# **Введение**

Компьютерное зрение входит в число актуальнейших направлений в сфере информационных технологий. Данное направление представляет собой комплекс методов, реализующих обработку двухмерных или трехмерных объектов, для обнаружения, отслеживания и классификации объектов. [1]

Системы компьютерного зрения находят широкое применение в задачах охраны объектов, борьбы с терроризмом, мониторинга дорожного движения, в контроле качества продукции, в робототехнике, авиастроении, промышленности и во многих других задачах. Все эти задачи затрагивают множество сторон жизни человека и являются чрезвычайно актуальными.

Искусственный интеллект становится неотъемлемым атрибутом современного мира, управляемого технологиями и данными [2]

Одним из ответвлений машинного зрения и машинного обучения является нейронные сети, позволяющие решить проблему контроля качества изделий на производстве. Это решение позволяет автоматически обнаруживать множество дефектов.

Цель данной работы – предложить один из способов реализации системы распознавания дефектов поверхностей металлов на изображениях с использованием нейронных сетей.

# **Машинное обучение и контроль качества**

1. **Дефекты поверхностей. Дефекты металлов**

При изготовлении и обработке деталей машин и приборов, а также при производстве различных конструкций в металле могут возникать дефекты, т. е. отдельные несоответствия продукции нормативным требованиям. [3] Дефекты снижают эксплуатационную способность наплавленного металла, а в некоторых случаях могут вызывать аварийный выход детали из строя.

Дефекты металлов на изображениях – не имеют фиксированной площади и обладают различной формой. Интенсивность дефекта может быть темнее или ярче фона. Все это усложняет процессы определения и классификации дефектов. Приходится обращаться к специальным процессам обнаружения и классификации дефектов – дефектоскопии [4]. Наиболее часто дефектоскопия применяется в металлургии для обнаружения дефектов поверхностей металлов в процессе неразрушающего контроля (НК). В настоящее время существуют в основном две черты для снимков поверхностей металлов с дефектами. Первая – это повторяющийся однородный фон, второе – наличие пикселей с неравномерной яркостью в районе дефекта.

* 1. **Неразрушающий контроль**

Неразрушающий контроль (НК) - контроль свойств и параметров объекта, при котором не должна быть нарушена его пригодность к использованию и эксплуатации. Выделяют несколько видов НК, одним из которых является визуальный и измерительный контроль (ВИК). Внешним осмотром обычно проверяют качество подготовки и сборки заготовок под сварку [5], качество выполнения швов в процессе сварки и качество готовых сварных соединений. Внешним осмотром выявляют, прежде всего, дефекты швов в виде трещин, подрезов, пор, свищей, прожогов, наплывов, непроваров в нижней части швов, а также влияние коррозии на металл. Типичные дефекты металлов показаны на рисунке 1

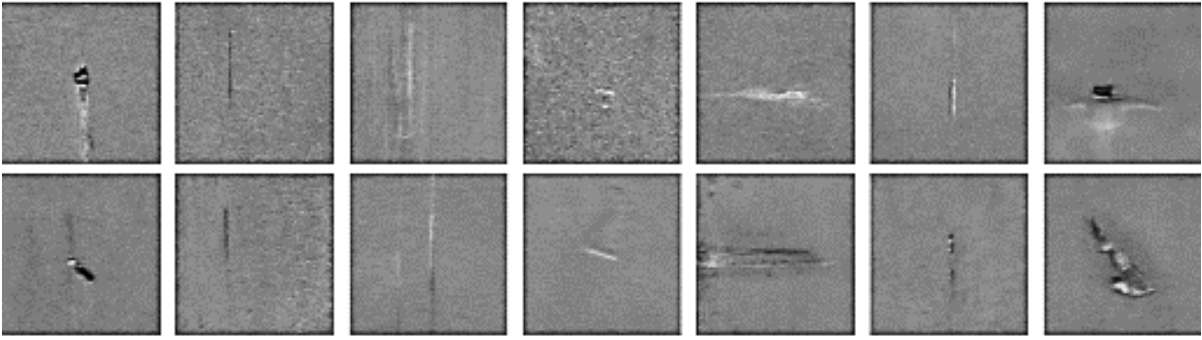


Рисунок 1 - Типичные дефекты металлов

Целью неразрушающего контроля качества материалов и изделий является не только обнаружение дефектов, но и их классификация вследствие различного уровня потенциальной опасности дефектов разного типа [6]. Например, для дефектов типа трещин устанавливается жесткий критерий браковки изделий, а для округлых дефектов (поры, шлаковые включения и т.п.) ― более мягкие нормы браковки [3, 4]. Классификация обнаруженных дефектов существенно повышает качество контроля, что нужно для оптимизации управления процессами производства и улучшения качества продукции. В этом может помочь компьютерное зрение.

# **1.3 Компьютерное зрение**

Термин «компьютерное зрение» имеет много синонимов: машинное зрение, распознавание зрительных объектов, анализ изображений, фотограмметрия и другие. Но смысл, скрывающийся за всеми этими определениями один – это попытка научить компьютер видеть мир глазами человека, выполнять различные действия так же, как делал бы это человек, тем самым подменяя или полностью исключая последнего. К сожалению, компьютерному зрению еще далеко до уровня распознавания человека, точные принципы анализа визуальной информации которого до конца не изучены. Решение задачи моделирования деятельности человеческого глаза и мозга, безусловно, ответило бы на большинство вопросов в области компьютерного зрения. Однако даже предварительные оценки показывают, что решение этой задачи в реальном времени потребует огромных вычислительных затрат.

Повышение уровня надежности и увеличение ресурса машин и других объектов техники возможно только при условии выпуска продукции высокого качества во всех отраслях машиностроения [8]. Это требует непрерывного совершенствования технологии производства и методов контроля качества. Одной из наиболее распространенных тенденций является внедрение полного неразрушающего контроль продукции на отдельных этапах производства. Зачастую, человеческий фактор играет далеко не последнюю роль. Это сказывается на качестве выпускаемой продукции. Потеря должного внимания, концентрации, может привести к пропуску дефектов и отправке некачественных изделий в дальнейшее использование. Конечно же, это далеко не полный перечень проблем, которые могут быть решены путем применения камер. Идеальным решением задач компьютерного зрения является создание универсальной самообучающейся системы, которая бы «росла» и «зрела» так же, как это с рождения происходит с любым человеком.

Перспективность таких систем очевидна, ведь источником информации для них являются электронные глаза, способные днем и ночью следить за людьми, дорогой, производством и другими интересующими объектами.

**1.4 Машинное обучение**

Нейронные сети являются одной из разновидностей алгоритмов машинного обучения (machine learning). Машинное обучение – это, в свою очередь, один из подразделов искусственного интеллекта [8]. Основной особенностью алгоритмов машинного обучения является их способность обучаться в процессе работы. Примером может служить алгоритм построения решающих деревьев, который, изначально не имея никакой информации о том, что представляют собой данные, для чего они нужны и какие в них есть закономерности, а только имея некоторый входной набор объектов и значения некоторых признаков для каждого из них вместе с меткой класса, в процессе работы сам выявляет нужные закономерности (обучается), а после обучения способен предсказывать класс для новых объектов, которые он не видел ранее.

В данной работе будет рассматриваться задача по классификации для различных видов дефектов поверхностей. Постановка задачи в случае задачи классификации такова: имея некоторый набор объектов, каждый из которых относится к одному из нескольких классов, необходимо определить, к какому из этих классов относится новый объект [9]. В нашем случае существует несколько видов дефектов – классов, и задача алгоритма – выявить и распознать дефект, то есть отнести его к одному из известных классов. Наиболее подходящим методом классификации будут нейронные сети.

**1.5 Искусственные нейронные сети**

Глубокое обучение — это раздел искусственного интеллекта, цель которого научить компьютеры обучаться с помощью нейронных сетей — технологии, созданной по образу и подобию человеческого мозга [10]

В основе искусственных многослойных нейронных сетей лежат персептроны (или, другими словами, искусственные нейроны).

Он может быть рассмотрен как простейший тип сетей прямого распространения, а также как бинарный классификатор, который отображает свой входной вектор на бинарный выход:

Персептрон схематично изображен на рисунке 1.4.

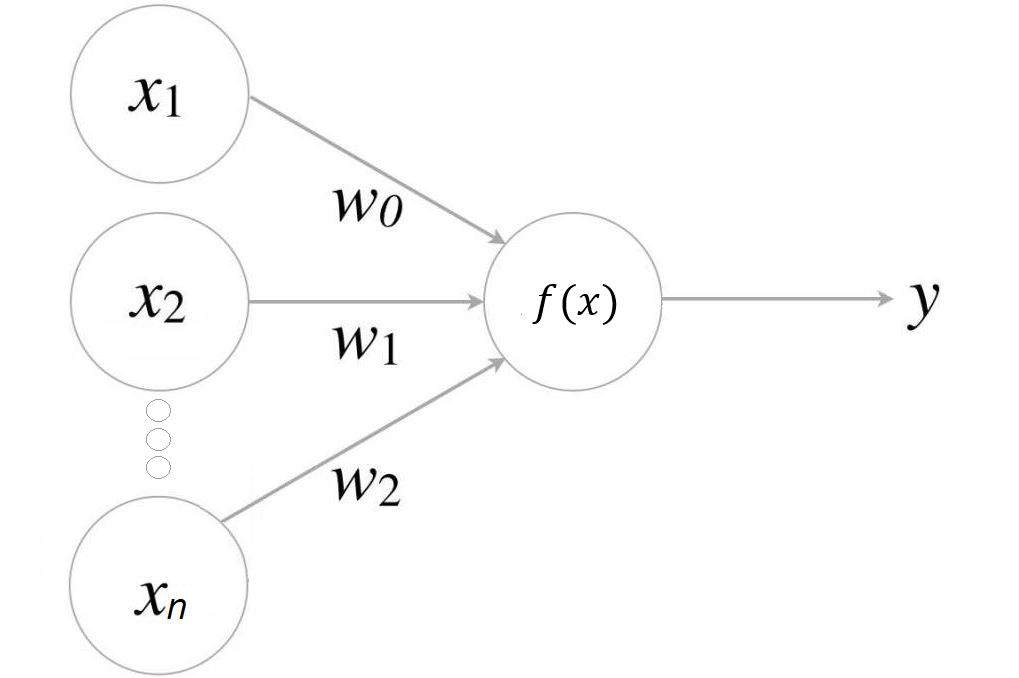


Рисунок 1.4 – Персептрон

Для начала нужно научить этот нейрон, как он должен реагировать на определенный входной вектор (паттерн). Этот процесс называется обучением (training). Когда персептрон обучается, его веса w и смещение b изменяются. После фазы обучения мы фиксируем веса и смещение, на этом обучение завершается. После обучения мы можем подать на вход персептрону любой входной вектор, даже если персептрон его никогда раньше не видел. Независимо от того, видел ли он этот входной паттерн или нет, персептрон вернет определенное значение. Для того, чтобы обучить персептрон, нам нужно подготовить обучающую выборку (training set). Обучающая выборка – это множество, состоящее из входных векторов признаков объектов и вектора ожидаемых ответов сети

### **1.6 Контролируемое обучение**

Контролируемое обучение (обучение с учителем, supervised learning) подразумевает использование помеченных наборов данных, содержащих входные данные и ожидаемые выходные результаты. Когда вы обучаете нейронную сеть с помощью контролируемого обучения, вы подаете как входные данные, так и ожидаемые выходные результаты.

Если результат, генерируемый нейронной сетью, неверен, она скорректирует свои вычисления. Это итерационный процесс, оканчивающийся тогда, когда сеть перестает совершать ошибки.

Примером задачи с контролируемым обучением является предсказание погоды. Нейросеть учится делать прогноз погоды с использованием исторических данных. Обучающие данные включают в себя входные данные (давление, влажность, скорость ветра) и выходные результаты (температура). [11]

Каждый нейрон имеет функцию активации. Ее смысл трудно понять без привлечения математических рассуждений. Одной из ее целей является «стандартизация» данных на выходе из нейрона.

После того, как набор входных данных прошел через все слои нейронной сети, функция активации возвращает выходные результаты через выходной уровень.

1. **Программа по распознаванию дефектов металлов**

**2.1 Выбор средств разработки**

При разработке системы было принято решение использовать следующие инструменты и технологии:

* Язык программирования Python
* Библиотека для машинного обучения tensorflow
* Библиотека для математических вычислений NumPy
* Библиотека для обработки изображений Matplotlib
* Библиотека для обработки и анализа структурированных данных Pandas
* библиотека для создания статистических графиковSeaborn

**2.2 Алгоритм решения задачи**

- Предварительная обработка изображения. Полученное от пользователя изображение преобразуется в полутоновое, удаляется шум с изображения. Изображение подготавливается для поиска дефектов на нем.

- Сегментация изображения, поиск предположительных дефектов. Обнаружение предположительных местоположений дефектов на изображении с целью дальнейшего их распознавания нейронной сетью [12].

- Распознавание дефекта. Использование обученной сверточной нейронной сети для распознавания дефектов. Алгоритм представлен на рисунке 2.1. Для упрощения работы нейросети и улучшения качества её работы, принято решение отправлять на вход изображения одного размера (100 х 100). Для этого нужно сегментировать исходное изображение таким образом, чтобы дефект максимально полно занимал часть изображение такой размерности. Другими словами, нужно найти прямоугольник, в который вписывается дефект. Эта процедура называется детектированием. Детектированный дефект при необходимости масштабируется, сегментируется и подается на вход нейронной сети.

Детектирование предположительного местоположения дефекта происходит в несколько этапов. Предобработка основана на предположении о кросскоррелированности подокон исходного изображения и шаблонных дефектов. Поэтому сначала производится свертка исходного изображения с шаблонным дефектом, а потом на результате свертки ищутся центры локальных максимумов, которые будут соответствовать предположительным местоположениям дефектов. Алгоритм детекции изображен. на рисунке 2

Начало

Получение изображения от пользователя

Детекция дефектов

Распознавание

Конец

Рисунок 2 – Алгоритм решения задачи распознавания

**2.3 Описание базы изображений**

Для тестирования разработанных алгоритмов распознавания была использована сгенерированная база данных, содержащая изображения дефектов и поверхностей без дефектов, а также соответствующие им метки. Для обучения классификаторов было использовано 1000 изображений, из которых 600 изображений использовались в качестве обучающей выборки, 400 изображений использовались в качестве тестовой выборки. Размер каждого изображения составляет 100×100 пикселя. Примеры дефекта изображены на рисунке 3.



Рисунок 3 – примеры дефекта (ржавчина)

База данных изображений состоит из двух наборов данных, для обучения и для тестирования.

Для правильной работы нейронной сети также необходимы данные для проверки, для этого набор данных для обучения делим на 2 части при помощи команды:

validation\_split=0.1,

Таким образом 10% набора данных для обучения будет использоваться в качестве проверочного набора данных.

**2.4 Работа нейронной сети**

Работа свёрточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное [13].

Первый скрытый слой — это сверточный слой. Этот слой имеет 16 карт функций, размер которых равен 5 × 5 и функции активации relu.

Затем мы определяем слой пулинга (подвыборки) с размером пула 2 × 2, который дает максимальные значения.

Следующий уровень — это уровень регуляризации Dropout. Он настроен на случайное исключение нейронов в слое, чтобы уменьшить переобучение. 4

Далее - слой, который преобразует данные двумерной матрицы в вектор, называемый. Он позволяет обрабатывать выходные данные стандартными полносвязными слоями.

Затем полносвязный слой с 128 нейронами и функцией активации relu Обучения сверточной нейронной сети занимает больше времени чем обучение простого персептрона. Однако ошибка значительно снижается по сравнению с персепроном. [14] Принцип работы сверточной нейронной сети изображен на рисунке 4

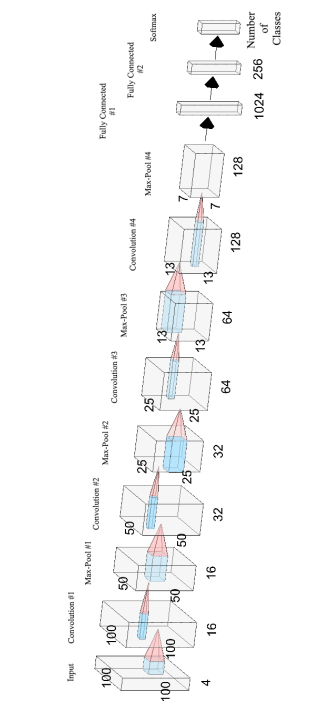


Рисунок 4 Сверточная нейронная сеть [15]

Особенностью разработанной нейронной сети является то, что в наборе данных для обучения находятся как сами изображения, так и правильный ответ.

Весь код программы описан в приложении А.

В результате 10 эпох обучения были получены следующие результаты:

Epoch 1/10

2/2 - 14s - loss: 33.7940 - accuracy: 0.2166 - val\_loss: 13.2342 - val\_accuracy: 0.2927 - 14s/epoch - 7s/step

Epoch 2/10

2/2 - 0s - loss: 8.7155 - accuracy: 0.4332 - val\_loss: 0.7221 - val\_accuracy: 0.7073 - 315ms/epoch - 158ms/step

Epoch 3/10

2/2 - 0s - loss: 0.8584 - accuracy: 0.7406 - val\_loss: 0.5964 - val\_accuracy: 0.6829 - 309ms/epoch - 155ms/step

Epoch 4/10

2/2 - 0s - loss: 0.6609 - accuracy: 0.7193 - val\_loss: 0.3358 - val\_accuracy: 0.7073 - 291ms/epoch - 145ms/step

Epoch 5/10

2/2 - 0s - loss: 0.5572 - accuracy: 0.7460 - val\_loss: 0.2758 - val\_accuracy: 0.9756 - 282ms/epoch - 141ms/step

Epoch 6/10

2/2 - 0s - loss: 0.4185 - accuracy: 0.8422 - val\_loss: 0.2584 - val\_accuracy: 0.9756 - 272ms/epoch - 136ms/step

Epoch 7/10

2/2 - 0s - loss: 0.3639 - accuracy: 0.8529 - val\_loss: 0.2123 - val\_accuracy: 0.9512 - 265ms/epoch - 133ms/step

Epoch 8/10

2/2 - 0s - loss: 0.3197 - accuracy: 0.8529 - val\_loss: 0.4513 - val\_accuracy: 0.7073 - 266ms/epoch - 133ms/step

Epoch 9/10

2/2 - 0s - loss: 0.5074 - accuracy: 0.7914 - val\_loss: 0.2366 - val\_accuracy: 0.9268 - 259ms/epoch - 130ms/step

Epoch 10/10

2/2 - 0s - loss: 0.4539 - accuracy: 0.8021 - val\_loss: 0.2127 - val\_accuracy: 0.9756 - 267ms/epoch - 133ms/step

Доля правильных ответов на последней эпохе на проверочном наборе данных почти 98%

Оценим качество обучения модели на тестовых данных:

scores = model.evaluate(test\_dataset, verbose=1)

print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1] \* 100, 4))

Доля верных ответов на тестовых данных, в процентах: 90.0

Графики качества обучения в процессе эволюции представлены на рисунке 5

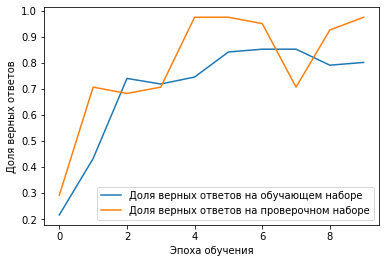
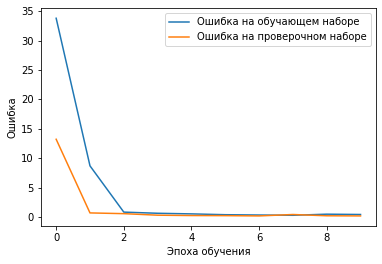
 

Рисунок 5 - Графики качества обучения

**Выводы**

В результате была получена система опознавания дефектов металла с применением глубокого обучения с хорошей точностью определения.

Программа может быть использована во многих отраслях промышленного производства.

Если говорить о дальнейшем развитии программы - её можно модифицировать для использования на реальных деталях в реальном времени. Также систему можно усовершенствовать обучением программы классифицировать виды дефектов поверхностей металлов.

# **Список используемой литературы**

1. Л. Шапиро, Дж. Стокман. Компьютерное зрение/Computer Vision. — М.: Лаборатория знаний, 2020. — 13 с.
2. Джоши, Пратик. Д42 Искусственный интеллект с примерами на Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2019. - 448 с. - Парал. тит. англ. ISBN 978-5-907114-41-8 (рус.)
3. Галимуллин Д.З, Денисов А.И, Иларионов С.Ю.Оценка эффективности обучения сверточных нейронных сетей на синтетических данных для распознавания символов государственных регистрационных знаков транспортных средств// Постулат 2017 с. 128
4. Дефекты металла: Учебное пособие по дисциплине «Материаловедение и ТКМ» для механических специальностей вузов / В.П. Расщупкин, М.С. Корытов. – Омск: Изд-во СибАДИ, 2006. – 37 с.
5. Александров С. Д., Забурненко Е. В. Акустическая дефектоскопия. Виды дефектоскопии. // статья, студенческий научный форум 2015
6. Приборы для неразрушающего контроля материалов и изделий. В 2-х книгах. Кн. 1. / Под ред. В.В. Клюева. – М.: Машиностроение, 1986. – 2-е изд., перераб. и доп.– 488 с.
7. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. ДМК Пресс, 2015. 402 с.
8. 2 Бархатов, В.А. Распознавание дефектов с помощью искусственной нейронной сети специального типа. // Дефектоскопия – 2006. – № 2. – с. 28―39.
9. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python, Москва, 2017. 393 с.
10. Бархатов, В.А. Обнаружение сигналов и их классификация с помощью распознавания образов. // Дефектоскопия – 2006. – № 4 – с. 14―27
11. Поляков А.В. О применении сверточных нейронных сетей при решении задачи неразрушающего контроля изделий //Eспехи современной науки – 2017 с. 204-207
12. Траск Эндрю Т65 Грокаем глубокое обучение. — СПб.: Питер, 2019. — 352 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-1334-7
13. Белых, А.А. Разработка системы политематической классификации на базе алгоритмов машинного обучения : А.А. Белых .— : [Б.и.], 2019 .— 86 с
14. Гафаров Ф.М Г12 Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.
15. Horea Mures¸an Fruit recognition from images using deep learning// Acta Univ. Sapientiae, Informatica, 10, 1 (2018) 26–42

**Приложение А – рабочая программа**

import os

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

from tensorflow.keras import utils

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

import matplotlib.pyplot as plt

from google.colab import files

%matplotlib inline

**Загрузка данных**

!wget !wget https://github.com/SichYeah/Metall\_Dataset\_test/archive/master.zip -O master.zip

!unzip master.zip

**Создание Tensorflow Dataset'ов**

batch\_size=256

image\_size=(100, 100)

**Набор данных для обучения**

train\_dataset = image\_dataset\_from\_directory('Metall\_Dataset\_test-main/Training',

                                             subset='training',

                                             seed=42,

                                             validation\_split=0.1,

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

**Проверочный набор данных**

validation\_dataset = image\_dataset\_from\_directory('Metall\_Dataset\_test-main/Training',

                                             subset='validation',

                                             seed=42,

                                             validation\_split=0.1,

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

**Названия классов в наборах данных.**

class\_names = train\_dataset.class\_names

class\_names

**Примеры изображений**

plt.figure(figsize=(8, 8))

for images, labels in train\_dataset.take(1):

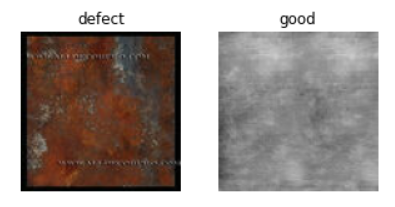
  for i in range(2):

    ax = plt.subplot(3, 3, i + 1)

    plt.imshow(images[i].numpy().astype("uint8"))

    plt.title(class\_names[labels[i]])

    plt.axis("off")



**Набор данных для тестирования**

test\_dataset = image\_dataset\_from\_directory('Metall\_Dataset\_test-main/Test',

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

**Настройки производительности TensorFlow DataSet'ов**

UTOTUNE = tf.data.experimental.AUTOTUNE

train\_dataset = train\_dataset.prefetch(buffer\_size=AUTOTUNE)

validation\_dataset = validation\_dataset.prefetch(buffer\_size=AUTOTUNE)

test\_dataset = test\_dataset.prefetch(buffer\_size=AUTOTUNE)

**Создание нейронной сети**

# Создаем последовательную модель

model = Sequential()

# Сверточный слой

model.add(Conv2D(16, (5, 5), padding='same',

                 input\_shape=(100, 100, 3), activation='relu'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(32, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(128, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Полносвязная часть нейронной сети для классификации

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.2))

# Выходной слой

model.add(Dense(100, activation='softmax'))

**Компилирование модели**

model.compile(loss='sparse\_categorical\_crossentropy',

              optimizer="adam",

              metrics=['accuracy'])

**Обучение нейронной сети**

history = model.fit(train\_dataset,

                    validation\_data=validation\_dataset,

                    epochs=10,

                    verbose=2)

# Оцениваем качество обучения модели на тестовых данных

scores = model.evaluate(test\_dataset, verbose=1)

print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1] \* 100, 4))

Доля верных ответов на тестовых данных, в процентах: 90.0

plt.plot(history.history['accuracy'],

         label='Доля верных ответов на обучающем наборе')

plt.plot(history.history['val\_accuracy'],

         label='Доля верных ответов на проверочном наборе')

plt.xlabel('Эпоха обучения')

plt.ylabel('Доля верных ответов')

plt.legend()

plt.show()

plt.plot(history.history['loss'],

         label='Ошибка на обучающем наборе')

plt.plot(history.history['val\_loss'],

         label='Ошибка на проверочном наборе')

plt.xlabel('Эпоха обучения')

plt.ylabel('Ошибка')

plt.legend()

plt.show()

model.save("metal\_model.h5")

files.download("metal\_model.h5")