Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа теоретической механики

Работа допущена к защите
директор ВШ ТМ, д. ф.-м. н., чл.-корр. РАН
А. М. Кривцов
«___»_____2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА
РАСПОЗНАВАНИЕ ВИДА ДВИГАТЕЛЬНОЙ АКТИВНОСТИ ЧЕЛОВЕКА С ПОМОЩЬЮ АКСЕЛЕРОМЕТРА

По направлению 01.04.03 «Механика и математическое моделирование»
по образовательной программе
01.04.03_03 Механика и цифровое производство

Выполнил
студент гр. 3640103/80301 А.Ф. Вакильева

Руководитель
Доц. ВШ ТМ, к.ф-м.н. М.Б. Бабенков

Санкт-Петербург
2020

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
Институт прикладной математики и механики
Высшая школа теоретической механики

УТВЕРЖДАЮ

Директор высшей школы

А.М. Кривцов

« » 20 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Вакильевой Адели Фиратовне 3640103/80301

1. Тема работы: Распознавание вида двигательной активности человека с помощью акселерометра

2. Срок сдачи студентом законченной работы: 8 июня 2020 г.

3. Исходные данные по работе: справочная литература, актуальные публикации по теме исследования.

4. Содержание работы (перечень подлежащих разработке вопросов): Исследовать принцип работы свёрточной нейронной сети. Подготовить данные с акселерометра к работе с нейронной сетью (балансировка, стандартизация). Задача классификации и разбиение выборки на обучающую и тестирующую. Написание алгоритма свёрточной нейронной сети, последующее её обучение на обучающей выборке. Оценка работы алгоритма.

5. Перечень графического материала (с указанием обязательных чертежей): не предусмотрены

6. Консультанты по работе: не предусмотрены

7. Дата выдачи задания: 22 января 2020 г.

Руководитель ВКР М.Б. Бабенков

Задание принял к исполнению 22 января 2020 г.

Студент А.Ф. Вакильева

РЕФЕРАТ

На стр. 34, 21 рисунок, 0 таблиц, 0 приложений, 11 литературных источника.

РАСПОЗНАВАНИЕ АКТИВНОСТИ, СВЁРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, ДАТЧИК, ГЛУБОКОЕ ОБУЧЕНИЕ, ЗАДАЧА КЛАССИФИКАЦИИ

В данной работе рассматривается решение проблемы распознавания двигательной активности человека с помощью методов глубокого обучения. Для этой проблемы предложен алгоритм свёрточной нейронной сети. Работа выполнена на языке Python, использованы библиотеки Pandas, NumPy, SciPy, TensorFlow, Keras, Scikit-learn, Seaborn.

THE ABSTRACT

34 pages, 21 pictures, 0 tables, 0 application, 11 references.

HUMAN ACTIVITY RECOGNITION, CONVOLUTED NEURAL NETWORK, SENSOR, DEEP LEARNING, STATISTICAL CLASSIFICATION

In this paper, we consider the solution to the problem of human activity recognition using deep learning methods. The convolutional neural network algorithm is proposed for this problem. The work was done in Python, the libraries Pandas, NumPy, SciPy, TensorFlow, Keras, Scikit-learn, Seaborn were used.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Регулярная физическая активность связана с пользой для здоровья, от поддержания или повышения физической подготовленности до снижения риска различных заболеваний. Существуют рекомендации о том, сколько и какой тип физической активности должны выполнять люди [6].

Основной целью распознавания человеческой активности является обеспечение правильного качества и количества физической активности путем постоянного мониторинга. Недавний прогресс в носимых технологиях делает небольшие, легкие, недорогие, мультимодальные и точные сенсорные блоки доступными. Поэтому ненавязчивый и мобильный мониторинг активности стал доступным.

Распознавание основных физических нагрузок (таких, как ходьба или катание на велосипеде) и позы (например, сидеть или стоять) хорошо изучены, тем самым доказывая, что хорошая производительность может быть достигнута даже при наличии только одного акселерометра и простых методов машинного обучения [9].

Тем не менее, многие практические проблемы остаются, в том числе: смещение датчика [2], персонализация, надежность в реальных сценариях, слияние различных модальностей сенсоров и сокращение необходимого количества помеченных обучающих данных.

Последнее очень актуально в распознавании человеческой активности, так как маркировка не может быть выполнена на необработанных данных (как, например, в классификации изображений). Маркировка требует дополнительных усилий (например, запись действий с синхронизированным видео) и, следовательно, очень много времени. Поэтому оценка новых алгоритмов (таких, как методы глубокого обучения) в отношении количества необходимых данных представляет высокую актуальность.

Большинство используемых в настоящее время алгоритмов распознавания человеческой активности основано на стандартных

классификаторах машинного обучения, таких как (усиленные) деревья решений или машины опорных векторов, в сочетании с шаблонами или функциями времени.

В других областях исследований, таких как распознавание речи и классификация изображений, эти методы машинного обучения были значительно превзойдены методами глубокого обучения.

В последнее время глубокое обучение развилось в целый ряд способов обучения модели, направленных на моделирование абстракций высокого уровня в данных. В глубоком обучении сложная архитектура с несколькими слоями построен для автоматизации извлечения признаков.

В частности, каждый уровень в глубокой архитектуре выполняет нелинейное преобразование на выходах предыдущего уровня, так что в моделях глубокого обучения данные представляются иерархией функций от низкого уровня до высокого уровня.

Некоторыми из хорошо известных моделей глубокого обучения являются свёрточная нейронная сеть, глубокие сети доверия и автоэнкодеры.

Хотя модели глубокого обучения достигают замечательных результатов в компьютерном зрении, обработке естественного языка и распознавание речи, они не были достаточно использованы в полевых условиях

В этой статье мы решаем проблему распознавания человеческой активности, используя конкретную модель глубокого обучения – свёрточную нейронную сеть. Ключевым атрибутом свёрточной нейронной сети является альтернативное проведение различных блоков обработки (например, свертка, объединение, сигмоидальное / гиперболическое касательное сжатие, и нормализация). Такое разнообразие блоков обработки может дать эффективное представление локальной значимости сигналов.

Глубокая архитектура свёрточной нейронной сети предполагает несколько слоев, эти блоки обработки должны быть организованы так, чтобы

эта модель глубокого обучения могла характеризовать значимость сигналов в различных масштабах. Следовательно, признаки, извлеченные свёрточной нейронной сетью зависят от задачи. Кроме того, эти признаки также обладают большей дискриминационной силой, поскольку свёрточная нейронная сеть может быть обучена с валидацией выходных значений. Все эти преимущества свёрточной нейронной сети будут более подробно рассмотрены в следующих разделах.

Таким образом, *целью* данной работы является создание нейросети методом глубокого обучения, определяющей различные виды двигательной активности человека.

Объектом исследования является распознавание двигательной активности человека.

Задачи, которые нужно выполнить для достижения данной цели:

1. Исследовать принцип работы свёрточной нейронной сети.
2. Подготовить данные с акселерометра к работе с нейронной сетью (балансировка, стандартизация, Label Encoding).
3. Задача классификации и разбиение выборки на обучающую и тестирующую.
4. Написание алгоритма свёрточной нейронной сети, последующее её обучение на обучающей выборке.
5. Оценка работы алгоритма.

ГЛАВА 1. НЕЙРОННЫЕ СЕТИ.

1.1. Введение в нейронные сети.

Искусственная нейронная сеть (ИНС) - это математическая модель, которая в значительной степени вдохновлена работой биологических нейронных сетей, т.е. сетей нервных клеток живого организма (таких, как мозг человека). ИНС в основном состоит из большого количества взаимосвязанных вычислительных узлов, называемых нейронами, в которых работа распределена между собой для коллективного обучения на входных данных, чтобы в дальнейшем оптимизировать конечный выход.

Базовая структура ИНС может быть смоделирована, как показано на рисунке 1.1. На вход поступают входные данные в форме многомерного вектора на входной слой, из которого они будут распределены по скрытым слоям. Затем скрытые слои будут принимать решения из предыдущего слоя и оценивать, как случайное изменение весов и значений внутри них ухудшает или улучшает конечный результат, что и называется процессом обучения. Наличие нескольких скрытых слоев, наложенных друг на друга, обычно называется глубоким обучением.

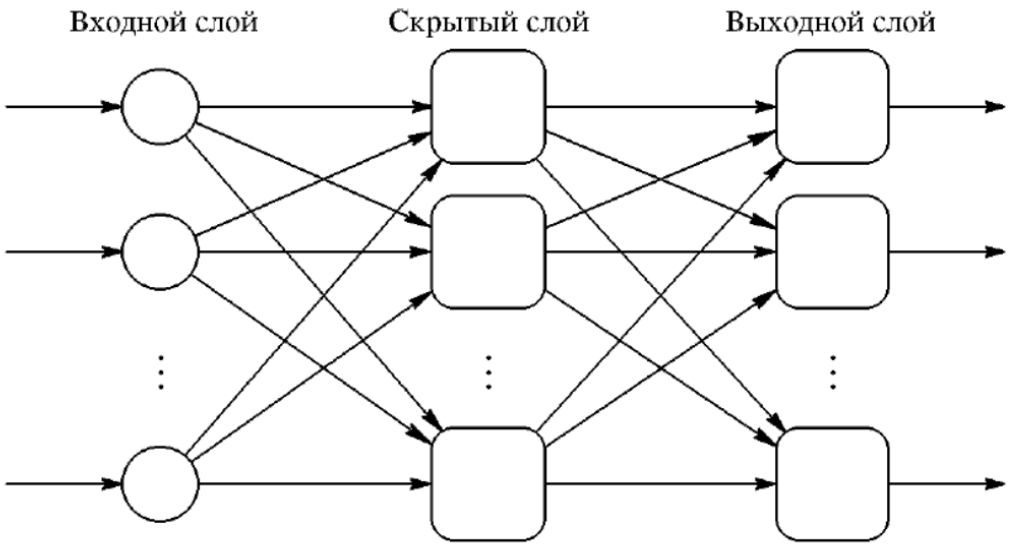

Рис. 1.1. Простая трехслойная нейронная сеть с прямой связью, состоящая из входного слоя, скрытого слоя и выходного слоя.

Эта структура является основой ряда общих архитектур ИНС, включая, но не ограничиваясь, нейронные сети с прямой связью, ограниченные машины Больцмана и рекуррентные нейронные сети (РНС).

Каждый нейрон получает от предыдущих значения, преобразует их и присваивает себе это новое преобразованное значение. Последующие нейроны следующего слоя получат это значения уже для своих вычислений. При этой структуре нейроны собирают значения предыдущего слоя вместе с весами, суммируя их и применяя нелинейные функции активации. Таких функций активации много, но самые популярные это:

- Sigmoid

$$S(x) = \frac{1}{1+e^{-x}} \tag{1.1}$$

- Tanh

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{1.2}$$

- ReLU.

$$ReLU(x) = max(0, x) \tag{1.3}$$

Нейроны не связаны внутри одного слоя, каждый нейрон сообщается и обменивается значениями только с соседними слоями.

Двумя ключевыми методами обучения в задачах машинного обучения являются контролируемое (обучение с учителем) и неконтролируемое (обучение без учителя) обучение.

Контролируемое обучение – это обучение через предварительно помеченные входные данные, которые действуют как цели. Для каждого обучающего примера будет набор входных значений (векторов) и одно или несколько связанных назначенных выходных значений. Цель этой формы обучения состоит в том, чтобы уменьшить общую классификационную погрешность моделей путем правильного расчета выходного значения.

Обучение без учителя отличается тем, что в обучающий набор не включены ярлыки. Успех обычно определяется тем, способна ли сеть уменьшить или увеличить связанную функцию затрат. Однако важно отметить, что большинство ориентированных на изображение задач по распознаванию образов обычно зависят от классификации с использованием контролируемого обучения.

1.2. Свёрточные нейронные сети.

Сверточные нейросети были разработаны в конце 1980-х — начале 1990-х годов франко-американским ученым Яном Лекуном и его коллегами из Bell Labs. Даже тогда их сверточные нейросети показывали превосходящие другие методы результаты, но могли быть использованы только для очень маленьких изображений, около 28*28 пикселей. Для таких изображений нейросеть, могла разобрать, какой рукописный символ какие изображения содержат.

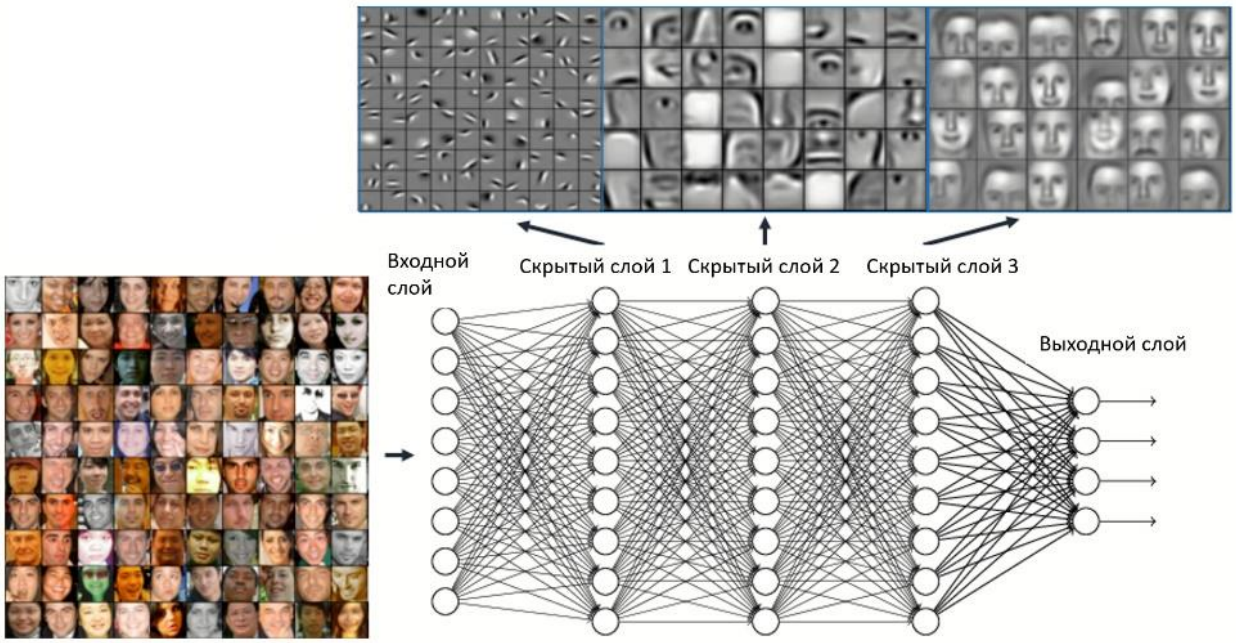

Рис. 1.2. Пример работы свёрточной нейронной сети.

Свёрточные нейронные сети (СНС) аналогичны традиционным ИНС тем, что состоят из нейронов, которые самооптимизируются посредством

обучения. Каждый нейрон будет по-прежнему получать входные данные и выполнять операцию (например, скалярное произведение, за которым следует нелинейная функция) – что и является основой работы ИНС.

От входных векторов необработанных изображений до конечного результата оценки класса вся сеть будет по-прежнему выражать одну функцию оценки восприятия (вес). Последний слой будет содержать функции потерь, связанные с классами.

Принципиальное различие между свёрточными нейронными сетями и традиционными ИНС состоит в том, что СНС использует внутри себя свёрточные слои и слои подвыборки.

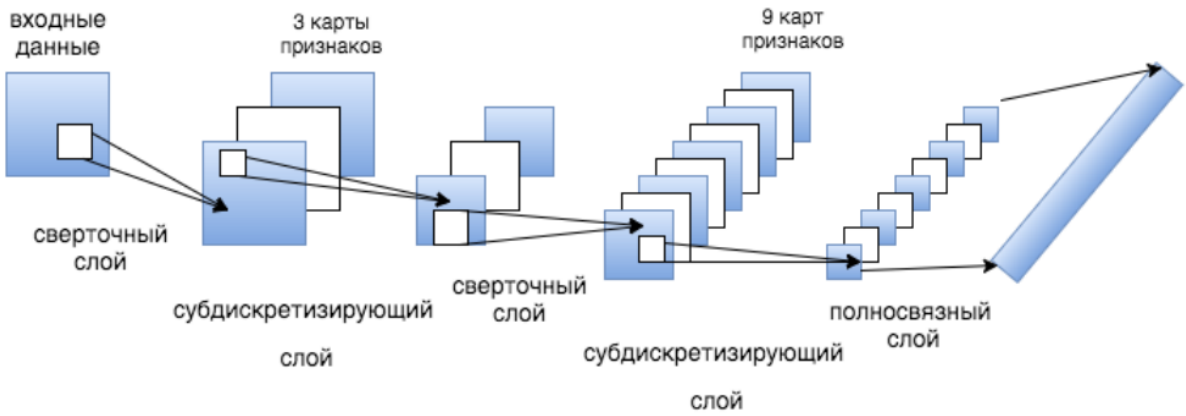

Рис. 1.3. Архитектура свёрточной нейронной сети.

Принцип свёртки относительно прост. Имеется матрица некоторого размера из вещественных элементов, и имеется матрица из весов. В исходной матрице выбирается подматрица (такого же размера, как и матрица весов) и поэлементарно умножается с матрицей весов, после чего результаты умножения складываются.

Получится некоторое значение – значение свёртки на подматрице исходной матрицы. Применяя свёртку к каждой из подматриц, получим множество вещественных чисел – результат применения свёртки к исходной матрице.

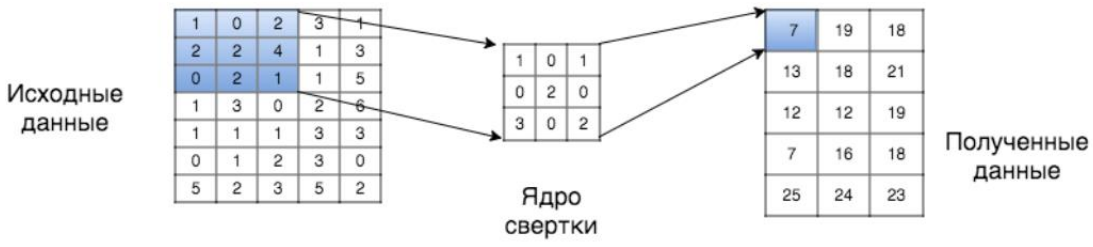

Рис. 1.4. Принцип свёртки.

Обычно в последовательную архитектуру свёрточной нейронной сети периодически вставляют промежуточный уровень подвыборки между слоями свёртки.

Его функция заключается в постепенном уменьшении пространственного размера представления, чтобы уменьшить количество параметров и вычислений в сети, и, следовательно, также контролировать перенастройку.

Слой подвыборки работает независимо на каждом срезе глубины входа и изменяет его размер в пространстве, используя операцию Maximum pooling (выдаёт наибольшее из результатов свёртки на определённом подмножестве результатов).

Операция подвыборки на результатах свертки - это преобразование этого множества результатов в меньшее количество элементов., в то время как mean pooling находит их среднее арифметическое.

Каждая свёртка кодирует некоторый признак, и если он встретится исходной матрице, то в том месте, где он есть, результат применения свёртки будет большим. Mean pooling позволяет оценить среднюю выраженность признака в матрице, к которой была применена свёртка.

К исходной матрице применяются несколько свёрток, тем самым проверяя наличие в матрице нескольких признаков.

Веса свёрток и соответственно признаки, соответствующие им, определяются в результате обучения нейронной сети так, чтобы повысить качество классификации настолько, насколько возможно. Типичная

свёрточная нейронная сеть имеет несколько свёрточных слоёв и слоёв подвыборки. В конце расположены полносвязные слои и softmax-слой, который подаёт на выход вероятности классов.

ГЛАВА 2. ПОДГОТОВКА ИСХОДНЫХ ДАННЫХ.

2.1. Постановка задачи.

Алгоритмы машинного обучения автоматически извлекают знания из машиночитаемой информации. К сожалению, их успех обычно зависит от качества данных, с которыми они работают. Если данные неадекватны или содержат постороннюю и неактуальную информацию, то алгоритмы машинного обучения могут давать менее точные и менее понятные результаты или могут вообще не найти ничего полезного.

Таким образом, предварительная обработка данных является важным шагом в процессе машинного обучения. Этап предварительной обработки необходим для решения нескольких типов проблем, включая шумные данные, избыточные данные, пропущенные значения данных и т. д.

Все индуктивные алгоритмы обучения в значительной степени зависят от результата этой стадии, который является окончательным тренировочным набором данных.

Имеются необработанные данные с мобильного акселерометра, собранные с помощью мобильного приложения. Испытуемому было предложено выполнить несколько повседневных действий, таких как сидение, спокойная ходьба, бег, подъём и спуск по лестнице и отметить меткой каждое из выполняемых действий. Записывающее показания акселерометра устройство находилось во внутреннем кармане верхней одежды.

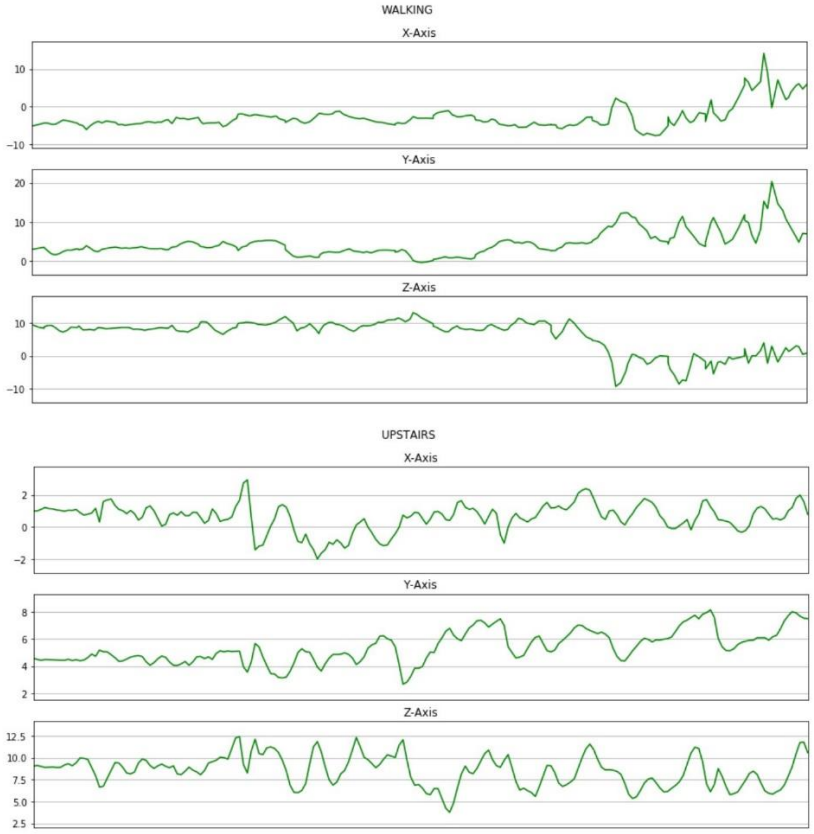

Рис. 2.1. Первые 10 секунд исходных данных (Ходьба, подъём по лестнице).

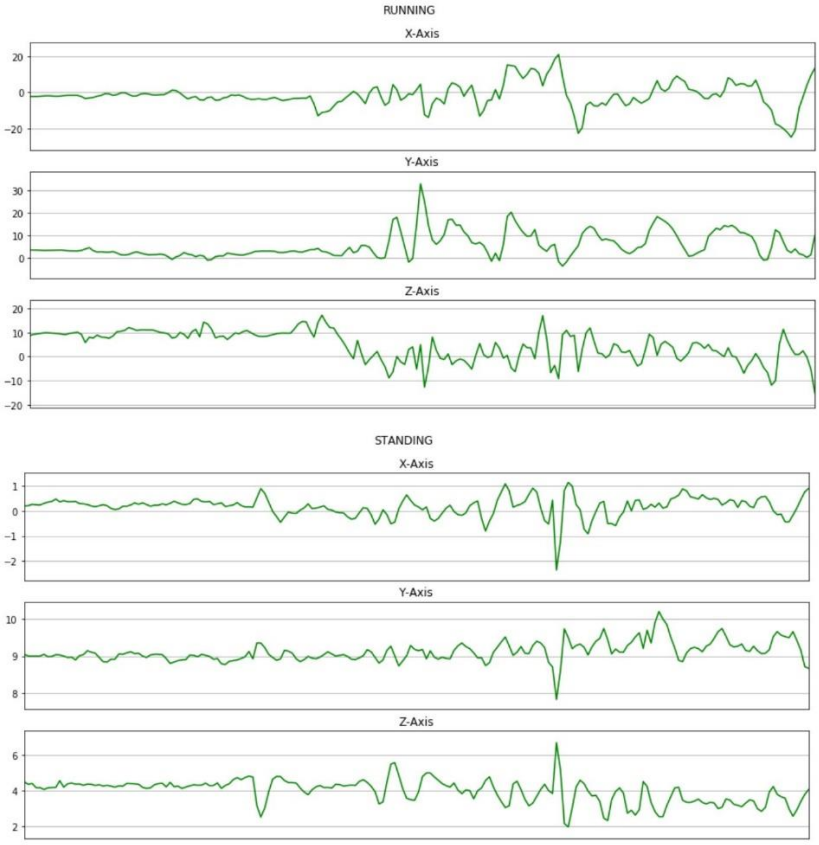

Рис. 2.2. Первые 10 секунд исходных данных (Бег, стояние).

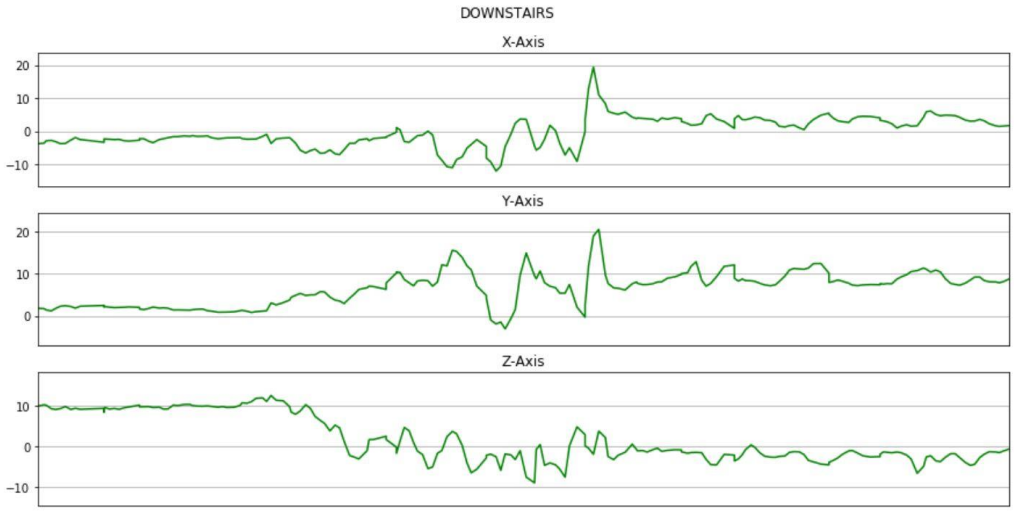

Рис. 2.3. Первые 10 секунд исходных данных (Спуск по лестнице).

2.2. Балансировка данных.

Несбалансированные данные относятся к ситуации, когда количество наблюдений не одинаково для всех классов в наборе классификационных данных.

Классификаторы машинного обучения не справляются с несбалансированными учебными наборами данных, поскольку они чувствительны к пропорциям различных классов. Как следствие, эти алгоритмы имеют тенденцию отдавать предпочтение классу с наибольшей долей наблюдений, что может привести к вводящей в заблуждение точности (рис. 2.4).

Это может быть особенно проблематично, когда мы заинтересованы в правильной классификации «редкого» класса, но мы находим высокую точность, которая фактически является продуктом правильной классификации класса большинства.

Несбалансированные данные

```
Epoch 10/10
399/399 [==============================] - 0s 225us/step - loss: 0.2441 - accuracy: 0.9073 - val_loss: 0.6553 - val_accurac
y: 0.7700
```

Сбалансированные данные

```
Epoch 10/10
176/176 [==============================] - 0s 312us/step - loss: 0.0758 - accuracy: 0.9830 - val_loss: 0.4710 - val_accurac
y: 0.9111
```

Рис. 2.4. Сравнительные потери и точность одной и той же модели на сбалансированных и несбалансированных данных.

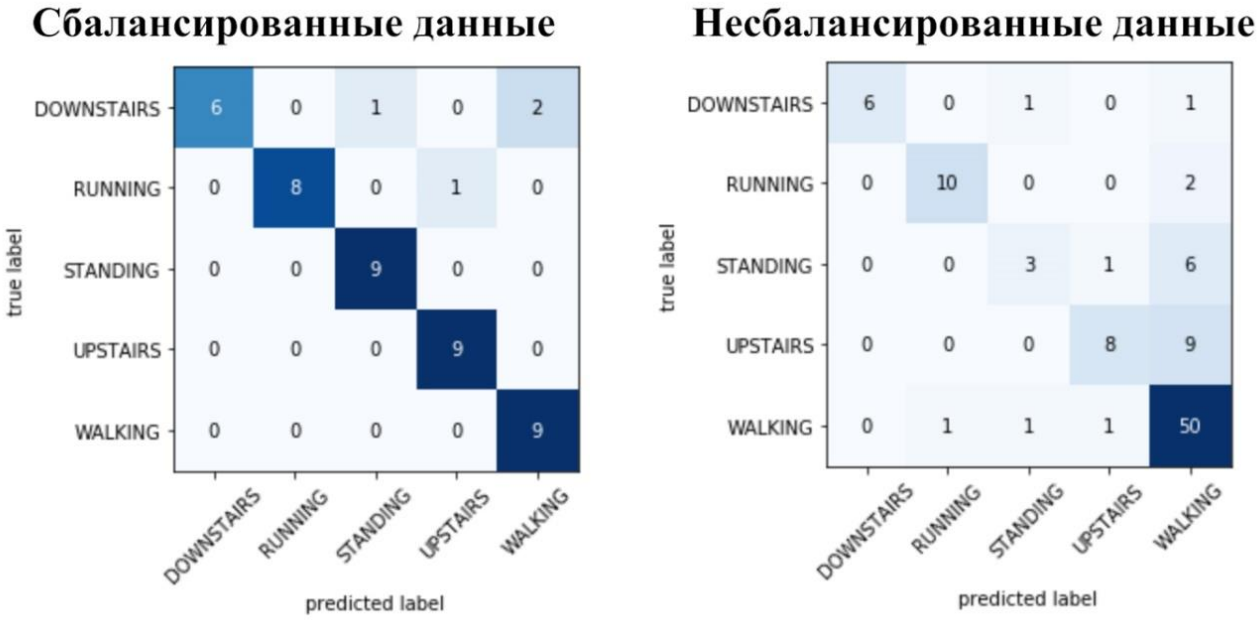

Рис. 2.5. Матрица ошибок одной и той же модели на сбалансированных и несбалансированных данных.

Учитывая, что эти алгоритмы направлены на минимизацию общего числа ошибок, вместо того, чтобы уделять особое внимание классу меньшинства, они могут не дать точного прогноза для этого класса, если не получат необходимое количество информации о нем. Таким образом, настоятельно рекомендуется выполнить балансировку данных.

Произведем подсчёт данных по имеющимся меткам двигательной активности и визуализируем (Рис. 2.6):

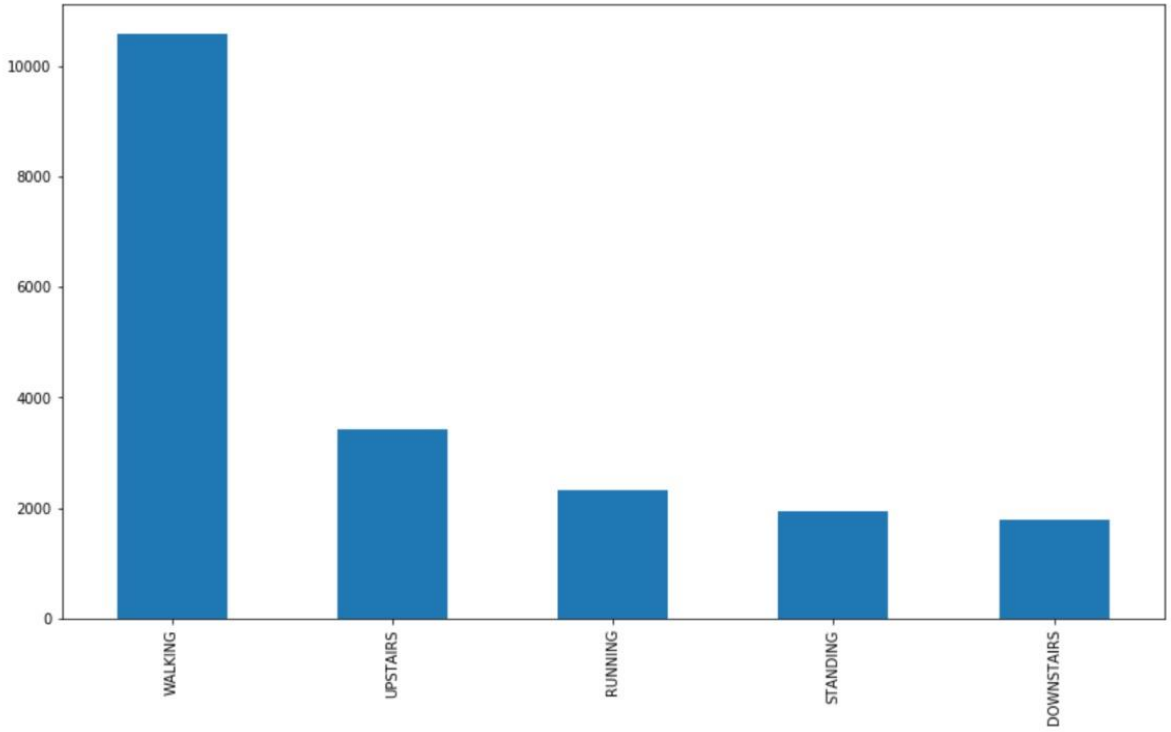

Рис. 2.6. Количество измерений по видам двигательной активности

Имеется много вариантов балансировки данных:

1. Пересмотр тренировочного набора

Помимо использования различных критериев оценки, можно также получить другой набор данных. Два подхода к созданию сбалансированного набора данных из несбалансированного - это недостаточная выборка и избыточная выборка.

2. Недостаточная выборка (Under-sampling)

Недостаточная выборка балансирует набор данных, уменьшая размер класса с обилием. Этот метод используется, когда количество данных достаточно. Сохраняя все выборки в редком классе и случайным образом выбирая равное количество выборок в богатом классе, можно получить сбалансированный новый набор данных для дальнейшего моделирования.

3. Чрезмерная выборка (Over-sampling)

Передискретизация используется, когда количество данных недостаточно. Он пытается сбалансировать набор данных путем увеличения размера редких образцов. Вместо того, чтобы избавляться от обильных

образцов, новые редкие образцы генерируются с использованием, например, повторение, начальная загрузка или SMOTE (методика избыточной выборки синтетического меньшинства).

Следует обратить внимание, что нет абсолютного преимущества одного метода передискретизации перед другим. Применение этих двух методов зависит от варианта использования, к которому он применяется, и от самого набора данных. Сочетание избыточной и недостаточной выборки также часто бывает успешным.

4. K-Fold Cross-Validation

Следует отметить, что перекрестная проверка должна применяться надлежащим образом при использовании метода избыточной выборки для решения проблем дисбаланса.

Стоит иметь в виду, что передискретизация берет наблюдаемые редкие выборки и применяет начальную загрузку для генерации новых случайных данных на основе функции распределения. Если после передискретизации применяется перекрестная проверка, то, в основном, мы делаем так, чтобы наша модель соответствовала конкретному результату искусственной начальной загрузки.

Вот почему перекрестная проверка всегда должна выполняться перед передискретизацией данных, так же как должен осуществляться выбор функций. Только путем повторной выборки данных можно случайно ввести в набор данных случайность, чтобы убедиться, что проблема переобучения не возникнет.

5. Объединение различных наборов данных с передискретизацией

Самый простой способ успешно обобщить модель - использовать больше данных. Проблема заключается в том, что готовые классификаторы, такие как логистическая регрессия или случайный лес, имеют тенденцию обобщать, отбрасывая редкий класс.

Одна из простых лучших практик - это построение n моделей, которые используют все выборки редкого класса и n-отличающиеся выборки заполненного класса. Учитывая, что вы хотите объединить 10 моделей, вы должны сохранить, например, 1.000 случаев редкого класса и случайная выборка 10.000 случаев обильного класса. Затем вы просто разбиваете 10 000 ящиков на 10 частей и тренируете 10 разных моделей.

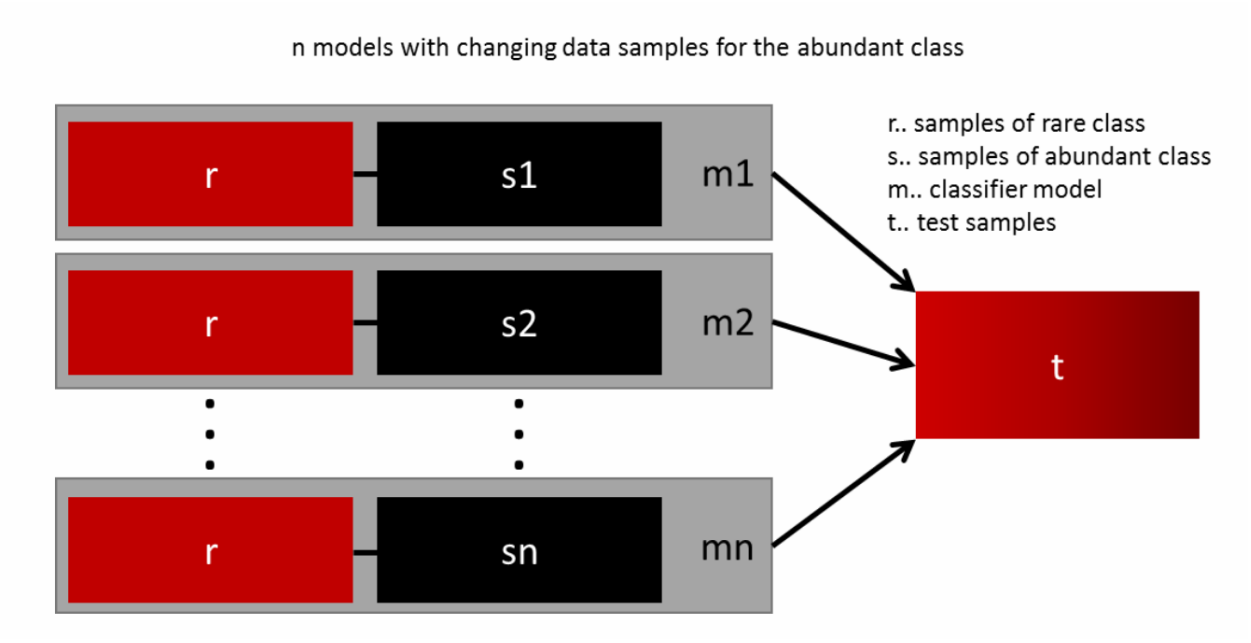

Рис. 2.7. Метод объединения данных

6. Несбалансированное изображение данных

Этот подход прост и идеально масштабируется по горизонтали, если у вас много данных, поскольку вы можете просто обучать и запускать свои модели на разных узлах кластера. Совмещённые модели также имеют тенденцию лучше обобщать, что облегчает использование этого подхода.

7. Повторная выборка с разными соотношениями

Предыдущий подход можно откорректировать, играя с соотношением между редким и обильным классом. Наилучшее соотношение во многом зависит от данных и используемых моделей. Но вместо того, чтобы тренировать все модели с одинаковым соотношением в совмещении, стоит попытаться объединить разные соотношения.

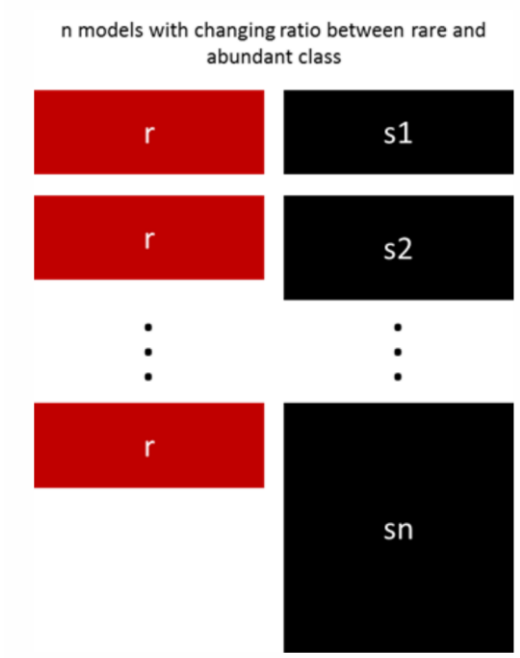

Рис. 2.8. Метод повторной выборки

Поэтому, если обучено 10 моделей, имеет смысл иметь модель с соотношением 1: 1 (редко: много) и еще одну с 1: 3 или даже 2: 1. В зависимости от используемой модели это может повлиять на вес, который получает один класс.

Подсчёт меток показывает следующие значения: WALKING – 10587 образцов, UPSTAIRS – 3414 образцов, RUNNING – 2332 образцов, STANDING – 1926 образцов, DOWNSTAIRS – 1779 образцов.

Из всего этого многобразия видов балансировки, обойдёмся простым – возьмем только первые 1779 образцов из каждого вида активности, балансируя по типу DOWNSTAIRS и объединим данные в новую выборку (Рис. 2.9).

```
In [75]:  WALKING = df[df['activity']=='WALKING'].copy()
          UPSTAIRS = df[df['activity']=='UPSTAIRS'].copy()
          RUNNING = df[df['activity']=='RUNNING'].copy()
          STANDING = df[df['activity']=='STANDING'].copy()
          DOWNSTAIRS = df[df['activity']=='DOWNSTAIRS'].copy()

In [76]:  bdf = pd.DataFrame()
          bdf = bdf.append([WALKING, UPSTAIRS, RUNNING, STANDING, DOWNSTAIRS])

In [77]:  bdf['activity'].value_counts()

Out[77]:  WALKING       10587
          UPSTAIRS       3414
          RUNNING        2332
          STANDING       1926
          DOWNSTAIRS     1779
          Name: activity, dtype: int64
```

Рис. 2.9. Сбалансированная выборка

2.3. Стандартизация данных и Label Encoding.

Стандартизация - это «масштабирование» преобразования функций. Внутри объекта часто существует большая разница между максимальным и минимальным значениями, например, 0,01 и 1000.

Когда стандартизация выполняется, величины величин и масштабируются до заметно низких значений. Это важно для многих нейронных сетей и алгоритмов. Два наиболее распространенных метода для этой области:

- Стандартизация min-max:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \quad (2.3.1)$$

- Стандартизация z-счета:

$$v' = \frac{v - mean_A}{stand\_dev_A} \quad (2.3.2)$$

где v - старое значение функции, а v' - новое.

Мы будем использовать стандартизацию min-max (формула (2.3.1)).

```
In [107]:  X = bdf[['x-axis','y-axis','z-axis']]
           y = bdf['activity']
```

```
In [108]:  scaler = StandardScaler()
           x = scaler.fit_transform(X)
           # создаем сдандартизированную выборку
           scaled_X = pd.DataFrame(data = X, columns = ['x-axis','y-axis','z-axis'])
           scaled_X['activity'] = y.values

           scaled_X
```

Out[108]:

	x-axis	y-axis	z-axis	activity
1779	-5.123591	3.035848	9.375694	WALKING
1780	-4.711789	3.323152	8.609549	WALKING
1781	-4.338294	3.572149	8.312668	WALKING
1782	-4.338294	3.485958	8.724471	WALKING
1783	-4.357447	2.624045	9.155427	WALKING
...	...	...	...	...
1774	-4.194642	3.198653	8.753201	DOWNSTAIRS
1775	-4.299986	3.189077	8.973468	DOWNSTAIRS
1776	-4.683059	3.236961	9.548077	DOWNSTAIRS
1777	-4.845864	3.064578	9.643845	DOWNSTAIRS
1778	-5.123591	2.987964	9.595961	DOWNSTAIRS

20038 rows × 4 columns

Рис. 2.10. Стандартизированная выборка

Перейдём к операции Label Encoding. Как можно видеть на рисунке 2.10, данные типа активности представлены типом данных string (строка текста). Невозможно использовать текст в данных для обучения модели. Поэтому, прежде чем будет возможно начать процесс, нужно подготовить данные.

Для преобразования подобных категорий в понятные модели числовые данные использован класс LabelEncoder.

Таким образом, всё что нужно сделать, чтобы получить признак для первого столбца, это импортировать класс из библиотеки sklearn, обработать колонку функцией fit_transform и заменить существующие текстовые данные новыми закодированными (Рис. 2.11).

```
In [109]:  from sklearn.preprocessing import LabelEncoder
           from sklearn.preprocessing import StandardScaler
```

```
In [110]:  label = LabelEncoder()
           bdf['label'] = label.fit_transform(bdf['activity'])
           bdf.head()
```

Out[110]:

	x-axis	y-axis	z-axis	activity	label
1779	-5.123591	3.035848	9.375694	WALKING	4
1780	-4.711789	3.323152	8.609549	WALKING	4
1781	-4.338294	3.572149	8.312668	WALKING	4
1782	-4.338294	3.485958	8.724471	WALKING	4
1783	-4.357447	2.624045	9.155427	WALKING	4

Рис. 2.11. Итоговый вид подготовленных данных

ГЛАВА 3. ЗАДАЧА КЛАССИФИКАЦИИ.

3.1. Задача классификации и метод скользящего окна.

Задача классификации - это метод обучения с широким контролем, который обучает программу категоризации новой, немаркированной информации на основе ее соответствия известным, маркированным данным. Реализуем задачу, используя распространённый метод прогнозирования для цифровых сигналов - метод скользящего окна.

Метод скользящего окна является классическим методом обработки сигналов и относится к разделению входного сигнала на временные сегменты. Границы сегментов тогда видны как разрывы, которые не соответствуют реальному сигналу.

Чтобы уменьшить влияние сегментирования на статистические свойства сигнала, применим управление окнами к временным сегментам. Чтобы смягчить «потери» по краям окна, отдельные наборы будут перекрываться во времени. Таким образом, управление окнами изменяет сигнал, но изменение разработано таким образом, чтобы его влияние на статистику сигнала было минимальным (Рис. 3.1).

```
In [84]:  import scipy.stats as stats

In [85]:  Fs = 20
          frame_size = Fs*4 #промежуток для прогнозирования, это 80 значений
          hop_size = Fs*2 #насколько сильно следующий промежуток накладывается на предыдущий.
          #иными словами - в новом промежутке у нас точно будет 40 значений, которые еще не рассматривались

In [86]:  def get_frames(df, frame_size, hop_size):

              N_Feat = 3

              frames = []
              labels = []
              for i in range(0, len(df) - frame_size, hop_size):
                  x = df['x-axis'].values[i: i + frame_size]
                  y = df['y-axis'].values[i: i + frame_size]
                  z = df['z-axis'].values[i: i + frame_size]

                  # Извлекаем самую часто используемую метку двигательной активности в этом промежутке прогнозирования
                  label = stats.mode(df['label'][i: i + frame_size])[0][0]
                  frames.append([x, y, z])
                  labels.append(label)
              # собираем массив из (промежуток прогнозирования, соответствующая ему метка двиг.активности)
              frames = np.asarray(frames).reshape(-1, frame_size, N_Feat) #пространство признаков двиг.активности (наша выборка)
              labels = np.asarray(labels) # целевая переменная, что мы пытаемся прогнозировать

              return frames, labels

In [87]:  X, y = get_frames(scaled_X, frame_size, hop_size)
```

Рис. 3.1. Реализация метода скользящего окна

Разобьём данные на обучающие и тестирующие. Как можно заметить из рисунка 14, в нашей обучающей выборке 176 «порций», в то время как в тестирующей 45 «порций».

```
In [216]:  X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 0, stratify = y)
           # разбили данные на обучающие и тестируемые

In [217]:  X_train.shape, X_test.shape

Out[217]: ((176, 80, 3), (45, 80, 3))
```

Рис. 3.2. Разбиение исходных данных на выборки

Также воспользуемся функцией reshape() и изменим форму нашего массива, не поменяв при этом его данные. Это необходимо для корректной работы алгоритма.

```
In [217]:  X_train.shape, X_test.shape

Out[217]: ((176, 80, 3), (45, 80, 3))

In [218]:  X_train[0].shape, X_test[0].shape

Out[218]: ((80, 3), (80, 3))

In [219]:  X_train = X_train.reshape(176, 80, 3, 1)
           X_test = X_test.reshape(45, 80, 3, 1)

In [220]:  X_train[0].shape, X_test[0].shape

Out[220]: ((80, 3, 1), (80, 3, 1))
```

Рис. 3.3. Изменение формы массива

ГЛАВА 4. НЕЙРОННАЯ СЕТЬ.

4.1. Архитектура нейронной сети.

Модель, представленная ниже, реализована в TensorFlow с использованием Keras. Процессор Intel Core i5-6300U, тактовая частота 2400 МГц и 8 ГБ ОЗУ, использовался для обучения и тестирования.

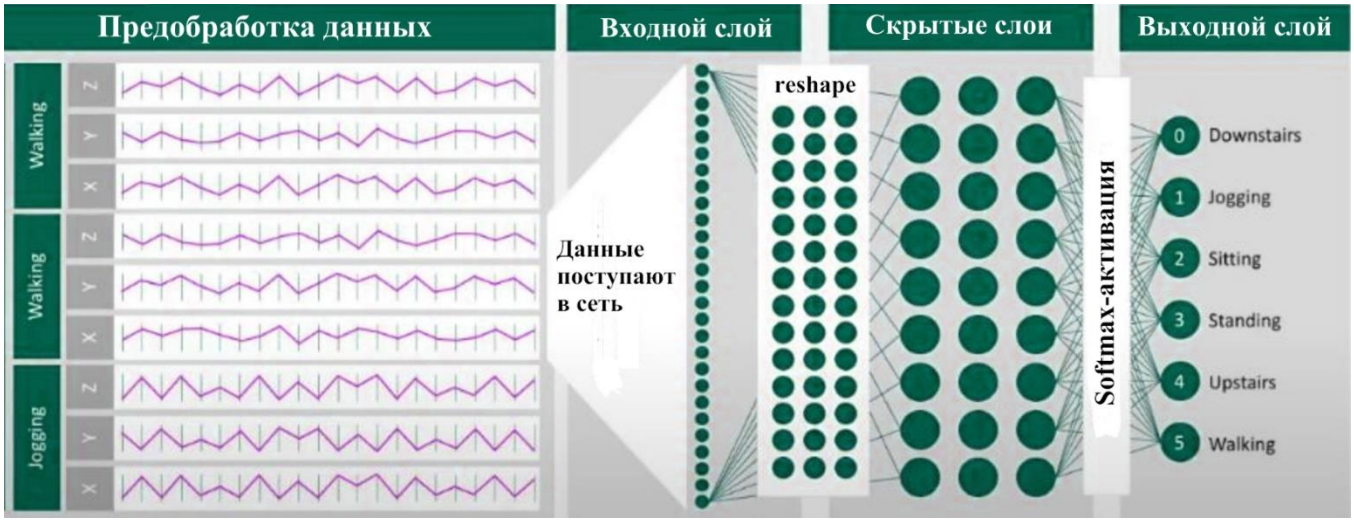

Рис. 4.1. Архитектура свёрточной нейронной сети

Представленная здесь сеть по своей архитектуре является свёрточной нейронной сетью с прямой связью и с тремя скрытыми слоями с функцией активации ReLU.

$$ReLU(x) = max(0, x) \tag{4.1.1}$$

Для предотвращения переобучения (слишком явной подгонки результатов прогнозирования) вводим для слоёв метод Dropout.

Метод Dropout отключает нейрон с вероятностью p и оставляет включенным с вероятностью $q = 1 - p$, при этом вероятность отключения одинакова для любого нейрона сети.

Пусть $a(x)$ — функция активации, тогда применение Dropout для i-ого нейрона выглядит так:

$$U_i = \Theta_i a(\sum_{k=1}^{H} \omega_k x_k + b), \tag{4.1.2}$$

где вероятность $P(\Theta_i = 0) = p$.

Эта формула используется на этапе обучения модели. Но так как на этом этапе нейрон остается включенным в сети с вероятностью q, на этапе тестирования необходимо эмулировать поведение нейронной сети,

использованного при обучении. Для этого результат выходного значения функции активации умножается на коэффициент q, то есть на этапе тестирования:

$$U_i = qa(\sum_{k=1}^{H} \omega_k x_k + b) \tag{4.1.3}$$

Последним слоем этой нейронной сети является слой логистической функции Softmax. Данная функция применяется в машинном обучения для задачи классификации в том случае, когда нужно определить принадлежность данных более чем к двум классам.

```
In [221]:
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from keras.optimizers import Adam
```

```
In [222]:
model = Sequential()
model.add(Conv2D(16, (2, 2), activation = 'relu', input_shape = X_train[0].shape))
model.add(Dropout(0,1))

model.add(Conv2D(32, (2, 2), activation = 'relu'))
model.add(Dropout(0,2))

model.add(Flatten())

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0,5))

model.add(Dense(6, activation = 'softmax'))
```

Рис. 4.2. Алгоритм свёрточной нейронной сети

4.2. Обучение алгоритма на обучающей выборке.

Для тренировки нейронной сети используются разнообразные алгоритмы оптимизации, многие из которых являются вариациями или усовершенствованиями стохастического градиентного спуска.

Как именно учится нейронная сеть? Тренировка нейронной сети заключается в подборе весов таким образом, чтобы штрафная функция на тренировочном наборе была минимальна.

$$L(\omega) = \sum_{i=0}^{N-1} L(\omega, x_i, y_i), \tag{4.2.1}$$

где $y_i \epsilon Y$ это метка класса для объекта $x_i \epsilon X$, а $\omega = \omega'$ это веса, которые должны быть подобраны таким образом, чтобы штрафная функция $L(\omega')$ принимала минимальное значение.

Инициируем модель, выбрав в качестве оптимизатора Adam.

Запустим процесс, вызвав метод model.fit и пронаблюдаем, как модель учится за 10 эпох. Прошедшая эпоха означает – что весь набор данных прошел через нейронную сеть.

```
In [223]:  model.compile(optimizer=Adam(learning_rate = 0.001), loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])

In [224]:  history = model.fit(X_train, y_train, epochs = 10, validation_data = (X_test, y_test), verbose = 1)

Train on 176 samples, validate on 45 samples
Epoch 1/10
176/176 [==============================] - 0s 2ms/step - loss: 1.7894 - accuracy: 0.3693 - val_loss: 1.1990 - val_accuracy:
0.5111
Epoch 2/10
176/176 [==============================] - 0s 272us/step - loss: 0.8437 - accuracy: 0.6705 - val_loss: 0.8822 - val_accurac
y: 0.7556
Epoch 3/10
176/176 [==============================] - 0s 278us/step - loss: 0.6103 - accuracy: 0.8125 - val_loss: 0.8197 - val_accurac
y: 0.6000
Epoch 4/10
176/176 [==============================] - 0s 278us/step - loss: 0.5011 - accuracy: 0.8011 - val_loss: 0.6922 - val_accurac
y: 0.8444
Epoch 5/10
176/176 [==============================] - 0s 357us/step - loss: 0.3921 - accuracy: 0.9034 - val_loss: 0.6290 - val_accurac
y: 0.8889
Epoch 6/10
176/176 [==============================] - 0s 408us/step - loss: 0.3099 - accuracy: 0.9261 - val_loss: 0.5404 - val_accurac
y: 0.8444
Epoch 7/10
176/176 [==============================] - 0s 448us/step - loss: 0.2484 - accuracy: 0.9091 - val_loss: 0.4612 - val_accurac
y: 0.8889
Epoch 8/10
176/176 [==============================] - 0s 283us/step - loss: 0.2090 - accuracy: 0.9318 - val_loss: 0.4341 - val_accurac
y: 0.8889
Epoch 9/10
176/176 [==============================] - 0s 221us/step - loss: 0.1520 - accuracy: 0.9659 - val_loss: 0.4636 - val_accurac
y: 0.9111
Epoch 10/10
176/176 [==============================] - 0s 306us/step - loss: 0.1265 - accuracy: 0.9545 - val_loss: 0.4943 - val_accurac
y: 0.8889
```

Рис. 4.3. Алгоритм проходит 10 эпох и учится.

В процессе обучения модели с каждой эпохой показывается метрика потерь (loss) и точности (accuracy). Эта модель достигает на тренировочных данных точности равной приблизительно 0.95 (95%), в то время как на проверочных данных точность составляет 0,88 (88%).

Небольшое различие между значениями точности на тренировочных и тестовых данных может являться свидетельством переобучения. Обычно переобучение возникает, если модель машинного обучения показывает на незнакомых ей данных худший результат, чем на тех, на каких производилось обучение.

4.3. Оценка работы алгоритма.

Чем меньше потери, тем лучше модель (если модель переучена на тренировочных данных). Потери рассчитываются при обучении и проверке, и их интерпретация определяет, насколько хорошо модель справляется с этими двумя наборами. В отличие от точности, потери не в процентах. Это сумма ошибок, сделанных для каждого примера в обучающих или проверочных наборах.

В случае нейронных сетей потери обычно имеют отрицательную логарифмическую вероятность и остаточную сумму квадратов для задач классификации и регрессии соответственно.

Естественно, что основной целью в модели обучения является уменьшение (минимизация) значения функции потерь по отношению к параметрам модели путем изменения значений вектора весовых коэффициентов с помощью различных методов оптимизации, таких как обратное распространение в нейронных сетях.

Значение потерь подразумевает, насколько хорошо или плохо ведет себя определенная модель после каждой итерации оптимизации. В идеале можно ожидать уменьшения потерь после каждой или нескольких итераций.

Точность модели обычно определяется после того, как параметры модели изучены и зафиксированы, а обучение не проводится. Затем тестовые образцы подаются в модель, и после сравнения с истинными целями регистрируется количество ошибок (потеря ноль один), которые делает модель. Затем рассчитывается процент ошибочной классификации.

Построим графики точностей (рис. 4.3) и потерь (рис. 4.4) алгоритма по обучающим и тестирующим выборкам.

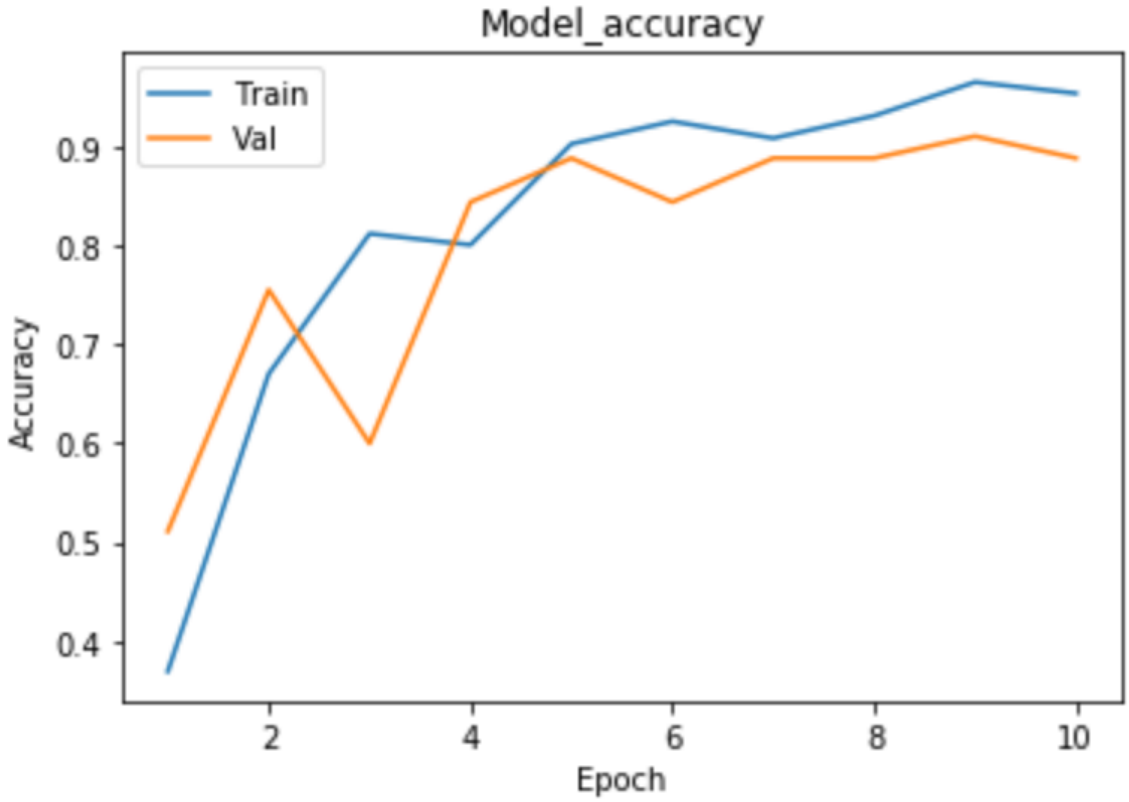

Рис. 4.3. Графики точностей.

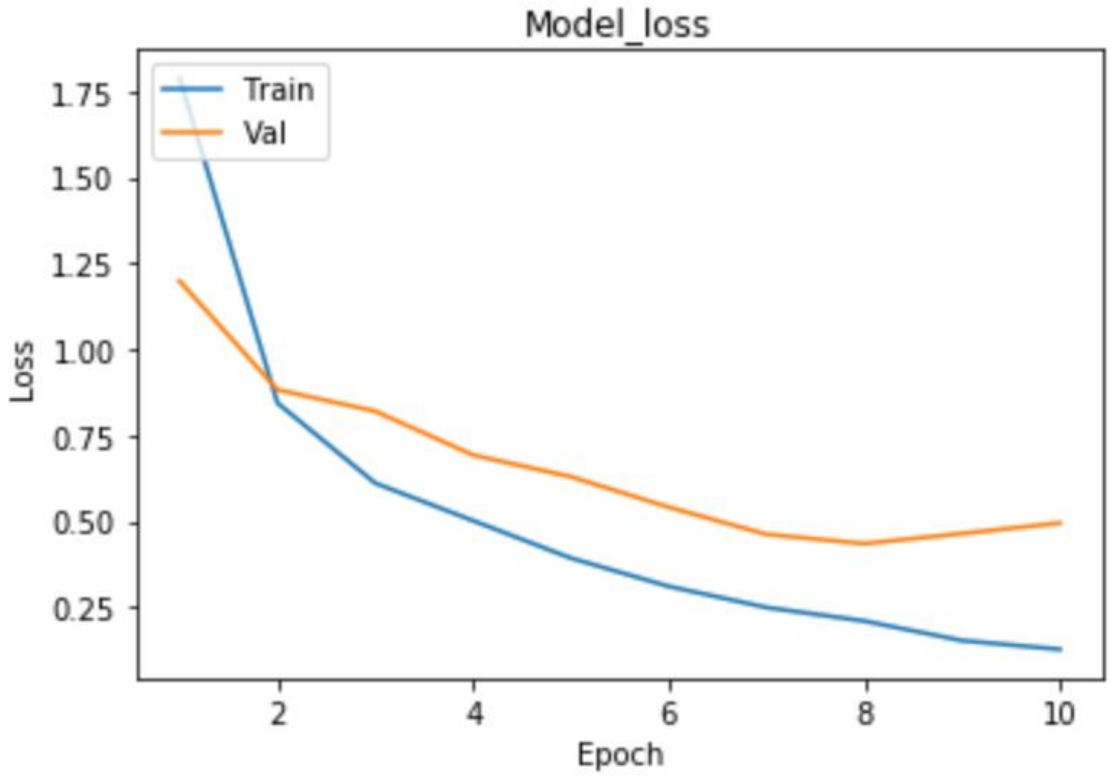

Рис. 4.4. Графики потерь

Еще одним удобным инструментом для оценки точности алгоритма является матрица ошибок (Confusion matrix). Количество правильных и неправильных прогнозов суммируется со значениями количества и разбивается по каждому классу.

Матрица ошибок показывает, каким образом наша модель классификации путается, когда она делает прогнозы. Это дает нам представление не только об ошибках, допущенных классификатором, но, что более важно, о типах ошибок, которые совершаются.

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Рис. 4.5. Матрица ошибок

Ключ к матрице ошибок:

- Положительный (P): Положительный прогноз.
- Отрицательный (N): Отрицательный прогноз.
- Истинно положительный (TP): Прогноз положителен, и прогнозируется, что он будет положительным.
- False Negative (FN): Прогноз положительный, но прогнозируется отрицательным.
- True Negative (TN): Прогноз отрицательный, и прогнозируется как отрицательный.
- Ложно положительный (FP): Прогноз отрицательный, но прогнозируется положительным.

Выведем матрицу ошибок нашей модели нейронной сети после итераций обучения и проверки (рис 4.6).

Рис. 4.6. Матрица ошибок свёрточной нейронной сети

Из матрицы ошибок можно увидеть, что нейронная сеть допустила 5 ошибок, больше всего ошибок было в типе активности DOWNSTAIRS – 3 ошибки. Это может быть свидетельством недостаточного количества данных для обучения, наши исходные данные были небольшими, всего 20 038 значений.

Если доступно только небольшое количество обучающих данных, рекомендовано провести обучение по нескольким моделям и выбрать лучшую модель на основе характеристик в наборе проверочных данных.

Альтернативный подходом может быть трансферное обучение, которое может быть использовано для уменьшения влияния на случайно инициализированные веса.

Известно, что методы глубокого обучения весьма ресурсоемки, и до сих пор зачастую сложно удовлетворить их вычислительные требования к

встроенным устройствам. Тем не менее, с учетом последних технологических достижений в области тензорных процессоров ожидается, что мобильные устройства в ближайшем будущем смогут реализовывать алгоритмы глубокого обучения. Сверточные нейронные сети имеют большой потенциал для идентификации различных характерных моделей сигналов двигательной активности человека.

ЗАКЛЮЧЕНИЕ

Традиционные подходы к распознаванию двигательной активности человека за последние годы достигли огромного прогресса. Однако эти методы часто в значительной степени основаны на эвристическом извлечении созданных вручную признаков, что может снизить производительность их обобщения. Кроме того, существующие методы являются недостаточными для неконтролируемых и дополнительных задач обучения.

В последнее время последние достижения в области глубокого обучения позволяют выполнять автоматическое извлечение признаков высокого уровня, что обеспечивает многообещающие результаты во многих областях. С тех пор методы, основанные на глубоком обучении, начинают применяться для задач распознавания двигательной активности на основе датчиков.

В результате выполнения исследовательской работы был реализован алгоритм свёрточной нейронной сети, обучающейся на выборке с данных мобильного акселерометра

Основным языком для работы над обработкой данных, алгоритмом и архитектурой нейронной сети был Python. В работе были решены поставленные задачи:

1) Произведен краткий литературный обзор о принципе работы свёрточной нейронной сети.
2) Были подготовлены данные для работы с нейросетью. Подробно объяснена важность балансировки, её виды. Произведена стандартизация данных и Label Encoding.
3) Была описана задача классификации и было произведено разбиение выборки на обучающую и тестирующую.
4) Представлен и реализован алгоритм свёрточной нейронной сети, прошедший последующее обучение и проверку.
5) Была произведена оценка работы алгоритма.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ba, J., and Kingma, D. Adam: A method for stochastic optimization, 2015.
2. Banos, O., Damas, M., Pomares, H., et al. A benchmark dataset to evaluate sensor displacement in activity recognition. (2012), 1026–1035.
3. Bleser, G., Steffen, D., Reiss, A., Weber, M., Hendeby, G., and Fradet, L. Personalized Physical Activity Monitoring Using Wearable Sensors. LNCS. 2015, 99–124.
4. Dieleman, S., Schlter, J., Raffel, C., et al. Lasagne: First release., Aug. 2015.
5. Hammerla, N. Y., Halloran, S. Deep, convolutional, and recurrent models for human activity recognition using wearables. (2016), 1533–1540.
6. Haskell, W. L., Lee, et al. Physical activity and public health: Updated recommendation for adults from the American College of Sports Medicine and the American Heart Association. Medicine and Science in Sports and Exercise 39, 8 (Aug. 2007), 1423–34.
7. Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167 (2015).
8. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25. Curran Associates, Inc., 2012, 1097–1105.
9. Lee, M.-h., Kim, J., Kim, K., et al. Physical activity recognition using a single tri-axis accelerometer. In Proceedings of World Congress on Engineering and Computer Science (WCECS) (2009).
10. Martinez, H. P., Bengio, Y., and Yannakakis, G. N. Learning deep physiological models of affect. IEEE Computational Intelligence Magazine 8, 2 (2013), 20–33. 1
11. Lane N. D., Georgiev P., and Qendro L. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In Ubicomp, pages 283–294. ACM, 2015.
12. LeCun Y., Bengio Y., and Hinton G. Deep learning. Nature, 2015.

13. Neverova N., Wolf C., Lacey G., Fridman L., Chandra D., Barbello B., and Taylor G. Learning human identity from motion patterns. arXiv:1511.03908, 2015.
14. Ordo´nez F. J., Roggen D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1):115, 2016.
15. Plotz T., Hammerla N. Y., Olivier P.. Feature learning for activity recognition in ubiquitous computing. In IJCAI, 2011.
16. Plotz T., Hammerla N. Y., Rozga A., Reavis A., Call N. Automatic assessment of problem behavior in individuals with developmental disabilities. In Ubicomp, 2012.
17. Rad N. M., Bizzego A., Kia S. M., Jurman G., Venuti P.. Convolutional neural network for stereotypical motor movement detection in autism. arXiv:1511.01865, 2015.