МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

ПРОГНОЗИРОВАНИЕ ДИАГНОЗОВ ЗАБОЛЕВАНИЙ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

Студентки 2 курса 248 группы
направления 09.04.03 — ПРИКЛАДНАЯ ИНФОРМАТИКА

механико-математического факультета
Рогачевой Арины Валентиновны

Научный руководитель
к. э. н., доцент _____ А. Р. Файзлиев

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2021

Введение. Задачи анализа рентгенографий, томографий, УЗИ принадлежит области под названием Medical Imaging.

Одна из основных тенденций в медицинском секторе - широкое использование томографий, что приводит к большим объемам рентгеновских снимков, УЗИ- сканирований и МРТ. В то время как практика использования томографий становится все обширнее, рынок испытывает острую нехватку специалистов. способных проанализировать результаты, радиологи испытывают нагрузку при оценке возросшего количества данных. Крупномасштабное исследование, проведенное в клинике Мейо показало что в течение десятилетия радиологи перешли с чтения трех изображений МРТ в минуту до 12 [1]. Поскольку в последние годы использование КТ и МРТ значительно возросло, необходимо проделать большую работу примерно с тем же числом радиологов. Авторы статьи отмечают риск увеличения количества ошибок из-за переутомления медицинских работников. Кроме того, важно помнить, что профессия рентгенолога выходит за рамки анализа изображений, и с увеличением рабочей нагрузки в рамках одной задачи, другим обязанностям становится все труднее уделять время. Описанные выше проблемы могут быть решены путем автоматизации обработки результатов с помощью моделей глубокого обучения [2].

Анализ томографий относится к области анализа изображения или Computer Vision. Задачи, выполняемые в этой области можно поделить на два класса: задача классификации и задача сегментации. В задаче классификации по входному изображению результатом будет предсказание болезни, полученное с помощью предсказательной модели глубокого обучения. Недостатком такого подхода несмотря на высокую точность, можно назвать низкую интерпретируемость процесса работы [3]. Данный подход может быть использован для полной автоматизации процесса анализа болезней. Результатом сегментации являются метки на изображении (например опухоль), что не является окончательным решением, но помогает специалисту обратить внимание на фрагмент снимка и ускоряет его работу.

Целью работы является построение системы автоматизации анализа диагностических изображений для решения практической задачи рентгенографии с использованием методов машинного обучения.

Структура ВКР. В данной работе будут рассмотрены существующие подходы к решению задач классификации и сегментации и предложен подход для каждой задачи. Эффективность подхода будет продемонстрирована на публичном наборе данных с учетом общепринятых метрик оценки качества. Работа имеет следующую структуру:

— В первой части работы рассмотрены основные задачи и виды машинного обучения;

— Приведена общая теория нейронных сетей и введение в сверточные нейронные сети.

Основная часть работы состоит из нескольких подразделов:

— В главе "Определения"дана общая формулировка задачи в терминах машинного обучения, а так же все необходимые в дальнейшем определения;

— Приведены и описаны классические решения для задач классификации и сегментации;

— Описан предлагаемый алгоритм классификации, перечислены различия с классическим решением и приведена оценка качества его работы с помощью метрик;

— Описана архитектура нейронной сети для задачи сегментации, продемонстрирован процесс проведения численного эксперимента и приведена оценка качества его работы с помощью метрик.

Основное содержание работы. За последние годы были предложены несколько архитектур нейронных сетей для задач классификации и сегментации. Часть архитектур были изначально созданы для решения задач Medical Imaging, другие были разработаны для задач вне области Medical Imaging, но применены к ним в дальнейшем. В основе данной работы лежит подход адаптации state-of-the-art моделей из других областей к задаче Medical Imaging с использованием публичных медицинских данных в качестве обучающей выборки. Ниже описаны модели классификации и сегментации, которые были взяты за основу разрабатываемой системы [4].

Задача классификации. В качестве основного набора данных для обучения, оценки качества и сравнения архтектур в задаче классификации вне области Medical Imaging общепринято используется набор данных Imagenet [5].

Датасет ImageNet представляет собой набор из 14197122 изображений, каждому из которых дана в соответствие одна из 21841 категорий (классов). Для сравнения архитектур используется выборка датасета ImageNet и производится разбиение на 1000 классов.

Одна из последних архитектур, представленных дял задачи классификации - ResNet впервые превзошла уровень человека на наборе ImageNet в задаче распредления картинок на 1000 классов. Относительно предыдущих подходов, авторами были предложены шаги по снижению количества параметров, оптимизации процесса тренировки сети и борьбы с проблемой "затухающих градиентов- [6]. Предложенные нововведения позволили увеличить число слоев, увеличив качество работы не влияя на стабильность и скорость сходимости модели.

Архитектура нейронной сети состоит из слоев свертки 3×3 двух типов:

— Количество фильтров не изменяется, если размер выхода не изменяется;

— Если размер выхода уменьшается вдвое, количество фильтров удваивается.

Для снижения размера используются свертки с шагом 2. После 34 сверток 1 и 2 типа применяется average pooling и полносвязный слой на 1000 выходов и последующим применением softmax [7]. Сеть обучалась на датасете ImageNet путем минимизации функции потерь cross entropy между выходами сети размера 1000 и меткой класса, представленного в виде one-hot-encoding размера 1000. Обучение проводилось с помощью метода стохастического градиентного спуска. Данная архитектура активно применяется для решения задач классификации в различных областях. В данной работы сеть была использована архитектура ResNet, предобученная на датасете ImageNet, после чего данная архитектура была адаптирована к задаче классификации медицинских снимков с помощью подхода Trasnfer Learning. Данный подход позволяет получить лучшее качество модели путем переиспользования параметров модели, обученной на большом наборе исходных данных (в данном случае ImageNet) для целевой задачи (в данном задачи Medical Imaging). Подробнее применение подхода Trasnfer Learning и способы адаптации модели описанны в разделе вычислительный эксперимент [8].

Задача сегментации. По аналогии с задачей классификации опишем наборы и формат данных, используемых в задачах сегментации. Поскольку разметка данных в задачах сегментации более сложная чем в задачах классификации, существуют общепринятые стандарты записи и хранения таких данных. Одним из самых популярных форматов является формат COCO [9].

Формат COCO представляет собой файл формата json, полями которого являются:

- imagepath - имя файла с изображением;
- annotations - массив с разметкой сегментации. Содержит в себе объекты вида:
  - category - класс объекта;
  - segmentation - массив, хранящий координаты многогранника, описывающего границы данного объекта.

Создатели формата также представили одноименный датасет, состоящий из 330000 изображений. С учетом наличия нескольких размеченных объектов накаждом изображении, общее число объектов достигает 1.5 млн [10].

Для задач Medical Segmentation в 2015 г. была разработана архитектура Unet, которая представляет собой архитектуру вида encoder-decoder. Encoder данной сети состоит из последовательности операций свертки 3×3 (без паддинга), с примененной функцией активации Relu. Операция понижения размерности производится с помощью слоев pooling 2×2 со сдвигом 2. Часть Decoder позволяет сгенерировать изображение из сжатого представления входной картинки путем последовательного применения слоев upconvolution и операцией свертки 3×3 [11].

Размер тензора на входе данной сети составляет 572×572×1. Выходом данной сети является тензор размера 388×388×N, где N - количество возможных классов на изображении. Каждый канал выходного тензора соответствует бинарному распределению пикселей для конкретного класса. Значения 0 или 1 сигнализируют о принадлежности данного пикселя классу [12].

Данная архитектура обширно применяется для задач сегментации в различных областях, оставаясь ведущим подходом в облести Medical Imaging и являясь основой для state of the art архитектур [13]. В данной работе была использована реализация Unet в одной из популярных библиотек для сег-

ментации segmentation models. Данная библиотека реализована с помощью фреймворка машинного обучения Keras, также используемого в данной работе. Для адаптации модели Unet, обученной на датасете COCO использовались подходы Transfer Learning, а также техники аугментации и progressive growing во время тренировки модели. Процесс поэтапно описано в разделе вычислительный эксперимент.

Вычислительный эксперимент. В ходе данной работы была построена система принятия медицинских решений для решения задач классификации на наборе данных ChestX-ray8. Данный датасет состоит из 108948 снимков размера 1024×1024. Каждому из изображений дан в соответствие один из 8 классов. Классы соответствуют следующему набору патологий:

— Atelactasis;
— Cardiomegaly;
— Enfitration;
— Mass;
— Nodule;
— Pneumonia;
— Pneumathorax.

Пример изображений каждого класса с аннотациями представлен на рисунке 1 Так же рассмотрена задача сегментации для определения местонахожде-

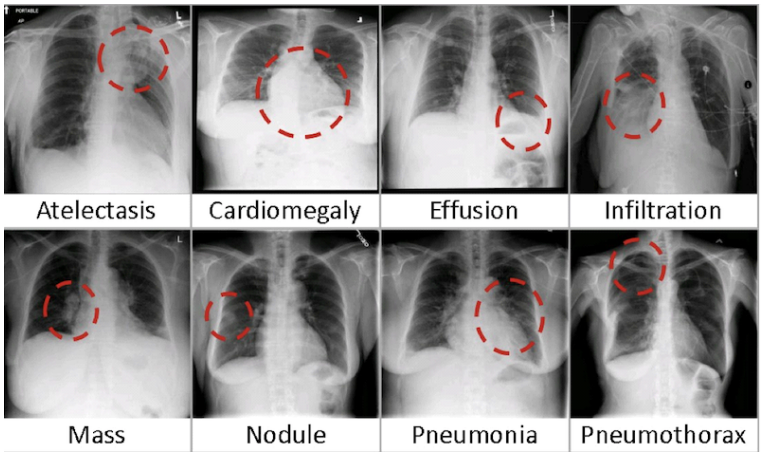

Рисунок 1 – Примеры снимков с метками классов.

ния одной из патологий на наборе данных mosmed, состоящий из 2049 снимков легких с бинарными масками, обозначающими местонахождение патологий. Примеры из набора данных представлены на изображении 2.

Рисунок 2 – Примеры снимка с бинарными масками сегментации. Слева направо: исходный снимок, маска пораженной области, маска области без патологий.

Датасеты были разбиты на тренировочную и тестовую части в соотношении 80/20. Подсчет качества итоговой модели был произведен на тестовой выборке, что позволяет оценить качество работы системы на новых данных в будущем. Разбиение было произведено с применение стратификации, суть которой в сохранении распределения и вариативности данных между тренировочной и тестовой выборками.

Для реализации данной системы были использованы:

— Язык программирования python;
— Фреймворк машинного обучения Keras;
— Библиотека предобученных моделей segmentation models;
— Библиотека для аугментации albumantations;
— Библиотека pydicom для работы с форматом Dicom медицинских снимков;
— Библиотеки matplotlib, numpy, scipy для визуализации результатов.

Процесс построения экспериментов разбит на несколько этапов и будет описан последовательно. Вначале данные из датасета для обучения считываются с помощью библиотеки pydicom и предварительно обрабатываются перед использованием для обучения нейронной сети. Процесс предобработки включает в себя исключение всех значений хаусфилда больше 500 и меньше 1500, и нормализации изображений путем вычета среднего значения пикселей и разделенное на среднеквадратическое отклонение. Подробная реализация представлена в Приложении А.

Для загрузки данных необходимо реализовать два интерфейса библиотеки Keras для загрузки данных - Dataset и Dataloader. Суть данных интерфейсов - предоставить возможно загружать данные по частям, ввиду ограничений оперативной и видеопамяти. Реализация предусматривает считывание

снимка с предварительно примененной нормализацией, нахождения правильного класса и маски сегментации для данного снимка и применение аугментаций, подробнее описанных в следующем пункте.

Для повышения устойчивости нейронной сети к афинным преобразованиям изображений, различий в сборе данных с различных приборов и повышения количества данных использовалась техника аугментации. Суть техники в изменении исходных изображений с помощью алгоритмов преобразования. В данном подходе использовались такие аугментации как:

— Поворот изображения от 0 до 360 градусов;
— Зеркальное отображение;
— Произвольное обрезание картинки.

После этого все изображения приводятся к единому размеру 512 на 512 пикселей. Подробный листинг реализации интерфейсов Dataset и Dataloader и построение аугментации представлено в Приложении Б.

Для задачи классификации использовалась вариация архитектуры ResNet под название ResNet-101. Модель была предобучена на датасете ImageNet. К данной архитектуре был применен подход transfer learning, в ходе которого были произведены следующие изменения: последний fully connected слой с входом 1024 и выходом 1000 был заменен на на fully connected слой с входом 1024 и выходом 8. Параметры нового слоя были инициализированы нормальным распределением с матожиданием 0 и дисперсией 1. Для всех слоев кроме fully connected было запрещено обновление параметров во время обучения вплоть до 10 эпохи.

Для задачи сегментации была выбрана и обучена модель Unet. В качестве реализации данной модели была взята за основу реализация библиотеки Segmentation Models. Энкодер данной модели были заменен с ResNet на архитектуру Efficientnet-b0, предобученная на датасете Imagenet. Данное изменения позволило достичь прироста ключевых метрик на 12%. Также, в данной реализацию были произведены изменения для ее адаптации к размеру и формату данных конкретной задачи. Реализация данных архитектур с учетом изменений представлена в Приложении В.

Обучение моделей происходило с помощью встроенных в фреймворк Keras инструментов с использованием реализованных интерфейсов Dataloader,

выбранного оптимайзера и функций ошибок. Подробнее оптимайзеры и функций ошибок будет описано ниже. Длительность обучения составила 30 эпох. В процессе обучения применялся progressive growing - первые 10 эпох модель обучалась на изображениях размера 256 и на изображениях размера 512 в дальнейшем.

В качестве оптимайзера использовался оптимайзер Adam с параметрами Learning Rate 0.001 и Beta 0.99. Для более быстрой сходимости был применен подход подход снижения learning rate во время обучения. Снижение производилось каждые 5 эпох на 0.0005. В качестве функции ошибки использовались Dice для подсчета ошибки сегментации между предсказанной бинарной маской патологии и размеченными экспертами данными. В качестве ошибки классификации использовалась категориальная кроссэнтропия. Подробный листинг процесса обучения с использованием описанных функции ошибки и оптимайзера представлено в Приложении Г.

Для оценки качества классификации и сегментации были использованы метрики Intesection Over Union для случая сегментации и метрики Accuracy и Fscore для оценки качества классификации. Данные метрики вычисляются по следующим формулам:

$$accuracy(a, X) = \frac{1}{L}\sum_{i=1}^{L}[a(x_i) = y]. \tag{1}$$

$$F = (1 + \beta^2)\frac{precision \times recall}{\beta^2 \times precision + recall}. \tag{2}$$

Процесс подсчета метрики сегментации Intesection Over Union схематично изображен на рисунке 3.

Оценка качества модели была произведена на отложенной выборке и оценена визуально. В ходе оценки были достигнуты показатели метрики **accuracy - 0.94**, **f-score - 0.91** для задачи классификации. В задаче сегментации метрика **IOU** достигла значения в **0.92**.

Заключение. В данной работе были рассмотрены две основные задачи области Medical Imaging: классификация и сегментация. Были рассмотрены современные подходы к решению такого рода задач и представлен собствен-

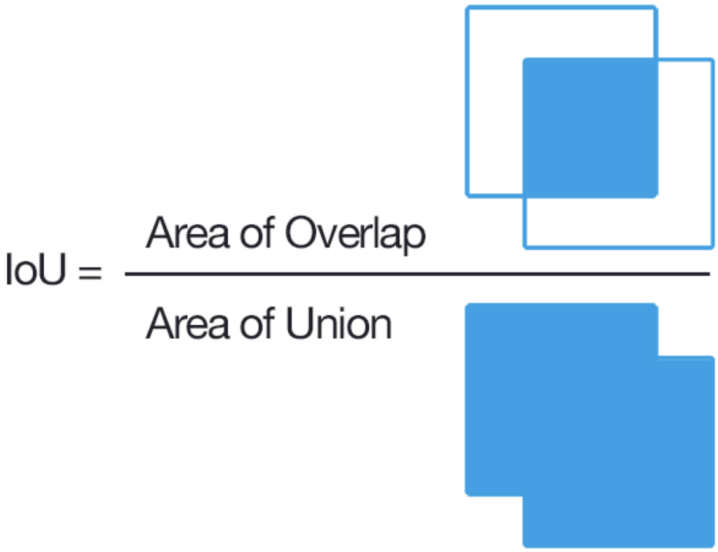

Рисунок 3 – Визуализация процесса вычисления метрики сегментации Intesection Over Union.

ный подход, показавший свою эффективность на публичных наборах данных.

В рамках работы были реализованы такие архитектуры как ResNet101 с применением техники Transfer Learning, а также архитектура Unet для сегментации, использущая сеть EfficientNet как энкодер и проводя обучение с помощью функции потерь Dice. Помимо этого, были использованы такие техники как progressive scaling и augmentation. Данные техники были применены ввиду ограниченного количества публичных данных для обучения и активно применяются при нехватке данных в других задачах. В ходе работы был изучен фреймворк машинного обучения Keras. В работе описана специфика данного фреймворка, а также в приложениях к данной работе представлена реализация всех частей реализованной системы и техник, примененных при обучении моделей.

Результаты численных экспериментов были оценены на отложенной выборке с помощью общепринятых в данных задачах метрик. Полученные оценки качества позволяют говорит о готовности системы к тестированию в медицинских учреждениях, и о стабильности работы моделей на новых данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Semantic segmentation of remote sensing image based on convolutional neural network and mask generation // *Mathematical Problems in Engineering.* — 06 2021. — Vol. 2021. — Pp. 1–13.

2 *Kingma, D. P.* Adam: A method for stochastic optimization / D. P. Kingma, J. Ba // *arXiv preprint arXiv:1412.6980.* — 2014.

3 *Panteleev, A.* Gradient optimization methods in machine learning for the identification of dynamic systems parameters / A. Panteleev, A. Lobanov // *Modelling and Data Analysis.* — 01 2019. — Vol. 09. — Pp. 88–99.

4 Tensorflow API Documentation.

5 *Sudharsan Ravichandiran Sean, S. R. S. Y. W.* Python reinforcement learning: Solve complex real-world problems by mastering reinforcement learning algorithms using openai gym and tensorflow / S. R. S. Y. W. Sudharsan Ravichandiran, Sean // *Paclt Publishing.* — 2019.

6 Keras Documentation, 2020.

7 *Burns Elizabeth A. Kenneth Korn, J. W. I.* Oxford American Hanbook of CLINICAL EXAMINATION AND PRACTICAL SKILLS / J. W. I. Burns Elizabeth A., Kenneth Korn. — Oxford University Press Inc, 2011.

8 *Glied, S.* Technological innovation and inequality in health / S. Glied, A. Lleras-Muney // *Demography.* — 09 2008. — Vol. 45. — Pp. 741–61.

9 *Rashid, T.* Create neural network / T. Rashid. — Williams, 2017.

10 *Eysenbach Gunther Deborah Greenwood, A. S.* Personalized telehealth in the future: A global research agenda / A. S. Eysenbach Gunther, Deborah Greenwood, E. Krupinski // *Journal of Medical Internet Research.* — 2016.

11 *Suzuki S, A. K.* Topological Structural Analysis of Digitized Binary Images by Border Following / A. K. Suzuki S. — Vol. 1. edition. — CVGIP, 1985.

12 *Krizhevsky, A.* Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton // Advances in Neural Information Processing Systems 25 / Ed. by F. Pereira, C. J. C. Burges,

L. Bottou, K. Q. Weinberger. — Curran Associates, Inc., 2012. — Pp. 1097–1105. https://bit.ly/3uTrxWr.

13 A survey on deep learning in medical image analysis / G. Litjens, T. Kooi, B. Ehteshami Bejnordi, A. Setio, F. Ciompi, M. Ghafoorian, J. van der Laak et al. // *Medical Image Analysis*. — 02 2017. — Vol. 42.

ПРИЛОЖЕНИЕ А

```
import numpy as np

prefix = '.'

images_radiopedia = np.load(os.path.join(prefix, 'images_radiopedia.npy')).astype(np.
masks_radiopedia = np.load(os.path.join(prefix, 'masks_radiopedia.npy')).astype(np.int

images_medseg = np.load(os.path.join(prefix, 'images_medseg.npy')).astype(np.float32)
masks_medseg = np.load(os.path.join(prefix, 'masks_medseg.npy')).astype(np.int8)
test_images_medseg = np.load(os.path.join(prefix, 'test_images_medseg.npy')).astype(np

images_mosmed = np.load(os.path.join(prefix, 'images_mosmed.npy')).astype(np.float32)
masks_mosmed = np.load(os.path.join(prefix, 'masks_mosmed.npy')).astype(np.int8)

def preprocess_images(images_arr, mean_std=None):
    images_arr[images_arr > 500] = 500
    images_arr[images_arr < -1500] = -1500
    min_perc, max_perc = np.percentile(images_arr, 5), np.percentile(images_arr, 95)
    images_arr_valid = images_arr[(images_arr > min_perc) & (images_arr < max_perc)]
    mean, std = (images_arr_valid.mean(), images_arr_valid.std()) if mean_std is None
    images_arr = (images_arr - mean) / std
    print(f'mean {mean}, std {std}')
    return images_arr, (mean, std)

images_radiopedia, mean_std = preprocess_images(images_radiopedia)
images_medseg, _ = preprocess_images(images_medseg, mean_std)
images_mosmed, _ = preprocess_images(images_mosmed, mean_std)
test_images_medseg, _ = preprocess_images(test_images_medseg, mean_std)
```

ПРИЛОЖЕНИЕ Б

```
import tensorflow
import albumentations
import cv2

SOURCE_SIZE = 512
TARGET_SIZE = 256

train_augs = albumentations.Compose([
    albumentations.Rotate(limit=360, p=0.9, border_mode=cv2.BORDER_REPLICATE),
    albumentations.RandomSizedCrop((int(SOURCE_SIZE * 0.75), SOURCE_SIZE),
                                   TARGET_SIZE,
                                   TARGET_SIZE,
                                   interpolation=cv2.INTER_NEAREST),
    albumentations.HorizontalFlip(p=0.5),

])

val_augs = albumentations.Compose([
    albumentations.Resize(TARGET_SIZE, TARGET_SIZE, interpolation=cv2.INTER_NEAREST)
])

class Dataset:
    def __init__(
            self,
            images,
            masks,
            augmentations=None
    ):
        self.images = images
        self.masks = masks
        self.augmentations = augmentations

    def __getitem__(self, i):
        image = self.images[i]
        mask = self.masks[i]

        if self.augmentations:
```

```
38              sample = self.augmentations(image=image, mask=mask)
39              image, mask = sample['image'], sample['mask']
40          return image, mask
41
42      def __len__(self):
43          return len(self.images)
44
45
46  class Dataloder(tensorflow.keras.utils.Sequence):
47      """Load data from dataset and form batches
48
49      Args:
50          dataset: instance of Dataset class for image loading and preprocessing.
51          batch_size: Integet number of images in batch.
52          shuffle: Boolean, if `True` shuffle image indexes each epoch.
53      """
54
55      def __init__(self, dataset, batch_size=1, shuffle=False):
56          self.dataset = dataset
57          self.batch_size = batch_size
58          self.shuffle = shuffle
59          self.indexes = np.arange(len(dataset))
60
61          self.on_epoch_end()
62
63      def __getitem__(self, i):
64
65          # collect batch data
66          start = i * self.batch_size
67          stop = (i + 1) * self.batch_size
68          images = []
69          masks = []
70          for j in range(start, stop):
71              image, mask = self.dataset[self.indexes[j]]
72              images.append(image)
73              masks.append(mask)
74
75          images = np.stack(images, axis=0)
76          masks = np.stack(masks, axis=0).astype(np.float32)
77
```

```
78         return (images, masks)
79
80     def __len__(self):
81         """Denotes the number of batches per epoch"""
82         return len(self.indexes) // self.batch_size
83
84     def on_epoch_end(self):
85         """Callback function to shuffle indexes each epoch"""
86         if self.shuffle:
87             self.indexes = np.random.permutation(self.indexes)
88
89 train_dataset = Dataset(train_images, train_masks, train_augs)
90 val_dataset = Dataset(val_images, val_masks, val_augs)
91
92 train_dataloader = Dataloder(train_dataset, batch_size=batch_size, shuffle=True)
93 val_dataloader = Dataloder(val_dataset, batch_size=batch_size, shuffle=False)
```

ПРИЛОЖЕНИЕ В

```
from segmentation_models import Unet
import segmentation_models as sm

classification_model = resnet101(weights="imagenet")

segmentation_model = Unet(backbone_name='efficientnetb0',
                  encoder_weights='imagenet',
                  classes=8,
                  activation='softmax')
```

ПРИЛОЖЕНИЕ Г

```
from tensorflow.keras.layers import Input, Conv2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint

model = Sequential([Input(shape=(TARGET_SIZE, TARGET_SIZE, 1)),
                    Conv2D(3, (1, 1)),  # map N channels data to 3 channels
                    base_model])

model.compile(Adam(learning_rate=0.001, amsgrad=True),
              loss=sm.losses.categorical_crossentropy,
              metrics=[accuracy, fscore])

checkpoint_callback = ModelCheckpoint('best_model',
                                      monitor='fscore',
                                      mode='max',
                                      save_best_only=True)

model.fit(
    train_dataloader,
    steps_per_epoch=len(train_dataloader) * 6,
    epochs=30,
    validation_data=val_dataloader,
    validation_steps=len(val_dataloader),
    callbacks=[checkpoint_callback],
    workers=4)
```