Ввиду этого можно ожидать повышение качества всей нейронной сети. Как показали эксперименты, Fractal of FractalNet удалось достичь примерно такого же качества, что и FractalNet. Таким образом, фрактальные свёрточные нейронные сети могут показывать хорошее качество при разных стратегиях усложнения архитектур, допускают эффективные реализации drop-out, и открывают дополнительные возможности для дальнейшего исследования.

### 3.2.4 Densely Connected Convolutional Networks (DenseNet)

Развитие идей прокидывания связей находится в статье [6]. Основной принцип архитектуры заключается в полном дополнении всех попарных связей между слоями сети. При объединении признаков, пришедших в один слой сети DenseNet, производится конкатенация, что способствует линейному росту признаков, обрабатываемой в слое. Это, в свою очередь, приводит к возможности сокращения параметров сети и вычислительным объёмам, о чём пойдёт речь позже. Следующая формула применяется для обработки очередной карты признаков на  $k$ -том слое:

$$x_k = H_k([x_0, x_1, \dots, x_{k-1}]), \quad (5)$$

то есть каждый набор признаков получается преобразованием всех предыдущих. В сравнении с правилами ResNet-архитектур данное выражение кажется гораздо сложнее, но в реализации DenseNet используются определённые свойства, позволяющие избегать большие вычислительные затраты. Каждый слой состоит из комбинации блоков: BN + ReLU + 3x3 Convolution +

dropout. На следующем рисунке приведена принципиальная схема 5-слойной сети DenseNet:

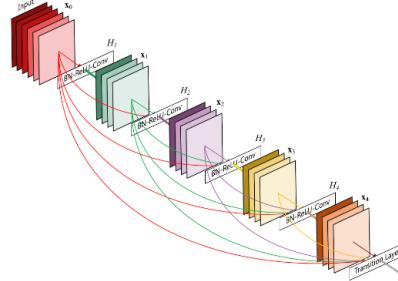


Рис. 18: Структура DenseNet [6]

Для нейронной сети с  $L$  слоями требуется провести  $L(L+1)/2$  связей. Каждый слой получает обработанную информацию со всех предыдущих. Одним из свойств такого метода является снижение требуемых параметров модели для качественного обучения. Дело в том, что получаемые на каждом слое данные аккумулируются, и появляется возможность из полученных слоем множества карт признаков получать фиксированное число карт (как правило это число равно 12 и является гиперпараметром модели), а оставшиеся – не подвергать изменениям. Несмотря на снижение числа преобразований, слои получают много информации и так или иначе модель принимает решение, опираясь на большое множество полезных признаков, накопленных за прямой проход по сети. Поэтому полное преобразование признаков необязательно. Отсюда следует, что и сеть DenseNet настраивает не много параметров в процессе обучения. Ещё одним преимуществом архитектуры можно назвать улучшение информационного потока между слоями, входными и выходными данными. Так как каждый слой имеет непосредственную вычислительную связь с началом и концом сети, то как прямое, так и обратное распространение можно производить путём прямого доступа (до входных данных и до значения функции потерь на батче соответственно). Для того, чтобы применять пулинг сеть подразделяется на Dense-блоки, в каждом из которых содержится описываемая замкнутая система слоёв. Это позволяет добиться снижения размерности (down-sampling) и корректно обрабатывать операцию объединения информации. Между блоками используется отдельные слои 1x1 свёртки и 2x2 пулинга:

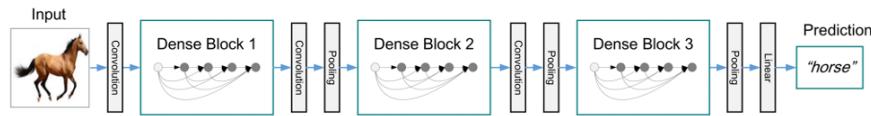


Рис. 19: Разделение сети на последовательные блоки [6]

Для повышения эффективности сети DenseNet используются дополнительно два архитектурных решения:

- Во-первых, для повышения вычислительной эффективности можно перейти к следующему представлению слоя:  $[BN + ReLU + Conv(1x1)] + [BN + ReLU + Conv(3x3)]$ . Основные вычисления остаются во второй половине слоя. Добавление свёртки  $1x1$  снижает количество входящих в слой карт признаков, что и обуславливает повышение вычислительной эффективности.
- Можно дополнительно регулировать долю карт признаков, переходящих от одного dense-блока к другому. За это отвечает ещё один гиперпараметр модели. Как правило, на последующий dense-блок передаётся половина полученных карт признаков на текущем блоке. Это аналогично позволяет повысить эффективность обучения.

Реализация, эксплуатирующая обе эти механики называется DenseNet-BC, только первую механику – DenseNet-B, только вторую DenseNet-C. На следующем графике (слева) представлено сравнение этих реализаций:

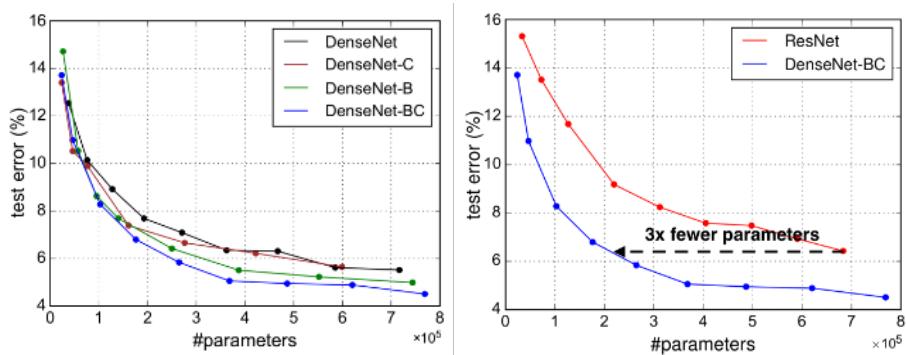


Рис. 20: Процесс обучения модификаций DenseNet (слева) и сравнение результатов обучения с ResNet (справа)[6]

Отсюда видно, что данные механики позволяют получить прирост в качестве модели, при этом, как написано выше, повышают эффективность обучения по вычислительным объёмам.

На рисунке 20 справа реализация DenseNet-BC сравнивается с результатами обучения ResNet.

При трёхкратном выигрыше в числе обучаемых параметров DenseNet-BC показывает то же качество, что и ResNet. При сближении количества параметров моделей DenseNet-BC демонстрирует относительное качество на 1.5%. Итак можно выделить следующие особенности DenseNet, выделяющие её преимущества:

- Компактность.

Как видно на графике выше, число параметров, настраиваемых сетью гораздо меньше, чем стандартной ResNet. В связи с этим повышается эффективность обучения.

- Непосредственная обработка входных данных и функции потерь.

Каждый слой в Dense-блоках напрямую связан с его входом и выходом. Благодаря этому при обратном распространении градиента этот учёт может дополнительно улучшить настройку весов модели.

- Переработка прошедших признаков.

Внутри одного Dense-блока слои действительно настраиваются на выходы многих предыдущих слоёв, что значит целесообразность использования архитектуры, допускающей такие полностью заполненные связями блоки. Данное свойство можно наблюдать по весам настроенной модели: их величина сигнализирует о важности используемых входных данных, полученных с нескольких предыдущих слоёв блока.

Итак, DenseNet является архитектурой свёрточных нейронных сетей, ориентированных на связь всех слоёв модели между собой. Это позволяет сократить параметрическую сложность моделей и, как следствие, повысить скорость обучения, снизить вычислительную её сложность, удерживая высокое качество предсказания.

### 3.2.5 WideResNet

Как указывается во многих статьях, стандартные глубокие свёрточные нейронные сети показывают высокое качество обработки изображений, однако вместе с этим допускается дальнейшее повышение их скорости обучения и точности предсказания посредством перехода от стратегии наращивания глубины к увеличению ширины архитектуры. В [7] авторы представляют результат такого решения – архитектуру WideResNet, представляющей более широкую конструкцию для ResNet-блоков. На рисунке 21 представлены два основных блока архитектуры ResNet. Ввиду того, что слой узкого горлышка (bottleneck) предназначен для сокращения входящей информации в следующий блок посредством увеличения числа карт признаков, то он приводит к сужению сети, и поэтому его роль снижается в WideResNet, и весь приоритет отдаётся базовому блоку (basic). Отметим, что на изображении неявно подразумеваются предварительные операции Batch-Normalization и ReLU.

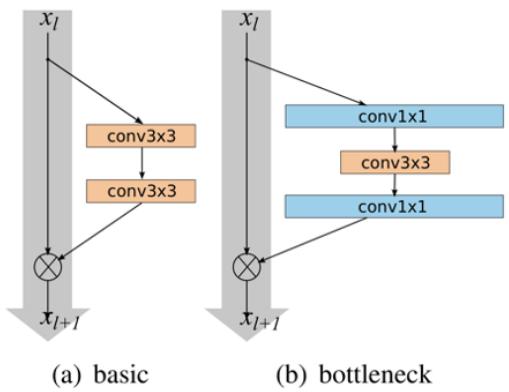


Рис. 21: Сравнение структуры блоков ResNet [7]

Для качественной модификации базового блока можно дополнить его несколькими операторами свёртки или вводом дополнительных признаковых наборов. Так же можно увеличить число фильтров свёрточных слоёв, что повысит ширину сети. Данный параметр регулируется введённым множителем  $k$ . В следующей таблице приведены слои архитектуры.

group name	output size	block type = $B(3, 3)$
conv1	$32 \times 32$	$[3 \times 3, 16]$
conv2	$32 \times 32$	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
conv3	$16 \times 16$	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$
conv4	$8 \times 8$	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
avg-pool	$1 \times 1$	$[8 \times 8]$

Рис. 22: Архитектура WideResNet [7]

Сеть состоит из входного свёрточного слоя  $3 \times 3$ , после которого идёт 3 группы по  $N$  блоков и операции average-pooling. Вторая, третья и четвёртая группы блоков допускают внешнюю настройку параметра расширения ( $k$ ). Он играет роль мультиплексора, на который домножается количество фильтров для свёрточного слоя. Как правило, достаточно использование небольших размеров фильтров. В ходе экспериментов варьировались параметр  $k$  и число свёрточных слоёв в блоке. Одним из наиболее эффективных (по числу параметров, времени обучения и доли ошибок) оказался блок с одним дополнительным  $1 \times 1$  свёрточным слоем. Самым эффективным является блок с двумя  $3 \times 3$  свёрточными слоями. Оптимальными мультиплексорами  $k$  являются значения 10 и 12. Как видно на следующем графике, WideResNet способна добиться улучшения качества относительно моделей класса архи-

тектурой ResNet. Помимо качества данная модель ускоряет процесс обучения благодаря расширению исходных сетей.

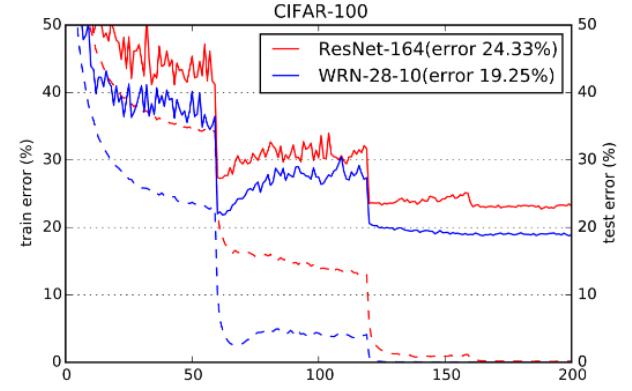


Рис. 23: Процесс обучения WideResNet [7]

Кроме того, как и в случае других сетей WideResNet допускает использование подхода drop-out, дополнительно повышающего качество модели благодаря регуляризации. В итоге сеть показывает следующие хорошие результаты на известных датасетах:

Dataset	model	dropout	test perf.
CIFAR-10	WRN-40-10	✓	3.8%
CIFAR-100	WRN-40-10	✓	18.3%
SVHN	WRN-16-8	✓	1.54%
ImageNet (single crop)	WRN-50-2-bottleneck		21.9% top-1, 5.79% top-5
COCO test-std	WRN-34-2		35.2 mAP

Рис. 24: Сравнение результатов обучения на разных датасетах [7]

Итак, WideResNet переиначивает взаимную структуру ResNet-блоков, ориентируясь на увеличение ширины сети. Это приводит к увеличению обучаемых возможностей модели и позволяет получать результат обработки быстрее и точнее.

### 3.2.6 EfficientNet

Отличным методом перестройки архитектуры для соблюдения баланса её объёма и эффективности является предложенный подход в работе [8]. Основная цель подхода заключается в сбалансированном упрощении сети по трём параметрам: глубине, ширине и мощности наборов обрабатываемых признаков (разрешение сети). Добиваясь снижения сложности сети можно ожидать значительный рывок в скорости обучения и вычислительным затратам. Задачу настройки масштабирования EfficientNet можно переписать в виде задачи условной максимизации точности на обучающих данных:

$$\begin{aligned}
& \max_{d,w,r} \quad \text{Accuracy}(\mathcal{N}(d, w, r)) \\
& s.t. \quad \mathcal{N}(d, w, r) = \bigodot_{i=1 \dots s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i}(X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i \rangle}) \\
& \quad \text{Memory}(\mathcal{N}) \leq \text{target\_memory} \\
& \quad \text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}
\end{aligned}$$

Рис. 25: Общая задача оптимизации [8]

Здесь варьируются три параметра, отвечающих за сложность нейронной сети:

- $d$  позволяет менять число одинаковых слоёв  $F_i$  в блоках сети, что влияет на её глубину;
- $w$  домножается на число каналов признаков, и регулирует ширину сети;
- $r$  влияет на размеры каждого канала, что позволяет варьировать разрешение пришедших в слой объектов;

Совокупность этих параметров определяет отличия новой, более эффективной модели от исходной. Проблема решения данной задачи является взаимосвязь между этими параметрами. При регулировки данных параметров независимо друг от друга получаются примерно одинаковые результаты: при усложнении сети по любому из параметров точность вместе с вычислительной сложностью повышается. Это видно на следующем рисунке:

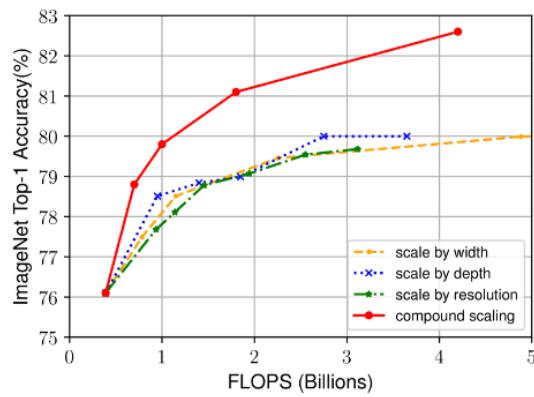


Рис. 26: Эффективность комплексной регулировки параметров сложности схемы [8]

Авторы статьи предлагают решать задачу комплексно, с помощью нового составного метода масштабирования сетей. Он задаётся параметром  $\phi$  и следующими соотношениями для параметров сложности схемы.

$$\begin{cases} \text{глубина : } d = \alpha^\phi, \\ \text{ширина : } w = \beta^\phi \\ \text{разрешение : } r = \gamma^\phi \\ \text{ограничение : } \alpha\beta^2\gamma^2 = 2 \end{cases}$$

Коэффициенты  $\alpha \geq 1$ ,  $\beta \geq 1$ ,  $\gamma \geq 1$  подбираются отдельно. Параметр  $\phi$  по семантике регулирует объём используемых ресурсов для реализации архитектуры. Коэффициент мультипликации 2 позволяет ввести ограничения на рост объёма операций работы модели при возрастании параметра  $\phi$ . Данное правило придерживается требования сохранять сбалансированность между параметрами модели и контролировать трудоёмкость вычислений.

Перейдём к алгоритму построения эффективной архитектуры. Он состоит из нескольких итераций, каждая из которых пересчитывает параметры  $\alpha, \beta, \gamma$  и  $\phi$  для улучшения модели. В качестве базовой модели была создана сеть EfficientNet-B0, состоящая из комбинации 9 наборов слоёв Batch-Normalization, пулинга, свёрток 3x3, 5x5. На каждом шаге алгоритма улучшения сначала фиксируется параметр  $\phi$  и перебираются  $\alpha, \beta, \gamma$  в соответствии с указанными условиями, затем в свою очередь они фиксируются и производится поиск оптимального значения  $\phi$ . На первой итерации параметр  $\phi$  считается равным 1.

В ходе экспериментов были получены следующие зависимости точности от сложности сети:

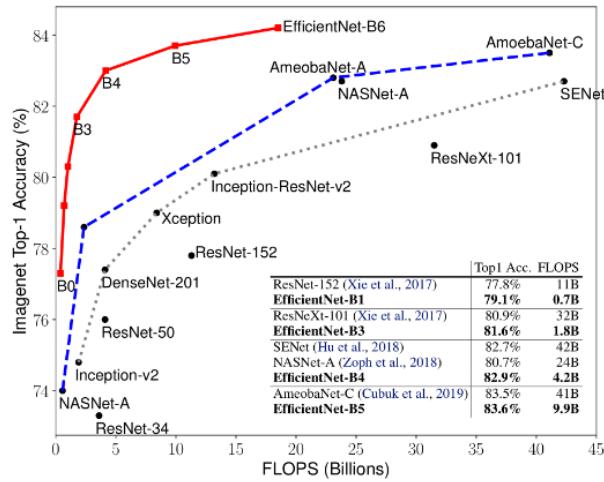


Рис. 27: Сравнение качества итерационных архитектур EfficientNet с другими моделями [8]

Видно, что настройка параметров в EfficientNet позволяет добиться значительного прироста в качестве. Но, что самое главное, эта архитектура требует малой вычислительной мощности благодаря соблюдению баланса между параметрами сложности сети. Кроме того, данный алгоритм настройки па-

раметров может быть использован и для других архитектур, аналогично повышая их производительность. Ещё одно доказательство преимущества комплексного сбалансированного усложнения сети проявляется на следующем сравнении наиболее важных участков изображения по анализу сети: именно совместная регулировка параметров позволяет получать наиболее важные участки для обработки, что сказывается на заметном росте качества.

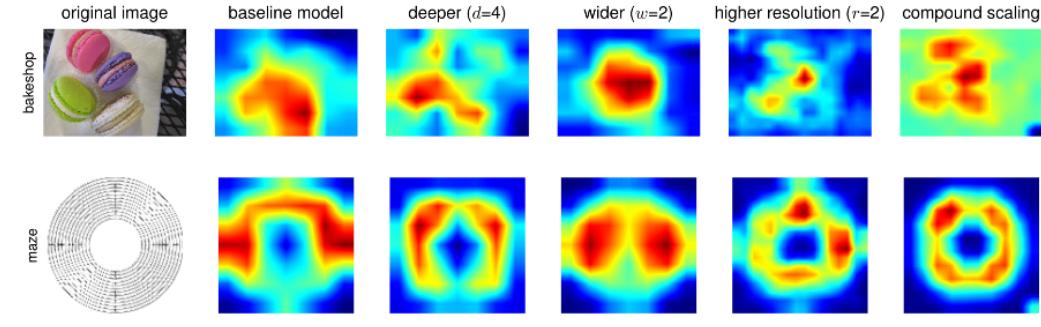


Рис. 28: HeatMap - распределение существенных образов при различных усложнениях модели [8]

Подытожим. Алгоритм для эффективного масштабирования моделей, предложенный в [8], доказывает свою состоятельность, позволяя получать точные и вычислительно более простые и производительные архитектуры нейронных сетей, что подтверждают эксперименты.

### 3.3 Снижение вычислительных ресурсов

Теперь рассмотрим группу архитектур, основной целью перед которыми стоит максимальное снижение числа параметров и требований мощности к используемой технике. Данные сети разрабатываются для лёгких вычислительных устройств и должны удовлетворять условиям быстрой работы и отклика. Для них допустимо небольшое снижение точности в угоду максимизации производительности.

#### 3.3.1 MobileNet

Специально для мобильных устройств, имеющих невысокий вычислительный потенциал и небольшие возможности памяти в работе [9] рассматривается архитектура MobileNet. Основной принцип её работы заключается в разделении обычного свёрточного слоя на две модифицированные составляющие: глубокую свёртку и точечную свёртку – Depthwise separable convolution. Если при обычной свёртке входящие в слой тензоры обрабатывались в общей совокупности разных каналов одним свёрточным ядром, то Depthwise separable convolution обрабатывает фильтрами каждый канал отдельно, а затем полученные тензоры “схлопывает” по числу каналов. Как показано на следующем

на рисунке, каждый из  $M$  слоёв обрабатывается своим ядром свёртки, после чего происходит поточечная свёртка с фильтром размера  $1 \times 1$  для регулирования числа каналов для каждого из  $N$  признаков выходных каналов.

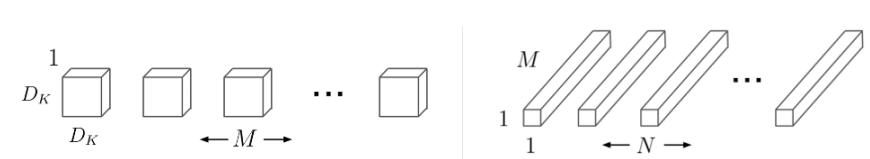


Рис. 29: Этапы свёртки DepthWise separable convolution [9]

Таким образом, обычный элемент свёртки в данной архитектуре усложняется до двух наборов операций: от последовательности  $\text{conv}(3 \times 3) + \text{BN} + \text{ReLU}$  происходит переход к элементу  $\text{Depthwise conv}(3 \times 3) + \text{BN} + \text{ReLU} + \text{Pointwise conv}(1 \times 1) + \text{BN} + \text{ReLU}$ .

Пусть  $-$  число каналов входного тензора,  $N$  – число каналов выходного тензора,  $D_K$  – линейный размер свёртки,  $D_F$  – линейный размер одного входного канала. Тогда стандартная операция свёртки оценивается по числу вычислений как  $D_K * D_K * D_F * D_F * M * N$ . При этом используемая модификация свёртки требует  $D_K * D_K * M * D_F * D_F + M * N * D_F * D_F$  вычислений, что на практике приводит к 8-9-кратному снижению вычислительных затрат.

Основная часть структуры сети состоит из таких свёрток за исключением первого стандартного свёрточного слоя. В конце сети присутствует оператор усреднённого пулинга, схлопывающий разрешение признаков (число каналов) до 1, после чего следует полносвязный свёрточный слой. В качестве функции нелинейности в первой версии использовалось преобразование ReLU6. Для дополнительного уменьшения множества параметров модели вводится два гиперпараметра-мультипликатора, отвечающие за снижение ширины ( $\alpha$ ) и числа обрабатываемых каналов сети ( $\rho$ ). При этом сложность вычислений модели можно переписать в следующем виде:

$$D_K * D_K * \alpha M * \rho D_F * D_F + \alpha M * \alpha N * \rho D_F * \rho D_F \quad (6)$$

Регулирование данных параметров может дополнительно повысить производительность нейронной сети. Эксперименты показали, что подход Depthwise separable convolution MobileNet способен работать с небольшим числом параметров, заметно ускоряя работу сети, и при этом показывать приемлемое качество обработки изображений:

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters	Layer/Modification	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3	Convolution	462	2.36
MobileNet	70.6%	569	4.2	Depthwise Separable Conv	52.3	0.27
				$\alpha = 0.75$	29.6	0.15
				$\rho = 0.714$	15.1	0.15

Рис. 30: Повышение компактности модели [9]

Настройка параметров  $\alpha$  и  $\rho$  приводит к дополнительному выигрышу по потребляемым вычислительным ресурсам модели. Благодаря этому сеть допускает использование и на небольших мобильных устройствах во многих приложениях.

После описанной архитектуры появлялись следующие версии MobileNet. Например, [10]. Во второй версии добавилось прокидывание связей, что увеличило пропускную способность сети и улучшило показатели качества сети. Кроме того, блок сети немного усложнился: добавилась одна  $1 \times 1$  линейная свёртка (узкое горлышко):

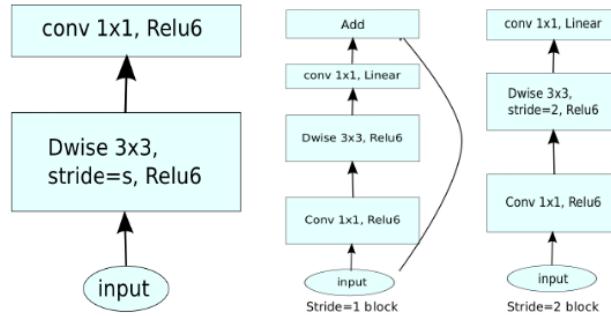


Рис. 31: Усложнение блоков MobileNet v2 [10]

Данные основные изменения положительно сказались на точности модели. Качество её работы увеличилось, что видно на следующем графике:

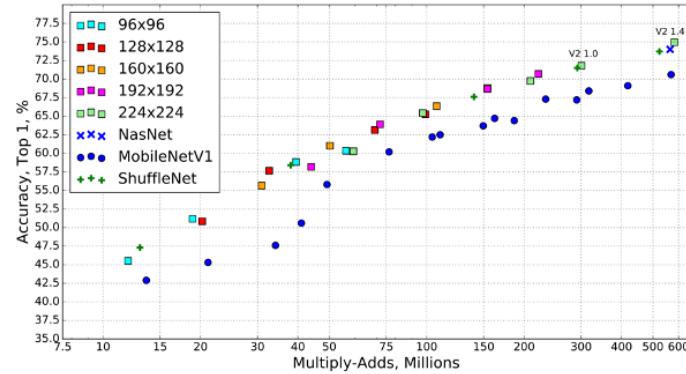


Рис. 32: Улучшение качества второй версии MobileNet [10]

В статье [11] авторы предлагают следующее развитие архитектуры. Дополнительные улучшения по эффективности коснулись преобразования некоторых сложно-вычислительных слоёв, а также заменена функция нелинейности (рисунок 33).

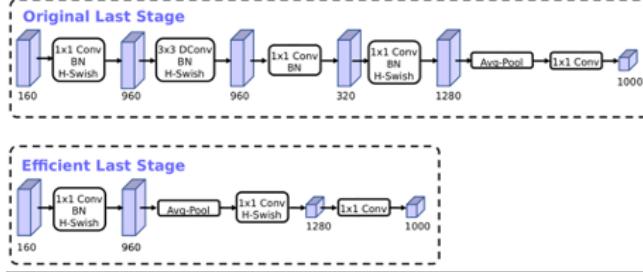


Рис. 33: Усложнение слоёв архитектуры третьей версии MobileNet [11]

Новая функция нелинейности вычисляется гораздо проще сигмоиды, при этом практически повторяет её форму. Плюс ко всему в архитектуру были добавлены модули squeeze-and-excitation, позволяющие среди глобальных признаков выделять информативные и эффективно использовать их для дальнейшей обработки. Такой блок является довольно гибким дополнением для свёрточных нейросетей. Данные изменения аналогично улучшили модель сравнении с предыдущей версией. Увеличилась точность модели, и уменьшилась задержка её отклика:

Backbone	mAP	Latency (ms)	Params (M)	MAdds (B)
V1	22.2	228	5.1	1.3
V2	22.1	162	4.3	0.80
MnasNet	23.0	174	4.88	0.84
V3	22.0	137	4.97	0.62
<b>V3<sup>†</sup></b>	22.0	119	3.22	0.51
V2 0.35	13.7	66	0.93	0.16
V2 0.5	16.6	79	1.54	0.27
MnasNet 0.35	15.6	68	1.02	0.18
MnasNet 0.5	18.5	85	1.68	0.29
V3-Small	16.0	52	2.49	0.21
<b>V3-Small<sup>†</sup></b>	16.1	43	1.77	0.16

Рис. 34: Сравнение версий MobileNet по длительности отклика и требуемым ресурсам [11]

MobileNet является ярким примером эффективных и лёгких свёрточных нейронных сетей, которые продолжают развиваться. Они потребляют небольшие вычислительные ресурсы и обладают свойством быстрого отклика. В этом проявляются их основные достоинства.

### 3.3.2 SqueezeNet

Следующая модель, описанная в статье [12], представляет реализацию нескольких архитектурных решений, значительно снижающих объём нейросети и сокращающих множество её параметров. При этих условиях удаётся поддерживать высокое качество обработки изображений на уровне известной свёрточной архитектуре AlexNet.

Среди самых главных используемых особенностей модели SqueezeNet выделяются следующие:

- Снижение размеров фильтров в слоях свёртки. Предлагается совершить замену 3x3-свёрток на 1x1.
- Помимо снижения сложности самих свёрток предварительно можно сократить число каналов, подаваемых на вход свёртки 3x3. Эта операция производится с помощью специальных сжимающих слоёв и аналогично снижает количество параметров сети (так как число параметров линейно зависит от этого количества каналов).
- Большие карты признаков, полученных после функции активации, могут при дальнейших вычислениях улучшить качество классификации. Ввиду этого, регулируя параметр stride для слоёв сети (ближе к концу сети), можно получить дополнительный прирост точности.

Для описания архитектуры SqueezeNet вводится в рассмотрение новый блок - Fire Module. Он состоит из двух частей: squeeze и expand. Squeeze производит предварительные “дешёвые” операции свёрток 1x1, после чего основная часть вычислений с использованием свёрток 1x1 и 3x3 выполняется в под-блоке expand. Результаты действий этих свёрток после функции активации конкатенируются и передаются на следующие слои сети. Число используемых свёрток разных типов регулируется отдельно. Функция активации ReLU производится после обоих под-блоков модуля. На рисунке представлен примерный вид Fire-модуля:

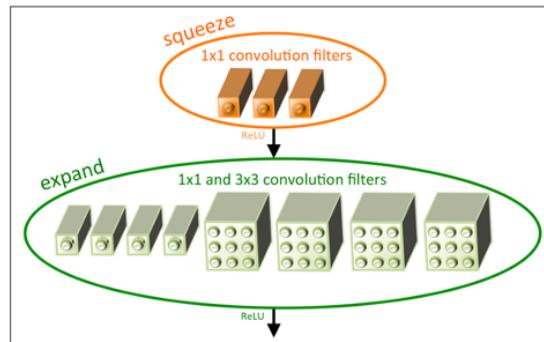


Рис. 35: Fire-модуль архитектуры SqueezeNet [12]

Реализация SqueezeNet состоит из 8 таких блоков, следующих за первым обычным свёрточным слоем. Данное решение позволяет обрабатывать карты признаков со значительным сокращением настраиваемых параметров, что заметно влияет на эффективность вычислений. Архитектура содержит только свёрточные слои (нет полно связной части – вместо неё выполняется операция глобального average pooling). Между группами разных Fire-модулей применяется операция max-pooling. Кроме того, существуют несколько модификаций SqueezeNet с развитием идеи прокидывания связей: simple/complex bypass. В реализации simple bypass прокинуты связи через некоторые fire-модули. В случае complex bypass добавлены новые связи с дополнительной операцией свёртки 1x1. В ходе экспериментов было выявлено, что наиболее оптимальная версия – simple bypass:

Architecture	Top-1 Accuracy	Top-5 Accuracy	Model Size
Vanilla SqueezeNet	57.5%	80.3%	4.8MB
SqueezeNet + Simple Bypass	<b>60.4%</b>	<b>82.5%</b>	4.8MB
SqueezeNet + Complex Bypass	58.8%	82.0%	7.7MB

Рис. 36: Сравнение модификаций с прокидыванием связей [12]

Следующей модификацией архитектуры является сеть SqueezeNext [13], в которой главную роль играет блок, представленный на следующем рисунке:

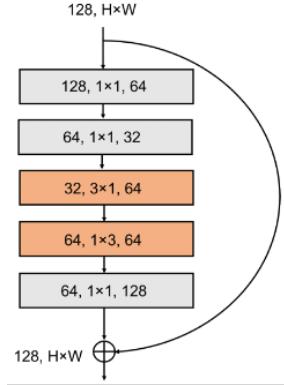


Рис. 37: Блок слоёв архитектуры SqueezeNext [13]

Здесь применяется факторизация свёртки с помощью пары одномерных свёрток, что позволяет снизить число параметров относительно использования обычных свёрток 3x3. Перед первым данным блоком в сети производится предварительная свёртка (+ stride=2 + max-pooling), сжимающее обрабатываемое изображение, этот подход обосновывается тем, что для мобильных структур не обязательно требование чёткого распознавания картинок с большим разрешением. Этим можно воспользоваться для дополнительного повышения эффективности. На следующем рисунке представлено сравнение

моделей SqueezeNext с некоторыми мобильными аналогами:

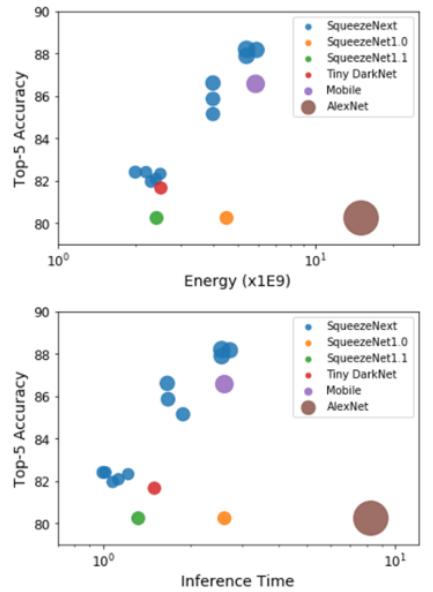


Рис. 38: Сравнение разных мобильных архитектур [13]

Как видно, модель SqueezeNext обладает хорошими показателями соотношения потребляемой энергии к точности, сравнимыми с MobileNet. Это свойство достигается многократным сокращением обучаемых параметров и необходимых вычислительных операций в процессе обучения и предсказания.

В общем случае достижение SqueezeNet заключается прежде всего в многократном уменьшении числа обучаемых сетью параметров и памяти, которую она занимает. При данных условиях модель способна показывать хорошее качество на уровне тяжёлых архитектур, таких как AlexNet.

### 3.3.3 ShuffleNet

Ещё одна реализация мобильных нейронных сетей описана в статье [14]. В данном случае используется идея групповых свёрток. Она заключается в том, что вычислительный объём архитектуры упрощается при использовании нескольких свёрток на подгруппах признаков из входных каналов (а не на всех каналах одновременно). Вместе с этим необходимо добиться распределения частей каналов по разным подгруппам для получения после свёртки полноценной картины (так как каждый канал может содержать независимую от других каналов информацию и необходимо при их обработке учитывать все доступные независимые наборы признаков). С этой задачей и призвана справиться архитектура ShuffleNet. Её основой являются блоки ShuffleNet Units, представленные на рисунке 39.

Данные блоки эксплуатируют DepthWise convolution (DWConv) и PointWise convolution (GConv), описанные ранее в докладе. Для корректного проки-

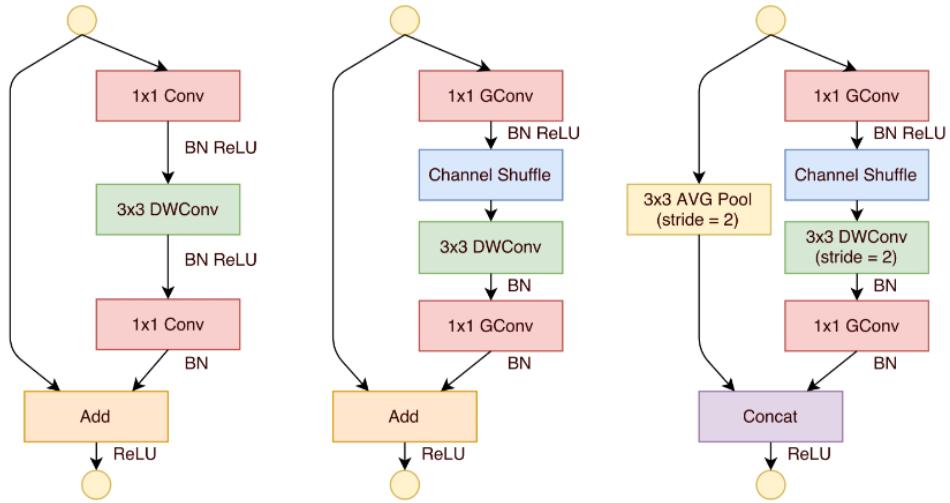


Рис. 39: Главные блоки ShuffleNet [14]

дывания связей используются  $1 \times 1$  свёртки, регулирующие число выходных каналов перед объединением тензоров. Второй и третий блоки обладают операторами для перемешивания частей каналов между групповыми свертками. Третий элемент дополнен опрацией stride (равного 2). Реализация перемешивания основывается на матричных трюках известных фреймворков с последовательным изменением размера тензора, посредством которой его элементы удаётся перемешать, как это требуется в основной идеи ShuffleNet. Из данных блоков и формируется рассматриваемая архитектура. Число разделённых групп признаков определено как гиперпараметр. Как правило, достаточно разделения признаков на 8 групп для достижения неплохого качества работы модели. Ниже производится сравнение этой архитектуры с другими моделями.

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 <sup>1</sup>	28.5	15300
ShuffleNet $2 \times (g = 3)$	26.3	<b>524</b>
GoogleNet	31.3	1500
ShuffleNet $1 \times (g = 8)$	32.4	<b>140</b>
AlexNet	42.8	720
SqueezeNet	42.5	833
ShuffleNet $0.5 \times (g = 4)$	41.6	<b>38</b>

Рис. 40: Облегчение модели по вычислительным ресурсам обучения [14]

Как видно, ShuffleNet показывает достойные результаты, как по вычислительной сложности, так и по доле ошибок на тестовом датасете. Кроме того, он демонстрирует многократное ускорение относительно работы крупногабаритных моделей. Более того, даже в сравнении с MobileNet данная архитекту-

ра демонстрирует более высокие показатели качества и производительности.

Существует модификация данной архитектуры, описанная в статье [15]. Основное отличие от основной версии заключается в небольшом изменении основного блока архитектуры.

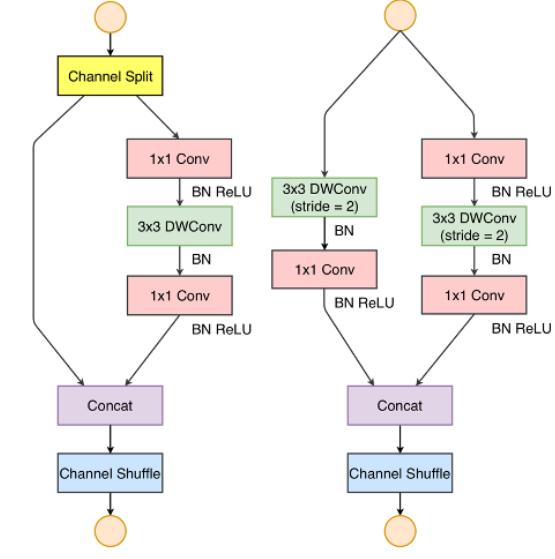


Рис. 41: Блоки второй версии ShuffleNet [15]

В данном случае часть признаков подаётся в конец блока без изменений, над другой частью производится операция свёртки. Оператор перемешивания каналов производится в конце блока после объединения признаков. Непосредственная передача части признаков в конец блока обеспечивает эффект повторного использования признаков, что может оказывать положительный эффект на точность дальнейшей обработки. Это свойство ведёт также к тому, что понятие групповой свёртки не используется в данном блоке – все операции свёртки производятся полностью на поданную часть входного тензора. Глобальное перемешивание позволяет повысить разнородность обрабатываемых признаков, сохраняя высокий темп вычислений. Данная версия способна показать ещё более хорошие показатели затрат вычислительных и мощностных ресурсов мобильных устройств.

### 3.3.4 FBNet

За последние годы набирает популярность новое направление в сфере построения нейронных сетей, называемое Neural Architecture Search. Его концепция связана на том, что оптимальную архитектуру нейронной сети может задать другая нейронная сеть. Описанная в работе [16] архитектура FBNet похожа по своим качествам на вторую версию MobileNet и получена представленной в статье модификацией NAS – Differentiable Neural Architecture Search – производящей поиск сети по конкретной решаемой задачи. Для поиска опти-

мальной архитектуры нейронной сети данный подход должен в общем случае решить задачу минимизации  $\min_{a \in A} (\min_{w_a} L(a, w_a))$ , то есть в пространстве допустимых архитектур А найти такую сеть, чтобы при обученных (полученных наилучших) весах достигался минимум функционала потерь рассматриваемой задачи. В качестве дополнительного условия требуется составить эффективный алгоритм поиска такой архитектуры. Для реализации данного алгоритма в начале вводится пространство допустимых сетей, как последовательность блоков, часть из которых можно выбирать из предопределённого множества наборов слоёв. У разных блоков-претендентов различаются типы свёрток, размер их фильтров, число разделений на группы для групповых свёрток. Конкретно в реализации работы [16] пространство архитектур содержит 22 произвольных слоя, каждый из которых может быть выбран из девяти вариантов. Это предоставляет огромное пространство для поиска лучшей сети. Для исключения полного перебора предлагается ввести генеральную функцию потерь, придерживаясь требований высокого качества сети и низкой задержки получаемой конструкции. Эти условия переходят в формулу потерь в следующем виде:

$$L(a, w_a) = CE(a, w_a) * \alpha \log(LAT(a))^\beta, \quad (7)$$

где СЕ – кросс-энтропийный критерий потерь на модели при фиксированных весах, LAT – характеристика длительности отклика. Параметры alpha и beta регулируют важность составляющей отклика в функционале потерь. Описав вычисление данных составляющих через параметры вероятностного распределения архитектурных блоков, удалось добиться дифференцируемости функционала потерь, как по весам модели, так и по данным параметрам распределения. После его оптимизации удалось получить архитектуру FBNet и её различные модификации. Сравнительная характеристика этих архитектур с другими существующими представлена в таблице:

Model	Search method	Search space	Search cost (GPU hours / relative)	#Params	#FLOPs	CPU Latency	Top-1 acc (%)
1.0-MobileNetV2	manual	-	-	3.4M	300M	21.7 ms	72.0
1.5-ShuffleNetV2	manual	-	-	3.5M	299M	22.0 ms	72.6
CondenseNet (G=C=8)	manual	-	-	2.9M	274M	28.4 <sup>‡</sup> ms	71.0
MnasNet-65	RL	stage-wise	91K* / 421x	3.6M	270M	-	73.0
DARTS	gradient	cell	288 / 1.33x	4.9M	595M	-	<b>73.1</b>
FBNet-A (ours)	gradient	layer-wise	216 / 1.0x	4.3M	<b>249M</b>	<b>19.8 ms</b>	73.0
1.3-MobileNetV2	manual	-	-	5.3M	509M	33.8 ms	<b>74.4</b>
CondenseNet (G=C=4)	manual	-	-	4.8M	529M	28.7 <sup>‡</sup> ms	73.8
MnasNet	RL	stage-wise	91K* / 421x	4.2M	317M	23.7 ms	74.0
NASNet-A	RL	cell	48K / 222x	5.3M	564M	-	74.0
PNASNet	SMBO	cell	6K <sup>†</sup> / 27.8x	5.1M	588M	-	74.2
FBNet-B (ours)	gradient	layer-wise	216 / 1.0x	4.5M	<b>295M</b>	<b>23.1 ms</b>	74.1
1.4-MobileNetV2	manual	-	-	6.9M	585M	37.4 ms	74.7
2.0-ShuffleNetV2	manual	-	-	7.4M	591M	33.3 ms	<b>74.9</b>
MnasNet-92	RL	stage-wise	91K* / 421x	4.4M	388M	-	74.8
FBNet-C (ours)	gradient	layer-wise	216 / 1.0x	5.5M	<b>375M</b>	<b>28.1 ms</b>	<b>74.9</b>

Рис. 42: Сравнение различных нейросетей с FBNet [16]

Как видно, подход DNAS позволяет искать эффективные по производительности архитектуры, показывающие высокое качество. При этом поиск таких сетей производится как решение отдельной задачи минимизации с помощью отдельной нейросети. Результаты полученных архитектуры схожи с MobileNet v2, что является интересным достижением, значащим, что в свою очередь и MobileNet может являться близкой к оптимальной конструкции сетью. Подход DNAS позволяет эффективно генерировать структуры для разного типа мобильных устройств, что безусловно является его преимуществом.

### 3.3.5 RevNet и iRevNet

Рассмотренные до этого момента сети концентрировались на решении вопроса снижения необходимых вычислительных ресурсов для продуктивной работы сети: число обучаемых параметров, функциональная и архитектурная сложность сети, требуемая мощность устройств для использования данных архитектур. Однако одним из других важных типов затрат на обучение сети является память, затрачиваемая на хранение накопленных в слоях входящей информации для обновления параметров модели на этапе обратного распространения. Такие данные могут занимать огромные ресурсы памяти, что так же необходимо контролировать или вовсе сокращать. Авторы работы [17] предлагают использовать подход обратимых вычислений для того, чтобы избавиться от необходимости затрачивать большие объёмы памяти на этапе обучения.

Идея метода описывается следующими выражениями:

$$\begin{aligned} y_1 &= x_1 + \mathcal{F}(x_2) & x_2 &= y_2 - \mathcal{G}(y_1) \\ y_2 &= x_2 + \mathcal{G}(y_1) & x_1 &= y_1 - \mathcal{F}(x_2) \end{aligned}$$

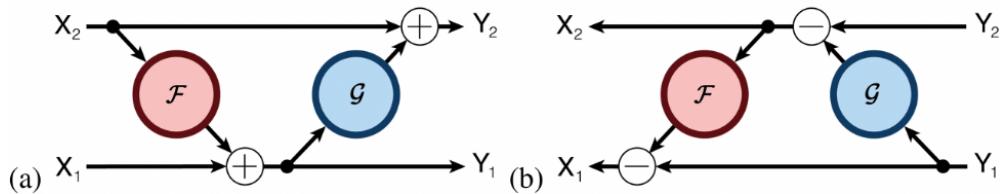


Рис. 43: Обратимые подходы в RevNet [17]

То есть на этапе обратного распространения предполагается восстанавливать входные данные через полученный с последующего слоя потока информации. В идеальном случае обратное распространение будет производиться без необходимости выделения памяти на прямом проходе сети. Естественно, такой вариант на практике неосуществим ввиду того, что из сетей нельзя выкидывать некоторые необратимые преобразования, такие как пулинг, свёртки с шагом. Например, содержание сети операторов down-sampling'a позволяют ей более эффективно обрабатывать множество признаков, и убирать их в

угоду обратимости вычислений было бы неразумно. Соответственно требуется оставить данные необходимые элементы (их не будет много и памяти на хранение потребуется не так много) и добавить эффект обратимости там, где это возможно. В качестве бейзлайна была взята сеть ResNet и по указанным формальным правилам в неё были внедрены участки обратимых вычислений. При заданном снижении необходимой памяти для обучения полученная сеть показала весьма хорошие результаты, немного уступив по качеству исходной модели:

Architecture	CIFAR-10		CIFAR-100	
	ResNet	RevNet	ResNet	RevNet
32 (38)	<b>7.14%</b>	7.24%	29.95%	<b>28.96%</b>
110	<b>5.74%</b>	5.76%	26.44%	<b>25.40%</b>
164	5.24%	<b>5.17%</b>	<b>23.37%</b>	23.69%

Рис. 44: Результаты работы (доля ошибок) [17]

Таким образом, использование обратимых вычислений способно значительно снизить требуемую память для этапа обратного распространения, и показывать высокую точность. На следующем графике представлен процесс обучения модели RevNet. По нему видно, что уже при недолгом обучении архитектура выдаёт практически такое же качество, как и в случае ResNet.

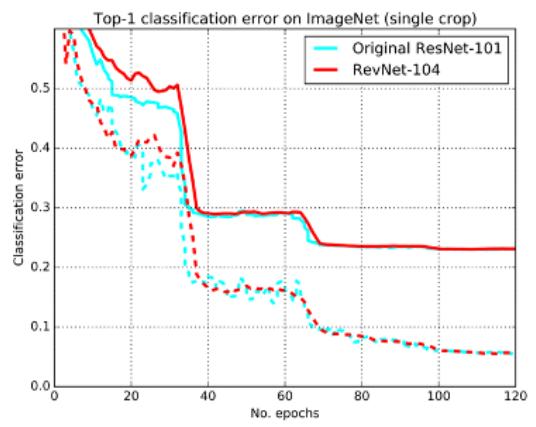


Рис. 45: Процесс обучения модели [17]

Дальнейшее развитие идей сети RevNet были реализованы в архитектуре iRevNet [18]. В данной работе подразумевается переход структуры сети на полностью обратимые вычислительные каналы. При данном переходе удается сохранить высокое качество обработки изображений и дополнительно сократить ресурсы требуемой памяти. По сути одновременно устройство архитектуры и алгоритма обучения схематично изображена на следующем рисунке:

Вход сети разделяется на две равные части –  $x_0$  и  $\tilde{x}_0$  с помощью линейного оператора. После этого над одним из тензоров производится преобразова-

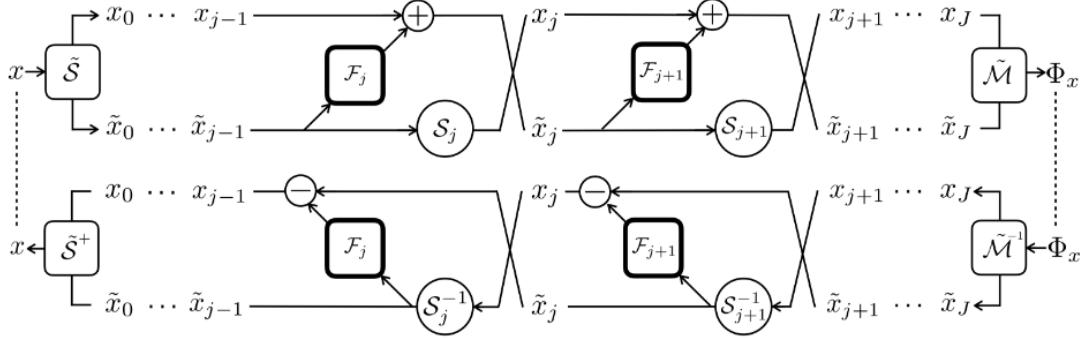


Рис. 46: Схема архитектуры iRevNet [18]

ние  $F_j$ , результаты объединяются и после дополнительного инвертируемого оператора  $S_j$  (выполняющего роль down-sampling'a) происходит дальнейшее деление необработанной части входного тензора. Блок  $F_j$  выполняет свёртку и нелинейные преобразования. В конце сети полученные обработанные признаки конкатенируются, результат усредняется вдоль пространственных размерностей и производится операция ReLU, после чего линейно преобразованные признаки подаются в классификатор. При обратном проходе сети используются следующие соотношения:

$$\begin{cases} x_{j+1} = \mathcal{S}_{j+1}\tilde{x}_j \\ \tilde{x}_{j+1} = x_j + \mathcal{F}_{j+1}\tilde{x}_j \end{cases} \iff \begin{cases} \tilde{x}_j = \mathcal{S}_{j+1}^{-1}x_{j+1} \\ x_j = \tilde{x}_{j+1} - \mathcal{F}_{j+1}\tilde{x}_j \end{cases}$$

Рис. 47: Соотношения для обратимых вычислений в iRevNet [18]

Вопрос, как оператор  $S_j$  способен обратимо снижать размерность признакового пространства, разрешается использованием разбиения карты признаков на непересекающиеся подмножества:

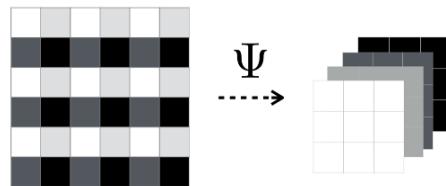


Рис. 48: Обратимая реализация down-sampling'a [18]

Данное преобразование сохраняет пространственную упорядоченность признаков, что позволяет избежать некорректное смешивание при последующих

действиях свёрточных слоёв. Авторами статьи было реализовано две модели: инъективная – более точная - и биективная – содержащая лишь небольшое число настраиваемых параметров. Сравнение этих моделей с основной RevNet произведено в следующей таблице:

<b>Architecture</b>	<b>Injective</b>	<b>Bijective</b>	<b>Top-1 error</b>	<b>Parameters</b>
ResNet	-	-	24.7	26M
RevNet	-	-	25.2	28M
<i>i</i> -RevNet (a)	yes	-	24.7	181M
<i>i</i> -RevNet (b)	yes	yes	26.7	29M

Рис. 49: Результаты работы и сравнение с RevNet [18]

Итак, архитектуры типа iRevNet демонстрируют хорошее качество результата, при этом в различных модификациях способны развивать идеи обратимых вычислений, позволяющих сократить расход памяти, необходимой для вычислений обратного прохода по сети во время её обучения.

### 3.4 Некоторые новые архитектуры

В данном небольшом подразделе кратко осветим новые архитектурные решения, разработанные в 2019-2020 годах. Многие из них модифицируются до новых версий, обходя по эффективности и качеству рассмотренные ранее нейросети.

#### 3.4.1 SpineNet

Свёрточные нейронные сети показывают отличные возможности в задаче классификации объектов на изображениях, однако довольно тяжело справляются с локализацией объектов. Авторы работы [20] предлагают модифицировать применение пары кодировщик-декодировщик для эффективного решения указанного вопроса. С помощью поиска эффективной архитектуры для задачи детектирования объектов была найдена архитектура, состоящая из двух подсетей: первая (базовая) сеть является фиксированной, вторая - предобучена и допускает масштабированную перестановку собственных слоёв. Блоки второй подсети входят в предопределённое множество допустимых слоёв.

Поиск эффективной архитектуры производится в несколько этапов.

Сначала ищется наилучшая перестановка допустимых блоков. Каждый из блоков сети имеет две связи с прошлыми блоками. На втором этапе производится поиск наиболее выгодных связей блока с прошлыми, и, таким образом, удаётся добиться более эффективной совместной обработки признаков с разных уровней сети. Далее для каждого блока дополнительно определяется его масштабная сложность и внутренняя структура: по типу узкого горлышка или обычного блока ResNet. Для соблюдения корректного объединения тен-

зоров на этапе масштабирования связей используется специальные операции пересемплирования.

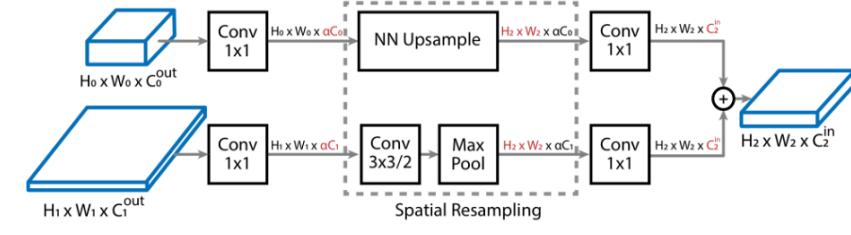


Рис. 50: Модуль пространственного ресемплинга (up-sampling'a верхнего и down-sampling'a нижнего тензоров перед их объединением) [20]

На следующем рисунке изображён пример перестановки блоков ResNet для повышения эффективности. Архитектура SpineNet допускает усложнение (по глубине) посредством дублирования комбинаций блоков.

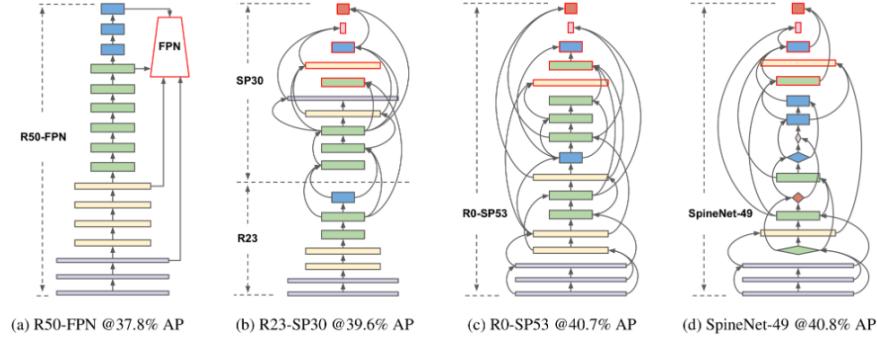


Рис. 51: Этап масштабируемой перестановки блоков [20]

Перестановочность и масштабируемость блоков сети, предложенные в данной статье, повышают эффективность обучения. Одну из определяющих ролей в таком результате играет дополнительная настройка блоков по их собственной сложности, что видно по статистике:

model	block adju.	#FLOPs	AP
R50-FPN	-	96.8B	37.8
R35-SP18	-	91.7B	38.7
R23-SP30	-	96.5B	39.7
R14-SP39	-	99.7B	39.6
R0-SP53	-	95.2B	40.7
SpineNet-49	✓	85.4B	40.8

Рис. 52: Снижение ресурсов для обучения относительно других сетей [20]

### 3.4.2 ThunderNet

Другим примером сети-детектора, опережающего по эффективности популярные легковесные сети, является описанным в [21] ThunderNet. Как утверждают авторы статьи, эта сеть способна работать в реальном времени и локализовать объекты потока изображений в 24 кадра в секунду.

Как и в прошлой архитектуре, описываемый подход эксплуатирует так называемые "магистральные" сети (backbone networks). Их принцип заключается в том, что ранние слои сети обрабатывают более крупные карты признаков, а для последующих слоёв карты признаков уменьшаются, обрабатывая более комплексные структуры объектов. Для задачи классификации как раз подходят более компактные карты признаков, содержащие более общую информацию о частях изображения. В то же время задачу детектирования объектов легче разрешить, используя более конкретные признаки, соответствующие первым слоям сети. В соответствии с этими идеями был взят блок ShuffleNetV2 и модифицирован до более лёгковесной версии, и в итоге благодаря добавлению эффективной обработки низкоуровневых признаков на первых слоях удалось повысить их значимость, что улучшает качество последующего после детектирования.

Часть детектирования сети использует модифицированный подход сетей семейства RPN (Region Proposal Networks). Данные сети обрабатывают результаты свёрточных слоёв для определения фрагментов изображения, содержащих целевые объекты. Такая сеть в статье сжимается с помощью замены обычных свёрток на DepthWise convolution - свёртки. Помимо этого, увеличивается размер ядра свёрточных слоёв, что позволяет агрегировать информацию более комплексно и затем более эффективно обнаруживать объекты. Так же в блоке детектирования используется модуль для увеличения роли контекста (Context Enhancement Module), агрегирующий информацию глобальных и низкоуровневых признаков, и модуль пространственного внимания (Spatial Attention Module). Он позволяет выделить объекты на переднем плане и снизить роль фона, что стабилизирует обучения используемых под-архитектур.

Общий вид архитектуры ThunderNet представлен на следующем рисунке. Сначала идёт "магистральная" часть сети, затем блок детектирования с использованием дополнительных описанных модулей.

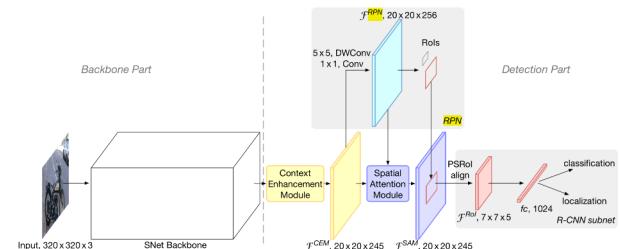


Рис. 53: Архитектура ThunderNet [21]

На следующем рисунке представлен результат работы сети. По нему видно, что качество обоих задач (детектирования и классификации) довольно высоко.

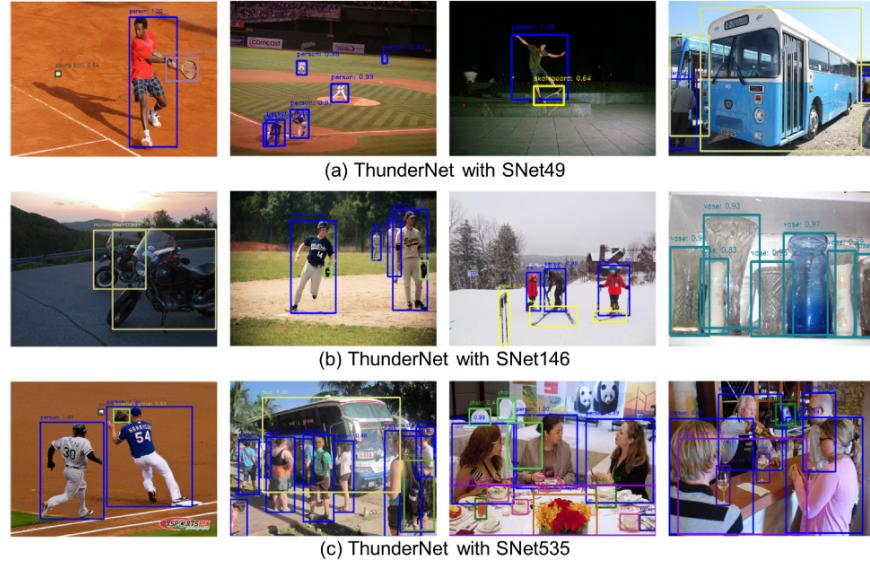


Рис. 54: Результаты детектирования и классификации [21]

### 3.4.3 CSPNet

Для повышения вычислительной эффективности свёрточных сетей в работе [22] предлагают модернизированную обработку специального набора признаков (пирамид признаков). Cross Stage Partial Network (CSPNet) обеспечивает возможность многосетевого распространения градиента при обучении. Эта идея обосновывается закономерностью, что при корректном разделении сети на несколько вычислительных путей и их конкатенации значения градиента из разных каналов будут заметно коррелировать. Это не только позволяет сократить объём вычислений при обучении сети, но и позволяет повысить точность благодаря более сложной обработке карт признаков с помощью различных градиентов функции потерь.

Для такого разделительного прокидывания связей в начале усложняется Dense-блок - добавляется разделение каналов входного тензора и обработка одной из них (рисунок 55). В новом виде он может увеличить число вычислительных путей для градиента, благодаря предварительному делению тензора соблюдать вычислительный баланс для слоя, а также снизить объём хранимой памяти.

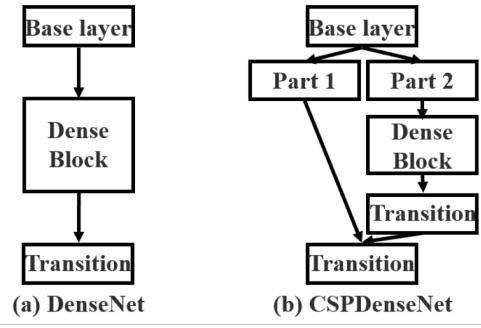


Рис. 55: Изменение Dense-блока [22]

Дополнительный слой перехода (transition layer) в этом блоке позволяет увеличить разницу между разными значениями градиентами. Это позволит предотвратить идентичное обучение для разных слоёв. Данная стратегия применима не только к блокам семейства DenseNet, но к другим архитектурам (ResNet и ResNext).

Представление признаков в виде пирамиды связано с возможностью повысить качество модели с помощью агрегированной обработки уровней этой пирамиды с помощью разных приходящих в них градиентов. Пирамида признаков содержит и низко- и высокоуровневые признаки, что при совместном эффективном анализе могут влиять на рост качества детектирования. Концептуально для каждого уровня пирамиды признаков происходит разделение путей извлекаемых признаков, что и приводит к разделённому обучению различными способами.

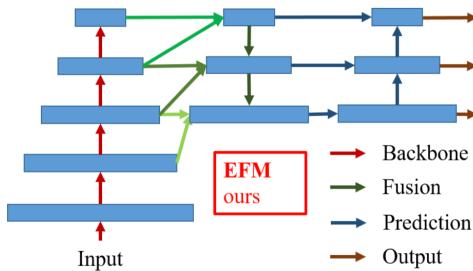


Рис. 56: Применяемая стратегия связей пирамидальной структуры признаков в CSPNet [22]

При использовании CSPNet вычислительная эффективность обучения повышается на 10%. При этом качество модели относительно бейзлайна не уменьшается и в некоторых случаях растёт. Кроме того, модель хорошо справляется с детектированием объектов в реальном времени.

Модификация CSPDenseNetb - самая быстрая из других детекторов в режиме реального времени.

Итак, благодаря разделению градиента на несколько вычислительных путей, способу пересечения связей в архитектуре для агрегированной обработки признаков разного уровня удалось облегчить модель, тем самым повысив скорость детектирования и обработки изображений в реальном времени.

#### 3.4.4 SAUNet

Примером свёрточной нейросети для решения задачи сегментации является описанная в статье [23] сеть SAUNet, применяющаяся медицине. На момент 2020 года она показывает наилучшие результаты для сегментации изображений МРТ.

Как говорят авторы, несмотря на высокую эффективность и точность свёрточных нейронных сетей, для использования в медицинской сфере они выдают недостаточно надёжные для интерпретируемости результаты, так как основываются больше на текстурах выделяемых объектах, а не их формах, что в медицине играет более важную роль. Для решения этой проблемы была представлена новая архитектура Shape Attentive U-Net (SAUNet), отдающая приоритет надёжности результатов.

Основной её принцип состоит в том, что параллельно с потоком данных, описывающих текстурные свойства объектов, модель работает с признаками формы. Помимо использования моделируемой сетью карт таких признаков для дальнейших преобразований, данные признаки более интерпретируемы, что полезно в данной задаче. За основу архитектуры взята U-Net, в декодер которой встроен специальный модуль для обработки пространственных и канальных преобразований.

Вся структура изображена на следующей схеме:

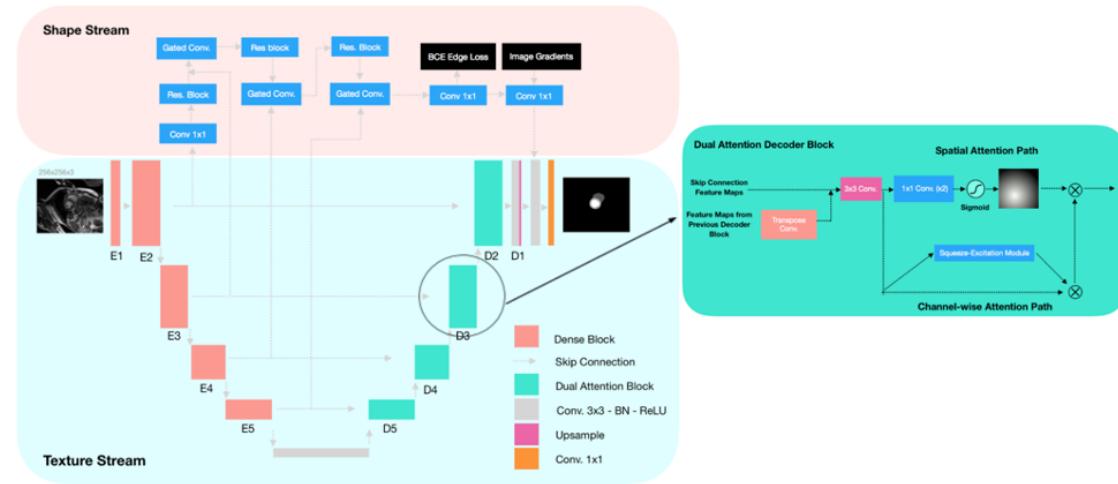


Рис. 57: Архитектура SAUNet [23]

SAUNet состоит из двух частей: одна часть обрабатывает текстуры (texture stream); вторая часть обрабатывает формы (gated shape stream). В кодиров-

щике текстур сети U-Net используются блоки семейства DenseNet, а в декодере в качестве слоев сети U-Net применяются указанные выше «блоки внимания» (dual attention decoder block). Обучение происходит по правилу минимизации функционала потерь, настраивающего сеть на качественную обработку формы изображений:

$$L_{total} = \lambda_1 L_{CE} + \lambda_2 L_{Dice} + \lambda_3 L_{Edge}, \quad (8)$$

где  $L_{Dice}$  характеризует качество пересечения истинных объектов с предсказанием детектирующей части модели.  $L_{CE}$  обозначает значение кросс-энтропийного критерия ошибки для задачи сегментации,  $L_{Edge}$  показывает несостыковку определения моделью границ выделенных объектов. Все совокупности данные слагаемые позволяют добиться укрепления требования надёжности анализа формы и текстур на изображениях для более качественного предсказания.

Эксперименты показали состоятельность данной архитектуры как в задаче сегментации, так и в вопросе надёжности результата для интерпретируемости действительно сложных изображений.

## 4 Выводы

В данном докладе были рассмотрены основные архитектуры свёрточных нейронных сетей, появившихся после основоположных моделей сферы обработки изображений. Данные сети представляют практический интерес, позволяя снижать затраты на процессе обучения и сохраняя высокую точность классификации. Были описаны архитектурные решения, предоставляющие возможность для сбалансированного изменения ширины, глубины, разрешения сети для наиболее эффективной реализации, сжатия сетей для значительного снижения числа их параметров и требуемой памяти, а также снижения требований мощности вычислительной техники, что позволяет использовать лёгкие нейросети на мобильных устройствах. Данные архитектуры продолжают развиваться, повышая эффективность сетей и дальше. В этом заключается их ценность.

## Список литературы

- [1] Min Lin, Qiang Chen, Shuicheng Yan, "Network In Network" , 2014
- [2] David Ha, Andrew Dai, Quoc V. Le, "HYPERNETWORKS" , 2016
- [3] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Q. Weinberger, "Deep Networks with Stochastic Depth" , 2016
- [4] Gustav Larsson, Michael Maire, Grefory Shakhnarovich, "FRACTALNET: ULTRA-DEEP NEURAL NETWORKS WITHOUT RESIDUALS" , 2016
- [5] Leslie N. Smith, Nicholay Topin, "DEEP CONVOLUTIONAL NEURAL NETWORK DESIGN PATTERNS" , 2016
- [6] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, "Densely Connected Convolutional Networks" , 2018
- [7] Sergey Zagoruyko, Nikos Komodakis, "Wide Residual Networks" , 2016
- [8] Mingxing Tan, Quoc V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" , 2019
- [9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" , 2017
- [10] Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks" , 2019
- [11] Andrew Howard Mark Sandler Grace Chu Liang-Chieh Chen Bo Chen Mingxing Tan Weijun Wang Yukun Zhu Ruoming Pang Vijay Vasudevan Quoc V. Le Hartwig Adam, "Searching for MobileNetV3" , 2019
- [12] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer, "SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE" , 2017
- [13] : Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, Kurt Keutzer EECS, UC Berkeley, "SqueezeNext: Hardware-Aware Neural Network Design" , 2018
- [14] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices" , 2017
- [15] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design" , 2018

- [16] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search" , 2018
- [17] Aidan N. Gomez, Mengye Ren, Raquel Urtasun, Roger B. Grosse "The Reversible Residual Network: Backpropagation Without Storing Activations" , 2017
- [18] Jörn-Henrik Jacobsen, Arnold Smeulders, Edouard Oyallon "i-REVNET: DEEP INVERTIBLE NETWORKS" , 2018
- [19] Викиконспекты // Свёрточные нейронные сети, URL: [https://neerc.ifmo.ru/wiki/index.php?title=Сверточные\\_нейронные\\_сети](https://neerc.ifmo.ru/wiki/index.php?title=Сверточные_нейронные_сети)
- [20] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, Xiaodan Song, "SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization" , 2019
- [21] Zheng Qin, Zeming Li, Zhaoning Zhang, Yiping Bao, Gang Yu, Yuxing Peng, Jian Sun, "ThunderNet: Towards Real-time Generic Object Detection" , 2019
- [22] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, "CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN" , 2019
- [23] Jesse Sun, Fatemeh Darbehani, Mark Zaidi, Bo Wang, "SAUNet: Shape Attentive U-Net for Interpretable Medical Image Segmentation" , 2020