Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ_
КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ__

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Анализ сегментации медицинских изображений на основе Unet»

Студент __ИУ5-33М____ _____ **Юй Шанчэнь**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **Ю. Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2022г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _______
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме «Исследование и разработка системы персонализированных музыкальных рекомендаций на основе коллаборативной фильтрации»

Студент группы ИУ5И-33М

_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание Вводится система музыкальных рекомендаций, анализируются классификация и существующие проблемы системы музыкальных рекомендаций. Рекомендация песни с различным анализом сходства алгоритмов совместной фильтрации.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на ___ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____
_____
_____
_____

Дата выдачи задания « ___ » Декабрь 2022 г.

Руководитель НИР _____ ___Ю. Е. Гапанюк____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ___Юй Шанчэнь____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

1. Описание проблемы

В медицинской диагностике важно использовать снимки КТ для определения местоположения и размера опухолей и определения того, произошло ли метастазирование. Цель этого случая состоит в том, чтобы сегментировать область опухоли от существующего изображения КТ, а область опухоли изображается профессиональным врачом и может быть использована в качестве метки (PS: Набор данных взят из 7-го «Чашки Тедди» Data Mining Challenge). Ситуация с данными показана на следующем рисунке

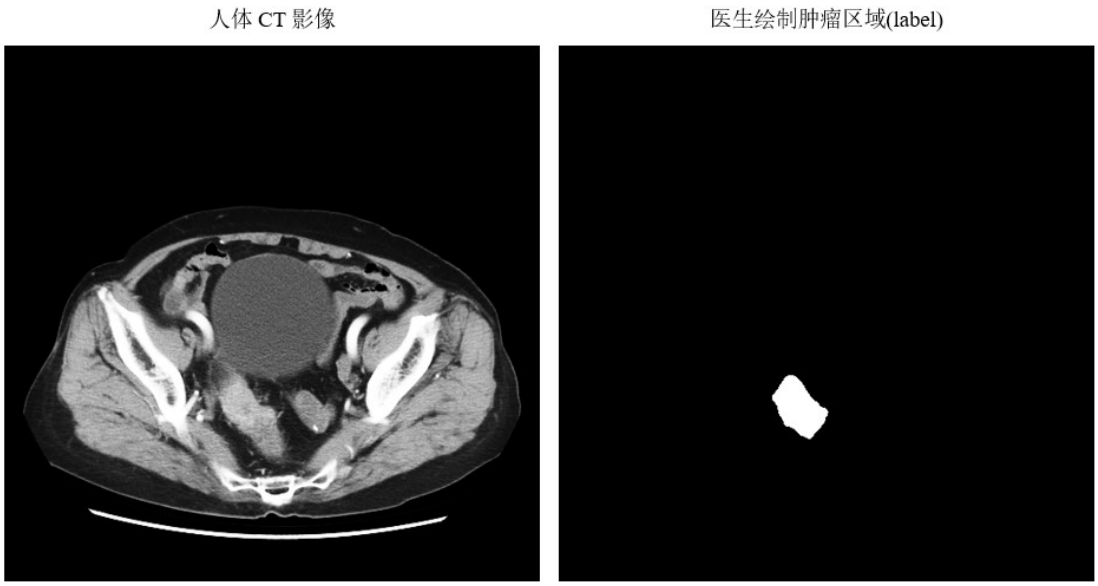


Рисунок 1 Диаграмма данных

Как показано на рисунке, левая картинка – это исходное изображение КТ-изображения, а правая картинка – это область опухоли, нарисованная врачом.

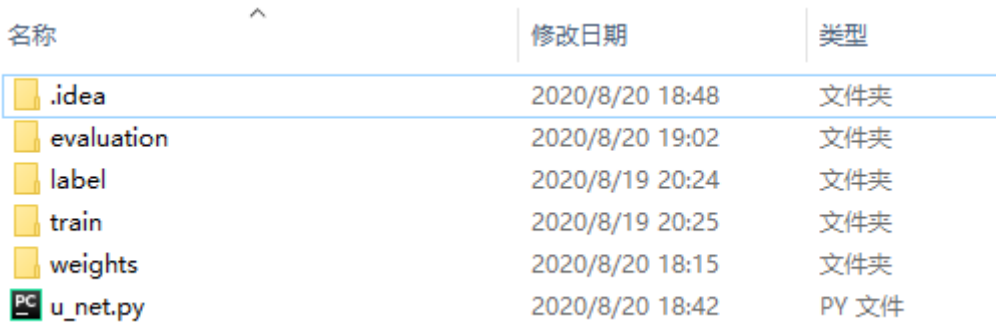
Требования к сегментации:

Сегментировать область опухоли максимально точно;

Для снимков КТ без опухоли результат сегментации сети должен быть полностью черным;

Оценочными показателями были точность и коэффициент игры в кости;

Описание исходного файла и папки:



поезд: размещена папка с компьютерными снимками 108 пациентов;

этикетка: Помещается маска опухоли, соответствующая TRAIN;

оценка: размещаются результаты сетевого теста;

веса: Сохранение веса сети;

u_net.py: Исполняемый файл для этого документа;

Отображение эффекта:

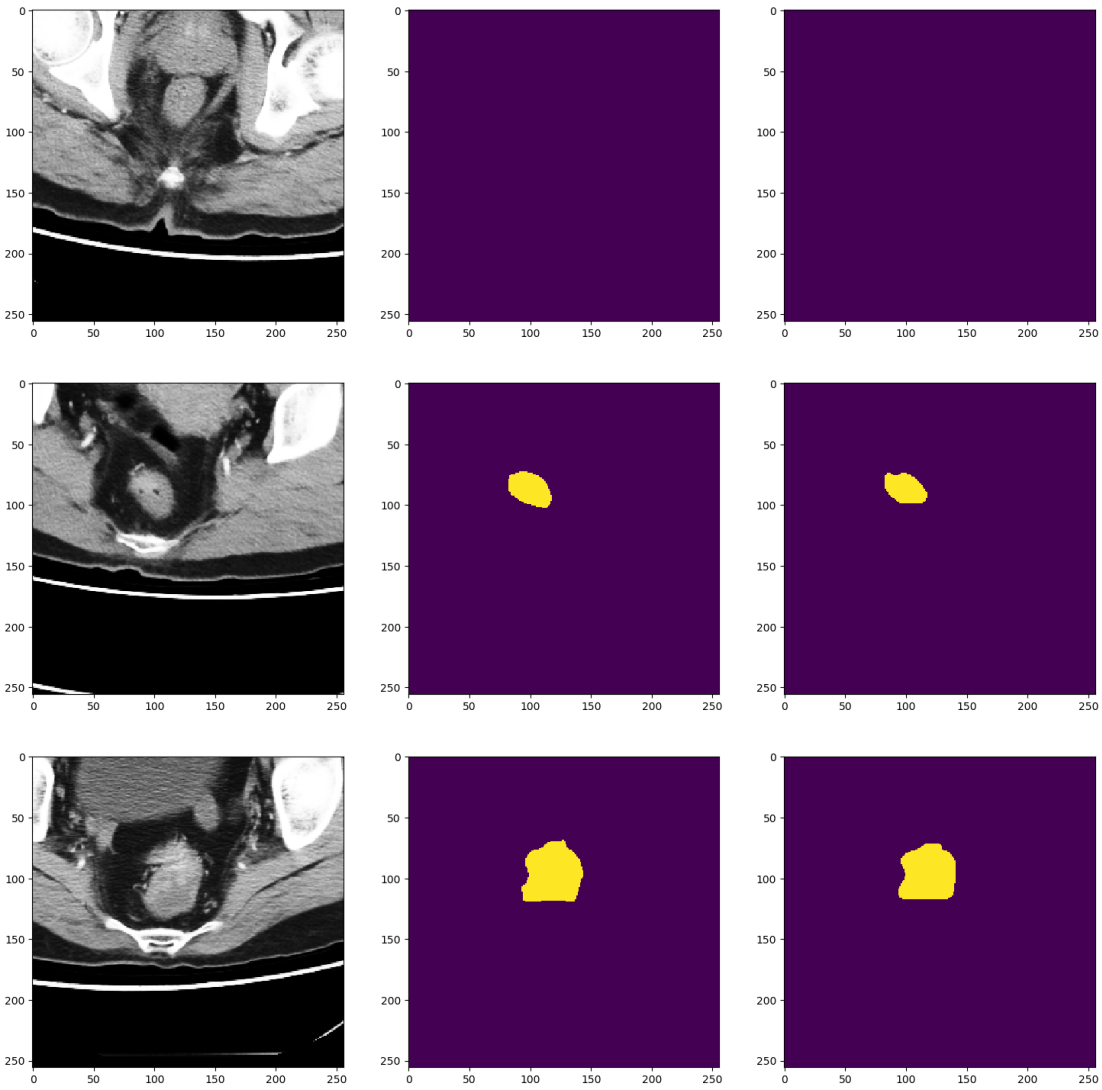


Рисунок 2 Принципиальная схема эффекта сегментации сети

2. Базовый процесс

Использование глубокого обучения для решения задач имеет относительно фиксированный процесс, который можно резюмировать следующим образом: - 1. Предварительная обработка данных -> 2. Построение сети -> 3. Компиляция сети -> 4. Обучение сети -> 5. Тестирование и оценка.

Предварительная обработка данных, включая подготовку набора данных, нормализацию данных, вырезку данных и т.д.;

Постройте сеть, постройте подходящую сеть в соответствии с потребностями конкретных задач, сеть, построенная в этом документе, является UNet;

Составить сеть, выбрать оптимизатор, скорость обучения, функцию потерь, функцию оценки и т.д. обучающей сети;

Обучить сеть, задать batch_size, эпоху, стоит ли тренироваться не по порядку, проверить заданный коэффициент сегментации и т.д.;

Протестируйте сеть, напишите тестовые функции, просмотрите эффект сегментации сети, рассчитайте соответствующие оценочные показатели и многое другое.

3. Предварительная обработка данных

Проблема сегментации опухоли UNet может быть разбита на модули, и аналогичным образом конкретная работа может быть подразделена под каждым модулем. Работа модуля предварительной обработки данных включает в себя: чтение изображения -> эффективную обрезку области изображения -> нормализацию изображения -> разделение обучающих и тестовых наборов -> сохранение данных.

Глубокое обучение для контролируемого обучения требует большого объема данных, поэтому работа по предварительной обработке данных очень важна, и подготовка наборов данных также станет основной работой после умелого использования глубокого

обучения на более позднем этапе.

3.1 Введение в наборы данных

Папка поезда содержит папку с изображениями КТ 108 пациентов, каждый пациент имеет около двадцати или тридцати изображений КТ, изначально изображение находится в формате DCM, после обработки оно было прочитано и сохранено как «». формат PNG"; Папка label содержит соответствующее изображение маски.

3.2 Чтение изображений

名称	修改日期	类型
1001	2020/6/12 9:15	文件夹
1002	2020/6/12 9:15	文件夹
1003	2020/6/12 9:15	文件夹
1004	2020/6/12 9:15	文件夹
1005	2020/6/12 9:15	文件夹
1006	2020/6/12 9:15	文件夹
1007	2020/6/12 9:15	文件夹
1008	2020/6/12 9:15	文件夹
1009	2020/6/12 9:15	文件夹
1010	2020/6/12 9:15	文件夹

Поскольку количество изображений CT в каждой папке в поезде отличается, вам нужно использовать функцию glob в Python для автоматического чтения имени папки и файла; После этого используйте Image.open для чтения изображения;

```
for file in glob('./train/*'):  # 获取文件夹名称
    for filename in glob(file + '/*'):  # 获取文件夹中的文件
        img = np.array(Image.open(filename), dtype='float32') / 255
```

Обратите внимание, что функция glob должна быть импортирована перед использованием (см. импорт модуля в начале

программы), эта часть кода и следующий код являются только фрагментами в функции, для целей интерпретации не могут быть запущены напрямую.

3.3 Обрезка изображения

Опухоли прямой кишки возникают только в прямой кишке, поэтому область, которая на самом деле нуждается в лечении, не нуждается в полной картине, а правильная обрезка может улучшить скорость сетевого обучения;

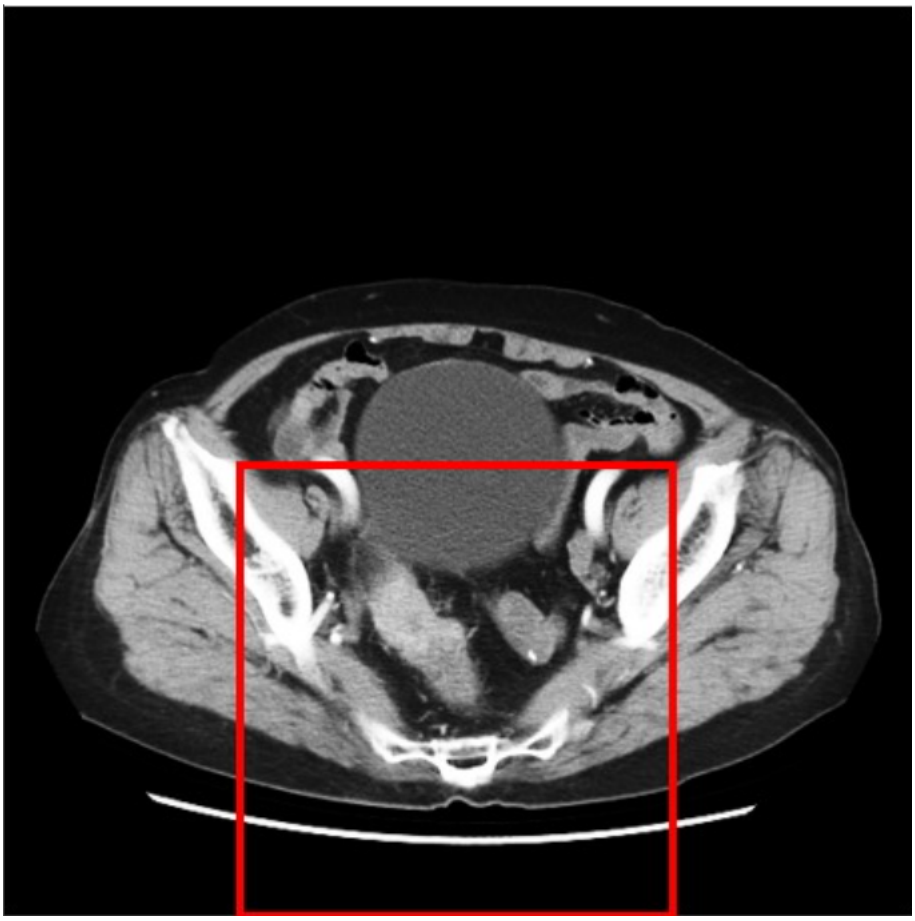


Figure 4 Принципиальная схема обрезки изображений

Метод реализации:

```
1 | x_train.append(img[256:, 128:384])
```

После того, как каждое изображение прочитано, оно расширяется до списка x_train, и когда все изображения пройдены, все данные изображения сохраняются x_train списка;

3.4 Разделите тренировочный набор и тест

Поскольку распределение опухоли непрерывно, чтобы избежать слишком концентрированных локальных признаков, набор данных здесь неупорядочен, а затем обучающие и тестовые наборы выделяются 9:1.

```
np.random.seed(116) # 设置相同的随机种子，确保数据匹配
np.random.shuffle(x_train) # 对第一维度进行乱序
np.random.seed(116)
np.random.shuffle(x_label)
# 图片有三千张左右，按9:1进行分配
return x_train[:2700, :, :], x_label[:2700, :, :], x_train[2700:, :, :], x_label[2700:, :, :]
```

После предварительной обработки данных их можно сохранить в данные .npy, чтобы не приходилось каждый раз выполнять операцию предварительной обработки.

4. Построить сеть

4.1 Введение в сеть UNet

Сети UNet — это сети, которые обеспечивают сквозное сопоставление, что делает их идеальными для достижения таких целей, как сегментация изображений, восстановление, улучшение и сверхразрешение. Конкретная структура сети показана на следующем рисунке.

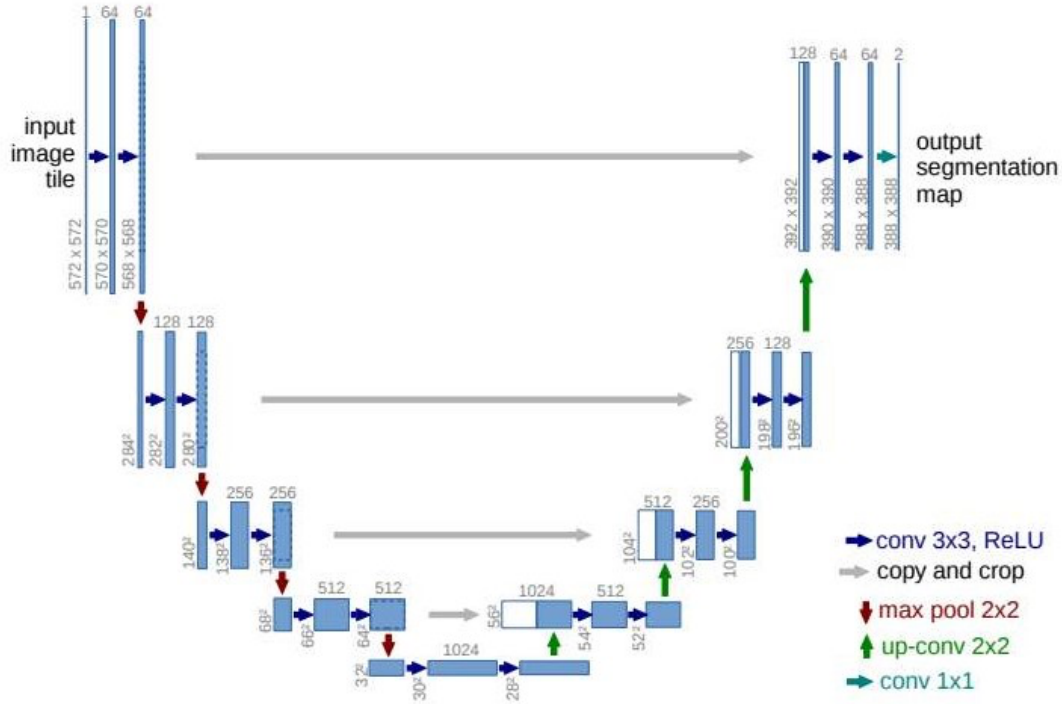


Рисунок 5 Принципиальная схема структуры сети UNet

Сеть UNet имеет несколько основных функций:

(1) Сеть может быть разделена на левую и правую части от середины, левая сторона - это путь сжатия, а модули понижения и свертки используются для извлечения признаков различных масштабов; Справа находится расширенный путь, использующий модули апсамплинга и свертки для восстановления масштаба и слияния предыдущих функций для постепенного восстановления изображения;

(2) Модуль свертки состоит из двух непрерывных сверточных слоев, которые могут осуществлять извлечение признаков большего размера и большей глубины; Даунсамплинг реализует уменьшение масштаба изображения; Апсамплинг (или деконволюционный слой)

для достижения большего масштаба изображения;

(3) Серая стрелка на рисунке указывает на скачок, который заключается в объединении объектов на пути сжатия в объекты на расширенном пути в том же масштабе, что может обеспечить лучшие ограничения на расширенном пути, и сеть может легче выводить наши желаемые результаты.

4.2 Создание сети UNet

Для того, чтобы создать сеть UNet, автор написал функцию, и следующий код будет разделен по порядку. Определите функции в соответствии с потребностями создания сети UNet:

```
1 | def build_unet(self, n_filters=16, dropout=0, batchnorm=True, padding='same'):
```

Существует самость во входе функции, потому что автор поместил функцию в класс (класс), который может реализовать гибкий вызов функции.

n_filters — количество ядер свертки, dropout — параметр выпадающего слоя, batchnorm определяет, следует ли стандартизировать, а заполнение определяет, изменяется ли размер изображения до и после свертки. Эти параметры будут использоваться позже в программе.

(1) Сначала определите повторно используемый модуль свертки (функцию)

Эта функция размещается внутри build_unet функции и является модулем (функцией), который необходимо использовать повторно. Компоненты сверочного модуля:

- Сверточный слой -> нормализующий слой -> функция активации -> сверточный слой -> нормализованный слой -> функция активации -> возвращает результат.

Описание заданных входных параметров функции:

input_tensor: входное изображение или результат вывода предыдущего слоя;

n_filters: Установка количества ядер свертки;

kernel_size: Установка размера ядра свертки;

batchnorm: если true, это означает, что используется слой нормализации (аналогично гауссовской нормализации, но есть две обучаемые переменные масштаба и смещения), если он ложный, нормализация не используется;

заполнение: Если оно «то же самое», это означает, что масштаб изображения одинаков до и после свертки, если он «действителен», изображение станет меньше после свертки.

Конкретная реализация:

```python
# 定义一个多次使用的卷积块
def conv2d_block(input_tensor, n_filters=16, kernel_size=3, batchnorm=True, padding='same'):
    # the first layer
    x = Conv2D(n_filters, kernel_size, padding=padding)(
        input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # the second layer
    x = Conv2D(n_filters, kernel_size, padding=padding)(x)
    if batchnorm:
        x = BatchNormalization()(x)
    X = Activation('relu')(x)
    return X
```

Среди них Conv2D — это функция в keras.layers TensorFlow, которая была импортирована в начале программы, поэтому ее можно вызывать без префикса. Эта функция строит сверточный слой, input_tensor — вход, x — выход сверочного слоя.

Аналогичным образом, BatchNormalization нормализует входные данные; Activation('relu')(x), который указывает, что функция активации relu выполняется на входе.

Повторите «сверточный слой -> нормализованный слой -> функцию активации» дважды, то есть создаются два сверточных слоя последовательно, которые являются широко используемым экстрактором признаков, и, наконец, вывод X.

(2) Путь конвергенции

Из структурной диаграммы UNet видно, что основным процессом пути конвергенции является

Входное изображение -> (1) (модуль свертки -> понижение дискретизации -> слой выпадения) -> (2) (модуль свертки ->

дауншамплинг -> слой выпадения) -> (3) (модуль свертки -> даунсэмплинг -> выпадающий слой) -> (4) (модуль свертки ->даксретизация -> слой выпадения).

Таким образом, программа может быть написана в логическом порядке:

```
# 构建一个输入
img = Input(shape=self.shape) # self.shape是图片维度大小

# contracting path
c1 = conv2d_block(img, n_filters=n_filters * 1, kernel_size=3, batchnorm=batchnorm, padding=padding)
p1 = MaxPooling2D((2, 2))(c1)
p1 = Dropout(dropout * 0.5)(p1)

c2 = conv2d_block(p1, n_filters=n_filters * 2, kernel_size=3, batchnorm=batchnorm, padding=padding)
p2 = MaxPooling2D((2, 2))(c2)
p2 = Dropout(dropout)(p2)

c3 = conv2d_block(p2, n_filters=n_filters * 4, kernel_size=3, batchnorm=batchnorm, padding=padding)
p3 = MaxPooling2D((2, 2))(c3)
p3 = Dropout(dropout)(p3)

c4 = conv2d_block(p3, n_filters=n_filters * 8, kernel_size=3, batchnorm=batchnorm, padding=padding)
p4 = MaxPooling2D((2, 2))(c4)
p4 = Dropout(dropout)(p4)
```

На этом запись программы сжатого пути завершается. Перейдите к векторному слою посередине:

```
c5 = conv2d_block(p4, n_filters=n_filters * 16, kernel_size=3, batchnorm=batchnorm, padding=padding)
```

(3) Путь расширения

Из структурной диаграммы UNet мы видим, что основным процессом расширения пути является

Входной выход предыдущего слоя -> (6) (апсамплинг -> слияние -> выпадающий слой -> модуль свертки) -> (7) (слияние функций апсамплинг -> -слияние -> выпадающий слой -> модуль свертки) -> (8) (слияние функций апсамплинг -> слияние -> выпадающий слой ->

модуль свертки) -> (9) (слияние -> функция слияния -> выпадающий слой - > модуль свертки).

Конкретная реализация:

```
# extending path
u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides=(2, 2), padding='same')(c5)
u6 = concatenate([u6, c4])
u6 = Dropout(dropout)(u6)
c6 = conv2d_block(u6, n_filters=n_filters * 8, kernel_size=3, batchnorm=batchnorm, padding=padding)

u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides=(2, 2), padding='same')(c6)
u7 = concatenate([u7, c3])
u7 = Dropout(dropout)(u7)
c7 = conv2d_block(u7, n_filters=n_filters * 4, kernel_size=3, batchnorm=batchnorm, padding=padding)

u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides=(2, 2), padding='same')(c7)
u8 = concatenate([u8, c2])
u8 = Dropout(dropout)(u8)
c8 = conv2d_block(u8, n_filters=n_filters * 2, kernel_size=3, batchnorm=batchnorm, padding=padding)

u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides=(2, 2), padding='same')(c8)
u9 = concatenate([u9, c1])
u9 = Dropout(dropout)(u9)
c9 = conv2d_block(u9, n_filters=n_filters * 1, kernel_size=3, batchnorm=batchnorm, padding=padding)
```

(4) Выходной слой

В зависимости от формата данных выходного изображения, наш выходной слой может быть определен:

```
output = Conv2D(1, (1, 1), activation='sigmoid')(c9)

return Model(img, output)
```

Первое 1 в Conv2D представляет собой количество ядер свертки, то есть количество каналов выходного изображения, если это цветное изображение, измените данные на 3; (1,1) указывает на размер ядра свертки, и имеется только один размер пикселя, что означает, что каждый пиксель изображения дискриминируется;

Функция активации — «сигмовидная», а выходной интервал — [0,1]. Наконец, вернитесь к созданной сети UNet: Model(img, output).

PS: Вышеуказанные программы могут быть подключены последовательно, чтобы получить полную программу.

На этом этапе мы определили функцию для создания сети UNet, а затем мы можем создать сеть, просто вызвав функцию.

```
self.unet = self.build_unet() # 创建网络变量
self.unet.summary()
```

После выполнения self.unet.summary(), если структура сети напечатана:

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 256, 256, 1) 0
_________________________________________________________________________________________________
conv2d (Conv2D)              (None, 256, 256, 16) 160         input_1[0][0]
_________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 256, 256, 16) 64          conv2d[0][0]
_________________________________________________________________________________________________
activation (Activation)      (None, 256, 256, 16) 0           batch_normalization[0][0]
_________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 256, 16) 2320        activation[0][0]
_________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 256, 256, 16) 64          conv2d_1[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 256, 256, 16) 0           batch_normalization_1[0][0]
_________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 128, 128, 16) 0           activation_1[0][0]
_________________________________________________________________________________________________
dropout (Dropout)            (None, 128, 128, 16) 0           max_pooling2d[0][0]
_________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 32) 4640        dropout[0][0]
_________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 128, 128, 32) 128         conv2d_2[0][0]
_________________________________________________________________________________________________
activation_2 (Activation)    (None, 128, 128, 32) 0           batch_normalization_2[0][0]
_________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, 128, 128, 32) 9248        activation_2[0][0]
_________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 128, 128, 32) 128         conv2d_3[0][0]
```

5. Скомпилируйте сеть

Сеть компиляции в основном задает базовую конфигурацию сети, такую как оптимизатор, скорость обучения, функция потерь, функция оценки и т. Д.;

1) Наиболее часто используемым оптимизатором является оптимизатор Адама:

```
# 优化器
optimizer = Adam(0.0002, 0.5)
```

、

2) Функция потерь часто использует 'mse', а функция оценки использует 'точность':

```
self.unet.compile(loss='mse',
                  optimizer=optimizer,
                  metrics=['accuracy'])
```

3).В этой статье в качестве оценочной функции используется пользовательская функция Dice

$$\mathrm{Dice}(A,B) = \frac{2|A \cap B|}{|A| + |B|}$$

Согласно формуле расчета Dice, определена следующая функция оценки, которая будет использоваться в процессе обучения сети, поэтому ее необходимо определить с помощью функции, связанной с tf.

```
def metric_fun(self, y_true, y_pred):
    fz = tf.reduce_sum(2 * y_true * tf.cast(tf.greater(y_pred, 0.1), tf.float32)) + 1e-8
    fm = tf.reduce_sum(y_true + tf.cast(tf.greater(y_pred, 0.1), tf.float32)) + 1e-8
    return fz / fm
```

Скомпилируйте сеть следующим образом:

```
self.unet.compile(loss='mse',
                  optimizer=optimizer,
                  metrics=[self.metric_fun])
```

6. Учебные сети

1) Импорт набора данных

```
# 获得数据
x_train, x_label, y_train, y_label = self.load_data()
```

x_train, x_label – это тренировочный набор; y_train, y_label является тестовым набором;

2) Установите режим обучения

```
# 设置训练的checkpoint
callbacks = [EarlyStopping(patience=100, verbose=2),
             ReduceLROnPlateau(factor=0.1, patience=20, min_lr=0.0001, verbose=2),
             ModelCheckpoint('./weights/best_model.h5', verbose=2, save_best_only=True)
```

Среди них EarlyStopping означает досрочное окончание, если проигрыш верификационного набора не падает 100 раз подряд, обучение прекращается; verbose представляет метод отображения;

ReduceLROnPlateau означает снижение скорости обучения, если потеря верификационного набора не уменьшается в течение 20 последовательных раз, скорость обучения становится 1/10;

ModelCheckpoint представляет контрольную точку (которая может обеспечить возобновление точки останова) и сохраняет только лучшую модель.

3) Начните обучение

```
# 进行训练
results = self.unet.fit(x_train, x_label, batch_size=32, epochs=200, verbose=2,
                        callbacks=callbacks, validation_split=0.1, shuffle=True)
```

validation_split=0,1, что означает, что данные обучающего набора разделяются на 0,1 в качестве проверочного набора (обратите внимание, что это не тестовый набор);

batch_size=32, что означает, что размер данных каждого пакета

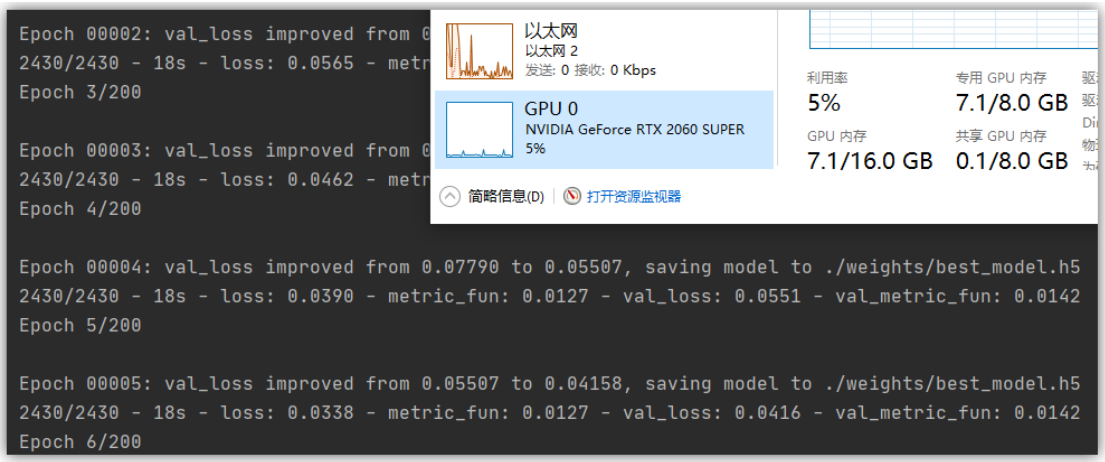
обучения составляет 32 изображения;

epochs=200, что означает в общей сложности 200 тренингов;

verbose=2, что означает, что обучающая информация печатается только после каждой эпохи обучения, или 0 и 1 могут быть взяты для обозначения различных методов печати информации;

shuffle=True, что означает, что каждая тренировка будет перетасовывать порядок тренировочной части тренировочного набора, и проверочный набор не будет затронут;

Учебный процесс:



4) Результаты обучения

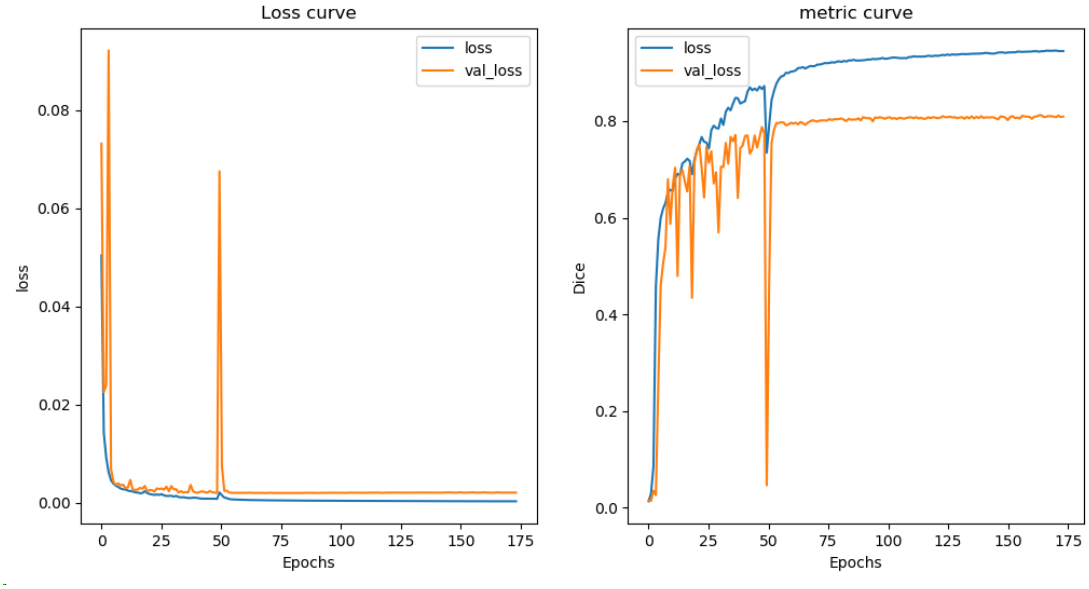


Рисунок 7 Кривая потерь и метрическая кривая тренировочного процесса

Из результатов на рисунке видно, что потеря проверочного набора (не участвующего в обучении) всегда больше, чем у обучающего набора, то есть эффект сегментации опухоли всегда хуже, чем у обучающего набора, что показывает, что мы оцениваем сеть для тестирования с помощью необученных данных, а сама сеть достигает тренировочного эффекта, постоянно подгоняя тренировочный набор, поэтому тестирование с данными обучающего набора не имеет большого значения. Коэффициент Dice набора проверки достигает 0,8, что является относительно высоким значением, указывающим на то, что эффект сетевой тренировки хороший.

Хотя мы установили количество тренировочных шагов до 200 шагов, средняя val_loss не спускалась 100 раз подряд, поэтому

тренировка была прекращена досрочно.

7. Тестирование и оценка

Для разных вопросов используемые оценочные показатели будут разными, и читатели могут построить свои собственные функции оценки, следуя методу настройки показателей оценки в этой статье. Ввиду проблемы сегментации опухоли в данной работе выделяют два основных показателя оценки, одним из которых является точность области сегментации, то есть коэффициент игральных костей; Второе – это точность сегментации опухоли, то есть точность.

7.1 Коэффициент игры в кости

Вычисление коэффициента Dice было введено ранее в пользовательскую функцию оценки, поэтому коэффициент Dice тестового множества прямо приведен здесь.

```
160/357 [============>.................] - ETA: 5s - loss: 0.0017 - metric_fun: 0.8275
192/357 [================>.............] - ETA: 3s - loss: 0.0017 - metric_fun: 0.8215
224/357 [==================>...........] - ETA: 2s - loss: 0.0019 - metric_fun: 0.8111
256/357 [====================>.........] - ETA: 1s - loss: 0.0020 - metric_fun: 0.8208
288/357 [========================>.....] - ETA: 1s - loss: 0.0020 - metric_fun: 0.8182
320/357 [==========================>...] - ETA: 0s - loss: 0.0021 - metric_fun: 0.8105
352/357 [============================>.] - ETA: 0s - loss: 0.0020 - metric_fun: 0.8174
357/357 [==============================] - 5s 15ms/sample - loss: 0.0020 - metric_fun: 0.8326
```

В более чем 300 тестовых наборах сегментированные опухоли достигли среднего коэффициента игральных костей 0,832; этот результат указывает на то, что сетевая сегментированная область опухоли очень похожа на маску врача.

7.2 Точность

Если в результате сегментации сети есть пиксели, равные 1, сеть считается имеющей выходное изображение с опухолью, а если результат сегментации сети равен 0, сеть считается имеющей выходное изображение без опухоли. Подсчитайте количество изображений, которые сеть определяет как правильные, а затем разделите на общее количество изображений, чтобы получить точность.

$$acc = \frac{n_{correct}}{n_{total}}$$

Метод реализации:

```
mask = self.unet.predict(x_train[index:index + batch_size]) > 0.1
mask_true = x_label[index, :, :, 0]
if (np.sum(mask) > 0) == (np.sum(mask_true) > 0):
    n += 1
```

Если сетевой вывод результата маскирует и реальное значение mask_true одной и той же ситуации, то есть обе опухоли или нет опухолей, то число составляет +1; окончательная точность тестового набора может быть получена:

```
schedule: 353/357
schedule: 354/357
schedule: 355/357
schedule: 356/357
the accuracy of test data is: 99.72%
```

7.3 Отображение эффекта сегментации сети

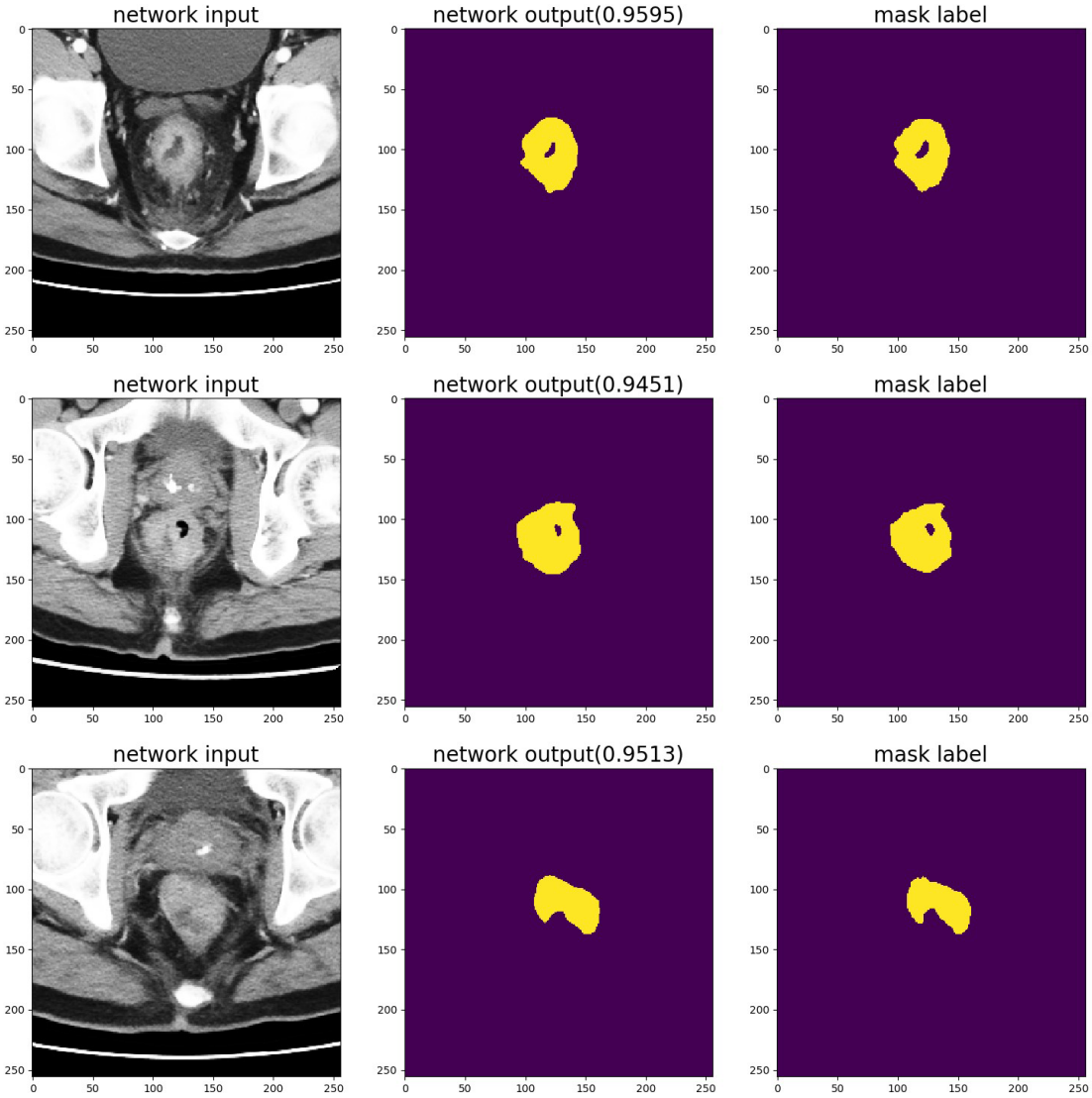


Рисунок 8 Влияние сетевой сегментации опухолей

Первый столбец – это вход сети, второй столбец – выход сети, а третий столбец – маска, предоставленная врачом. Тестовая оценка в этом разделе основана на тестовом наборе для обучения, что еще раз показывает, что сетевое обучение очень успешно.

Заключение

Чтобы помочь врачам определить местоположение и размер опухолей и определить, произошло ли метастазирование с помощью изображений КТ, модель сегментации U-net используется для выполнения ряда процессов, включая предварительную обработку данных, построение сети и компиляцию сети. Обучайте сеть, тестируйте и оценивайте. Наконец, через тест тестового набора доказано, что получается сеть с очень хорошим тренировочным эффектом.

Литература

[1] VAN ASCH C J J，LUITSE M J A，RINKEL G J E，et al. Incidence, case fatality, and functional outcome of intracerebral haemorrhage over time, ccording to age, sex, and ethnic origin: a systematic review and meta-analysis. Lancet Neurol，2010，9(2): 167–176.

[2] WANG W， JIANG B， SUN H， et al. Prevalence, incidence, and mortality of stroke in China clinical perspective. Circulation，2017， 135(8): 759–771.

[3] GARCIA GARCIA A，ORTS ESCOLANO S，OPREA S O，et al. A review on deep learning techniques applied to semantic seg- mentation [EB / OL]. [2017 - 04 - 22]. https：//arxiv. org / abs / 1704.06857.