

Санкт-Петербургский государственный университет

Кафедра технологий программирования

Ежов Федор Валерьевич

Выпускная квалификационная работа бакалавра

**Сравнение архитектур нейронных сетей в задаче
сегментации фигуры человека**

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005.2016 «Прикладная
математика, фундаментальная информатика и программирование»

Профиль: «Математическое и программное обеспечение вычислительных
машин»

Научный руководитель,
старший преподаватель
Стученков А. Б.

Содержание

Введение	2
Постановка задачи	4
Обзор литературы	5
Глава 1. Теоретические основы применяемых нейросетевых технологий	7
1.1. Fully Convolution Network	7
1.2. Архитектура FCN	8
1.3. DeepLabV3	11
1.4. Atrous Convolution	12
1.5. Архитектура DeepLabV3	15
1.6. DeepLabV3 Plus	17
1.7. Архитектура DeepLabV3 Plus	18
1.8. Датасеты	19
1.9. Метрики	21
1.10. Выводы	23
Глава 2. Особенности реализации решения	24
2.1. Используемые технологии и средства разработки	24
2.2. Программа	25
2.3. Система тестирования	27
Глава 3. Результаты тестирования полученного решения	29
Выводы	33
Заключение	34
Список литературы	35

Введение

В настоящее время в мире становится все больше разной информации. Человек использует ее для получения различного рода выводов, прогнозирования будущего и нахождения закономерностей.

Из-за большого роста данных с каждым днем, становится очевидно что нужна автоматизация обработки этих данных. Человек не может “переварить” весь информационных поток, который только нарастает, людям нужен инструмент, который позволит делать выводы основываясь на данных. Такой инструмент предоставляют алгоритмы машинного обучения и нейронные сети. Так же как в свое время машины заменили людской грубый труд на фабриках, эти алгоритмы способны заменить людей в задачах по работе с данными.

Computer Vision (CV) - очень важное направление по работе с данными исследуемое сейчас. Алгоритмы CV помогают автоматизировать работу с неструктурированными данными такими как изображение и видео. Алгоритм, “просматривая” изображение, может молниеносно обработать его и сделать соответствующие выводы. Например, можно научить алгоритм распознавать разные объекты на изображении или же находить точное местоположение этих объектов на изображении, отслеживать машины в видеопотоке, распознавать лица людей и так далее. Все эти алгоритмы используются в таких важных проектах, как умный магазин, автоматизированный транспорт, разблокировка телефона с

помощью камеры, отслеживания преступников с помощью камер наблюдения в больших городах и в других не менее значимых задачах.

Семантическая сегментация - очень важная и интересная задача в области CV. В рамках данной задачи алгоритму требуется найти так называемую “маску” объекта на изображении, то есть нужно выделить только те пиксели, которые принадлежат интересующему нас объекту, для дальнейшего оперирования ими. Данную задачу решают в таких проектах, как автоматического управления транспортом, обводки контуром фигуры человека на камерах слежения, замены фона за человеком на фотографии или видео и других.

Постановка задачи

Целью научно-исследовательской работы является изучение и сравнение существующих нейросетевых решений задачи сегментации фигуры человека, их модификаций и возможность использования этих алгоритмов в реальном времени. Поставленная цель определила следующие задачи:

1. Ознакомиться с существующими решениями задачи семантической сегментации.
2. Отобрать несколько нейросетевых моделей для сравнения.
3. Изучить преимущества и недостатки каждой модели. Проверить возможность приемлемой работы каждой модели в реальном времени.
4. Собрать тестовый датасет для проверки качества работы выбранных нейронных сетей.
5. Реализовать программу на языке Python, позволяющую провести тесты и сравнение нейронных сетей на тестовых изображениях и видео в задаче сегментации фигуры человека.
6. Провести тестирование реализованного решения.

Обзор литературы

В статье [8] (К. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014) описана 16-слойная сверточная нейронная сеть VGG16. Данная сеть часто используется в качестве encoder для извлечения ключевой информации из изображения. В статье подробно рассмотрена архитектура нейронной сети, метод тренировки и результаты ее работы на различных датасетах.

В работе [9] (He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, 2016) рассмотрены сети семейства ResNet и их фирменными Residual Blocks. ResNet используются для решения задач классификации, а также в качестве encoder. В отличие от VGG16, ResNet могут достигать до 152 слоев в глубину. Обучать такие большие сети становится возможным с помощью Residual Blocks. В статье подробно рассмотрена архитектура сетей ResNet, обучение и результаты на различных датасетах.

В статье [13] (Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML, 2015) подробно рассмотрен слой Batch Normalization, который позволяет ускорить вычисления нейронной сети и избежать переобучения.

В статье [16] (Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR, 2017) рассмотрена нейронная сеть

Хception и слой Depthwise Separable Convolution. Сеть Хception используется как encoder в DeepLabV3 Plus, все слои свертки заменены на слои Depthwise Separable Convolution. Данная свертка ускоряет выполнение вычислений.

В статьях [4] (L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” in ICLR, 2015) и [5] (L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv:1606.00915, 2016) рассмотрены сети DeepLab и DeepLabV2. Подробно описаны архитектура нейронных сетей, слой atrous convolution, а также результаты работы сетей с разными параметрами на различных датасетах.

Глава 1. Теоретические основы применяемых нейросетевых технологий

Для решения поставленной задачи были решено рассмотреть перспективные нейронные сети Fully Convolution Network (FCN) [3], DeepLabV3 [6] и DeepLabV3 Plus [7].

1.1. Fully Convolution Network

Fully Convolution Network (FCN) - как следует из названия полностью сверточная нейронная сеть. Изначально данная сеть была ориентирована на решение задач классификации изображений. Но в 2014 году была представлена ее адаптация, которая позволила решать задачи семантической сегментации.

FCN - позволяет находить “маски” объектов end-to-end, то есть без предобработки и постобработки изображения. Так как сеть полностью сверточная, это делает ее инвариантной по отношению к размеру изображения. Нейронную сеть можно разделить на две части. Первая часть выделяет важную информацию с помощью сверточных слоев, вторая же с помощью слоев “upsampling” увеличивает размерность полученной карты признаков до размера изначального изображения. В качестве первой части могут использоваться предобученные нейронные сети, такие как VGG16 [8], ResNet50 [9], ResNet101 [9].

Нейронная сеть для решения поставленной задачи была обучена на датасете Pascal VOC 2012 [1]. Сеть обучалась “с учителем”. В качестве функции потерь FCN использует попиксельную кросс-энтропию.

1.2. Архитектура FCN

FCN - изначально решала задачи классификации изображений. В классификации, как правило, входное изображение уменьшается и проходит через сверточные слои (convolution layers) и полносвязанные слои (fully connected layers), а также выводит одну предсказанную метку для входного изображения, как показано ниже (рис. 1):

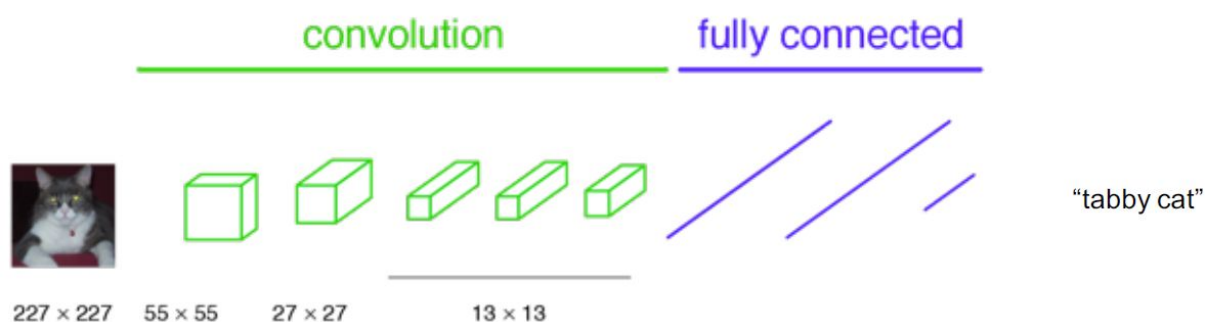


Рис. 1. Архитектура FCN для решения задач классификации изображений

Заменяя полносвязные слои на сверточные слои размера 1×1 , вместо одной метки для всего изображения мы получим тензор $W \times H \times N$. Где W - это ширина, а H - высота получившейся карты признаков, а N - количество классов, которое модель способна сегментировать. Ширина и высота будут меньше, чем изначальное изображение, в зависимости от того, через

сколько сверточных слоев пройдет изображение. Теперь в полученном тензоре будут содержаться “маски” объектов, но так как ширина и высота меньше, чем исходное изображение, тензор нужно “растянуть” так, чтобы его ширина и высота совпали с изначальным изображением. Для этого в конце нейросети были добавлены слои “upsampling” (рис. 2).

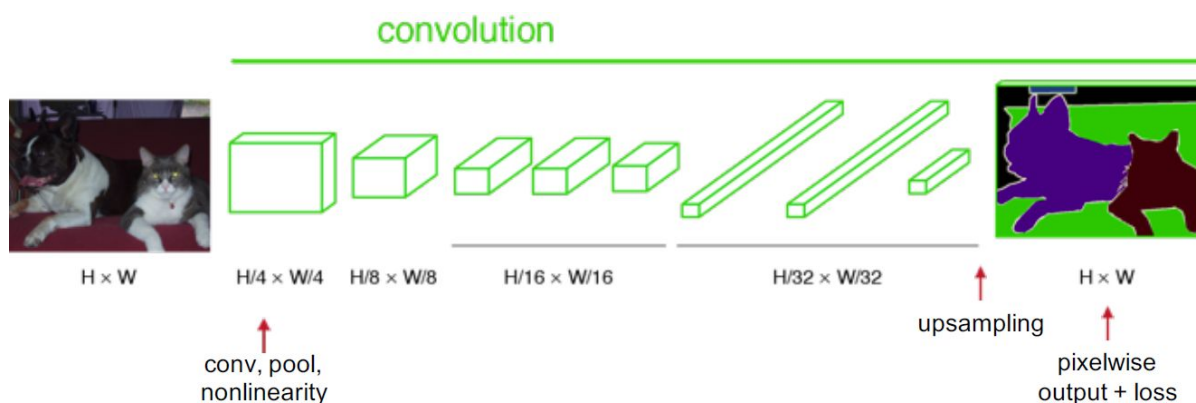


Рис. 2. Архитектура FCN для решения задач семантической сегментации

Сверточные слои в первой половине нейронной сети извлекают наиболее ценную информацию и помещают ее в тензоры меньшей размерности, чем изначальное изображение. Слои “upsampling” позволяют развернуть полученную информацию в первоначальный размер, что дает возможность попиксельно сравнить получившуюся “маску” с истинно верной и таким образом посчитать функцию потерь.

В других версиях FCN применяется операция слияния выходных тензоров. После прохождения седьмого сверточного слоя (conv7), как показано ниже, выходной размер невелик, а затем выполняется “upsampling by 32”, чтобы выходные данные имели одинаковый размер с

входным изображением. Но это также делает финальную карту признаков неточной. Такая сеть называется FCN-32s (рис. 3):

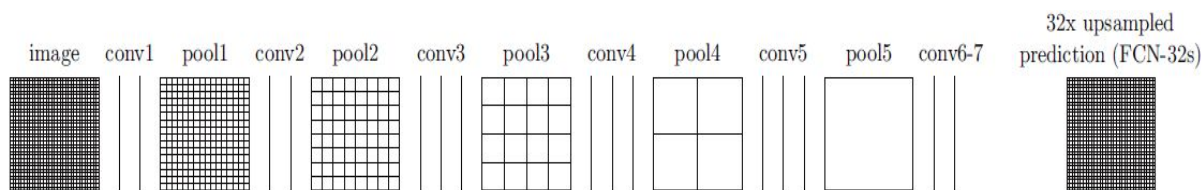


Рис. 3. Архитектура сверточной нейронной сети FCN-32s

Это происходит потому, что сложные карты признаков могут быть получены при прохождении большего количества слоев, а пространственная информация о местоположении объектов теряется при прохождении глубоких слоев нейронной сети. Это означает, что выходные данные из более ранних слоев имеют больше информации о местоположении объектов. Если объединить и то, и другое, то можно улучшить результат.

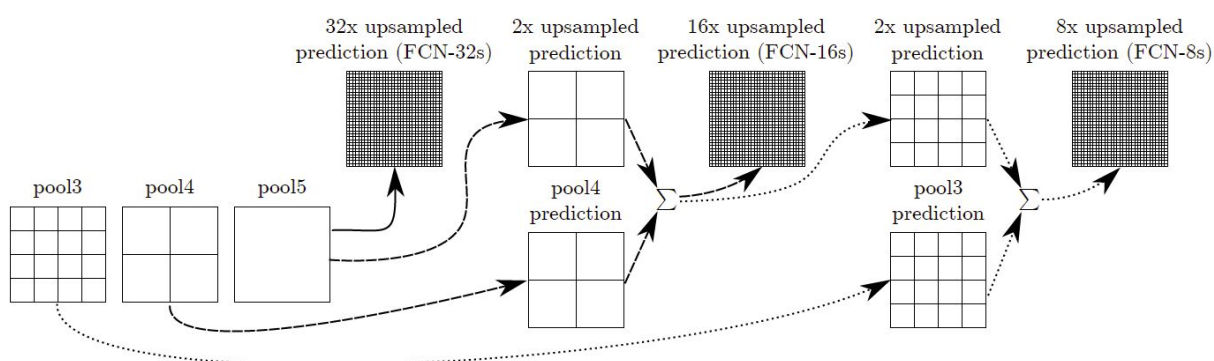


Рис. 4. Архитектура сверточной нейронной сети FCN-16s

В итоге получаем FCN-16s: выход из “pool5” проходит через “upsampling by 2” и объединяется с выходом “pool4” и проходит через “upsampling by 16”, в итоге получается финальный тензор, содержащий “маски” для

объектов. Аналогичные операции выполняются и для FCN-8s, как показано на рисунке выше (рис. 4).

В итоге получается три похожие нейронные сети, сравнение их работы показано на рис. 5. Исходя из результатов, можно заметить, что FCN-8s показывает наилучший результат, так как данная сеть использует наибольшее количество информации. Напротив, FCN-32s, которая не использует информацию, полученную из карт признаков на ранних этапах, показывает себя намного хуже.

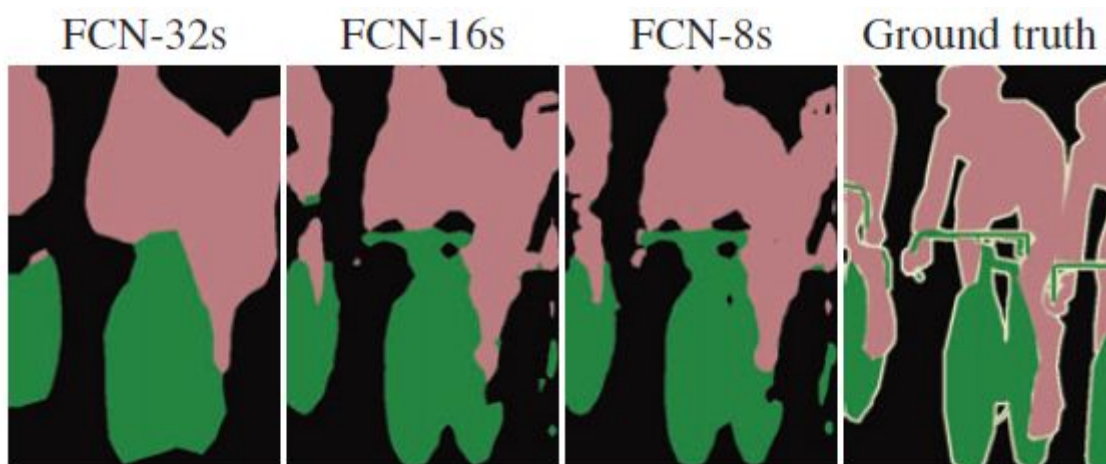


Рис. 5. Сравнение разных архитектур FCN

1.3. DeepLabV3

DeepLab [4] - семейство сетей разработанных для решения задачи семантической сегментации. Первая версия сети была представлена в 2014 году. Отличительной особенностью DeepLab [4] является слой *atrous convolution*. Этот термин взят из вейвлет-анализа и звучал как "*algorithme*

à trous", где *trous* означает дыра. Еще этот слой называется *dilated convolution*. Данный слой позволяет получать больше полезной информации с изображения, делая результаты работы нейронной сети в задаче семантической сегментации лучше.

1.4. Atrous Convolution

Atrous convolution - визитная карточка нейронных сетей DeepLab [4]. Он отличается от слоев обычной свертки тем, что фильтр умножается не на оригинальный сигнал, а на сигнал, взятый с определенным показателем *rate*. Это то же самое, что использовать фильтр, прореженный нулями. Такой вид свертки позволяет получать информацию на разных масштабах изображения.

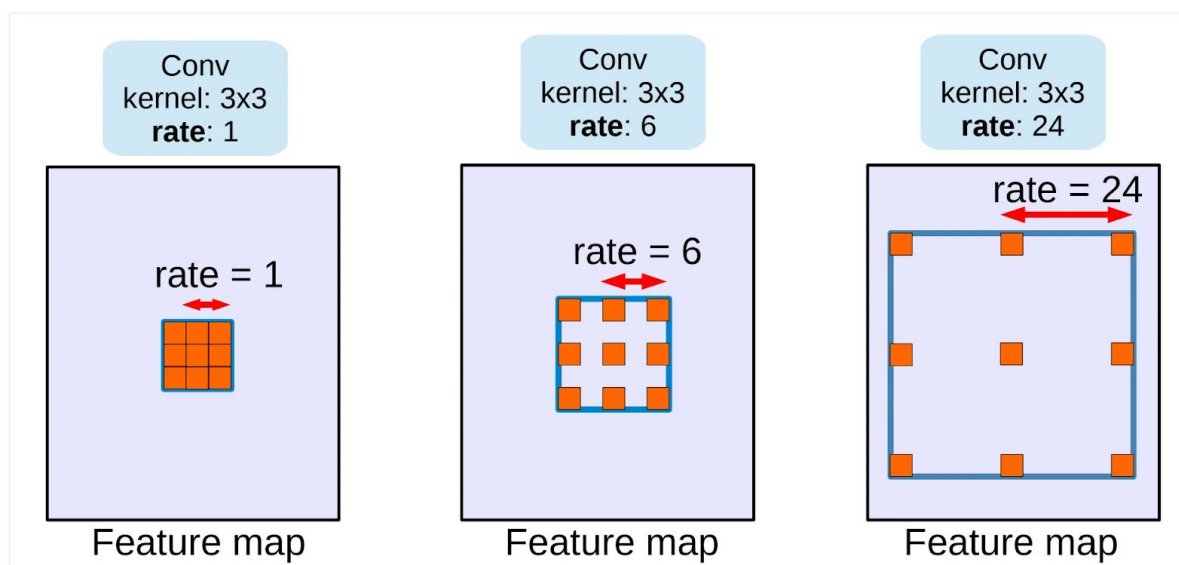


Рис. 6. Слой Atrous Convolution

Как видно на рис. 6, при rate равным 1, atrous convolution превращается в обычный слой свертки. В случае, когда rate больше 1, слой atrous извлекает полезную информацию в широком спектре. Так, благодаря разным показателям rate, atrous convolution позволяет нам расширить поле зрения фильтров, чтобы включить более широкий контекст, без затрат в производительности, так как количество вычисляемых параметров не зависит от показателя rate. Таким образом, atrous предлагает эффективный механизм управления полем зрения и находит наилучший компромисс между точной локализацией (малое поле зрения) и ассимиляцией контекста (большое поле зрения).

Также использование слоя atrous делает выходной сигнал больше, что положительно сказывается на результатах в рамках задачи сегментации.

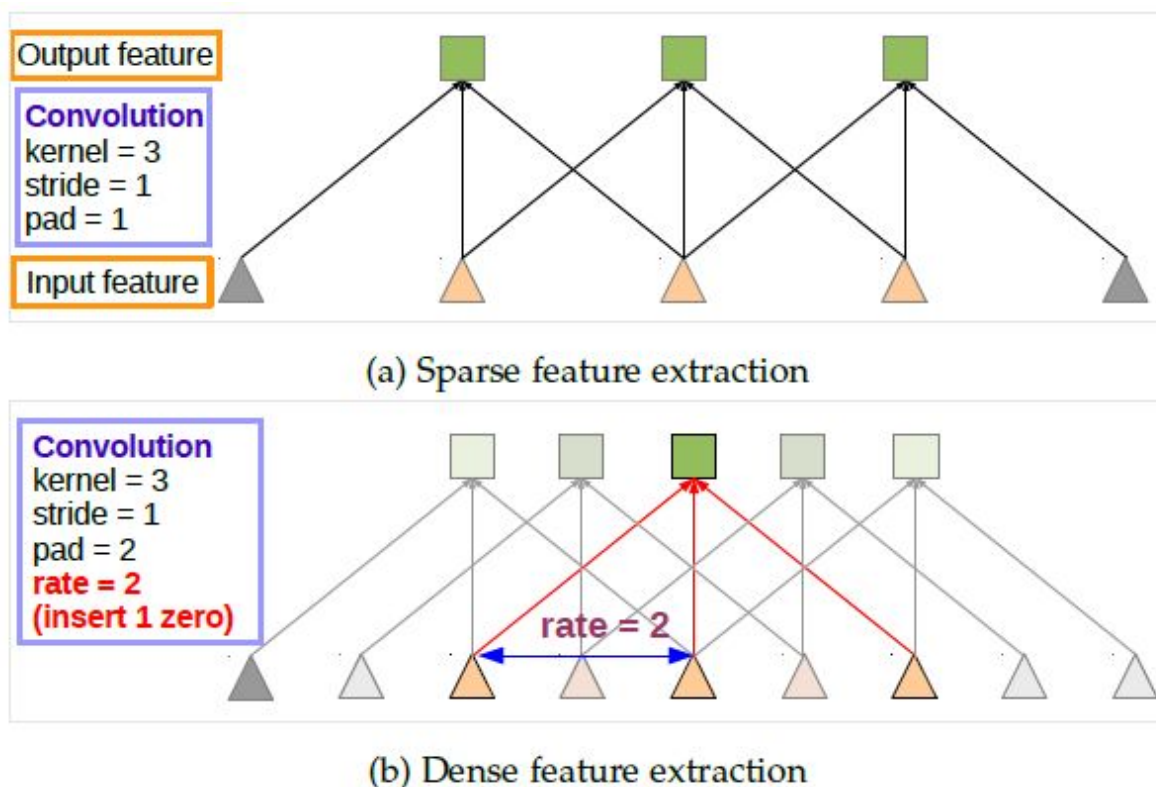


Рис. 7. Сравнение работы слоев convolution и atrous convolution

На изображении выше (рис. 7) показана работа обычного сверточного слоя с параметром $\text{pad} = 1$ (Верхняя часть изображения. Пункт “а”). Зеленым показан выходной сигнал. В нижней части изображения показана работа слоя *atrous* с показателями $\text{pad} = 2$ и $\text{rate} = 2$. Применение слоя *atrous* делает карту признаков больше. Что, как уже говорилось, положительно сказывается на результатах работы нейронной сети.

DeepLabV2 [5] появилась в свет в 2016 году в ее архитектуру была внедрена так называемая *Atrous Spatial Pyramid Pooling* (работает похожим образом как и *Spatial Pyramid Pooling* [10, 11, 12]).

В 2017 году миру представили DeepLabV3 [6]. В отличие от своих предшественников DeepLabV3 - сеть *end-to-end*, то есть может получать финальный результат без каких-либо постобработок. В ее архитектуре можно использовать предобученные сети, такие как: VGG16 [8], ResNet50 [9], ResNet101 [9] и другие. DeepLabV3 обучается с учителем. В качестве функции потерь используется попиксельная кросс-энтропия.

1.5. Архитектура DeepLabV3

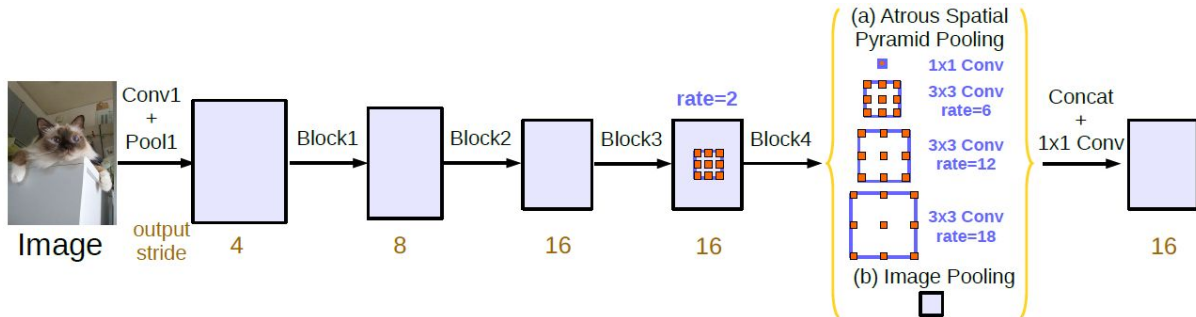


Рис. 8. Архитектура DeepLabV3

Как видно на рисунке 8, исходное изображение проходит через несколько блоков слоев обычной свертки. После “block3” применяется слой atrous с показателем $rate = 2$. Блоки до “block3” могут быть любой предобученной нейронной сетью (например из семейства ResNet [9]), умеющей решать задачу классификации изображений. Эти блоки отвечают за извлечение полезной информации из изображения. Побочным явлением свертки является уменьшение размерности изображения и, как следствие, потеря информации. На картинке под каждым выходом после блоков подписан так называемый Output Stride (OS). Чем больше значение OS, тем меньше получается карта признаков после блока (логично, что чем больше слоев свертки пройдет изображение, тем меньше по размерности будет результат).

После “block3” используются слои atrous и Atrous Spatial Pyramid Pooling (ASPP), что делает OS финальной карты признаков равным 16. То есть, если мы имели изображение 224×224 , то в конце нейронной сети мы получим карту признаков 14×14 .

ASPP - была представлена в DeepLabV2 [5]. В третьей версии в пирамиду включен слой Batch Normalization (BN) [13]. Смысл ASPP заключается в параллельном использовании слоев atrous с разными показателями rate, а результаты с каждого слоя объединяются в одну карту признаков. Причина использования ASPP заключается в том, что по мере увеличения параметра rate число допустимых весов фильтров (т. е. весов, которые применяются к допустимой области объектов, а не к дополненным нулям) становится меньше. Так же, поскольку объекты одного и того же класса могут иметь разные масштабы на изображении, ASPP помогает учитывать различные масштабы объектов, что может повысить точность. ASPP имеет 1 слой сверточный слой 1x1 и 3 слоя atrous с показателями rate = [6, 12, 18] при OS = 16. Когда OS = 8 все показатели rate удваиваются. В итоге всего получается 256 фильтров и слоев BN [13].

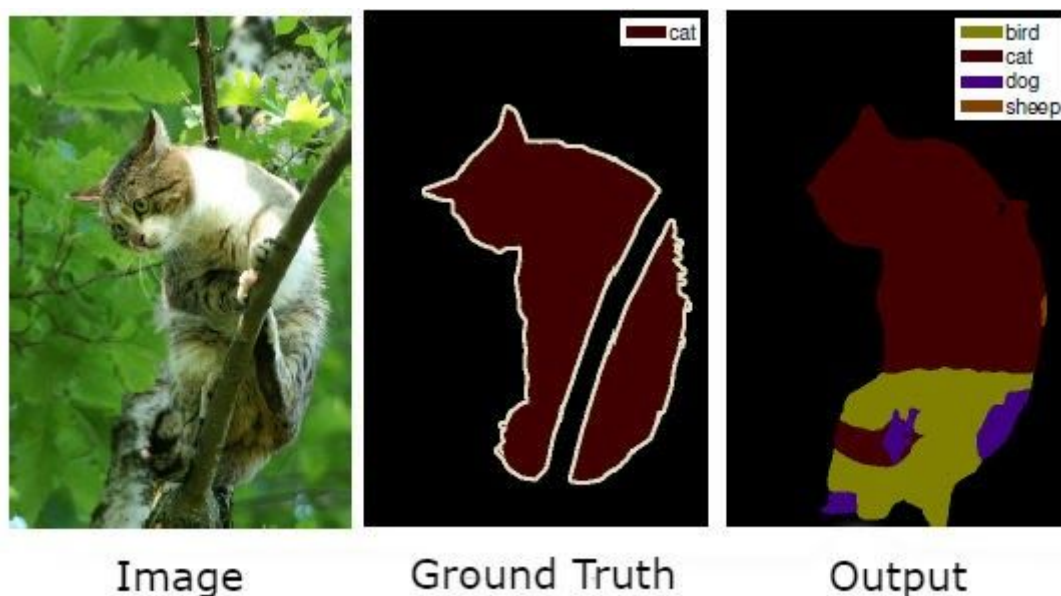


Рис. 9. Пример работы DeepLabV3

Также в DeepLabV3 информация уровня всего изображения объединяется с ASPP (image-level feature). Что позволяет учитывать глобальный контекст, и, например, отличить кошку от птицы, как показано на рисунке 9.

После того, как карты признаков со всех “веток” объединяются в одну, она проходит через 2 слоя свертки с ядром 1x1 и в итоге получается финальная карта признаков, которая идет на слой “upsampling”.

1.6. DeepLabV3 Plus

В 2018 году была представлена новая нейронная сеть DeepLab. Она не получила название DeepLabV4. DeepLabV3 Plus [7] стала модификацией предшественницы. Новая сеть была представлена с архитектурой encoder-decoder [14, 15]. В качестве слоев для кодирования изображения, как и в случае DeepLabV3, можно было использовать предобученные сети. Сеть так же обучалась с учителем и имела одинаковую функцию потерь. На данный момент DeepLabV3 Plus является возможно лучшим end-to-end решением в задаче семантической сегментации.

1.7. Архитектура DeepLabV3 Plus

Как уже было упомянуто, версия Plus использует архитектуру encoder-decoder [14, 15]. В качестве encoder используется предыдущая нейронная сеть DeepLabV3 [6]. OS на выходе encoder равен 16 (или 8) для более плотного извлечения признаков. Полная архитектура сети изображена на картинке ниже (рис. 10):

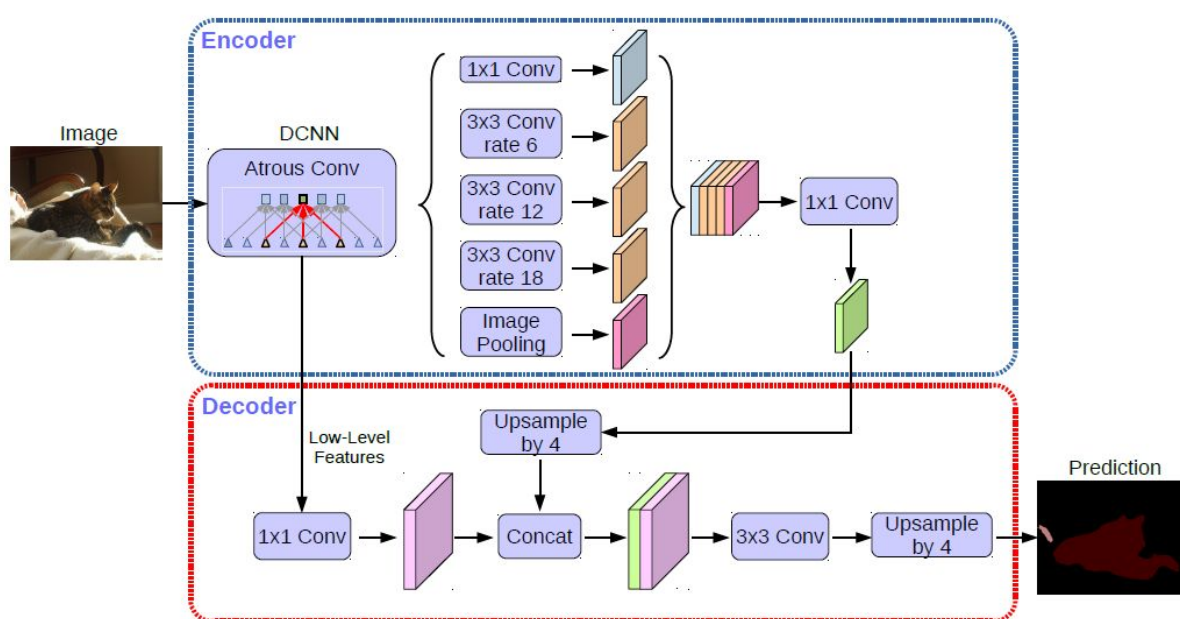


Рис. 10. Архитектура DeepLabV3 Plus

В decoder карты признаков сначала проходят через слой “upsampling by 4”, а затем объединяются с соответствующими низкоуровневыми картами признаков. Перед конкатенацией существует свертка 1x1 на низкоуровневых картах признаков, чтобы уменьшить количество каналов,

так как соответствующие низкоуровневые карты признаков обычно содержат большое количество каналов (например, 256 или 512). После конкатенации применяется несколько сверток 3×3 для уточнения карт признаков, за которыми следует еще один слой “upsampling by 4”. Такой подход намного лучше, чем один билинейный слой “upsampling by 16”.

Также для извлечения признаков в encoder DeepLabV3 Plus [7] использует нейронную сеть Xception [16], в которой вместо обычных слоев свертки применяются слои Separable Depthwise Convolution [16], что увеличивает производительность, за счет снижения количества вычисляемых параметров. Также в DeepLabV3 Plus [7] используется Atrous Separable Convolution, что, как и в случае с обычными слоями convolution, помогает разбить процесс свертки на 2 этапа и таким образом снизить количество вычисляемых параметров, что приводит к увеличению производительности.

1.8. Датасеты

Все сети, которые были отобраны для сравнения, были обучены на подвыборке датасета COCO [2], которая содержит все категории - датасете Pascal VOC [1].

Pascal Visual Object Classes [1] - очень известный датасет, позволяющий тренировать с учителем нейронные сети для разных задач области Computer Vision. Последняя версия данного датасета была опубликована в 2012 году. Датасет составлен из пар: изображение и метка, где метка - это

заранее известный правильный ответ для соответствующего изображения. Для каждой задачи метки разные. В случае задачи семантической сегментации метки - это маски интересующих нас объектов на изображении, как показано на рисунке 11.



Рис. 11. Пример датасета в случае решения задачи семантической сегментации

Всего “интересующих” объектов в Pascal VOC [1] 20 штук, с этими категориями объектов нейронная сеть и учится работать (например, распознавать, есть ли на изображении одна из двадцати категорий - это задача классификации изображения).

Датасет COCO [2] - тоже очень известный и “свежий” датасет. Последняя его версия была опубликована в 2019 году, также предназначен для решения задач Computer Vision. Содержит 80 различных категорий для задач распознавания и детекции объектов и 91 категорию для задач

семантической сегментации. Суммарно содержит 330 тысяч различных изображений.

1.9. Метрики

Обычно результаты работы нейронных сетей в задаче сегментации оценивают с помощью метрик Pixel Accuracy. Рассмотрим их ниже поподробнее.

Pixel Accuracy - интуитивно понятная метрика. Состоит в том, чтобы просто сообщить процент пикселей в изображении, которые были правильно классифицированы. Считается она похожим образом как и Accuracy в классическом ML.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Рис. 12. Метрика Accuracy

Mean Intersection over Union (mIoU) - измеряет общее количество пикселей между целевой и прогнозной масками, деленное на общее количество пикселей, присутствующих в обеих масках. IoU высчитывается для каждого класса отдельно, далее полученные результаты усредняются, так и получается метрика mean IoU.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Рис. 13. Метрика IoU

Ниже в таблице представлены данные, взятые из открытых источников. Они показывают метрику mean IoU, полученную на тестовой выборке датасета COCO [2].

Neural Networks	Mean IoU
FCN	63.7
DeepLabV3	67.4
DeepLabV3 Plus	76.3

В дальнейшем мы будем производить сравнение нейронных сетей, используя только метрику Pixel Accuracy, так как мы имеем в нашем датасете всего одну категорию - “человек”.

1.10. Выводы

Исходя из детального обзора нейронных сетей, описанных в ранних главах, для решения поставленной задачи были выбраны FCN-8s, DeepLabV3, DeepLabV3 Plus. Данные сети хорошо зарекомендовали себя в решении задачи семантической сегментации.

В качестве тестового датасета было выбрано использовать датасет из 100 пар изображений и масок. Все пары были выбраны из датасета PascalVOC автором статьи. После обработки “маскок” в качестве объектов остался лишь класс “человек”.

Результат работы нейронных сетей будет измеряться в метрике Pixel Accuracy.

Глава 2. Особенности реализации решения

2.1 Используемые технологии и средства разработки

В реализации программы использовался язык программирования Python3. Данный язык славится большим количеством библиотек, которые легко устанавливаются. Благодаря им, проводить различного рода исследования на этом языке значительно проще. В данной задаче были использованы следующие библиотеки:

OpenCV - библиотека для работы с изображением и видео. Весьма гибкая библиотека, предоставляющая удобные инструменты для работы в области Computer Vision. Имеет множество простых алгоритмов для работы с изображениями.

PyTorch и TensorFlow - две библиотеки для работы с нейронными сетями. Их отличие состоит в разном подходе подсчета производных, нужных для обратного распространения ошибки. В этих библиотеках содержится все, что только может потребоваться для разных задач связанных с работой с нейронными сетями.

Pathlib - библиотека для упрощенной работы с путями, файлами и директориями.

Logging - стандартная библиотека логирования Python. Предоставляет большие возможности в ведении логов, что полезно для сбора информации и отладки приложений.

Numpy - это библиотека, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой

высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

Python Imaging Library (PIL) - библиотека для обработки графики в Python. Предоставляет широкие возможности для работы с изображениями.

Argparse - модуль для обработки аргументов командной строки.

2.2. Программа

Опишем объектную модель созданного приложения (с исходным кодом можно ознакомиться по следующей ссылке:

<https://github.com/morememes/ComparingNetworks>)

Основную часть функциональности, а именно: обработку одного изображения и обработку видео, было принято реализовать в классе Predictor.

Конструктор класса принимает на вход 3 аргумента: model, mode, gpu. Через параметр model передается предобученная модель, написанная с помощью библиотеки PyTorch или TensorFlow. В параметре mode указывается, с помощью какой библиотеки была написана сеть. Имеет два возможных значения: "torch" и "tf". Параметр gpu по умолчанию установлен в положение False. Отвечает за использование графического ускорителя для ускорения вычислений, если параметр mode = "torch". В случае mode = "tf" все вычисления будут производиться на gpu, если это возможно.

Метод `process_image(image_path, save_path)` - принимает на вход 2 аргумента. `image_path` - путь до входного изображения. `save_path` - путь до директории, в которую сохраняются результаты. После отработки алгоритма в `save_path` сохранится маска и изначальное изображение с примененной маской на нем (то есть, если на изображении присутствует человек, после применения маски на результирующем изображении останется человек на черном фоне). Большие изображения не помещаются в память. Для их обработки они нарезаются на маленькие куски пересекающихся изображений - Tile (тайлы). Тайлы обрабатываются по очереди, затем склеиваются в одну большую маску для изначального изображения. Изображения маленького размера обрабатываются напрямую без применения тайлов.

Метод `process_video(video_path, save_path, newfps = None)` - принимает на вход 3 аргумента. `video_path` - путь до входного видео. `save_path` - путь в которое сохранится обработанное видео (так же как и в случае с изображением, все кадры нового видео будут кадрами старого видео с примененными соответствующими масками на них). Метод по очереди обрабатывает каждый кадр видео и сохраняет по пути `video_path` (звуковая дорожка не сохраняется). `newfps` - параметр по умолчанию в положении `None`. Предназначен для снижения количества кадров в секунду в выходном видео. `newfps` должен быть всегда меньше значения `fps` входного видео.

2.3. Система тестирования

Для реализации тестов был разработан скрипт `test.py` и несколько утилит, поддерживающую работу скрипта. Скрипт позволяет проводить вычисление метрик Pixel Accuracy и mean IoU на различных датасетах.

Пример запуска скрипта:

```
python test.py --model [model_name] --datapath [path] --logpath [path] [--gpu]
```

Параметр `--model` принимает на вход название одной из трех выбранных нейронных сетей. Возможные варианты: `fcn`, `deeplabv3`, `deeplabv3plus`.

Параметр `--datapath` принимает путь до директории в которой храниться датасет. В директории должны быть 2 папки `img` и `mask` в которых лежат изображения и соответствующие маски к ним. Соответствующие изображения и маски должны быть названы одинаково, изображения имеют расширение `“jpg”`, маски - `“png”` (то есть, изображение `0001.jpg`, лежащие в директории `“img”`. Маска `0001.png`, лежащая в директории `“mask”`). Также должен быть текстовый файл `imagelist.txt` в котором на каждой строчке написано название изображения без расширения. (то есть на строке к примеру будет записано `0001`).

Параметр `--logpath` - путь до директории в которой будет сохраняться файл с логами. По умолчанию это директория `“./log/”`.

Параметр `--gpu`. Если написан в запуске, значит скрипт будет использовать графический ускоритель для обработки изображений. По

умолчанию стоит в положении False. (Работает только для моделей, написанных на PyTorch)

После запуска скрипт тестирует модель на всех изображениях в датасете, считая по ним метрику Pixel Accuracy и mean IoU. Изменение метрик после каждого изображения фиксируется “логгером” в Терминале, а также в лог-файле, который сохраняется в директорию указанную в параметре --log.

Глава 3. Результаты тестирования полученного решения

Тесты выбранных нейронных сетей (FCN [3], DeepLabV3 [6] и DeepLabV3 Plus [7]) проводились на небольшом датасете (данный датасет был собран автором данной статьи из изображений и масок к ним датасета PascalVOC [1]) из 100 пар изображений и “правильных” масок к ним. На всех изображениях представлены люди на различном фоне. Для каждой нейронной сети считалась метрика Pixel Accuracy на описанном выше датасете. Ниже в таблице представлен результат.

Neural Networks	Pixel Accuracy
FCN	93.8
DeepLabV3	94.0
DeepLabV3 Plus	97.6

На рис. 14, 15, 16, 17 и 18 приведено несколько примеров работы нейронных сетей.

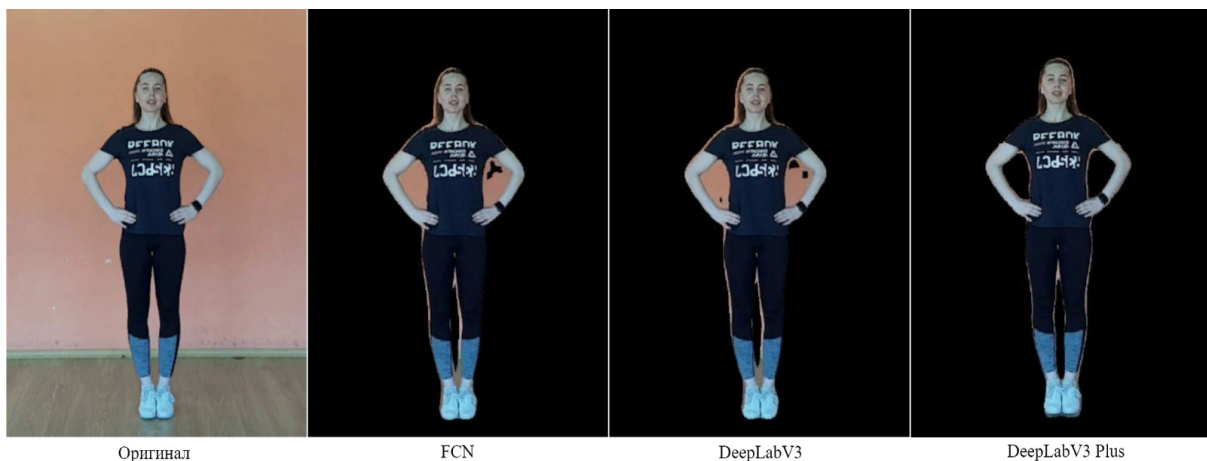


Рис. 14. Пример 1.



Рис. 15. Пример 2.



Рис. 16. Пример 3.



Рис. 17. Пример 4.

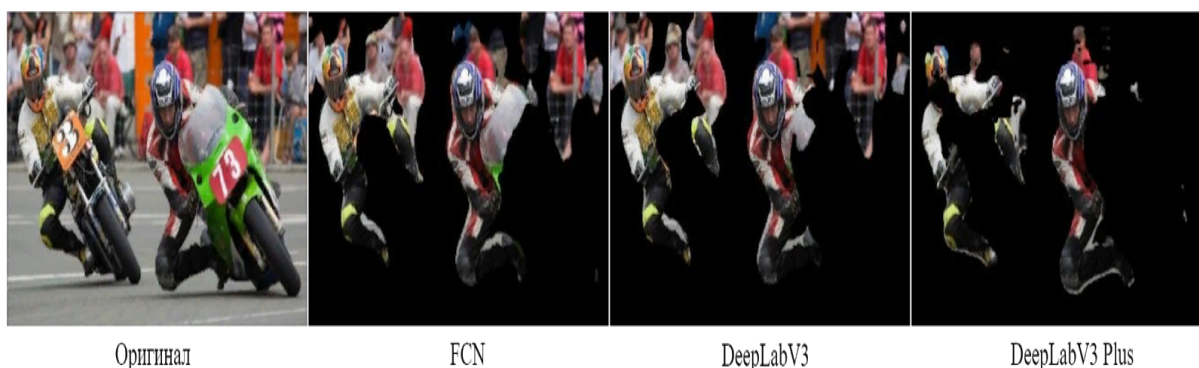


Рис. 18. Пример 5.

Как видно по таблице и на примерах, наилучший результат показывает DeepLabV3 Plus [7]. Данная нейросеть выдает наиболее чистые маски без фона и контуров. FCN [3] и DeepLabV3 [6] показывают себя примерно одинаково, что также видно по метрике Pixel Accuracy. Обе нейронные сети выдают грубую, но все же достаточно точную маску. На маске может присутствовать фон. Очень часто в маске можно увидеть контур у объекта.

Также нейронные сети были протестированы в скорости работы на видео формата mp4. Разрешение видео 1920 на 1080, количество кадров в секунду - 60. Все видео состоит из 409 кадров и идет около 4 секунд.

Для нейронной сети ставилась такая задача: обработать все видео и сохранить результат. Финальное видео должно быть в 30 кадров в секунду (30 кадров в секунду - минимальный порог для многих задач real time). Ниже представлена таблица скорости работы каждой сети на видео, описанном выше:

Neural Networks	Seconds
FCN	730.61
DeepLabV3	815.06
DeepLabV3 Plus	1113.93

Как видно из таблицы, использовать данные нейронные сети для обработки в задачах режима real time не представляется возможным на аппаратуре, имеющейся в распоряжении автора (Все тесты и вычисления проводились с использованием графического ускорителя NVIDIA GeForce GTX 1060). Скорее всего, имея более мощный GPU, данные нейронные сети могли бы обрабатывать 30 кадров в секунду, что дает возможность применять их в режиме реального времени.

Выводы

В ходе работы была разработана программа для сравнения различных моделей нейронных сетей в рамках решения задачи сегментации фигуры человека.

Были решены следующие задачи:

1. Изучены несколько архитектур нейронных сетей для тестирования.
2. Собран свой тестовый датасет из 100 пар изображений и меток.
3. Реализован алгоритм обработки одного изображения.
4. Реализован алгоритм обработки видео.
5. Реализован алгоритм тестирования нейронных сетей.
6. Проведено сравнение выбранных нейронных сетей.
7. Проверено возможная работа выбранных нейронных сетей в условиях реального времени.

Код разработанного решения доступен на GitHub по ссылке:

<https://github.com/morememes/ComparingNetworks>.

Заключение

Как видно из проведенного исследования, нейронные сети хорошо справляются с задачей сегментации. Все модели в работе показали приемлемое качество работы. Также DeepLabV3 Plus [7] показал удивительную точность в построении масок, что ставит эту модель на первое место среди рассмотренных в данной работе. Но все же по скорости обработки DeepLabV3 Plus [7] проигрывает своим предшественникам.

В будущем планируется нарастить тестовый датасет, а также увеличить количество нейронных сетей для сравнения. Кроме того, представляется перспективным попробовать легковесные нейронные сети (с маленьким количеством параметров) в задачах real time.

Список литературы

1. Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 2015.
2. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. Microsoft COCO: Common Objects in Context. In: *ECCV*, 2014.
3. Jonathan Long, Evan Shelhamer, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In: *CVPR*, 2015.
4. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” in *ICLR*, 2015.
5. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.
6. Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv: 1706.05587*, 2017.
7. Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *arXiv preprint arXiv: 1802.02611*, 2018.
8. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, 2016.
10. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: ICCV, 2005.
11. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, 2006.
12. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: ECCV, 2014.
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML, 2015.
14. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI, 2015.
15. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. PAMI, 2017.
16. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR, 2017.