Московский государственный технический университет им.Н.Э.Баумана
кафедра "Системы обработки информации и управления"

"Методы машинного обучения"

к Домашнее задание №1

«Методы машинного обучения»

Инструктор：Юрий Гапанюк

Email:2623859464@qq.com

Студент: Ван Чаочао

группа ИУ5И-22М

2022/06/10

Домашнее задание

по дисциплине «Методы машинного обучения»

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

1. выбор задачи;
2. теоретический этап;
3. практический этап.

Этап выбора задачи предполагает анализ ресурса paperswithcode. Данный ресурс включает описание нескольких тысяч современных задач в области машинного обучения. Каждое описание задачи содержит ссылки на наиболее современные и актуальные научные статьи, предназначенные для решения задачи (список статей регулярно обновляется авторами ресурса). Каждое описание статьи содержит ссылку на репозиторий с открытым исходным кодом, реализующим представленные в статье эксперименты. На этапе выбора задачи обучающийся выбирает одну из задач машинного обучения, описание которой содержит ссылки на статьи и репозитории с исходным кодом.

Теоретический этап включает проработку как минимум двух статей, относящихся к выбранной задаче. Результаты проработки обучающийся излагает в теоретической части отчета по домашнему заданию, которая может включать:

- описание общих подходов к решению задачи;
- конкретные топологии нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения, предназначенных для решения задачи;
- математическое описание, алгоритмы функционирования, особенности обучения используемых для решения задачи нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения;
- описание наборов данных, используемых для обучения моделей;
- оценка качества решения задачи, описание метрик качества и их значений;
- предложения обучающегося по улучшению качества решения задачи.

Практический этап включает повторение экспериментов авторов статей на основе представленных авторами репозиториев с исходным кодом и возможное улучшение обучающимися полученных результатов. Результаты проработки обучающийся излагает в практической части отчета по домашнему заданию, которая может включать:

- исходные коды программ, представленные авторами статей, результаты документирования программ обучающимися с использованием диаграмм UML, путем визуализации топологий нейронных сетей и другими способами;
- результаты выполнения программ, вычисление значений для описанных в статьях метрик качества, выводы обучающегося о воспроизводимости экспериментов авторов статей и соответствии практических экспериментов теоретическим материалам статей;
- предложения обучающегося по возможным улучшениям решения задачи, результаты практических экспериментов (исходные коды, документация) по возможному улучшению решения задачи.

Выбор задачи

Я ссылаюсь на две статьи следующим образом:《Deep Residual Learning for Image Recognition》 и《Learning Interpretable Anatomical Features Through Deep Generative Models: Application to Cardiac Remodeling 》,
Ссылка на статью:
https://paperswithcode.com/paper/deep-residual-learning-for-image-recognition
https://arxiv.org/pdf/1807.06843v1.pdf

и проанализируйте его на основе того, что я узнал,
Рекуррентные нейронные сети более склонны к задачам последовательности, распознаванию речи, обработке естественного языка и т. д., в то время как сверточные нейронные сети более склонны к приложениям для распознавания изображений. По сравнению с Googlenet и VGG, VGG является более общим и может быть напрямую трансплантирован. сетевые уровни очень близки к более чем 20 уровням, и частота ошибок также очень близка.RESNET, изобретенный Microsoft Research под руководством Хэ Кайминга, имеет поразительные 152 сетевых слоя и поразительную частоту ошибок 3,57.

Трудности с традиционным «глубоким обучением»
Текущий прогресс глубокого обучения зависит от методов: выбора начального веса, локального рецептивного поля, распределения веса и т. д., но при использовании более глубоких сетей (например, > 100) он по-прежнему сталкивается с традиционными трудностями, такими как исчезновение градиента при обратном распространении,
Проблема деградации: чем больше слоев, тем **выше** частота ошибок обучения и частота ошибок теста.

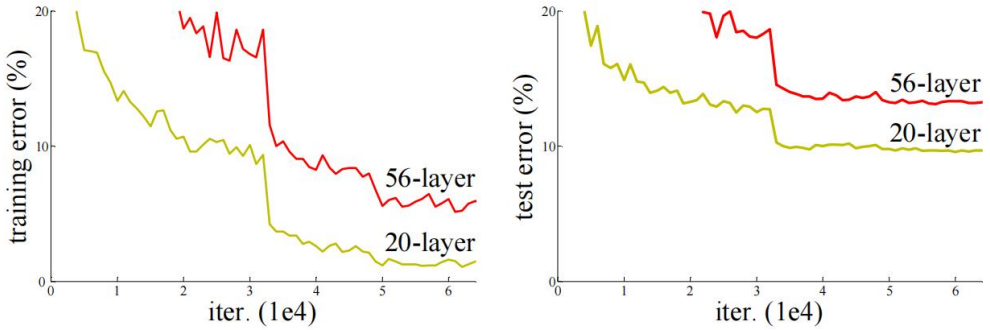


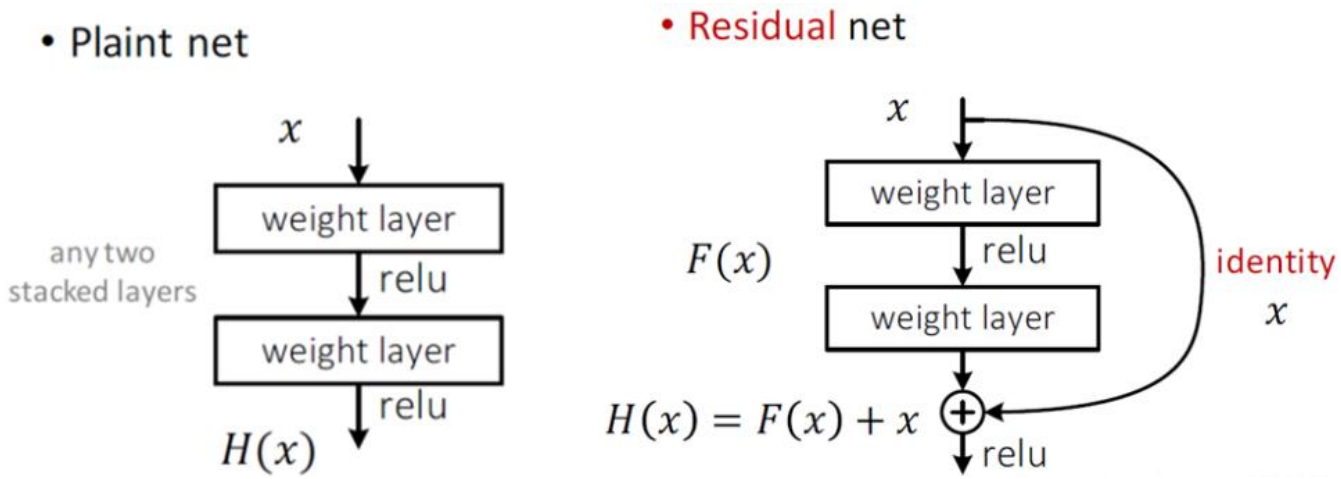
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Как решить вышеуказанную проблему?

Теоретический этап

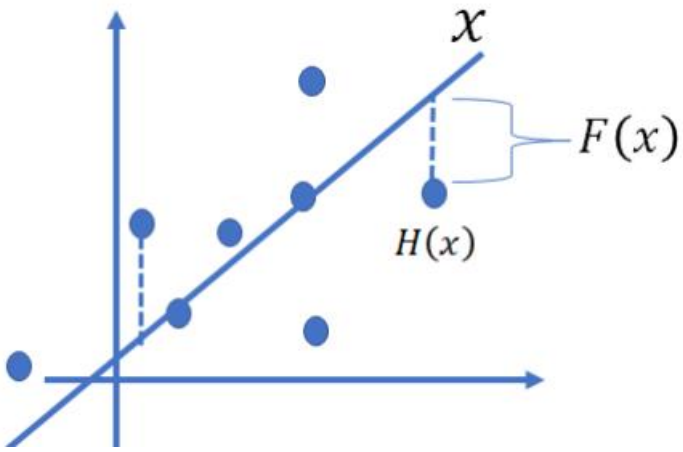
Идея глубокой остаточной сети

Введение «ярлыков» может предотвратить проблему исчезновения градиента. На самом деле, до выхода ResNet некоторые люди уже изучали этот аспект. «Дорожная сеть» и «обучение очень глубокой сети» Шриваставы и др. похожи. методы.Они считают, что более глубокая сеть также должна быть легко оптимизируемой.Чем больше слоев, тем выше точность, а метод обучения не сильно изменится по сравнению с «традиционной» глубокой сетью.Основываясь на этом, команда Хэ Кайминга предложила новая глубокая остаточная сеть, следующая Давайте сначала рассмотрим основные части сети:



Функция активации слева принимает функцию RELU, чтобы избежать проблемы исчезновения градиента.Правая половина является базовой единицей остаточной сети.Ввод X в начале накладывается в соответствии с обычной нейронной сетью, а затем проходит через функцию активации.После наложения весов входной сигнал и выходной сигнал в это время накладываются друг на друга, а затем проходят через функцию активации. А в остатке есть еще один способ.

Остаток в линейной подгонке относится к разнице между точками данных и значением функции подобранной прямой линии, тогда мы можем провести здесь аналогию, где X — функция нашей подгонки, а H (x) — конкретная точка данных. , затем я добавляю подобранное значение F(x) посредством обучения, чтобы получить значение конкретной точки данных, так что это F(x) является остатком, как показано ниже:



Общая сеть показана на рисунке:

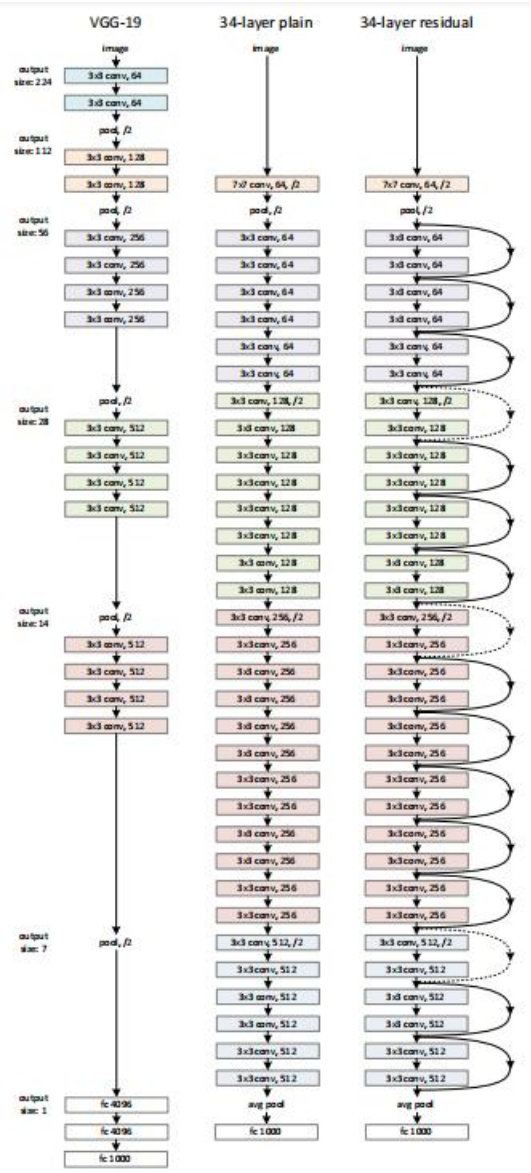
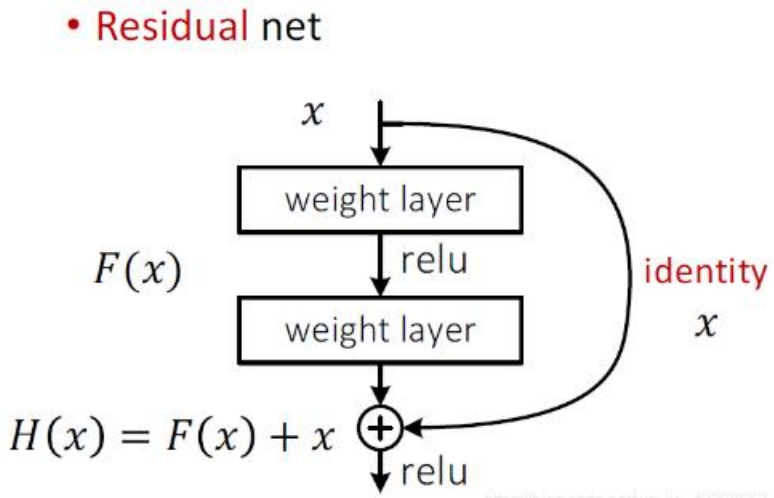


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Через приведенную выше сеть вы можете увидеть сравнение между сетью и VGG.Принцип обучения VGG-19 в основном такой же, как и у обычной сверточной нейронной сети.Разница в том, что размер его ядра свертки составляет все 3× 3. , размер слоя пула 2x2, такую модель легко трансплантировать и применять, в то время как Resnet состоит из множества слоев высокоскоростных каналов, а Resnet состоит из множества слоев высокоскоростных каналов, начиная с перспектива извлечения признаков. Объясните, если первый слой должен извлекать глобальные признаки, то есть мелкие признаки, а глубокие слои извлекают глубокие признаки. Можем ли мы сказать, что при выводе каждого слоя он объединяет глубокие и мелкие признаки, чтобы судить, так что градиент не исчезает легко.

Математический анализ:



Эта сеть похожа на метод обучения свертки.Шоссе не влияет на обратное распространение ошибки.Давайте объясним с помощью математики:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

Разумно было бы сказать, что весовая матрица есть, но исходная бумага прямо равна 1,

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \quad (1)$$

$$\mathbf{x}_{l+1} = f(\mathbf{y}_l). \quad (2)$$

Здесь отображаются английские правила RELU, используемые f(), поэтому их можно получить напрямую:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l). \quad (3)$$

Повторяя:

$$\mathbf{x}_{l+2} = \mathbf{x}_{l+1} + \mathcal{F}(\mathbf{x}_{l+1}, \mathcal{W}_{l+1}) = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) + \mathcal{F}(\mathbf{x}_{l+1}, \mathcal{W}_{l+1}),$$

В соответствии с принципом обратного распространения мы предполагаем$\mathcal{E}$, что ошибка равна ее частной производной от Xl:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)\right). \quad (5)$$

Приведенная выше формула дает обратное распространение функции ошибок, если

$$h(\mathbf{x}_l) = \lambda_l \mathbf{x}_l$$

на данный момент:

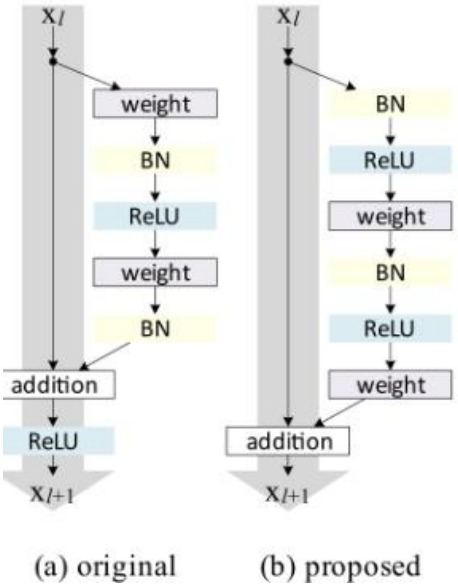
$$\mathbf{x}_{l+1} = \lambda_l \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \tag{6}$$

Подставляя в (5), обратное распространение ошибки можно получить как:

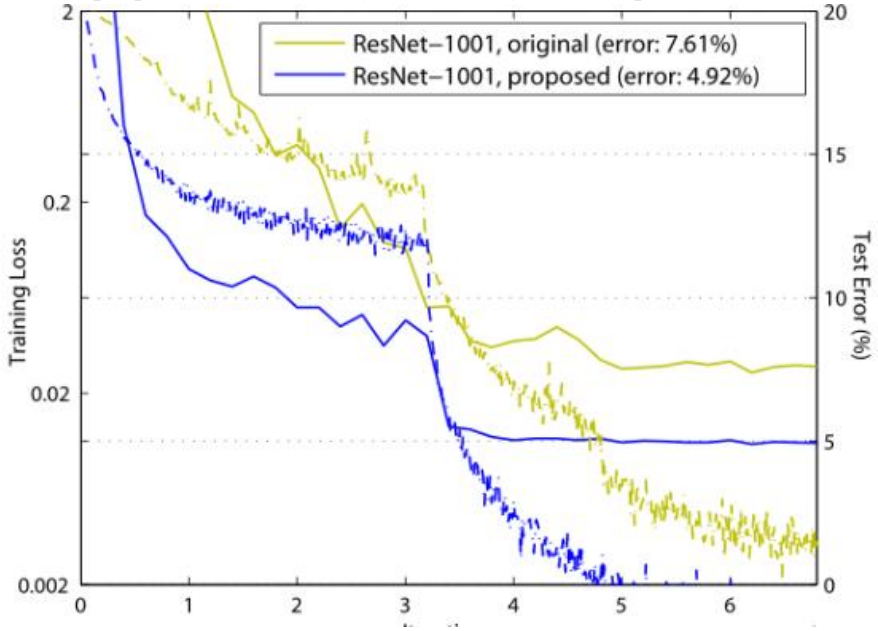
$$\mathbf{x}_L = (\prod_{i=l}^{L-1} \lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i), \tag{7}$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left((\prod_{i=l}^{L-1} \lambda_i) + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i) \right). \tag{8}$$

С помощью формулы (8) мы можем видеть, как, когда коэффициент больше 1 или меньше 1, будет вызвана проблема взрыва или исчезновения градиента, поэтому сразу 1 является лучшим, а именно (5). На основе данных они также объясняют, почему прямое картирование является лучшим, следующим образом:



(a) original (b) proposed



Из приведенного выше рисунка видно, что если вес увеличен, а положение пакетной нормализации другое, правильность другая.Они выступают за то, чтобы упорядочить сравнение перед RELU, но даже это все еще имеет коэффициент ошибок 4,9 %, а значение веса A, равное 1, соответствует 3,57 %.

Развертывание конструкции:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Первая строка представляет разные слои, а именно 18 слоев, 34 слоя, 50 слоев, 101 слой и 152 слоя структур RESNET.Как рассчитать количество слоев? Здесь для примера возьмем 152 этажа, скоростные дороги здесь все три этажа, глубина слишком глубокая, а количество слоев слишком мало, а объем расчета слишком большой. Возьмем, к примеру, 152 слоя. Количество расчетных слоев: (3 + 8 + 36 + 3) x3 = 150, плюс два полносвязных слоя — это 152 слоя

Почему на скоростных автомагистралях всего два этажа? Можно ли иметь четыре слоя, в чем разница между большим количеством слоев. сравнить результаты. Или может ли шоссе продолжать распространяться перекрестно. Мы можем сделать это сами через код.

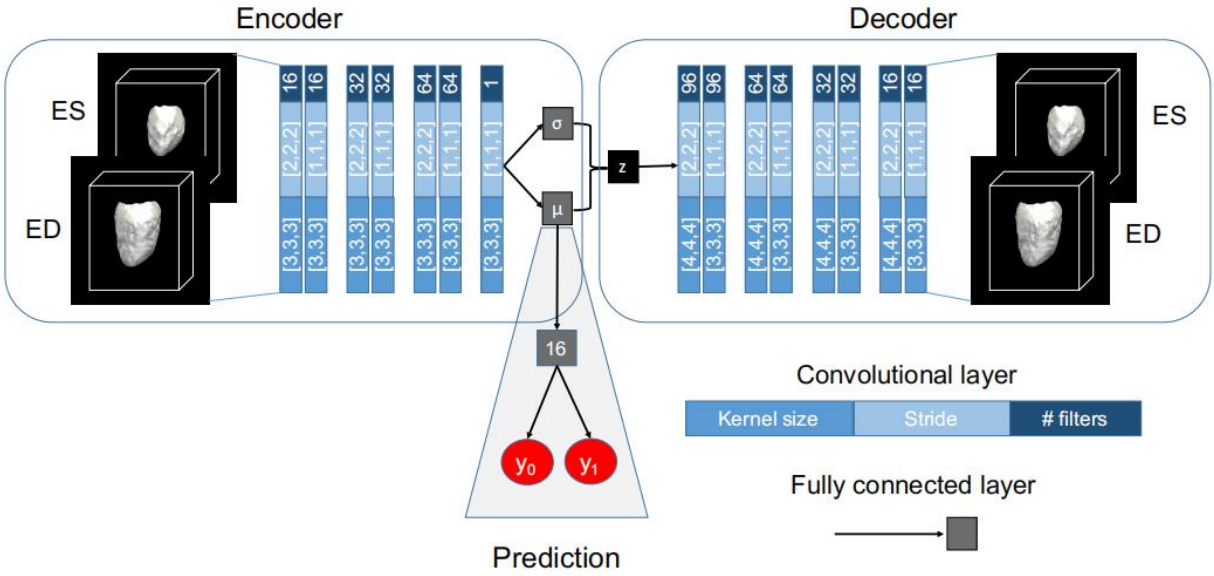
Статья 2

Анализ модели

Принципиальная схема модели, предложенной в данной статье, представлена на рис. 1. Вход X представляет собой 3D-сегментацию миокарда левого желудочка (3D-сегментацию миокарда левого желудочка) в конечно-диастолическом (ED) и конечно-систолическом (ES) сегментах с двухканальным входом. Используя трехмерный сверточный VAE, d-мерное распределение вероятностей, представляющее входной сегмент X в скрытом пространстве, изучается через сеть кодера, а скрытое распределение параметризуется как d-мерное нормальное распределение N (μ_i, σ_i), где μ_i представляет собой среднее значение, σ_i — стандартное отклонение. Во время обучения сеть декодера учится восстанавливать аппроксимацию входных данных X путем выборки вектора z из изученного скрытого d-многообразия. Между тем, дискриминационная сеть, состоящая из многослойных персептронов (MLP) (называемая в этой статье предсказанием сети предсказания), связана со средним вектором μ и обучена отличать здоровых добровольцев (HVols) от гипертрофических субъектов с гипертрофической кардиомиопатией (HCM). Сквозное обучение с использованием следующей функции потерь:

где L_rec представляет потери при реконструкции, которые можно рассчитать, введя потери в кости Соренсена между X и реконструкцией. L_KL - это потеря расхождения Кульбака-Лейблера, цель которой - максимально приблизить N (μ, σ) к его предыдущему распределению N (0, 1). L_MLP — кросс-энтропийная потеря для задачи классификации MLP. Размер скрытого пространства d=64.

На этапе тестирования каждый входной сегмент восстанавливается путем передачи предсказанного μ в z (без выборки из скрытого пространства), и, наконец, задача классификации выполняется на этапе обучения.



Архитектура модели, предложенная в этой статье, позволяет визуализировать особенности, изученные сетью, в исходном пространстве сегментации. Используя веса, полученные MLP, частная производная метки классификации болезни C(y_C) вычисляется путем обратного распространения градиента от метки классификации C к μ_i с использованием цепного правила. Учитывая случайно выбранную форму здоровой ткани, полученные градиенты можно использовать для сдвига латентного представления субъекта в

направлении лежащей в основе закодированной вариабельности с использованием итеративного алгоритма для максимизации вероятности отнесения этой вариабельности к классу C. Начиная со среднего скрытого представления здоровой формы, μ_i итеративно обновляется на каждом шаге t с использованием:

$$\mu_{i,t} = \mu_{i,t-1} + \lambda \frac{\partial y_1}{\partial \mu_{i,t-1}}, \quad \forall i = 1, \ldots d \qquad (1)$$

В этой статье выбрано значение λ=0,1. Наконец, каждое скрытое представление μ_t на каждом шаге t может быть декодировано путем передачи его в z для получения сегментированного пространства, позволяющего визуализировать соответствующий реконструированный сегмент.

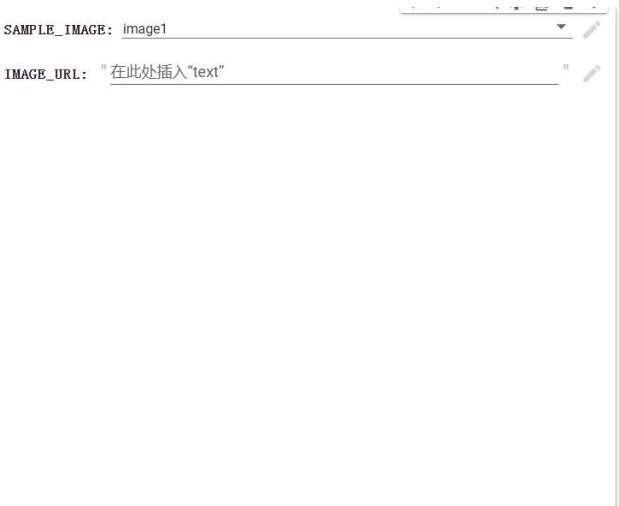
Практический этап

Подробности смотрите во вложении, здесь показаны только некоторые результаты。

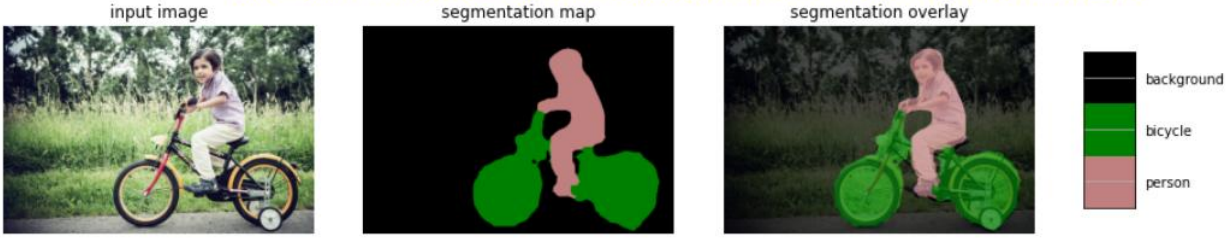
Поэкспериментируйте с обученной моделью
Четыре обучающие модели

```
MODEL_NAME = 'mobilenetv2_coco_voctrainaug'
MODEL_NAME = 'mobilenetv2_coco_voctrainval'
MODEL_NAME = 'xception_coco_voctrainaug'
MODEL_NAME = 'xception_coco_voctrainval'
```

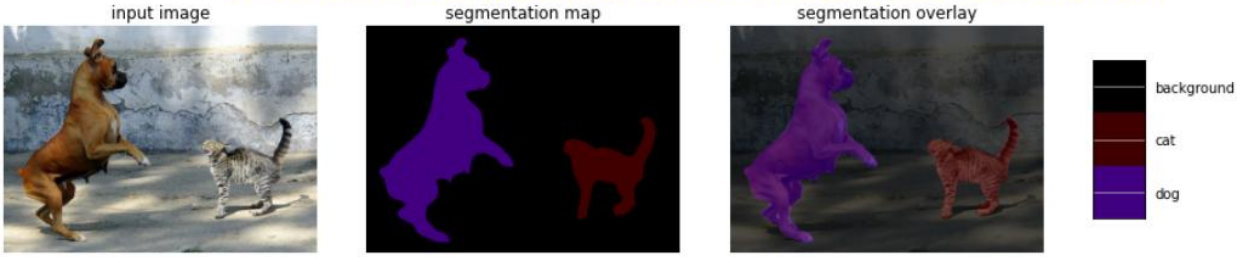
```
SAMPLE_IMAGE = "image1" #@param ["image1", "image2", "image3", "image4"]
IMAGE_URL = ''  #@param {type:"string"}

_SAMPLE_URL = ('https://github.com/tensorflow/models/blob/master/research/'
               'deeplab/g3doc/img/%s.jpg?raw=true')


def run_visualization(url):
  """Inferences DeepLab model and visualizes result."""
  try:
    f = urllib.request.urlopen(url)
    jpeg_str = f.read()
    original_im = Image.open(BytesIO(jpeg_str))
  except IOError:
    print('Cannot retrieve image. Please check url: ' + url)
    return

  print('running deeplab on image %s...' % url)
  resized_im, seg_map = MODEL.run(original_im)

  vis_segmentation(resized_im, seg_map)


image_url = IMAGE_URL or _SAMPLE_URL % SAMPLE_IMAGE
run_visualization(image_url)
```



running deeplab on image https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/img/image1.jpg?raw=true...



running deeplab on image https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/img/image2.jpg?raw=true...



running deeplab on image https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/img/image3.jpg?raw=true...