МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Кафедра: высокопроизводительных вычислений и системного программирования

Направление подготовки: «Прикладная математика и информатика»

Профиль подготовки: «общий профиль»

ОТЧЕТ

по учебной практике

на тему:

«Семантическая сегментация мультимодальных изображений растений с помощью глубокого обучения»

Выполнил:
студент группы ПМоп3
Смирнов Илья Кириллович

_____
Подпись

Научный руководитель:
доктор технических наук,
профессор кафедры ВВСП
Турлапов Вадим Евгеньевич

_____
Подпись

Нижний Новгород
2024

Оглавление

Введение

Применение сегментации изображений.

Сегментация изображения — это метод, при котором цифровое изображение разбивается на различные подгруппы, называемые сегментами изображения, что помогает уменьшить сложность изображения и упростить дальнейшую обработку или анализ изображения. Сегментация — это присвоение меток пикселям. Все элементы изображения или пиксели, принадлежащие к одной и той же категории, имеют общую метку, назначенную им. Сегментация изображений — ключевая задача в компьютерном зрении, разделяющая визуальные данные на сегменты для специализированной обработки. Применяется в робототехнике, медицинской визуализации, автономных транспортных средствах и видеоаналитике.

Основными областями применения сегммментации изображений являются: медицина (рентгеновская и КТ-сегментация, сегментация стоматологических и патологических экземпляров), умные города (обнаружение пешеходов и автомобилей, аналитика трафика и распознавание номерных знаков, видеонаблюдение), самоуправляемые автомобили (семантическая сегментация дорожного покрытия, обнаружение автомобилей, пешеходов, объектов в салоне и выбоин).

Рассматриваемая мной модель сегментации данных будет применяться на наборе данных для агрокультуры. Сегментация изображений в агрокультуре играет важную роль в повышении эффективности сельскохозяйственного производства и обеспечении продовольственной безопасности. Среди областей применения есть:

1. Обнаружение и подсчёт растений для контроля за ростом и развитием культур.

2. Определение состояния растений: здоровье, зрелость и наличие вредителей.

3. Мониторинг состояния почвы: определение уровня влажности, плотности и наличия питательных веществ. Эта же область включает и работу с мультиспектральными данными, о которых будет рассказано позже.

4. Алгоритмы сегментации могут использоваться для автоматического обнаружения и подсчёта сорняков на полях.

5. Обнаружение дефектов продукции.

Формат EXR

В данной работе будут применяться изображения в формате .exr. Специфика этих данных будет рассмотрена позже, а в текущем разделе мы сосредоточимся на особенностях указанного формата.

Целью формата EXR является точное и эффективное представление данных сцен с высоким динамическим диапазоном в линейном формате. Это значительное отличие от большинства форматов изображений, которые хранят данные, готовые для отображения. Программное обеспечение, работающее с изображениями в формате OpenEXR, может требовать их обработки иначе, чем изображения в других форматах, таких как JPEG.

Основные особенности:

1) Широкий динамический диапазон.

Формат EXR сохраняет данные в высокоточном формате с плавающей точкой, что позволяет хранить больше деталей как в светлых, так и в темных областях изображения. Это особенно полезно для сложных сцен с высокими контрастами.

2) Поддержка нескольких каналов.

EXR поддерживает произвольное количество цветовых и дополнительных каналов (например, глубина, альфа-каналы), что может быть использовано для улучшения сегментации.

3) Прецизионное представление данных.

EXR позволяет сохранять изображения с 32-битной точностью на канал, что обеспечивает более точное представление градиентов, текстур и других мелких деталей. Это важно для задач сегментации, требующих высокого уровня точности.

Цели и задачи работы

Цель работы: Оценка и доработка модели сегментации изображений на основе архитектуры SegNet, с применением изображений в формате EXR, для демонстрации возможностей сегментации данных в агрокультуре.

Задачи работы:

1. Подготовка данных:
    - Сбор и предварительная обработка изображений в формате EXR.
    - Нормализация и преобразование данных в пригодный для работы с моделью вид.
2. Реализация модели SegNet:
    - Изучение архитектуры SegNet и ее адаптация под задачи сегментации изображений EXR.
    - Настройка гиперпараметров модели, таких как количество слоев, размер фильтров, функция активации и другие.
3. Обучение и тестирование модели:
    - Разделение данных на обучающую, валидационную и тестовую выборки.
    - Обучение модели с использованием данных EXR.
    - Оценка производительности модели с помощью метрик сегментации.
4. Подготовка выводов и предложений:
    - Формулировка выводов о применимости SegNet для сегментации EXR-изображений.
    - Рекомендации по дальнейшему улучшению модели и использованию формата EXR в других задачах.

Обучение модели

Описание данных.

Исходная модель и данные были заимствованы с платформы Kaggle: Synthetic RGB-D data for plant segmentation. Набор данных включает 50 тысяч файлов в формате EXR. Он разделен на пять подпапок: rgb, depth, semantic, instances masks и node. Каждое растение классифицируется по пяти классам: лист, черешок, стебель, плод и фон.

Так как данные представнены в EXR, то разберем подробнее какие существуют каналы в мультиспектральных данных и какие представлены в нашем датасете.

Основные типы каналов и их назначения:

1) Видимый спектр (RGB-каналы)
2) Инфракрасный спектр (NIR, SWIR, LWIR):

- Ближний инфракрасный (NIR, Near Infrared) используется для определения состояния растительности, анализа содержания воды, изучения почвы.
- Средний инфракрасный (SWIR, Short-Wave Infrared) применяется для обнаружения влажности, минералов и анализа поверхности Земли.
- Дальний инфракрасный (LWIR, Long-Wave Infrared) позволяет измерять тепловое излучение. Используется в тепловизорах для диагностики тепловых процессов.

3) Ультрафиолетовый спектр (UV) позволяет выявлять материалы с флуоресценцией, обнаруживать биологическую активность.
4) Каналы глубины и расстояния
5) Тепловые каналы показывают распределение температуры на поверхности объектов.
6) Альфа-канал. Дополнительный канал для прозрачности, часто используется в визуализации и композитной графике.

Обработка мультиспектральных каналов:

- Нормализация данных: Приведение значений пикселей к одному масштабу (например, 0, 1 или 0, 255).
- Удаление шумов: Устранение шумов с помощью фильтрации (например, медианный фильтр, Gaussian Blur).
- Сокращение количества каналов путем анализа важности (например, метод главных компонент, PCA).

Для работы с файлами формата EXR используется функция exr_to_jpg, представленная в Листинге 1 в приложении. Изображения содержат три канала, что можно определить, вызвав метод header() у открытого изображения. После преобразования изображения в массив его значения нормируются для обеспечения возможности обработки моделью.

Перейдем к обучению модели. В процессе обучения для обеспечения разумного времени обучения было решено разделить набор данных на три части: обучающую выборку (4000 изображений), проверочную выборку (1000 изображений) и контрольную выборку (1000 изображений).

Для данного датасета аугментации данных не применялись, поскольку предполагается, что имеющегося разнообразия данных достаточно для обучения модели. Аугментации данных используются для увеличения вариативности и повышения обобщающей способности модели.

Модель SegNet.

SegNet — это архитектура глубокого обучения, разработанная для семантической сегментации изображений, где целью является классификация каждого пикселя изображения в предопределенную категорию. Архитектура включает сеть кодировщика и соответствующую сеть декодера, за которыми следует финальный полносвязный слой классификации.

Сеть кодировщика в SegNet состоит из 13 сверточных слоев, повторяющих первые 13 сверточных слоев сети VGG16, которая изначально была разработана для классификации объектов. Ключевые моменты сети кодировщика:

1. Предустановленные веса: использование предустановленных весов VGG16 позволяет эффективно инициализировать и ускорить сходимость при обучении.
2. Сверточные слои.
3. Пакетная нормализация: за каждым сверточным слоем следует пакетная нормализация, чтобы стабилизировать и ускорить процесс обучения.
4. Функция активации ReLU.
5. Max-pooling: используется max-pooling с окном 2×2 и шагом 2, уменьшая пространственное разрешение карт признаков вдвое. Этот шаг помогает достичь инвариантности к сдвигу при малых пространственных смещениях.

Операция max-pooling приводит к потере информации об изображении, особенно в терминах граничных деталей, которые важны для задач сегментации. Чтобы смягчить эту потерю, сохраняются местоположения максимальных значений признаков в каждом окне пулинга (max-pooling индексы). Эта информация позже используется в сети декодера для точного восстановления масштаба.

Сеть декодера состоит из 13 слоев, каждый из которых соответствует слою кодировщика. Процесс декодирования предназначен для восстановления масштаба карт признаков до исходного разрешения изображения. Ключевые шаги в сети декодера:

1. Восстановление масштаба с использованием индексов max-pooling.
2. Конволюция с обучаемыми фильтрами. Разреженные карты признаков свертываются с обучаемыми фильтрами декодера для получения плотных карт признаков.

3. Пакетная нормализация. Как и в кодировщике, пакетная нормализация применяется к каждому слою сети декодера.

4. Классификатор Softmax. Финальный выход сети декодера проходит через многоклассовый классификатор softmax, который присваивает вероятности класса каждому пикселю. Предсказанная сегментация получается путем выбора класса с наибольшей вероятностью для каждого пикселя.

Параметры обучения

Функция потерь: dice_coef_loss

Для обучения сети используется функция потерь, основанная на коэффициенте Дайса (Dice Coefficient). Эта метрика измеряет схожесть между предсказанными и истинными масками.

Функция потерь определяется как $Loss = 1 - Dice\ Coefficient$, где коэффициент Дайса стремится к 1 для идеально совпадающих предсказаний и истинных меток. Чем больше совпадение между предсказанной и истинной масками, тем меньше значение функции потерь.

Оптимизатор: adam

Используется алгоритм оптимизации Adam (Adaptive Moment Estimation), который: комбинирует преимущества двух методов: AdaGrad (хорошо работает на разреженных данных) и RMSProp (хорошо работает при нестабильных градиентах).

Метрики: accuracy и dice_coef.

Для мониторинга качества модели используются две метрики:

1. accuracy:
    - Стандартная метрика точности. Она измеряет долю правильно предсказанных пикселей (в бинарной классификации — попиксельно).
    - Подходит для общей оценки, но может быть нечувствительной к небольшим ошибкам в сегментации.
2. dice_coef:

    Коэффициент Дайса, рассчитываемый как:

    $Dice\ Coefficient = \frac{2*|A \cap B|}{|A|+|B|}$, где A — предсказанная маска, B — истинная маска.

    Он измеряет баланс между ложноположительными и ложноотрицательными пикселями, что делает его особенно полезным для несбалансированных данных (например, когда объекты занимают небольшую часть изображения).

Особенности реализации

Изначальный исходный код проекта не был адаптирован для работы с новой версией TensorFlow с использованием GPU. Для решения этой проблемы пользовательские слои MaxPoolingWithArgmax2D и MaxUnpooling2D, которые изначально применялись в модели, были заменены на стандартные слои MaxPooling2D и UpSampling2D, встроенные в библиотеку Keras. Данная модификация позволила значительно сократить время обучения модели: с 5–6 часов на CPU до одного часа на GPU.

Для повышения производительности при работе с датасетом генератор, реализующий его, был преобразован в объект типа tf.data.Dataset. Это позволило использовать встроенные механизмы оптимизации, предоставляемые TensorFlow. Также была модифицирована исходная функция data_gen_small, представляющая собой Python-генератор, который обрабатывает изображения из датасета. В обновленной версии добавлена поддержка параллельной обработки изображений, что существенно ускоряет выполнение операций. Реализация данной функции приведена в Приложении в Листинге 2.

Дополнительно для минимизации задержек во время обучения была внедрена предварительная загрузка (prefetching) следующих изображений в память. Это позволяет эффективно использовать время простоя процессора или графического ускорителя. Данный механизм реализован с использованием встроенной функции dataset.prefetch(buffer_size=tf.data.AUTOTUNE), которая автоматически определяет оптимальный размер буфера для предзагрузки.

Помимо этого, была выявлена ошибка в исходном методе считывания изображений. Формат EXR некорректно интерпретировался функцией imageio.imread(). Для устранения этой проблемы изображения были преобразованы с использованием библиотеки OpenEXR. Дополнительно была выполнена нормализация значений пикселей: значения каждого пикселя делились на его максимальное значение. Это позволило улучшить восприятие изображений моделью. Реализация функции считывания и нормализации приведена в приложении к работе (Листинг 1).

Обучение

Первоначальное обучение модели с использованием стандартных настроек оказалось неэффективным: максимальная точность сегментации составляла около 30% (подробности представлены в приложении).

Для улучшения качества работы модели был проведен эксперимент с различными оптимизаторами. Применение RMSprop позволило увеличить точность на обучающем наборе данных до 70%, однако модель быстро переобучалась, поскольку точность на валидационных данных оставалась на уровне 30%. Использование оптимизатора Adagrad так же не дало значительных улучшений.

Интересные результаты были получены при комбинированном подходе: применение предварительно обученных весов после обучения с RMSprop в сочетании с оптимизатором Adagrad позволило повысить точность модели и коэффициент Дайса до 70%.

Для дальнейшего улучшения модели были внесены изменения в её архитектуру. В частности, были добавлены остаточные связи и прореживание (dropout).

Смысл остаточных связей заключается в том, что входные данные слоя или блока слоев добавляются в его выходные данные. Остаточные связи действуют как короткие пути для распространения информации в обход деструктивных блоков или блоков, вносящих существенные искажения, позволяя информации градиента ошибок проходить по глубокой сети без искажений. Остаточные связи обеспечивают более эффективный градиентный поток через глубокие слои модели, что позволяет избегать затухания градиентов. Прореживание, в свою очередь, уменьшает риск переобучения за счет регуляризации, отключая случайные нейроны в процессе обучения.

В результате внедрения указанных модификаций удалось достичь 85% в точности и значения функции потерь равное 0.25 (соответственно 75% по коэффициенту Дайса), что свидетельствует о значительном улучшении качества сегментации (см. рис).

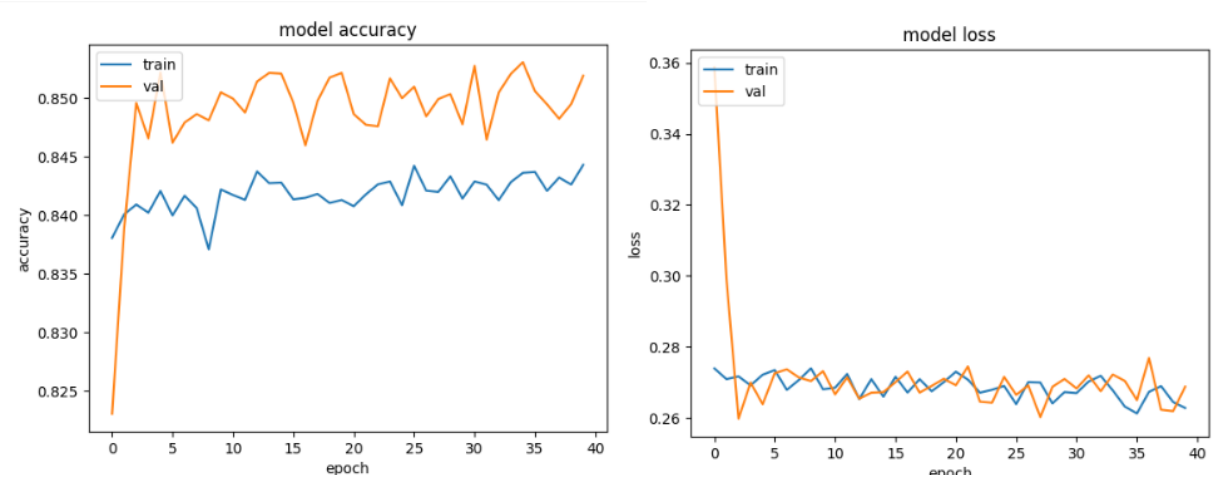


Рис.1

Заключение

В данной работе проведен анализ работы модели SegNet для задачи сегментации на данных в формате EXR. Полученная точность сегментации маски составила 75%, что свидетельствует о том, что модель способна эффективно решать поставленную задачу. Это подтверждает применимость SegNet для обработки сложных изображений с высоким динамическим диапазоном.

Однако в ходе работы были выявлены несколько проблем, требующих внимания. Во-первых, модель демонстрировала снижение точности при обработке областей с мелкими деталями, что указывает на необходимость улучшения способности модели к захвату локальных особенностей изображения. Во-вторых, использование EXR-данных связано с увеличенной нагрузкой на память и вычислительные ресурсы, что может ограничивать масштабируемость подхода на больших датасетах.

Тем не менее, результаты имеют значимость, так как применение SegNet для сегментации EXR-данных открывает перспективы в областях, где точность сегментации и работа с мультимодальными изображениями критически важны, например, в компьютерной графике, визуализации или инженерном моделировании.

Для дальнейшего улучшения результатов можно рассмотреть следующие направления:

1. Использование современных архитектур, таких как U-Net++ или DeepLab, которые лучше захватывают детали изображения.

2. Постобработка масок путем использования методов сглаживания или доработки границ для повышения точности маски.

Таким образом, достигнутые результаты демонстрируют перспективность предложенного подхода, но также указывают на возможные пути для повышения эффективности и точности сегментации. Сами результаты сегментации представлены в приложении (Рис и Рис).

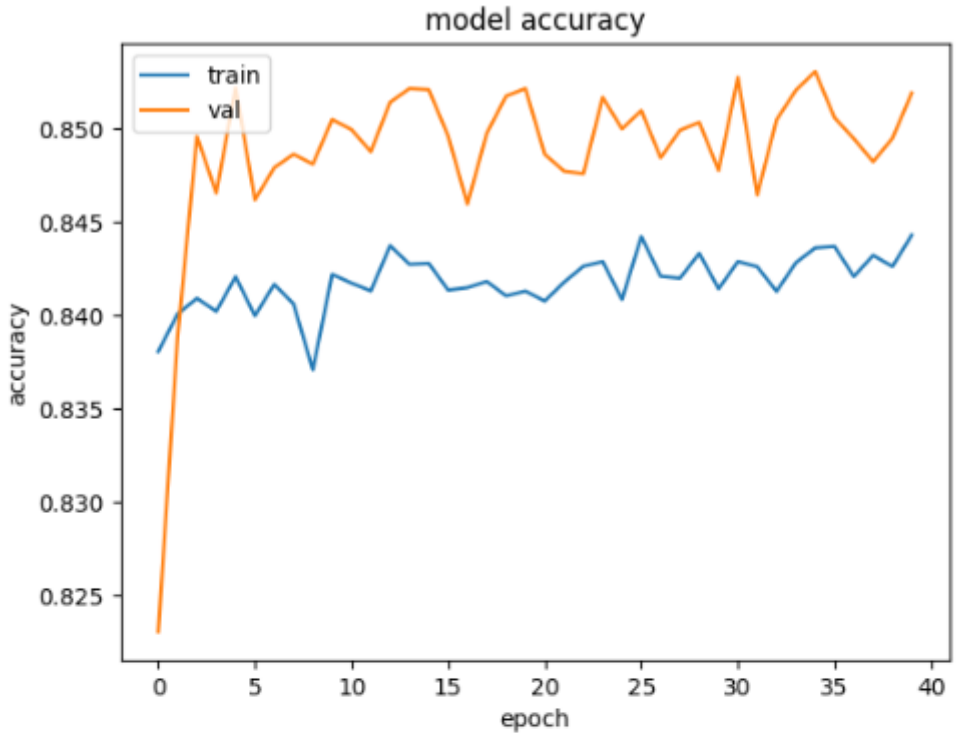
Приложение

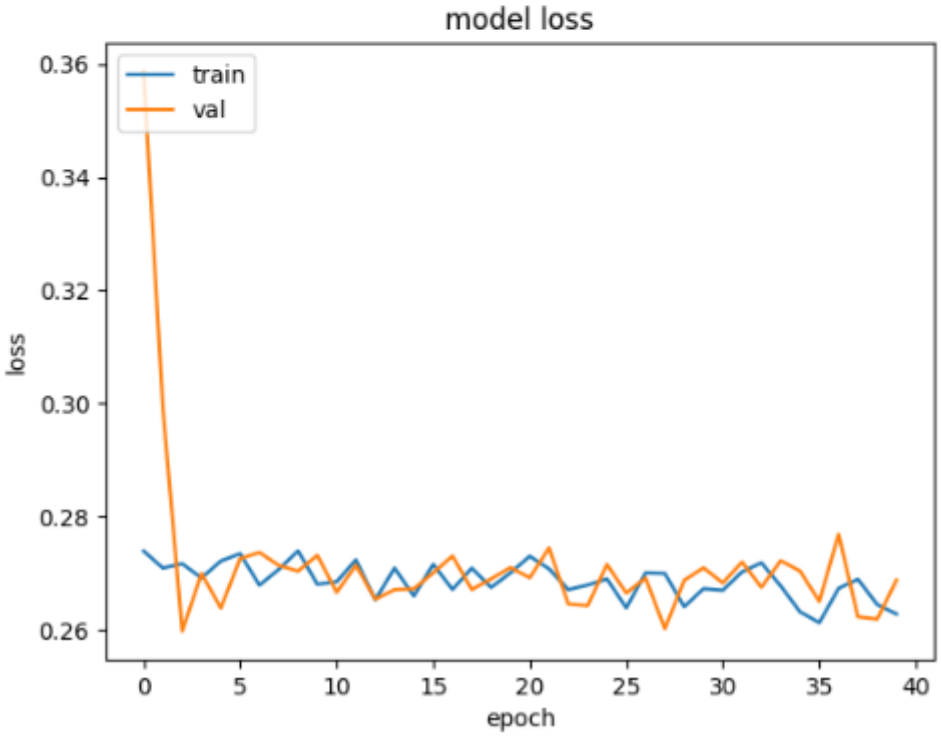
```
def exr_to_jpg(path):
  exr_file = OpenEXR.InputFile(path)
  header = exr_file.header()
  data_window = header['dataWindow']
  width = data_window.max.x - data_window.min.x + 1
  height = data_window.max.y - data_window.min.y + 1
  channels = ["R","G","B"]
  data = [np.frombuffer(exr_file.channel(channel, Imath.PixelType(Imath.PixelType.FLOAT)), dtype=np.float32) for channel in channels]
  array = [np.clip(channel.reshape(height, width) / (np.max(channel)+ 1e-7), 0, 1).astype(np.float32) for channel in data]
  image = np.dstack(array)
  return image
```

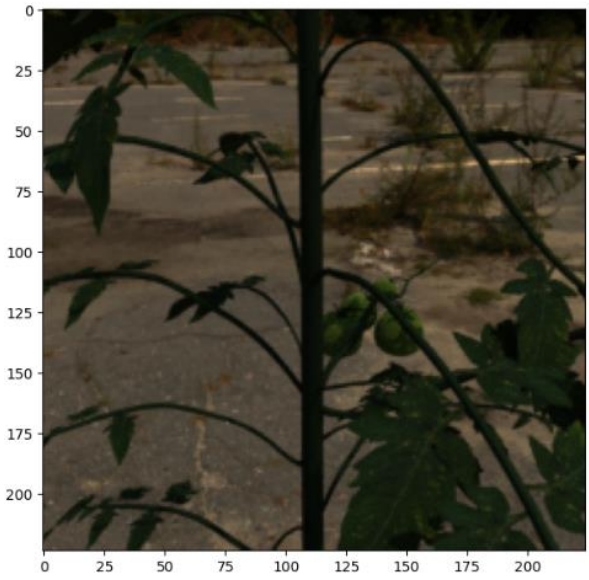
Листинг 1. Считывание и нормализация изображений.

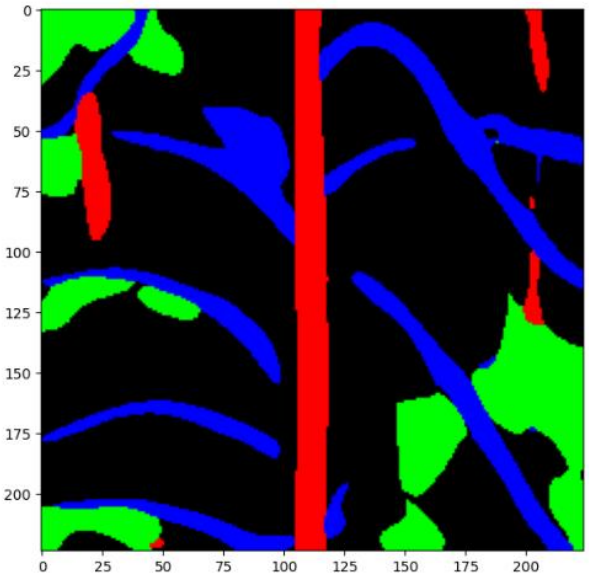
```
import concurrent.futures

def process_file(index, img_dir, mask_dir, dims, n_labels):
  img_path = img_dir[index]
  original_img = exr_to_jpg(img_path)
  array_img = img_to_array(original_img) / 255.0

  mask_path = mask_dir[index]
  original_mask = cv2.imread(mask_path, cv2.IMREAD_ANYCOLOR | cv2.IMREAD_ANYDEPTH)
  array_mask = category_label(original_mask[:, :, 0], dims, n_labels)

  return array_img, array_mask

def data_gen_small(img_dir, mask_dir, depth_dir, liste, batch_size, dims=(224, 224), n_labels=5):
  with concurrent.futures.ThreadPoolExecutor() as executor:
    while True:
      ix = np.random.choice(liste, batch_size)
      results = list(executor.map(
        lambda index: process_file(index, img_dir, mask_dir, dims, n_labels), ix)
      )
      imgs, labels = zip(*results)
      yield np.array(imgs), np.array(labels)
```

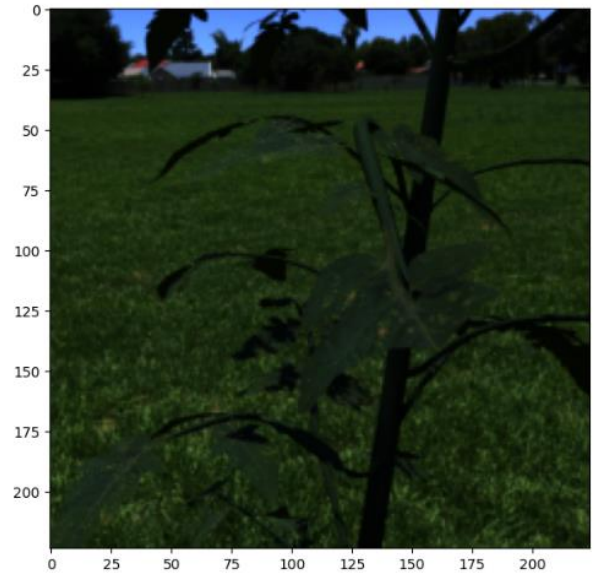
Листинг 2. Параллельная генерация изображений.

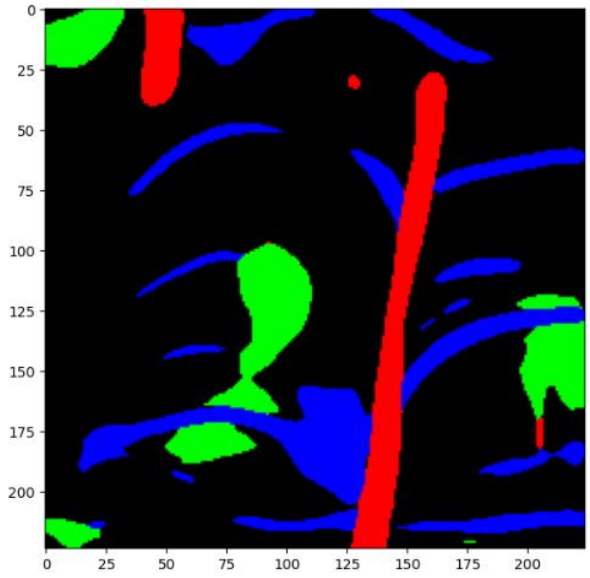

model accuracy
train
val
accuracy
0.850
0.845
0.840
0.835
0.830
0.825
0
5
10
15
20
25
30
35
40
epoch


model loss
train
val
loss
0.36
0.34
0.32
0.30
0.28
0.26
0
5
10
15
20
25
30
35
40
epoch


0
25
50
75
100
125
150
175
200
0 25 50 75 100 125 150 175 200


0
25
50
75
100
125
150
175
200
0 25 50 75 100 125 150 175 200


0
25
50
75
100
125
150
175
200
0 25 50 75 100 125 150 175 200


0
25
50
75
100
125
150
175
200
0 25 50 75 100 125 150 175 200