Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Радиотехники и Компьютерных Технологий
Кафедра инфокоммуникационных систем и сетей

Направление подготовки / специальность: 03.03.01 Прикладные математика и физика
Направленность (профиль) подготовки: Радиотехника и компьютерные технологии

ЗАДАЧА МНОГОКЛАССОВОЙ КЛАССИФИКАЦИИ СИГНАЛОВ. ОПРЕДЕЛЕНИЕ ЗАБОЛЕВАНИЙ ПО ЭКГ

(бакалаврская работа)

Студент:
Филиппов Иван Михайлович

_____
(подпись студента)

Научный руководитель:
Платонов Евгений Николаевич,
канд. физ.-мат. наук, доц.

_____
(подпись научного руководителя)

Консультант *(при наличии)***:**
_____
_____
(подпись консультанта)

Москва 2022

Содержание

1 Введение **3**

1.1 Актуальность 3

1.2 Кардиография 3

1.3 Краткий экскурс в нейронные сети 5

1.3.1 Основные принципы 5

1.3.2 Свёрточные сети 8

1.3.3 Рекуррентные сети 11

1.4 Обзор литературы 12

1.5 Обзор используемых данных 13

1.6 Источники протестированных сетей 15

2 Постановка задачи **16**

3 Эксперименты **17**

3.1 Программная реализация 17

3.2 Сводные таблицы 18

3.3 Попытки улучшить архитектуру 20

3.3.1 Encoder 20

3.3.2 Inception 21

3.3.3 Zolotyh 21

3.3.4 Результаты изменений 22

3.4 Ансамбли 24

4 Заключение **27**

5 Список литературы **28**

1 Введение

1.1 Актуальность

Заболевания сердечно-сосудистой системы являются самой распространённой причиной смерти во всём мире [1], в России за последние 30 лет доля смертей от болезней системы кровообращения составляла от 44 до 57 процентов [2]. Очевидна необходимость внедрения новых методов диагностики и предупреждения заболеваний этого класса.

На данный момент ЭКГ (электрокардиограмма) является одним из основных средств диагностирования заболеваний сердца. Сама процедура проведения электрокардиографии [3] не требует от работника высокой квалификации, в отличие от анализа результатов – это может делать только врач. Сейчас во многих направлениях медицины разрабатываются способы автоматического распознавания потенциальных больных для их последующего направления к реальному врачу с целью снятия нагрузки со специалистов. Электрокардиография, конечно, не исключение.

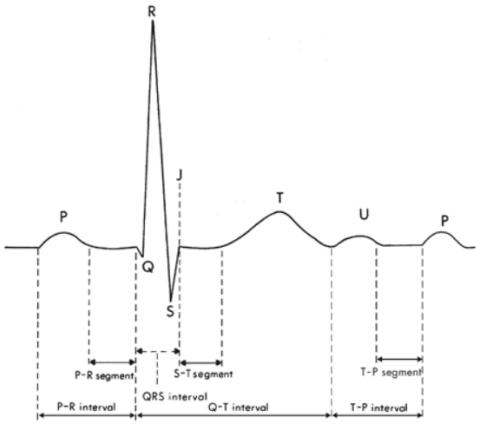
В данной работе будет рассмотрена задача многоклассовой классификации заболеваний по данным ЭКГ с использованием нейронных сетей [4]. При этом исследованы различные архитектуры, которые реализуют общепринятые подходы, такие как свёрточные [5] и рекуррентные [6–10] сети. Лучшие из них будут взяты за основу конечной модели.

1.2 Кардиография

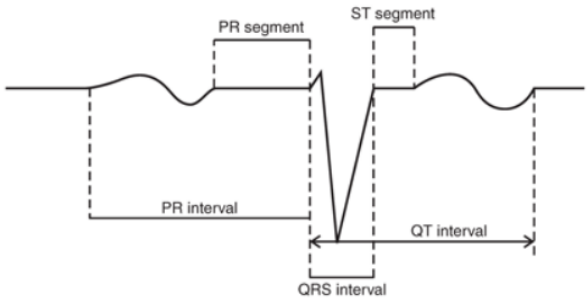
Электрокардиография – медицинский метод изучения состояния сердца. Он основан на измерении электрических потенциалов на теле человека, которые изменяются во времени в процессе работы сердца. Начало электрокардиографии было положено в конце 19го века, первый прибор для регистрации ЭКГ был создан в 1900х годах Виллемом Эйнтховеном, за что тот был удостоен нобелевской премии в 1924 году.

Результатом кардиографии является кардиограмма – один или несколько графиков зависимости потенциала от времени. Обычно эти графики печатаются на термопринтере, выдающем бумажную ленту с измеренными

колебаниями. Также измерения могут сохраняться в электронным виде на компьютере. Примеры кардиограмм на рисунках 1 и 2-3.



(a) Отведение II
(вдоль длинной оси сердца)



(b) Отведение V1
(вдоль короткой оси сердца)

Рис. 1: Эскизы кардиограмм разных отведений [3]

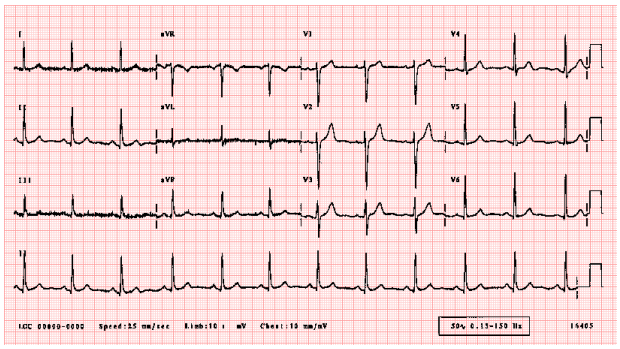


Рис. 2: ЭКГ здорового человека [11]

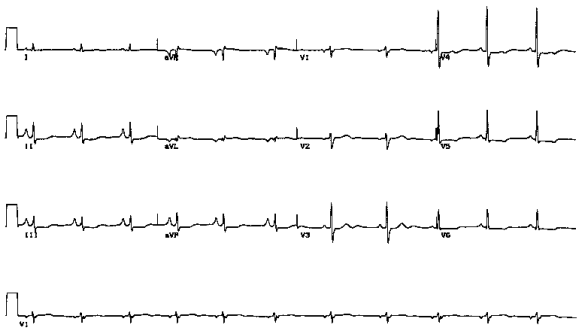


Рис. 3: Гипертрофия правого предсердия [11]

Каждый график отображает разность потенциалов, которую называют отведением (lead). Разности потенциалов измеряются при помощи электродов, крепящихся на тело пациента и смазанных токопроводящим гелем или спиртом. Чаще всего используется схема с 12ю отведениями, 4 электрода крепятся на конечности и ещё 6 на грудь в районе сердца. На рис 4 показаны характерный вид сигналов, снимаемых с отведений, и места крепления электродов на грудную клетку. При этом 6 грудных отведений именуются с V1 по V6, а из 6ти отведений с конечностей (I, II, III, aVR, aVL, aVF) только 2 являются линейно независимыми.

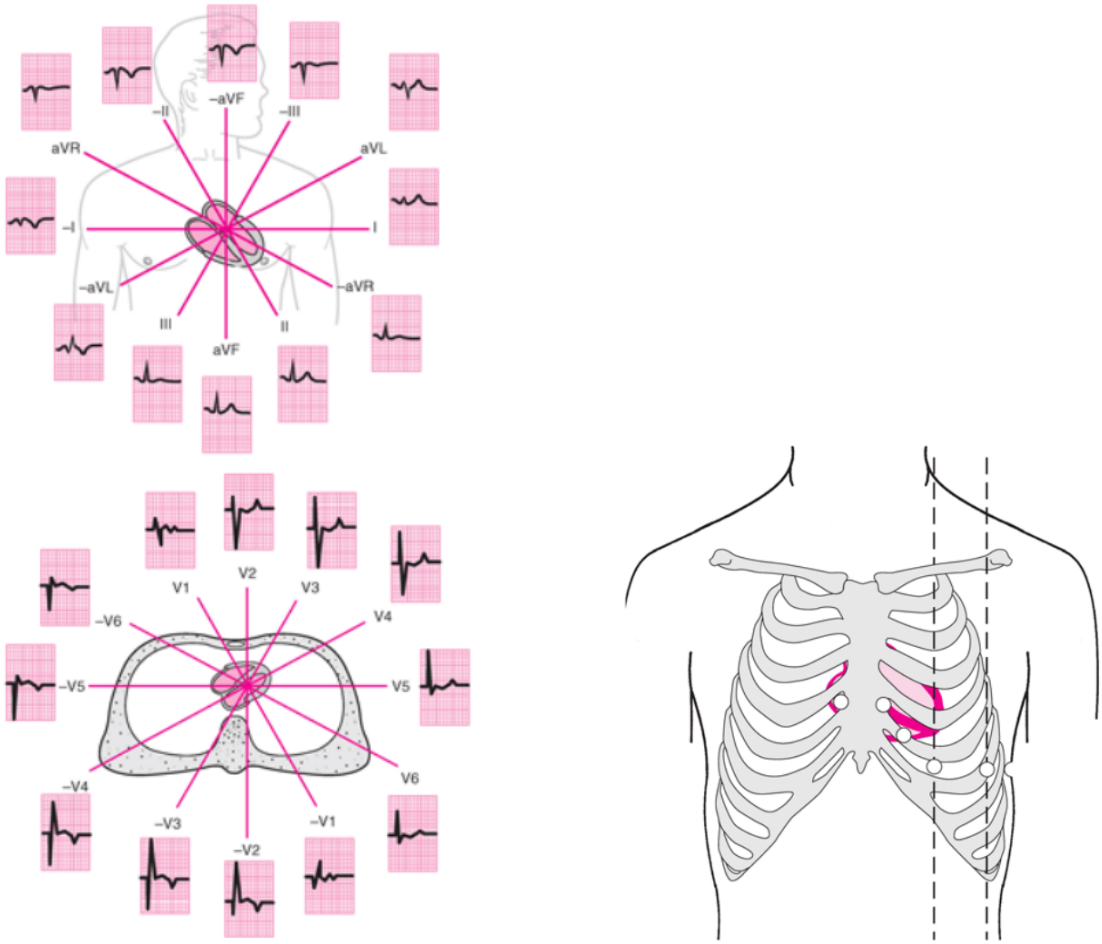


Рис. 4: 12 отведений и места крепления электродов V1-V6 [3]

1.3 Краткий экскурс в нейронные сети

1.3.1 Основные принципы

Нейронная сеть – модель, главный принцип работы которой позаимствован у природы – нейронов мозга. Идея заключается в следующем. Каждый нейрон в сети вычисляет значение некоторой функции, заложенной в него. После этого к результатам применяется функция активации. Эта функция как бы определяет пойдёт ли сигнал дальше по сети – как в настоящем нейроне, для этого она обязана быть существенно нелинейной. При конструировании нейронных сетей термин «нейрон» обычно не используют, сети задают сразу «слоями». Слой – это набор нейронов с одинаковыми функциями отображения, при этом нейроны одного слоя не соединены друг с другом связями. Рассмотрим самый простой пример.

Введём линейный слой. Функция этого слоя:

$$y = Wx + b,$$

где $x \in \mathbb{R}^n$ – входной вектор, $W \in \mathbb{R}^{m \times n}$ – матрица весов, $b \in \mathbb{R}^m$ – вектор сдвига, а $y \in \mathbb{R}^m$ – выходной вектор. Обычно этот слой обозначают как `Linear(n, m)`, `Dense(n, m)` или `FC(n, m)` (Fully Connected).

Второй вид слоёв – слой активации. Иногда его не выделяют в отдельный слой, включая в предыдущий. Первыми функциями активации служили tanh (гиперболический тангенс) и sigmoid (сигмоида). Однако сейчас чаще всего используется функция ReLU.

$$ReLU(x) = \max(0, x)$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

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

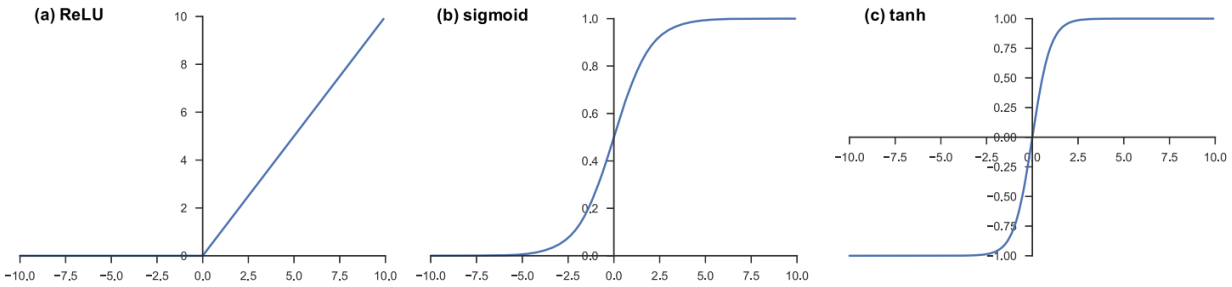


Рис. 5: Функции активации

Главный механизм обучения нейронных сетей – обратное распространение ошибки. Принцип его работы состоит в следующем. Выходом нейронной сети является некоторый набор значений, на его основе вычисляется значение функции потерь. После этого веса последнего слоя корректируются, чтобы уменьшить её – сдвигаются в сторону, противоположную направлению градиента функции потерь по весам слоя. Далее это передаётся по всем слоям к началу сети, при этом имеет место «цепное правило». В итоге в каждом слое вычисляется производная функции потерь по весам в слое. Рис 6 и выкладки ниже наглядно демонстрируют этот механизм, называемый «обратное распространение ошибки».

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \cdot \frac{\partial out_{h1}}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

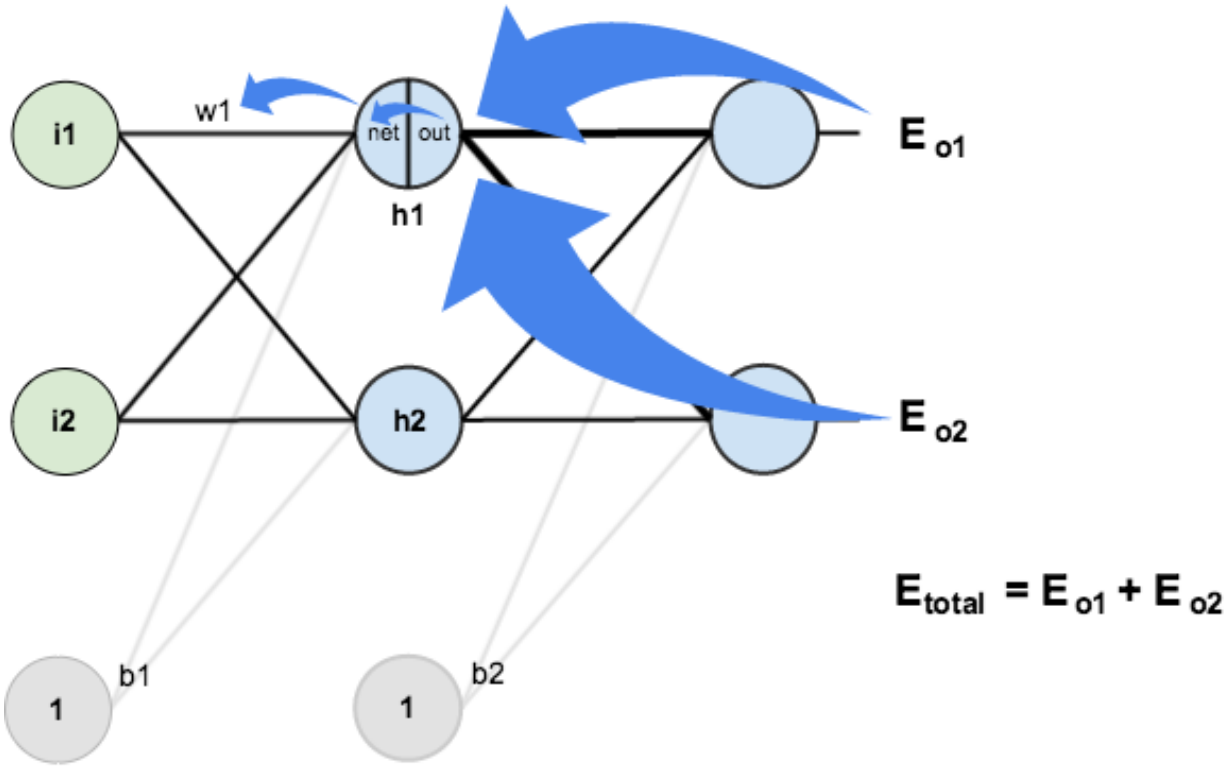


Рис. 6: Обратное распространение ошибки [12]

Например, простейшая сеть:

`Linear(2, 2) Sigmoid() Linear(2, 1)`

решает так называемую «проблему исключающего или», описанную, например, в [4].

Для определения качества предложенной модели классификации используются метрики. Стандартными метриками являются accuracy, precision, recall, F_1-score, F_β-score. Для их определения рассматривается таблица 1. Здесь True/False означают правильность предсказания, а Positive/Negative – метки классов в бинарной классификации. То есть, например, TN – количество объектов, которые правильно распознаны как Negative.

В этих терминах вышеупомянутые метрики определяются как:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Таблица 1: Матрица несоответствий

		predicted	
		positive	negative
true	positive	True Positive (TP)	False Negative (FN)
	negative	False Positive (FP)	True Negative (TN)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

Коэффициент β в F_β означает «важность» *Recall*'а по сравнению с *Precision*. В данной работе внимание будет уделено метрикам *Accuracy* и F_2. *Accuracy* – так как это простая и понятная метрика, которая показывает долю правильных ответов. F_2 же выбрана, потому что в задачах диагностирования заболеваний лучше поставить диагноз здоровому человеку для последующих исследований, чем отпустить больного, не распознав его болезнь. Таким образом *Recall* важнее, поэтому и выбрана $\beta > 1$.

Далее рассмотрим чуть подробнее стандартные подходы, которые довольно часто используются в нейронных сетях. Сконцентрируемся на тех, которые подходят под наш формат данных – временные ряды, то есть последовательность измерений.

1.3.2 Свёрточные сети

Обычно при помощи свёрточных сетей обрабатываются изображения. Рассмотрим, как это происходит.

Начнём со слоёв сжатия (подвыборки, субдискретизации, пулинга, англ. Pooling layer). Мы можем сжать, например, каждые 4 пикселя в один, взяв среднее значение, или максимальное, или как-то ещё. Это позволяет снизить число обучаемых признаков, так как количество информации в

изображениях с высоким разрешением зачастую избыточно и размерности матриц будут сильно замедлять обучение.

Главная идея состоит в, собственно, свёрточных слоях (англ. Convolutional layer) или просто свёртках. На вход подаются данные размерности $M \times N \times C$, где C – количество каналов, $M \times N$ – размер изображения. К изображению применяется D ядер (фильтров, kernel, filter) размерами $K \times K \times C$. Ядро «прикладывается» к картинке в каждой точке и вычисляется насколько изображение в ядре совпадает с участком исходной картинки. Можно интерпретировать это как шаблон, который мы ищем на изображении (см. рисунок 7). На выходе получаем данные размерности $M-K+1 \times N-K+1 \times D$.

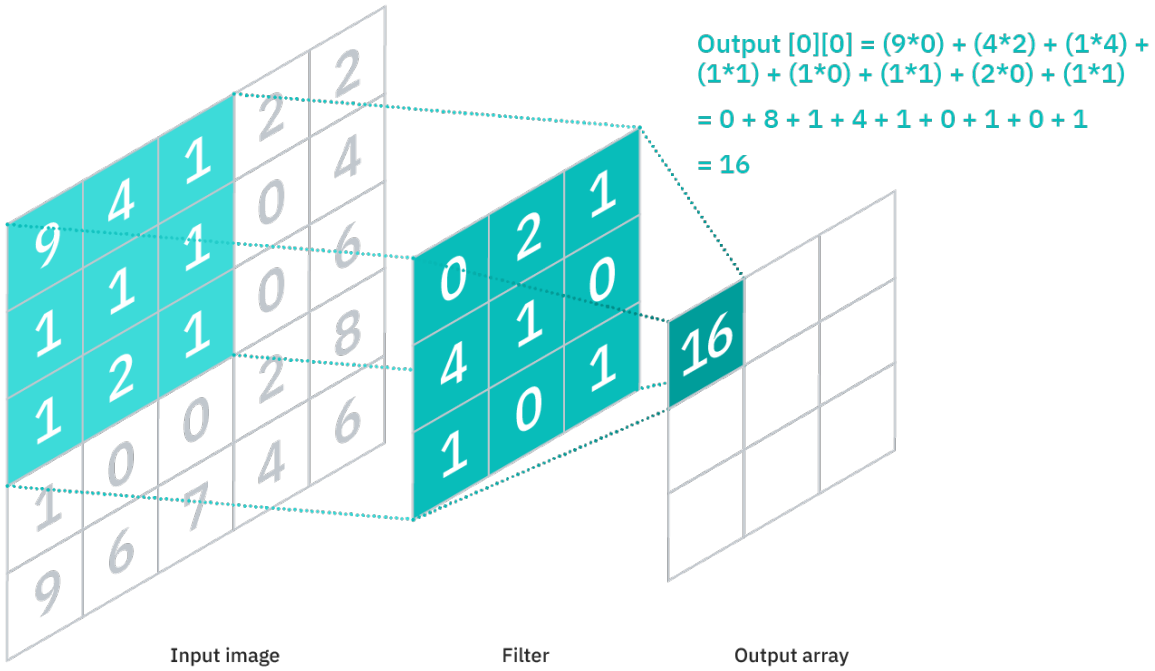


Рис. 7: Демонстрация работы ядра свёртки

Чем больше свёрток последовательно применено к изображению, тем более сложные структуры могут быть выделены при помощи ядер. Это продемонстрировано на рисунках 8 и 9. Странные цвета на изображениях вызваны тем, что это отображение многоканальных данных в приемлемый для восприятия вид. Можно видеть, что на первых слоях свёрток фильтры научились различать примитивные структуры – границы объектов, линии, точки. Далее фигуры усложняются – видны некие конструкции из множества линий. Ещё дальше в фильтрах становятся различимы сложные объекты, например, морда животного и колесо автомобиля.

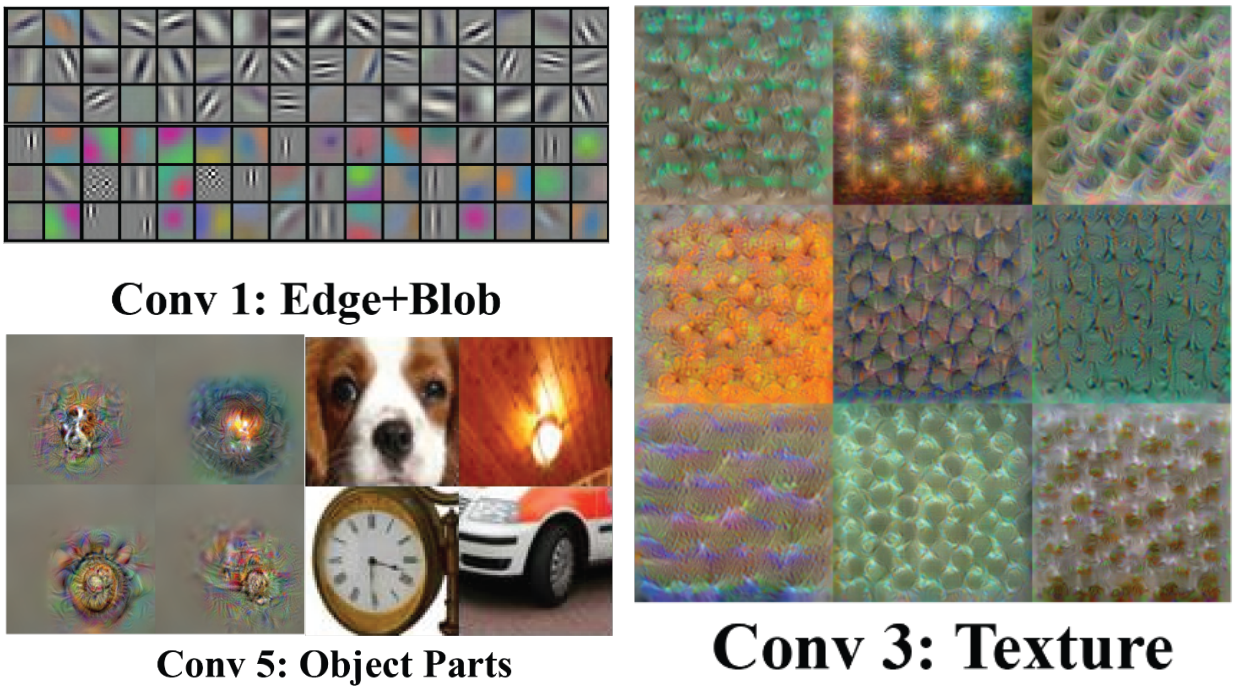


Рис. 8: Ядра свёрток разных слоёв [13]

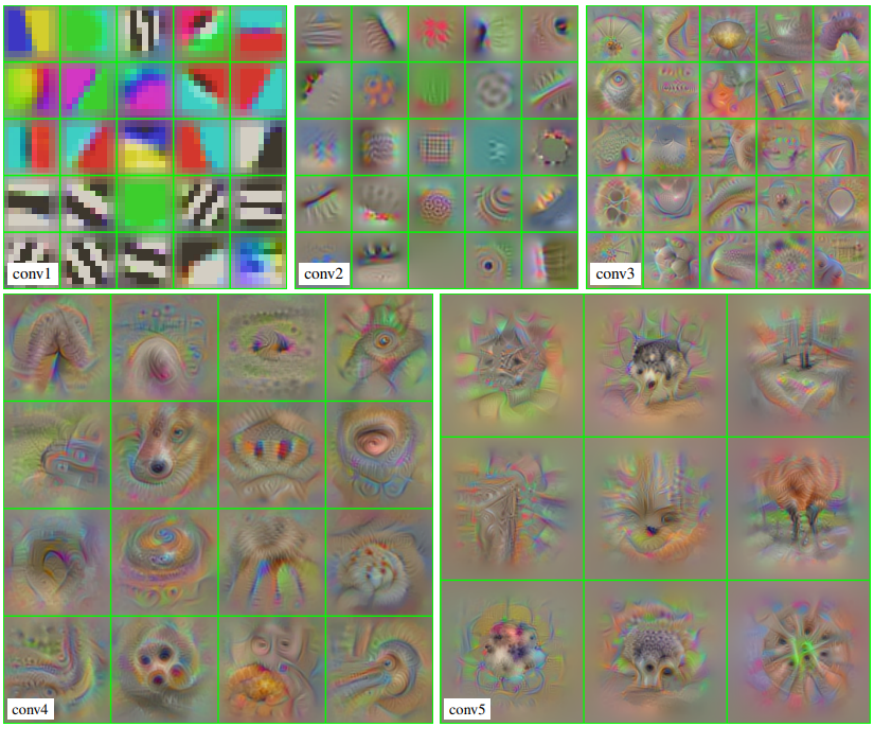


Рис. 9: Ядра свёрток разных слоёв [14]

Здесь для наглядности механизм работы свёрток описан для двумерных изображений. Однако он так же показал себя эффективным для использования в задачах о временных рядах. При этом свёртки не обязаны быть двумерными, а могут применяться и вдоль одной оси. Интуитивно

понятно, почему этот подход применим для временных рядов. Они представляют собой некоторую последовательность измерений, которую можно представить как график функции. У этой функции могут быть отличительные или повторяющиеся шаблоны поведения. Особенно логичным это кажется в контексте кардиограммы. Модель будет искать не изображение, похожее на морду собаки, а характерную последовательность пиков, свойственную какому-либо заболеванию – тоже некий шаблон.

1.3.3 Рекуррентные сети

Рекуррентные сети полезны для обработки последовательных данных. Часто их используют при анализе естественного языка или временных рядов. Идея рекуррентного слоя такова: данные обрабатываются поэлементно, слой хранит некоторое внутреннее состояние, которое дополняется каждым новым входным элементом, при этом размер скрытого состояния постоянен.

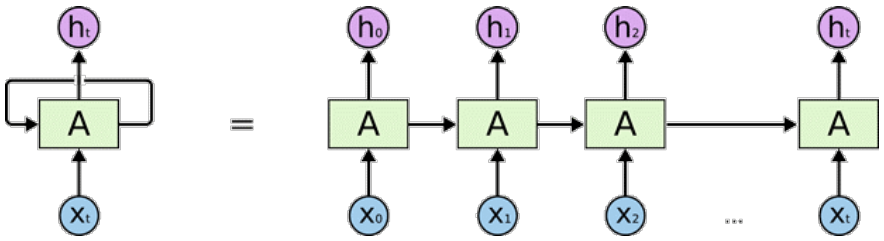


Рис. 10: Рекуррентный слой

На каждом шаге вычисляется также и выходное значение. Это значение на следующей итерации может быть использовано в качестве входа. Например, так можно заставить нейросеть по одному входному слову генерировать предложения.

Самые понятные примеры применения таких систем – распознавание спама или определение является ли отзыв положительным или отрицательным. В таком случае необходимо понять какую информацию несёт в себе текст. При том что количество слов может быть различным и смысл может быть разным от контекста применения того или иного выражения. Аналогично с текстом можно рассуждать и о последовательных данных временного ряда – нужно уловить «смысл» этого ряда. Сами значения элементов

последовательности могут оказаться не так важны, как их относительное расположение, изменение величин со временем, периодичности и так далее. Как в тексте смысл слова зависит от контекста, так и для временного ряда имеет значение «контекст», в котором находятся значения измеряемых величин.

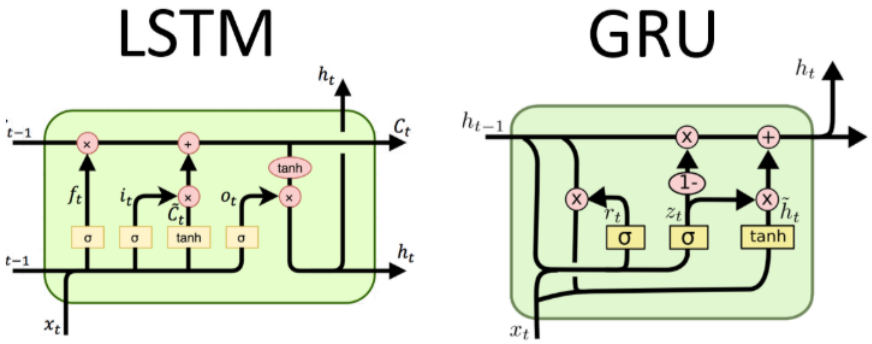


Рис. 11: Схемы рекуррентных слоёв LSTM и GRU

Чаще всего используются две реализации рекуррентных нейронных сетей: GRU [7] (Gated Recurrent Units) и LSTM [8, 9] (Long Short-Term Memory), см. рисунок 11. Несмотря на то, что LSTM обучается чуть дольше, его используют чаще, так как в нём лучше реализовано «запоминание» информации, которая была давно. GRU быстрее «забывает» старые данные, отдавая больший приоритет новым.

1.4 Обзор литературы

В последние годы публикуется довольно много статей по классификации временных рядов. В них приводятся сравнения разных моделей, а также результаты экспериментов с новыми моделями, которые зачастую тестируются на данных UEA & UCR Time Series Classification Repository [15] (далее - UCR/UEA).

Например, в статье [16] исследователи получили, что ResNet [17] превосходит другие модели на большинстве датасетов (наборов данных), однако для классификации ЭКГ лучше себя показал FCN [18].

Также интересны статьи про модели BOSS [19] (Bag-of-SFA-Symbols) и семейство COTE [20–22] (Collective of Transformation-based Ensembles).

Они упомянуты на сайте архива UCR/UEA как демонстрирующие наибольшую точностью для датасетов ECG200 и ECG5000. Однако это модели, не относящиеся к нейронным сетям.

Ещё одна предложенная архитектура – LSTM-FCN [23] (Long Short Term Memory Fully Convolutional Network) – показывает хорошие результаты на множестве различных датасетов. Хотя в приведённых результатах эта модель уступает на единственном представленном датасете с ЭКГ.

1.5 Обзор используемых данных

В работе были использованы 9 датасетов из архива UCR/UEA. Два из них многоканальные, остальные – одноканальные. Данный набор датасетов очень удобен в использовании по сравнению с другими, потому что данные в нём одинаковой длины, хранятся в едином формате, который легко прочитать из файлов. Также, в этих датасетах явно прописаны классы, соответствующие каждому объекту данных.

Эти 9 датасетов полезны в разной степени. Только три из них разбиты по классам именно по заболеваниям – это ECG200, ECG5000 и AtrialFibrillation. Также обратим внимание на датасеты с самой большой тестовой выборкой – это ECG5000 и два датасета NonInvasiveFetalECGThorax. Это позволяет иметь бо́льшую уверенность в устойчивости получаемых результатов. У этих же датасетов к тому же наибольшие обучающие выборки. Ниже представлена сводная таблица 2 с краткой информацией по каждому датасету. А в таблице 3 указаны принципы разбивок на классы.

Таблица 2: Параметры использованных датасетов

датасет	размер обучающей выборки	размер тестовой выборки	число каналов	длина ряда	количество классов
CinCECGTorso	40	1380	1	1639	4
ECG200	100	100	1	96	2
ECG5000	500	4500	1	140	5
ECGFiveDays	23	861	1	136	2
NonInvasiveFetalECGThorax1	1800	1965	1	750	42
NonInvasiveFetalECGThorax2	1800	1965	1	750	42
TwoLeadECG	23	1139	1	82	2
AtrialFibrillation	15	15	2	640	3
StandWalkJump	12	15	4	2500	3

Таблица 3: Разбивка датасетов на классы

датасет	классы
CinCECGTorso	4 разных человека
ECG200	нормальное сердечное сокращение и сокращение при инфаркте миокарда
ECG5000	нормальное сердечное сокращение, преждевременное сокращение желудочков «R на T» вида, преждевременное сокращение желудочков, суправентрикулярное преждевременное сокращение и неклассифицированное сокращение
ECGFiveDays	2 различные даты, в которые прводились измерения у одного пациента
TwoLeadECG	2 разных отведения
AtrialFibrillation	мерцательная аритмия, которая (1) не заканчивается минимум час после измерений, (2) не заканчивается минимум минуту после измерений, (3) заканчивается в течение секунды после измерений
StandWalkJump	разные активности человека - стоит, бежит, прыгает

1.6 Источники протестированных сетей

Реализации сетей для первоначального анализа были взяты с нескольких источников, они указаны в таблице 4.

Таблица 4: Источники сетей

hfawaz	lxdv	Luc Nies	tsai		
fcn_dl4tsc	EcgResNet34	lstms	FCN	ResCNN	TSTPlus
mlp_dl4tsc	ZolotyhNet		FCNPlus	ResNet	XceptionTime
resnet_dl4tsc	HeartNet1D		TCN	ResNetPlus	XceptionTimePlus
tlenet	HeartNet2D		InceptionTime	RNN	XCM
encoder			InceptionTimePlus	RNNPlus	XCMPlus
mcdcnn			MLP	RNN_FCN	XResNet1d
cnn			gMLP	RNN_FCNPlus	XResNet1dPlus
inception			mWDN	TransformerModel	TSPerceiver
			OmniScaleCNN	TST	TSiTPlus

hfawaz – псевдоним пользователя GitHub. Был использован его код, сопровождавший исследование по временным рядам [24]. Изначально была идея взять этот код за основу проекта, но в итоге вся структура была почти полностью переделана. Были сохранены архитектуры модели и некоторые элементы сбора статистики.

lxdw – также псевдоним на GitHub. Представленный в [25] код является частью магистерской выпускной квалификационной работы по свёрточным сетям применительно к ЭКГ.

Luc Nies в [26] реализовал простую рекуррентную сеть, предназначавшуюся для другого датасета ЭКГ. Вызвало интерес, сможет ли такая простая модель показать высокие результаты. Более того, была предпринята попытка использовать датасет, задействованный в данной реализации, а также схожие с ним. Однако данные в этих датасетах оказались неудобными для обработки и обучения моделей. Было принято решение остановиться на датасетах, описанных в пункте 1.5.

tsai – крупный репозиторий [27], в котором реализовано множество современных моделей для классификации временных рядов. Эти модели оказались самыми удобными в использовании на практике, встраивание их в проект оказалось наиболее простым.

2 Постановка задачи

Целью данной работы является конструирование нейронной сети для многоклассовой классификации временных рядов ЭКГ.

Формально задача многоклассовой классификации формулируется так:

X – множество объектов,

C – конечное множество возможных меток классов объектов.

Существует неизвестное отображение $\hat{f} : X \to C$.

При этом на конечном наборе данных X_{train} (обучающая выборка) известны значения $\hat{f}(x_i) = y_i$, где $x_i \in X_{train}$, а $y_i \in C$.

Требуется создать алгоритм, который может классифицировать любой объект $x \in X$.

Конкретно для временных рядов X имеет вид $\mathbb{R}^{(L \times T)}$, где L – число каналов (для ЭКГ это количество отведений), а T – количество измерений величины (разности потенциалов между разными точками тела) во времени.

Для создания архитектур сетей-кандидатов будут проведены эксперименты с нейронными сетями из пункта 1.6. Лучшие модели будут взяты за основу. Качество моделей будет сравниваться по метрикам *Accuracy* и F_2.

3 Эксперименты

3.1 Программная реализация

В рамках данной работы под единый интерфейс были объединены 40 моделей нейронных сетей. Они были реализованы разными людьми, с разным интерфейсом, форматом данных, на разных библиотеках [28–30]. Многие их этих моделей необходимо было предварительно адаптировать для комфортного использования. С этой целью был реализован родительский класс и все модели были переработаны или завёрнуты в класс-прослойку для расширения этого класса. Таким образом был достигнут единый интерфейс работы со всеми моделями. При этом есть возможность масштабирования для новых моделей.

Помимо унификации интерфейса была реализована возможность эти самые единообразные модели запускать. В одном файле, который условно можно назвать конфигурационным, выбираются настройки запуска, а также модели и датасеты, на которых необходимо провести эксперимент. В этом же файле можно задать какие модели будут входить в ансамбль.

Все измеренные данные сохраняются в файлы. Сюда входят матрицы несоответствий (можно увидеть пример в пункте 3.3.4), графики процесса обучения и история этого процесса, метрики и время обучения автоматически сводятся в один общий Excel-файл. Кроме этого, сохраняются также веса обученных моделей. Это позволяет, например, измерять качество работы ансамблей (пункт 3.4) не обучая заново каждый классификатор, а переиспользовав веса уже исследованных моделей.

Большая часть экспериментов проводилась на графических картах в сервисе Google Colaboratory, меньшая - на кластере МФТИ. Было измерено время обучения исходных моделей и их модификаций. Эти данные вместе с исходным кодом можно найти в GitHub репозитории [31]. Всего на обучение моделей было потрачено более 12ти суток.

3.2 Сводные таблицы

f2	ECG5000	ECG200	AtrialFibrillation	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	TwoLeadECG	StandWalkJump	CinCECGTorso	ECGFiveDays
mlp_dl4tsc	0,549	0,862	0,369	0,742	0,818	0,677	0,318	0,835	0,912
cnn	0,557	0,752	0,318	0,859	0,909	0,926	0,332	0,776	0,883
encoder	0,554	**0,931**	0,494	0,942	0,944	0,942	**0,622**	0,900	0,995
fcn_dl4tsc	0,505	0,870	0,406	0,954	0,945	**1,000**	0,238	0,801	0,986
resnet_dl4tsc	0,551	0,866	0,238	0,957	0,951	**1,000**	0,238	0,797	**1,000**
inception	**0,603**	0,884	0,329	**0,962**	0,954	0,999	0,350	0,816	**1,000**
tlenet	0,175	0,449	0,238	0,003	0,003	0,417	0,238	0,156	0,416
mcdcnn	0,548	0,861	0,466	0,923	0,920	0,961	0,558	0,919	0,969
lstms	0,407	0,848	0,333	0,729	0,787	0,664	0,538	0,812	0,928
FCN	0,461	0,866	0,238	0,935	0,934	0,999	0,238	0,782	0,983
FCNPlus	0,498	0,866	0,238	0,951	0,939	0,999	0,202	0,711	0,990
TCN	0,419	0,809	0,238	0,855	0,872	0,648	0,238	0,479	0,990
InceptionTime	0,527	0,874	0,238	0,948	0,953	0,996	0,238	0,786	**1,000**
InceptionTimePlus	0,519	0,887	0,238	0,947	0,948	0,995	0,318	0,710	**1,000**
MLP	0,389	0,879	0,315	0,890	0,912	0,855	0,302	0,779	0,976
gMLP	0,539	0,783	0,238	0,860	0,874	0,814	0,238	0,921	0,880
mWDN	0,529	0,887	0,238	0,871	0,869	0,962	0,238	0,904	0,964
OmniScaleCNN	0,537	0,853	0,196	0,931	0,937	0,999	0,342	0,839	0,998
ResCNN	0,395	0,857	0,238	0,958	0,947	0,998	0,156	0,708	0,978
ResNet	0,536	0,840	0,241	0,946	0,942	0,999	0,238	0,644	0,983
ResNetPlus	0,554	0,880	0,245	0,938	0,943	0,996	0,303	0,639	0,966
RNN	0,494	0,800	0,320	0,495	0,643	0,680	**0,622**	0,314	0,870
RNNPlus	0,489	0,814	**0,603**	0,501	0,677	0,715	0,196	0,353	0,900
RNN_FCN	0,550	0,853	0,318	0,956	0,940	0,999	0,312	0,665	0,992
RNN_FCNPlus	0,516	0,840	0,401	0,949	0,935	0,999	0,368	0,672	0,995
TransformerModel	0,398	0,710	0,346	0,556	0,561	0,903	0,364	0,510	0,794
TST	0,562	0,809	0,303	0,842	0,855	0,933	0,373	0,940	0,994
TSTPlus	0,564	0,866	0,196	0,887	0,922	0,910	0,364	0,924	0,944
XceptionTime	0,388	0,884	0,238	0,954	**0,958**	0,999	0,469	0,795	0,992
XceptionTimePlus	0,391	0,836	0,202	0,955	0,955	0,999	0,317	0,862	0,992
XCM	0,545	0,866	0,241	0,877	0,919	0,934	0,238	0,794	**1,000**
XCMPlus	0,599	0,874	0,318	0,873	0,892	0,971	0,238	0,746	**1,000**
XResNet1d	0,459	0,874	0,245	0,948	0,946	0,877	0,467	0,797	0,879
XResNet1dPlus	0,445	0,857	0,238	0,949	0,954	0,936	0,202	0,825	0,878
TSPerceiver	0,311	0,701	0,238	0,073	0,557	0,975	0,432	0,397	0,803
TSiTPlus	0,488	0,814	0,238	0,002	0,002	0,574	0,493	0,324	0,779
EcgResNet34	0,335	0,859	0,238	0,849	0,910	0,914	0,145	0,294	0,947
ZolotyhNet	0,503	0,887	0,364	0,923	0,913	0,987	0,462	0,913	0,834
HeartNet1D	0,523	0,690	0,238	0,904	0,919	0,747	0,333	0,921	0,976
HeartNet2D	0,563	0,909	0,238	0,905	0,912	0,960	0,238	**0,981**	0,983

Таблица 5: Результаты измерений F_2

accuracy	ECG5000	ECG200	AtrialFibrillation	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	TwoLeadECG	StandWalkJump	CinCECGTorso	ECGFiveDays
mlp_dl4tsc	0,941	0,880	0,400	0,737	0,811	0,668	0,333	0,828	0,912
cnn	0,935	0,780	0,333	0,858	0,912	0,925	0,333	0,770	0,883
encoder	0,942	**0,940**	0,467	0,941	0,947	0,942	**0,600**	0,896	0,995
fcn_dl4tsc	0,939	0,880	0,400	0,956	0,948	**1,000**	0,333	0,800	0,986
resnet_dl4tsc	0,940	0,880	0,333	0,959	0,954	**1,000**	0,333	0,796	**1,000**
inception	0,942	0,880	0,333	**0,963**	0,956	0,999	0,333	0,816	**1,000**
tlenet	0,584	0,640	0,333	0,028	0,028	0,500	0,333	0,249	0,497
mcdcnn	0,940	0,870	0,467	0,923	0,921	0,961	0,533	0,916	0,969
lstms	0,923	0,860	0,333	0,726	0,784	0,662	0,533	0,804	0,927
FCN	0,928	0,880	0,333	0,938	0,938	0,999	0,333	0,780	0,983
FCNPlus	0,930	0,880	0,333	0,954	0,944	0,999	0,267	0,705	0,990
TCN	0,928	0,830	0,333	0,854	0,871	0,644	0,333	0,494	0,990
InceptionTime	0,937	0,880	0,333	0,950	0,955	0,996	0,333	0,783	**1,000**
InceptionTimePlus	0,922	0,890	0,333	0,948	0,950	0,995	0,333	0,703	**1,000**
MLP	0,927	0,890	0,333	0,890	0,913	0,854	0,333	0,773	0,976
gMLP	**0,943**	0,800	0,333	0,859	0,874	0,810	0,333	0,920	0,880
mWDN	0,932	0,890	0,333	0,868	0,864	0,962	0,333	0,901	0,964
OmniScaleCNN	0,931	0,860	0,267	0,931	0,938	0,999	0,333	0,839	0,998
ResCNN	0,848	0,870	0,333	0,961	0,951	0,998	0,200	0,703	0,978
ResNet	0,884	0,860	0,267	0,949	0,946	0,999	0,267	0,631	0,983
ResNetPlus	0,906	0,900	0,333	0,941	0,946	0,996	0,333	0,632	0,966
RNN	0,921	0,820	0,333	0,490	0,645	0,680	**0,600**	0,354	0,870
RNNPlus	0,910	0,820	**0,600**	0,497	0,675	0,711	0,267	0,328	0,899
RNN_FCN	0,937	0,860	0,333	0,958	0,945	0,999	0,333	0,659	0,992
RNN_FCNPlus	0,931	0,860	0,400	0,950	0,938	0,999	0,400	0,667	0,995
TransformerModel	0,888	0,760	0,333	0,542	0,549	0,903	0,400	0,491	0,784
TST	0,922	0,830	0,333	0,839	0,854	0,933	0,400	0,940	0,994
TSTPlus	0,933	0,870	0,267	0,887	0,923	0,910	0,400	0,923	0,944
XceptionTime	0,924	0,900	0,333	0,955	**0,960**	0,999	0,467	0,794	0,992
XceptionTimePlus	0,931	0,860	0,267	0,957	0,957	0,999	0,333	0,861	0,992
XCM	0,934	0,870	0,267	0,877	0,920	0,933	0,333	0,788	**1,000**
XCMPlus	0,939	0,880	0,333	0,873	0,893	0,971	0,333	0,736	**1,000**
XResNet1d	0,891	0,880	0,333	0,949	0,950	0,872	0,467	0,796	0,879
XResNet1dPlus	0,932	0,870	0,333	0,951	0,956	0,936	0,267	0,820	0,878
TSPerceiver	0,771	0,760	0,333	0,080	0,559	0,975	0,467	0,397	0,801
TSiTPlus	0,927	0,820	0,333	0,021	0,021	0,541	0,467	0,318	0,777
EcgResNet34	0,815	0,850	0,333	0,849	0,913	0,912	0,133	0,310	0,947
ZolotyhNet	0,935	0,900	0,400	0,924	0,915	0,987	0,467	0,912	0,834
HeartNet1D	0,880	0,590	0,333	0,906	0,921	0,746	0,333	0,920	0,976
HeartNet2D	0,938	0,910	0,333	0,906	0,915	0,960	0,333	**0,980**	0,983

Таблица 6: Результаты измерений *Accuracy*

В таблице 5 собраны значения метрики F_2, а в таблице 6 – *Accuracy*. Лучшие результаты на каждом датасете выделены жирным, топ-10 подсвечены. Цвет подсветки обозначает приоритет датасета, это обсуждалось в пункте 1.5.

На основе этих измерений с учётом важности каждого датасета (описанной в 1.5) определены модели, представляющий наибольший интерес. Они выделены жирным. Критерий отбора был следующим: два из трёх

первых столбцов подсвечены или три из пяти первых столбцов подсвечены.

По метрике F_2 лучше других себя показали модели: `encoder`, `fcn_dl4tsc`, `resnet_dl4tsc`, `inception`, `ResNetPlus`, `XceptionTime`, `XCMPlus`, `ZolotyhNet`, `HeartNet2D`.

По метрике *Accuracy* из этого списка выбыл `ResNetPlus` и добавились `mlp_dl4tsc`, `mcdcnn`, `InceptionTime`.

3.3 Попытки улучшить архитектуру

Были предприняты попытки изменить архитектуру некоторых сетей для улучшения метрик.

Для этого были выбраны следующие модели: `encoder`, `inception`, `ZolotyhNet`. Первые две из них очень хорошо себя показали на важных для нас датасетах. Последняя же, хоть и была выделена жирным в обеих таблицах, показала себя хуже других конкурентов. Однако эта сеть отличилась скоростью работы (полные таблицы результатов с указанием времени обучения доступны в GitHub репозитории [31]).

3.3.1 Encoder

Данная модель основана на [32].

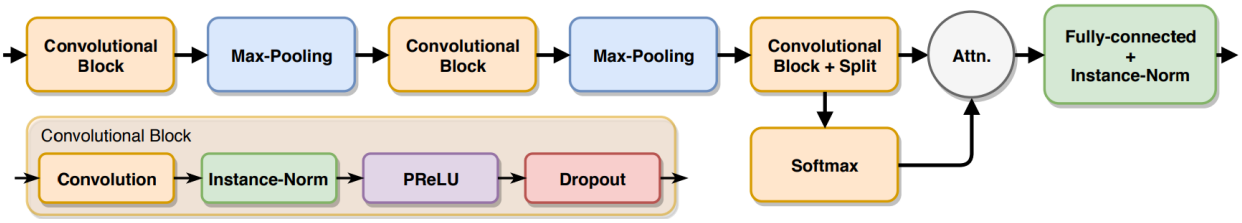


Рис. 12: Структура модели `encoder`

Были реализованы следующие вариации этой сети:

- слои Max-Pooling удалены, уменьшение размерности перенесено на этап свёртки (ядро применяется с шагом 2),
- слои Max-Pooling заменены на Convolutional Block с ядром 2x2 и шагом применения 2,

- слой Instance-Norm заменён на Batch-Norm,
- добавлено ещё одно повторение Convolutional Block + Max-Pooling.

3.3.2 Inception

Данная модель основана на [33].

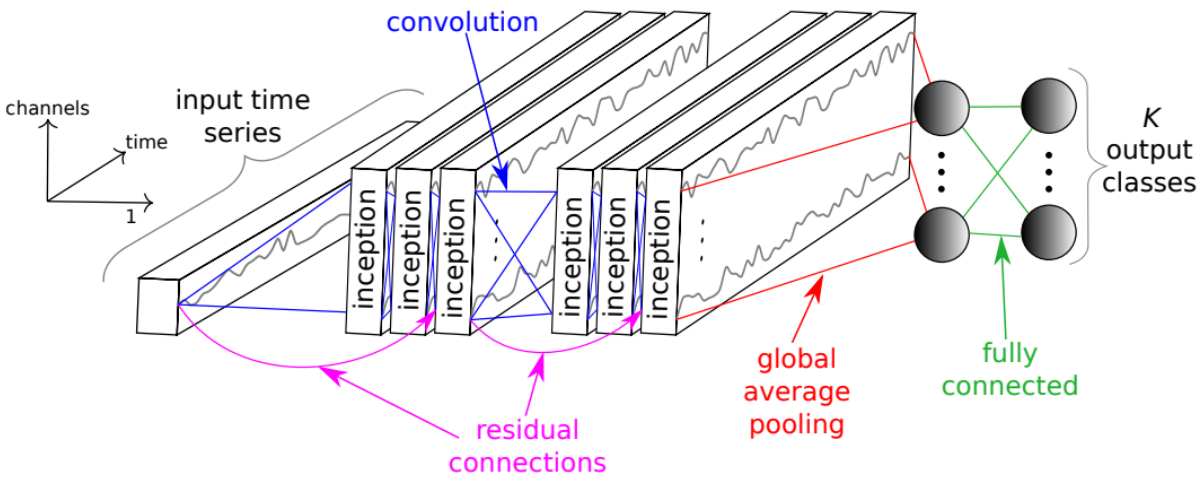


Рис. 13: Структура модели `inception`

Были реализованы следующие вариации этой сети:

- количество Inception блоков увеличено с 6ти до 8ми (по 4 в группе),
- количество Inception блоков увеличено с 6ти до 9ти (по 3 в группе),
- слои Max-Pooling внутри Inception блоков заменены на свёртки с ядром 2x2 и шагом применения 2,
- слой global average pooling заменён на свёртку.

3.3.3 Zolotyh

Данная модель основана на [34].

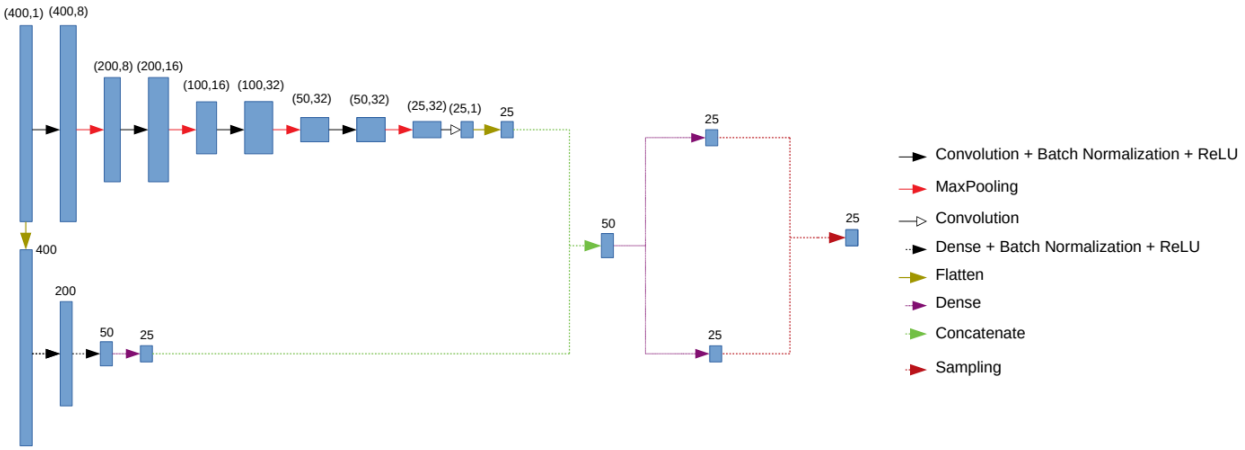


Рис. 14: Структура модели `ZolotyhNet`

Были реализованы следующие вариации этой сети:

- в «верхнюю» часть сети добавлен ещё один блок вида Convolution + Batch-Norm + ReLU + MaxPooling,
- добавлена третья «подсеть», являющаяся «средним» между двумя имеющимися,
- слои Max-Pooling заменены на Convolution + Batch-Norm + ReLU с ядром свёртки 2x2 и шагом применения 2.

3.3.4 Результаты изменений

Результаты экспериментов с изменёнными моделями представлены в таблице 7.

	ECG5000		ECG200		AtrialFibrillation		NonInvasive1		NonInvasive2				
model	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy			
ENCODER_ORIG	**0,567**	**0,943**	0,891	0,900	0,330	**0,333**	**0,947**	**0,947**	0,937	**0,940**	**3**	6	2
ENCODER_NO_POOLING	0,539	**0,943**	0,900	0,910	**0,469**	**0,467**	0,938	0,938	0,932	0,933	**2**	3	1
ENCODER_CONV_INSTEAD_POOLING	0,553	**0,943**	**0,918**	**0,930**	**0,336**	**0,333**	0,925	0,926	**0,938**	**0,940**	**2**	**7**	2
ENCODER_BATCH_NORM	0,563	0,938	**0,913**	**0,920**	0,238	**0,333**	0,661	0,635	0,620	0,609	0	3	0
ENCODER_EXTRA_LAYER	**0,564**	**0,944**	0,909	**0,920**	0,311	**0,333**	**0,941**	**0,941**	**0,940**	**0,942**	**3**	**8**	**4**
INCEPTION_ORIG	**0,605**	**0,949**	**0,892**	0,890	0,196	0,267	0,952	0,952	**0,956**	**0,959**	**3**	**5**	3
INCEPTION_EXTRA_LAYER	**0,614**	0,940	**0,913**	**0,920**	0,296	0,333	0,951	0,952	**0,953**	**0,956**	**3**	**5**	3
INCEPTION_BIG_STEP	0,568	0,940	0,887	**0,900**	0,444	**0,467**	**0,953**	**0,954**	0,951	0,954	1	**4**	1
INCEPTION_CONV_INSTEAD_POOLING	0,552	**0,942**	0,778	0,800	**0,477**	**0,467**	**0,955**	**0,957**	0,952	0,955	**4**	**5**	**4**
INCEPTION_CONV_INSTEAD_GAP	0,432	0,926	0,859	0,850	**0,477**	**0,467**	0,002	0,018	0,002	0,018	2	2	0
ZolotyhNet_ORIG	**0,511**	**0,928**	**0,880**	**0,900**	**0,238**	**0,333**	**0,926**	**0,927**	0,925	0,927	**5**	**8**	3
ZolotyhNet_CONV_INSTEAD_POOLING	**0,442**	0,924	0,831	0,840	0,208	0,267	0,893	0,892	0,892	0,896	0	1	0
ZolotyhNet_EXTRA_LAYER	**0,442**	**0,930**	0,848	0,860	**0,238**	**0,333**	0,909	0,909	**0,933**	**0,935**	2	**6**	1
ZolotyhNet_EXTRA_SUBNET	0,437	0,927	**0,884**	**0,900**	**0,303**	**0,333**	**0,921**	**0,923**	**0,939**	**0,941**	**6**	**8**	**4**

Таблица 7: Результаты сравнения вариаций моделей

Полученные значения метрик были обработаны следующим образом. Сравнения проводились между модификациями каждой модели отдельно, то есть вариации различных моделей не конкурировали друг с другом. В данном случае, цель – выявить лучшую модификацию для каждой сети. В группах жирным выделены значения двух лучших метрик. Значение лучшей метрики выделено увеличенным шрифтом.

В двух отдельно стоящих столбцах подсчитаны количество записей в строке, выделенных крупным шрифтом, и количество записей в строке, выделенных жирным. Далее эти значения внутри групп размечены аналогично значениям метрик. В последнем столбце стоит количество баллов, определённое на основе двух столбцов по правилу: за значение, выделенное крупным шрифтом, даётся 2 балла, а за значение, выделенное жирным, -1 балл.

Таким образом, для каждой модели была определена модификация, набравшая наибольшее количеств баллов. Ими стали:

- `Encoder` с дополнительным повторением Convolutional Block + Max-Pooling,
- `Inception` со свёртками вместо Max-Pooling внутри Inception блоков,
- `ZolotyhNet` с третьей подсетью.

Ниже на рисунках 15 и 16 приведены сравнения матриц несоответствий для пар, где отличие будет наиболее заметно – Encoder для ECG200 и ECG5000. Для датасета ECG200 точность определения одного из классов значительно возросла при сохранении точности по второму.

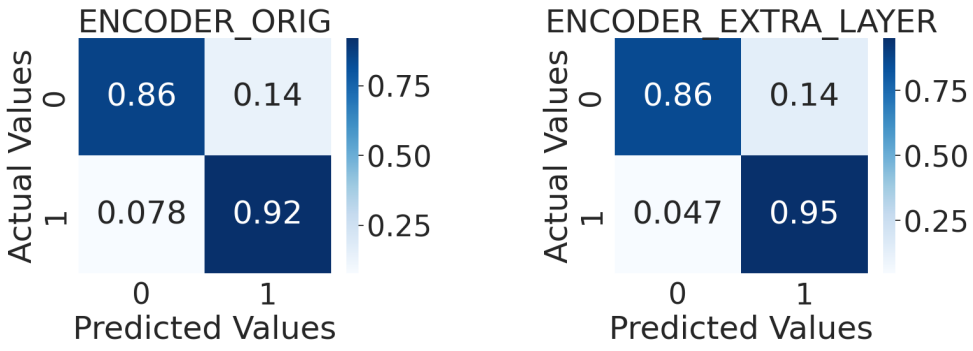


Рис. 15: Нормированные матрицы несоответствий для датасета ECG200

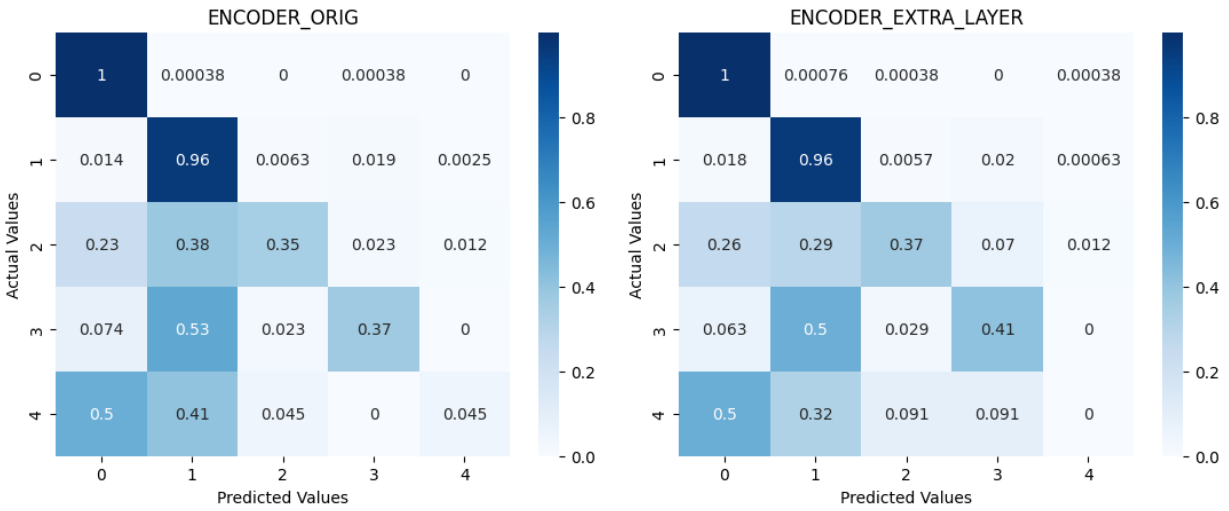


Рис. 16: Нормированные матрицы несоответствий для датасета ECG5000

Плохое качество работы сетей на 3х классах ECG5000 объясняется тем, что эти классы представлены гораздо меньшим числом объектов. Это можно увидеть на рисунке 17.

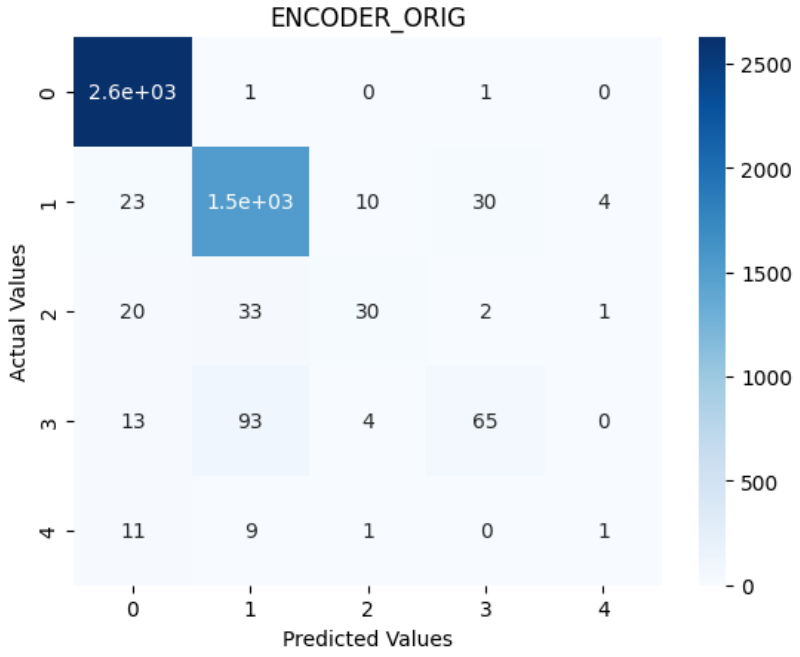


Рис. 17: Ненормированная матрица несоответствий для датасета ECG5000

3.4 Ансамбли

Ещё один способ улучшить результаты – объединить несколько моделей в одну (в ансамбль). В данном случае был реализован самый простой

вариант такого объединения. Каждая модель на выходе даёт распределение вероятностей по классам – её предсказания. Итоговый класс – это класс с наибольшей предсказанной вероятностью. Можно сложить предсказанные вероятности нескольких моделей и взять максимум от суммы. Это может улучшить метрики. Наиболее эффективно такой подход работает, если модели имеют разные внутренние структуры. Однако, в нашем случае большинство из них основаны на свёртках, но это всё равно может дать улучшение.

	ECG5000		ECG200		AtrialFibrillation		NonInvasive1		NonInvasive2	
model	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy
ENSEMBLE_f2_orig	0,579	0,948	**0,913**	**0,920**	**0,487**	**0,467**	**0,973**	**0,974**	0,968	0,970
ENSEMBLE_f2_mod	0,577	0,947	**0,913**	**0,920**	0,480	**0,467**	0,971	0,973	**0,969**	**0,972**
ENSEMBLE_acc_orig	0,549	0,946	0,904	0,910	0,425	0,400	0,969	0,970	0,967	0,969
ENSEMBLE_acc_mod	0,547	0,946	0,904	0,910	0,480	**0,467**	0,968	0,969	**0,969**	**0,972**
ENSEMBLE_encoder_inception_orig	**0,600**	**0,950**	0,900	0,900	0,359	0,333	0,963	0,964	0,962	0,965
ENSEMBLE_encoder_inception_mod	0,588	0,947	0,891	0,900	0,190	0,200	0,960	0,961	0,959	0,961
ENSEMBLE_encoder_inception_mcdcnn_orig	0,589	0,947	0,900	0,900	0,408	0,400	0,962	0,962	0,964	0,965
ENSEMBLE_encoder_inception_mcdcnn_mod	0,555	0,946	0,891	0,900	0,255	0,267	0,961	0,961	0,959	0,961
ENSEMBLE_encoder_fcn_mcdcnn_orig	0,546	0,943	0,904	0,910	0,406	0,400	0,960	0,961	0,958	0,961
ENSEMBLE_encoder_fcn_mcdcnn_mod	0,545	0,943	**0,913**	**0,920**	0,469	**0,467**	0,962	0,963	0,955	0,958
	inception	gMLP	encoder		RNNPlus		inception		XceptionTime	
best of models	0,603	0,943	0,931	0,940	0,603	0,600	0,962	0,963	0,958	0,960

Таблица 8: Результаты сравнения ансамблей

Ансамбли, названия которых заканчивается на «`orig`», состоят из оригинальных реализаций моделей. Ансамбли, заканчивающиеся на «`mod`», вместо `encoder`, `inception` и `ZolotyhNet` содержат в себе их лучшие модификации данных моделей, выделенные в пункте 3.3.4.

Ансамбли «`f2`» и «`acc`» включают в себя модели, выделенные как лучшие в пункте 3.2 для метрик F_2 и *Accuracy* соответственно. Остальные ансамбли содержат модели, указанные в названиях.

В таблице 8 указаны полученные метрики для ансаблей. Видно, что лучше всего себя показал ансамбль `f2_orig` – его можно считать оптимальным выбором, однако `encoder_inception_orig` для датасета ECG5000 подходит лучше. Для сравнения ниже приведены метрики, которые были получены оригинальными моделями, выбран лучший результат для каждой пары датасет-метрика. Как можно видеть, `f2_orig` удалось улучшить значения половины этих метрик. На датасетах NonInvasive *Accuracy* и F_2 увеличились примерно на процент, что является неплохим результатом. Красным отмечены значения, которые ансамблям не удалось превзойти.

	ECG5000		ECG200		AtrialFibrillation		NonInvasive1		NonInvasive2	
	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy	f2	accuracy
encoder	0,554	0,942	**0,931**	**0,940**	0,494	0,467	0,942	0,941	0,944	0,947
inception	**0,603**	0,942	0,884	0,880	0,329	0,333	0,962	0,963	0,954	0,956
gMLP	0,539	0,943	0,783	0,800	0,238	0,333	0,860	0,859	0,874	0,874
RNNPlus	0,489	0,910	0,814	0,820	**0,603**	**0,600**	0,501	0,497	0,677	0,675
XceptionTime	0,388	0,924	0,884	0,900	0,238	0,333	0,954	0,955	0,958	0,960
ENSEMBLE_f2_orig	0,579	**0,948**	0,913	0,920	0,487	0,467	**0,973**	**0,974**	**0,968**	**0,970**
ENSEMBLE_encoder_inception_orig	0,600	0,950	0,900	0,900	0,359	0,333	0,963	0,964	0,962	0,965

Таблица 9: Сравнения ансамблей с лучшими моделями

В таблице 9 приведено сравнение метрик ансамбля `f2_orig` и моделей, которые давали максимальные метрики. Также здесь представлен `encoder_inception_orig` как наиболее подходящий для ECG5000. Тут можно видеть, что нет идеальной модели для всех датасетов. Однако в среднем `f2_orig` показывает себя весьма хорошо. Ни одна изначальная модель не может этим похвастаться на различных датасетах, они хорошо подходят только к конкретным датасетам. Таким образом, можно сделать вывод, что на новом неизвестном датасете ансамбль `f2_orig` может быть хорошим начальным выбором.

4 Заключение

В данной работе были проведены множественные эксперименты с нейронными сетями в приложении к многоклассовой классификации одноканальных и многоканальных временных рядов ЭКГ. Произведены измерения качества сорока реализаций нейронных сетей на девяти наборах данных, подобраны оптимальные параметры, собраны и проанализированы результаты их работы. Выделены лидеры среди исследованных моделей, предложено и реализовано 11 вариантов модификаций, произведены измерения качества их работы. Также проведены эксперименты с десятью ансамблями различных наборов сетей и выявлен ансамбль, получающий метрики выше любой из протестированных одиночных сетей на некоторых датасетах при высоком среднем значении метрик на других датасетах.

Для проведения экспериментов был написан код на языке Python, позволяющий единообразно использовать модели, реализованные с применением разных библиотек. При этом запуск тестов можно осуществлять централизованно сразу на группе моделей и/или датасетов. Также реализована возможность конфигурировать некоторые параметры запуска, такие как количество собираемой статистики. Весь исходный код и всю собранную статистику можно найти в GitHub репозитории [31].

5 Список литературы

[1] *Ritchie, Hannah.* Causes of death. https://ourworldindata.org/causes-of-death.

[2] Число умерших по основным классам и отдельным причинам смерти за год. https://www.fedstat.ru/indicator/31620.

[3] *Strauss, Galen S Wagner; David G.* Marriott's practical electrocardiography. / Galen S Wagner; David G Strauss. — Philadelphia : Wolters Kluwer Health/Lippincott Williams & Wilkins, 2014.

[4] *Haykin, S.* Neural Networks: A Comprehensive Foundation / S. Haykin, S.S. Haykin, S.A. HAYKIN. International edition. — Prentice Hall, 1999. https://books.google.ru/books?id=bX4pAQAAMAAJ.

[5] Convolutional neural networks: an overview and application in radiology / Rikiya Yamashita, Mizuho Nishio, Richard K. G. Do, Kaori Togashi // *Insights into Imaging.* — 2018. — Vol. 9. — Pp. 611 – 629.

[6] *Abdulwahab, Saddam.* Deep Learning Models for Paraphrases Identification: Ph.D. thesis. — 2017. — 09.

[7] *Cho, Kyunghyun.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. — 2014.

[8] *Hochreiter, Sepp.* Long Short-term Memory / Sepp Hochreiter, Jürgen Schmidhuber // *Neural computation.* — 1997. — 12. — Vol. 9. — Pp. 1735–80.

[9] *Olah, Christopher.* Understanding LSTM Networks. — 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[10] *Chung, Junyoung.* Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. — 2014.

[11] *Jenkins, Dr Dean.* ECG Library. https://ecglibrary.com/ecghome.php.

[12] *Mazur, Matt.* A Step by Step Backpropagation Example. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

[13] Introduction to Computer Vision. https://faculty.cc.gatech.edu/~hays/compvision2017/proj6/.

[14] *Mahendran, Aravindh.* Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images / Aravindh Mahendran, Andrea Vedaldi. — 2015.

[15] *Anthony Bagnall Jason Lines, William Vickers.* The UEA & UCR Time Series Classification Repository. http://timeseriesclassification.com/dataset.php.

[16] Deep learning for time series classification: a review / Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber et al. — 2018.

[17] *He, Kaiming.* Deep Residual Learning for Image Recognition. — 2015.

[18] *Wang, Zhiguang.* Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. — 2016.

[19] *Schäfer, Patrick.* The BOSS is concerned with time series classification in the presence of noise / Patrick Schäfer // *Data Mining and Knowledge Discovery.* — 2015. — 11. — Vol. 29.

[20] Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles / Anthony Bagnall, Jason Lines, Jon Hills, Aaron Bostrom // *IEEE Transactions on Knowledge and Data Engineering.* — 2015. — 09. — Vol. 27. — Pp. 1–1.

[21] *Bagnall, Anthony.* A tale of two toolkits, report the third: on the usage and performance of HIVE-COTE v1.0. — 2020.

[22] *Middlehurst, Matthew.* HIVE-COTE 2.0: a new meta ensemble for time series classification. — 2021.

[23] Multivariate LSTM-FCNs for Time Series Classification / Fazle Karim, Somshubra Majumdar, Houshang Darabi, Samuel Harford. — 2018.

[24] Deep learning for time series classification: a review / Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber et al. // *Data Mining and Knowledge Discovery*. — 2019. — Vol. 33, no. 4. — Pp. 917–963.

[25] *А.Ю., Лящук*. ECG Arrhythmia classification. https://github.com/lxdv/ecg-classification.

[26] *Nies, Luc*. Using LSTMs to classify ECG signals in several different heart diseases. https://github.com/LucNies/ecg-classification.

[27] *Oguiza, Ignacio*. tsai - A state-of-the-art deep learning library for time series and sequential data. — Github. — 2022. https://github.com/timeseriesAI/tsai.

[28] Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // *Journal of Machine Learning Research*. — 2011. — Vol. 12. — Pp. 2825–2830.

[29] PyTorch: An Imperative Style, High-Performance Deep Learning Library / Adam Paszke, Sam Gross, Francisco Massa et al. // Advances in Neural Information Processing Systems 32 / Ed. by H. Wallach, H. Larochelle, A. Beygelzimer et al. — Curran Associates, Inc., 2019. — Pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-li
pdf.

[30] *Falcon, William*. PyTorch Lightning. — 2019. — 3. https://github.com/Lightning-AI/lightning.

[31] *Иван, Филиппов*. Исходный код пректа настоящей работы. https://github.com/FilippovIvan19/bachelor_paper.

[32] *Serrà, Joan*. Towards a universal neural network encoder for time series. — 2018.

[33] InceptionTime: Finding AlexNet for Time Series Classification / Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier et al. — 2019.

[34] *Kuznetsov, V. V.* Electrocardiogram Generation and Feature Extraction Using a Variational Autoencoder. — 2020.

[35] *Ф.М. Гафаров, А.Ф. Галимянов.* Искусственные нейронные сети и приложения: учеб. пособие / А.Ф. Галимянов Ф.М. Гафаров. — Казань: Изд-во Казан. ун-та, 2018. — P. 121.

[36] Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky et al. // *Journal of Machine Learning Research.* — 2014. — 06. — Vol. 15. — Pp. 1929–1958.