Дефектоскопия тканей

Северюхина Анастасия
гр. 5040102/30201

Общее описание задачи

Необходимо построить универсальную нейронную сеть для распознавания дефектов текстильных материалов. Автоматизация контроля качества продуктов легкой промышленности на основе анализа параметров качества продукции позволит обеспечить качественное изменение процесса изготовления текстильных изделий, а также повысить качество продукции и производительности оборудования.

https://spbpu.com/novosti/sistemu-raspoznavaniya-defektov-tkani-obuchayut-dlya-raboty-s-gladkokrashenym-materialom/

Модель распознавания

В качестве базы для разработки алгоритмов обнаружения и локализации дефектов применяется модель бинарной классификации пикселей. Т.о есть генерируется маска предсказания: есть ли в данном пикселе дефект или нет.
Далее решается задача семантической сегментации – поиск пикселей изображения, которые относятся к объекту одного типа. Для решения задачи в данный момент используется архитектура YOLOv8

Результаты работы YOLOv8s

Результат работы нейронной сети представляет собой маску по контурам найденных дефектов для изображений, поступивших с камер на вход нейронной сети. Выполняется расчет координат непосредственно на кадре и также пересчет в глобальные координаты рулона. Таким образом формируется запись о том, что дефект найден, а также во сколько, какой и какие у него координаты.

Пример локализации дефекта на изображении

JSON-файл с информацией по изображению

После отработки алгоритма по поиску дефекта, получаем json файл с информацией по каждому изображению.

Эти данные содержат:

- координаты точек, ограничивающих площадь дефекта
- класс дефекта
- время обнаружения и т.д.

Реализация препроцессинга

Есть набор тестовых данных с установки. Данные содержат изображения, полученные с камер при тестировании установки, а также файлы с характеристиками изображения (размер, наличие/отсутствие дефекта, класс дефекта и т.д.)

В качестве препроцессинга необходимо было создать набор сценариев для того, чтобы разделить полный датасет на части с заданным набором параметров.

JSON-файл для препроцессинга

Был создан json-файл, на верхнем уровне которого находятся задачи (бизнес-процессы), например, обработка больших дефектов.

В рамках каждой задачи формулируется набор гипотез о том, какой вариант препроцессинга будет лучше решать поставленную задачу. Каждая гипотеза содержит набор константных и изменяемых параметров препроцессинга.

Структура json с гипотезами. Пример


```
"task_name" : "Обработка маленьких дефектов",
"task_directory": «…\\small_defects",
"constant_parameters":{
                "filter_defects": {
                                "exclude_tags": null,
                                "include_only_tags": null,
                                "no_tag_exclude": true,
                                "area_threshold": 1000,
                                "threshold_up_direction": false},
"hypotheses": [
 {"name": "Значения параметров по-умолчанию",
   "directory": «..\\default_parameters_values",
   "parameters": {
                "YOLOPipeline": {
                                "test_dst_fld": «..\\test_dst_fld",
                                "stat_fld": «..\\stats"},
                "rescale": {
                                "reduce_coef": 4,
                                "scale_size": null},
                "split_on_tiles": {
                                "tile_size": 512}
                                }]
```

Препроцессинг

Данные из json считываются и передаются на вход программе препроцессинга, в рамках которой происходит формирование и заполнение данными дирректорий для каждой задачи и далее внутри для каждой гипотезы. В том числе формирование и заполнение json файла со статистическими данными по датасету гипотезы.

Дополнительно написана программа для сбора статистики по датасету – подсчету количества изображений датасета, в которых есть дефекты.

Структура json со статистикой. Пример


```
{
"tags_stat": {"C.2.4": 19, "C.2.1": 94},
"defects_number": 101,
"img_number": 504,
"min_defect_area": 3.1426,
"max_defect_area": 690.8607,
"mean_defect_area": 72.0877,
"median_defect_area": 35.9519
}
```

Разметка данных

Необходимо реализовать программу, которая будет обрабатывать информацию о локализации дефектов и обрезать изображение, которое далее будет обрабатываться классификатором.

Один из методов препроцессинга разбивает изображение на определенное количество тайлов. Необходимо определить набор тайлов, в котором содержатся дефекты.

Создание датасета тайлов

Реализована программа, которая сохраняет тайлы изображения в отдельную директорию и формирует csv-файл с информацией о наличии/отсутствии дефекта в данном тайле.

tile_name	tile_label
..\tiles_imgs\frame_1654787049813000_id-0019692213Cropped_tile_0.jpg	0
..\tiles_imgs\frame_1654787049813000_id-0019692213Cropped_tile_1.jpg	0
..\tiles_imgs\frame_1654787049813000_id-0019692213Cropped_tile_2.jpg	0
..\tiles_imgs\frame_1654787049813000_id-0019692213Cropped_tile_3.jpg	1
..\tiles_imgs\frame_1654787054146000_id-0019692239Cropped_tile_0.jpg	1

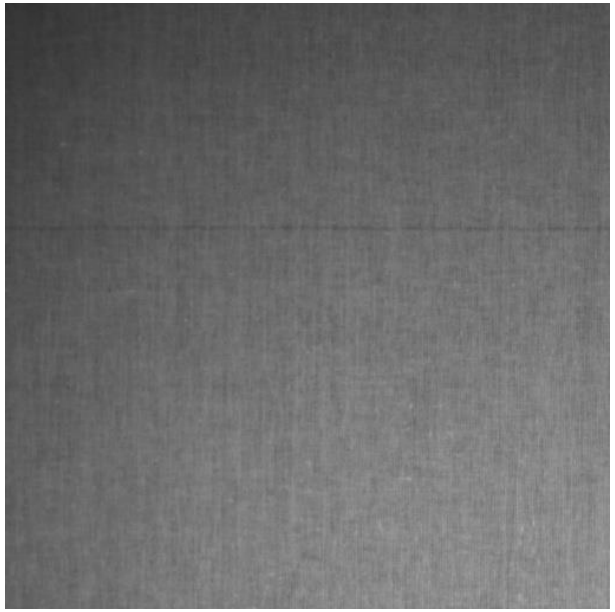
Пример данных из csv-файла с информацией о директории и наличии/отсутствии дефекта в тайле

Формирование датасета

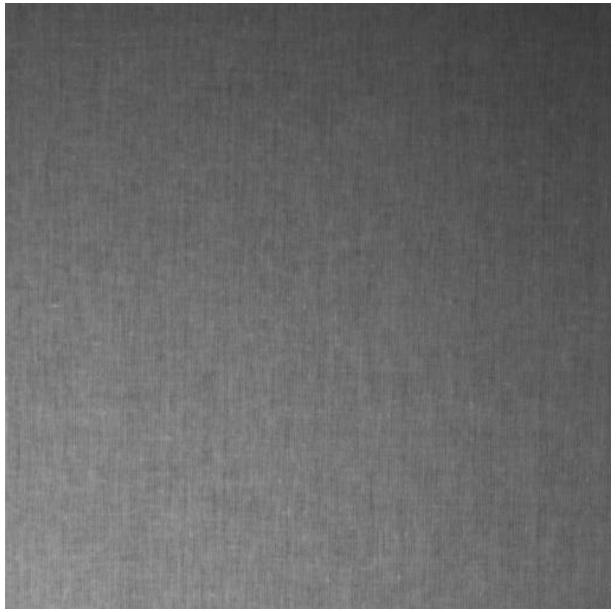
Набор изображений был переорганизован следующим образом:

tiles_images:
- train:
  - img_with_defects
  - img_without_defects
- test:
  - img_with_defects
  - img_without_defects

Пример изображения с дефектом



Пример изображения без дефектов



Подготовка данных для пакетной загрузки в модель

```
transforms = transforms.Compose([transforms.Resize(224),
                                 transforms.ToTensor()])

img_data = datasets.ImageFolder(tiles_path, transform=transforms)

def prepare_dataloader(dataset, current_gpu_index, num_gpus, batch_size):
    sampler = torch.utils.data.distributed.DistributedSampler(
        dataset,
        num_replicas=num_gpus,
        rank=current_gpu_index,
        shuffle=False
    )
    dataloader = torch.utils.data.DataLoader(
        dataset,
        sampler=sampler,
        batch_size=batch_size,
        shuffle=False
     )
     return dataloader
```

Модуль torchvision.transforms PyTorch предоставляет множество распространённых методов переобразования данных.

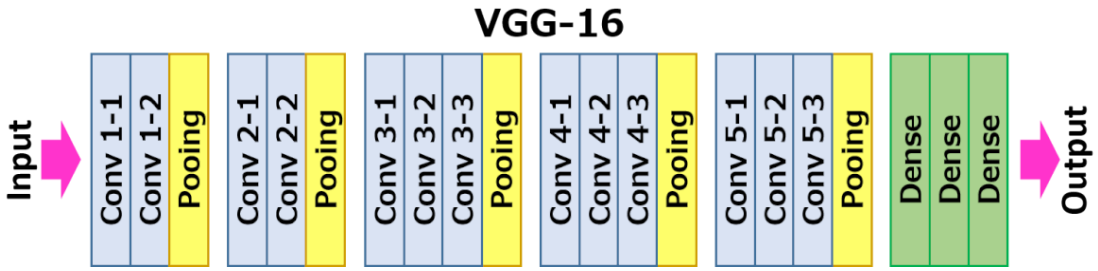
DataLoader используется для пакетной загрузки данных для обучения и проверки.

Разметка датасета с изображениями

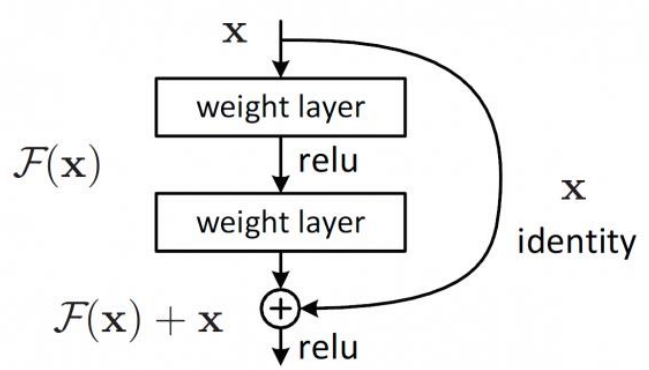
Для разметки данных, была написана программа реализации обучения для каждой из трех следующих моделей нейронных сетей: ResNet50, VGG16 и VGG19 из открытой библиотеки Keras.

Также был использован фреймворк PyTorch. Он поддерживает распределённые вычисления с помощью модуля DistributedDataParallel и позволяет обучать модель на нескольких графических процессорах или машинах.

Для использования распределённых вычислений в программе используется torch.utils.data.distributed.DistributedSampler.



Архитектура нейронной сети VGG16



Shortcut connections из архитектуры нейронной сети ResNet

Обучение модели

```
dataset = img_data
model = get_model_resnet50()

# prepare the dataloader
dataloader = prepare_dataloader(dataset, current_gpu_index,
num_gpu, batch_size)

# Instantiate the torch optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

# Instantiate the torch loss function
loss_fn = torch.nn.CrossEntropyLoss()

# Put model on device
model = model.to(current_gpu_index)
ddp_model = torch.nn.parallel.DistributedDataParallel(
    model, device_ids=[current_gpu_index],
output_device=current_gpu_index)

train_model(ddp_model, dataloader, num_epochs, optimizer,
loss_fn)
```

Оптимизатор Adam широко используется в различных моделях глубокого обучения, благодаря адаптивной скорости обучения, коррекции смещения и эффективности использования памяти.

Функция потерь CrossEntropyLoss() измеряет разницу между предсказанными вероятностями классов и истинными метками классов.

DistributedDataParallel (DDP) — это модуль в PyTorch, который позволяет параллелизовать модель.

Результаты

- Создан json файл для реализации препроцессинга данных. В рамках каждой задачи формулируются гипотезы о том, какой вариант препроцессинга будет лучше решать поставленную задачу.
- Реализовано разделение датасета на части, соответствующие разным гипотезам. Это поможет в дальнейшем лучше обучить классификатор.
- Реализовано создание датасета тайлов изображений с разметкой данных (с информацией о наличии/отсутствии дефектов).
- Проанализирована структура сверточных нейронных сетей и написана программа для реализации разметки данных с применением cuda для распределенных вычислений на GPU.

Задачи для дальнейшей работы

Улучшение модели предсказания классов дефектов.

- Одни дефекты встречаются чаще, другие реже, поэтому нужно использовать модель с весами для каждого класса
- К решению задачи семантической сегментации добавить решение задачи экземплярной сегментации. Это поможет улучшить работу модели в случае, когда на изображении присутствует много дефектов одного класса.