«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	01.03.02 Прикладная математика и информатика
Профиль	Без профиля
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой Кринкин К.В.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ТЕМА: Применение нейросетевых автокодировщиков для выделения признаков из серии спектров флуоресценции

Студент		_____ *подпись*	Гоголев Е.Е.
Руководитель	к.т.н., доцент	_____ *подпись*	Геппенер В.В.
Консультанты	ассистент	_____ *подпись*	Жангиров Т.Р.
		_____ *подпись*	Житенева М.И.
	к.т.н.	_____ *подпись*	Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой МО ЭВМ

_____ Кринкин К.В.

«___»_____2022 г.

Студент Гоголев Е.Е. Группа 8383

Тема работы: Применение нейросетевых автокодировщиков для извлечения признаков из серии спектров флуоресценции

Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

Исходные данные (технические требования):
ОС Windows, Python, Jupyter Notebook

Содержание ВКР:
Обзор предметной области, Материалы и методы, Описание решения, Экономическое обоснование ВКР.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал.

Дополнительные разделы: Экономическое обоснование ВКР.

Дата выдачи задания «20»__апреля_____2022 г.

Дата представления ВКР к защите «20»___июня_____2022 г.

Студент _____ Гоголев Е.Е.

Руководитель *(к.т.н., доцент)* _____ Геппенер В.В.

Консультант *(ассистент)* _____ Жангиров Т.Р.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___»_____2022 г.

Студент Гоголев Е.Е. Группа 8383

Тема работы: Применение нейросетевых автокодировщиков для извлечения признаков из серии спектров флуоресценции

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	20.04 – 27.04
2	Построение моделей автокодировщиков	28.04 – 10.05
3	Выбор модели	11.05 – 14.05
4	Изучение применимости выделенных признаков	15.05 – 17.05
5	Изучение выделенных признаков	17.05 – 22.05
6	Оформление пояснительной записки	22.05 – 09.06
7	Оформление иллюстративного материала	10.06 – 16.06
8	Предзащита	17.06 – 20.06

Студент(ка) _____ Гоголев Е.Е.

Руководитель _____ Геппенер В.В.
(к.т.н., доцент)

Консультант _____ Жангиров Т.Р.
(ассистент)

РЕФЕРАТ

Пояснительная записка 67 стр., 35 рис., 5 табл., 9 ист.

Тема выпускной квалификационной работы – «Применение нейросетевых автокодировщиков для извлечения признаков из серии спектров флуоресценции».

Главной целью данной работы является выделить признаки из спектров флуоресценции, чтобы их можно было в дальнейшем использовать в различных прикладных задачах. Данная цель будет решаться при помощи особой архитектуры искусственных нейронных сетей – автокодировщика.

Для выполнения данной цели были построены различные модели автокодировщика, изучено влияние параметров моделей на качество их работы, выявлена оптимальная модель, изучены выделенные признаки и возможность их применения в прикладных задачах. Также приведены расчёты экономического обоснования, которые показывают целесообразность воплощения данного проекта.

В результате получена модель автокодировщика, способная автоматически выделять полезные для решения экологических задач признаки, что облегчит работу экологов.

ABSTRACT

The topic of this thesis is "Application of neural network autoencoders to extract features from a series of fluorescence spectra".

The main goal of this work is to extract features from fluorescence spectra, so they can be further used in various applications. This goal will be solved with the help of a special architecture of artificial neural networks - autoencoder.

To achieve this goal, we have built different models of autoencoder, studied the influence of model parameters on the quality of their work, found the optimal model, studied the selected features and the possibility of their use in applied tasks. Also, calculations of economic justification, which show the expediency of the embodiment of this project are given.

As a result, an autoencoder model was obtained, capable of automatically selecting signs useful for solving environmental problems, which will facilitate the work of ecologists.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

PCA	Метод главных компонент (англ. principal component analysis) — один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации.
DAE	Денуасифицирующий автокодировщик (denoising autoencoder) – модель автокодировщика, которая принимает частично испорченный входной сигнал и обучаются восстанавливать исходный неискаженный входной сигнал.
CAE	Сверточный автокодировщик (convolutional autoencoder) расширяет базовую структуру обыкновенного автокодировщика путем замены полностью связанных слоев на слои свертки. Как и в простом автокодировщике, размер входного слоя такой же, как и выходных слоев, но сеть кодировщика меняется на слои свертки.
ReLU	Rectified Linear Unit — это наиболее часто используемая функция активации при глубоком обучении. Данная функция возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число.
Dropout	Исключение или дропаут — метод регуляризации искусственных нейронных сетей, предназначен для уменьшения переобучения сети за счет предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения.

ВВЕДЕНИЕ

Для определения загрязнения водоемов биологическими методами чаще всего используются данные о видовом многообразии организмов, формирующих сообщества. Самыми важными из них являются цианобактерии, вызывающие цветения открытых водоемов. В связи с этим возникает задача анализа цианобактерий. Самый удобный способ их анализа – спектрофотометрические методы, являющиеся быстрыми и дешевыми. При этом, порождаемые ими серии спектров собственной флуоресценции цианобактерий имеют довольно большую размерность, что затрудняет обработку этих данных, в частности, их классификацию.

Исходя из этого, выделение признаков из серий спектров флуоресценции является важной и актуальной проблемой в области экологии.

Главной целью данной работы является выделить признаки из спектров флуоресценции, чтобы их можно было в дальнейшем использовать в различных прикладных задачах. Данная цель будет решаться при помощи особой архитектуры искусственных нейронных сетей – автокодировщика.

Для выполнения данной цели потребуется построить различные модели автокодировщика, изучить влияние параметров моделей на качество их работы, выявить оптимальную модель, изучить выделенные признаки и возможность их применения в прикладных задачах.

В результате мы должны получить модель автокодировщика, способную автоматически выделять полезные для решения экологических задач признаки, что облегчит работу экологов.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Искусственные нейронные сети

Изучение и использование искусственных нейронных сетей, в принципе, началось давно – в начале 20 века, но по-настоящему широкое распространение они получили позже. Это связано в первую очередь с тем, что стали появляться передовые (для того времени) вычислительные устройства, достаточно мощные для работы с искусственными нейронными сетями.

В настоящее время любой персональный компьютер может легко смоделировать нейронную сеть средней сложности.

Нейронная сеть – это набор нейронов, соединенных друг с другом определенным образом. Нейрон – это элемент, который вычисляет выходной сигнал (по определенному правилу) из набора входных сигналов. Другими словами, основная последовательность действий одного нейрона выглядит следующим образом:

- Прием сигналов от предыдущих элементов сети
- Комбинирование входных сигналов
- Вычисление выходного сигнала
- Передача выходного сигнала следующим элементам нейронной сети

Нейроны могут быть соединены друг с другом абсолютно разными способами, это определяется структурой конкретной сети. Но суть нейронной сети всегда остается неизменной. Комбинация сигналов, поступающих на вход сети, порождает выходной сигнал (или несколько выходных сигналов) на выходе.

Каждое соединение в нейронной сети может быть охарактеризовано тремя факторами:

- элемент, от которого исходит связь
- элемент, к которую связь направлена
- вес связи.

Вес связи определяет, будет ли усилен или ослаблен сигнал, передаваемый по данной связи. Выход нейрона можно представить следующим образом:

$$net_j = \sum_{i=1}^{N} x_i * w_{ij}$$

Где $netj$ – это результат комбинирования всех входных сигналов для нейрона j (комбинированный ввод нейрона), N – количество элементов, передающих свои выходные сигналы на вход сигнала j, w_{ij}– вес связи, соединяющей нейрон i с нейроном j.

При суммировании всех взвешенных входных сигналы, получается комбинированный вход элемента сети.

Формальную модель нейрона можно представить следующим образом (рис. 1):

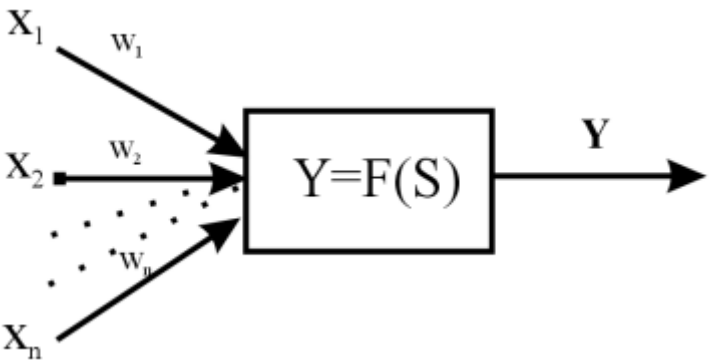

Рисунок 1 - Формальная модель нейрона

Здесь введены следующие обозначения:

- X1, X2, …, Xn - входной сигнал (паттерн)
- w1, w2,…, wn – весовые коэффициенты
- b - порог нейрона

Сначала нейрон вычисляет взвешенную сумму S, далее применяя функцию активации F(S) вычисляет выходной сигнал Y.

Функция активации нейрона – это функция, которая вычисляет выходной сигнал нейрона. Входом этой функции является сумма всех

произведений сигналов и весов этих сигналов. Наиболее часто используемые функции активации:

- Пороговая функция
- Линейный порог
- Сигмоидальная функция или сигмоида (sigmoid)
- Гиперболический тангенс (hyperbolic tangent, tanh).

1.2 Задача выделения признаков

В реальной жизни все полученные в ходе экспериментов данные в подавляющем большинстве случаев находятся в больших объемах. Чтобы понять эти данные, необходимо их обработать. Вручную обработать их невозможно. Именно здесь и возникает концепция извлечения признаков.

Извлечение признаков – это часть процесса уменьшения размерности, в ходе которого исходный набор необработанных данных разделяется и сокращается до более управляемых групп. Это упрощает обработку данных.

Наиболее важной характеристикой больших наборов данных является то, что они содержат большое количество переменных. Анализ с большим количеством переменных обычно требует много памяти и вычислительной мощности, и это может привести к тому, что алгоритмы задачи классификации будут переобучаться относительно обучающей выборки.

Извлечением признаков называют методы построения комбинаций переменных, позволяющих обойти эти проблемы и при этом описывающих данные с достаточной точностью.

Таким образом, методы извлечения признаков полезны, когда имеется большой набор данных и необходимо сократить его объем без потери важной или актуальной информации, и помогают:

- уменьшить количество избыточных данных
- избежать проблемы переобучения
- упростить архитектуру модели

1.3 Снижение размерности пространства

Избыточное количество информации может привести к снижению эффективности анализа данных. Существует даже термин "проклятие размерности", описывающий проблемы работы с высокоразмерными данными.

Неинформативные атрибуты являются источником дополнительного шума и влияют на точность оценки параметров модели. Кроме того, наборы данных с большим количеством атрибутов могут содержать группы коррелирующих переменных. Наличие таких групп признаков означает дублирование информации, что может повлиять на качество оценки ее параметров.

Можно выделить два направления в снижении размерности признакового пространства [2] по принципу используемых для этого переменных:

- отбор признаков из имеющегося исходного набора
- формирование новых признаков путем трансформации первоначальных данных.

Выбор информативных признаков заключается в нахождении наилучшего подмножества из множества всех исходных переменных.

Наилучшее подмножество задает либо наивысшее качество моделирования для заданной размерности пространства признаков, либо наоборот – наименьшую размерность данных, при которой возможно построение модели заданного качества. В идеальном случае сокращенное представление данных должно иметь размерность, соответствующую размерности, внутренне присущей данным (intrinsic dimensionality).

Прямое решение задачи создания наилучшей модели связано с перебором всех возможных комбинаций признаков, что обычно представляется чрезмерно трудоемким. Поэтому, как правило, используется прямой или обратный отбор признаков.

В процедурах прямого отбора переменные из исходного набора последовательно добавляются до тех пор, пока не будет достигнуто желаемое качество модели. Алгоритмы последовательного сокращения исходного пространства признаков (обратный отбор) предполагают постепенное удаление наименее информативных переменных до тех пор, пока информативность модели не снизится до приемлемого уровня.

Таким образом, процесс формирования нового признакового пространства обычно приводит к меньшему набору реально информативных переменных. На их базе может быть построена более качественная модель как основанная на меньшем числе наиболее информативных признаков.

Рассмотрим основные способы выделения признаков.

1.3.1 Конструирование признаков

Конструирование признаков — это процесс использования предметной области данных для создания признаков, которые нужны для обучения машин. Конструирование признаков является фундаментом для приложений машинного обучения, а также процессом трудным и затратным. Необходимости ручного конструирования признаков можно избежать при автоматизации прикладного обучения признакам.

Данный способ может быть представлен следующим образом:

- Метод мозгового штурма или проверка признаков;
- Решение, какие признаки создавать;
- Создание признаков;
- Проверка, какие признаки работают с моделью;
- Улучшение признаков, если требуется;
- Возврат к методу мозгового штурма/создание других признаков, пока работа не будет завершена.

1.3.2 Метод главных компонент

Метод главных компонент [2, с. 1] — это технология многомерного статистического анализа, используемая для сокращения размерности пространства признаков с минимальной потерей полезной информации.

С математической точки зрения метод главных компонент представляет собой ортогональное линейное преобразование, которое отображает данные из исходного пространство признаков в новое пространство меньшей размерности.

При этом первая ось новой системы координат строится таким образом, чтобы дисперсия данных вдоль неё была бы максимальна. Вторая ось строится ортогонально первой так, чтобы дисперсия данных вдоль неё, была бы максимальной их оставшихся возможных и т.д. Первая ось называется первой главной компонентой, вторая — второй и т.д.

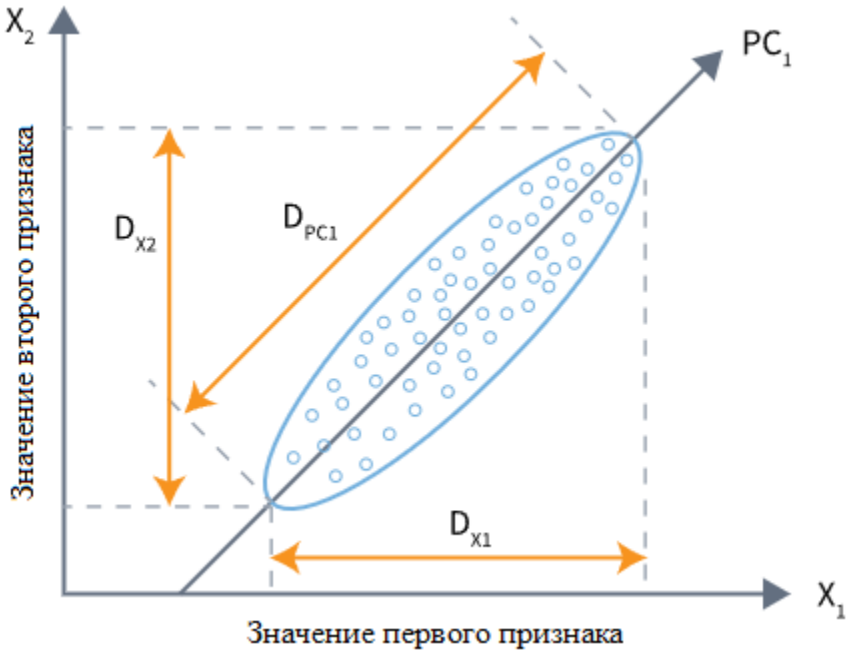

Рисунок 2 - Пример PCA

На рисунке 2 показано снижение размерности исходного 2-мерного пространства с помощью метода главных компонент до 1-мерного. Первая главная компонента ориентирована вдоль направления наибольшей

вытянутости эллипсоида рассеяния точек объектов исходного набора данных в пространстве признаков, т.е. с ней связана наибольшая дисперсия.

Задача метода главных компонент заключается в том, чтобы построить новое пространство признаков меньшей размерности, дисперсия между осями которой будет перераспределена так, чтобы максимизировать дисперсию по каждой из них.

Метод главных компонент применим всегда. Однако метод не всегда эффективно снижает размерность при заданных ограничениях на заданную точность. Прямые и плоскости не всегда обеспечивают хорошую аппроксимацию. Например, данные могут с хорошей точностью следовать какой-нибудь кривой, а эта кривая может быть сложно расположена в пространстве данных. В этом случае метод главных компонент для приемлемой точности потребует нескольких компонент (вместо одной), или вообще не даст снижения размерности при приемлемой точности.

1.3.3 Сингулярное разложение

Сингулярное разложение значений, или SVD, возможно, является наиболее популярным методом уменьшения размерности при разреженных данных. SVD широко используется как для вычисления других матричных операций, таких как инверсия матрицы, так и в качестве метода сокращения данных в машинном обучении.

Разреженные данные относятся к рядам данных, в которых многие значения равны нулю. Это часто встречается в некоторых областях, таких как рекомендательные системы, где пользователь имеет рейтинг для очень немногих фильмов или песен в базе данных и нулевой рейтинг для всех остальных случаев.

Рассмотрим матрицу входных данных порядка m x n, т.е. матрица имеет m строк и n столбцов. Например, если рассматривать эту матрицу как матрицу документов, то каждая строка представляет документ, а каждый столбец - слово.

Идея SVD заключается в том, чтобы взять эту матрицу порядка m x n и представить ее в виде произведения трех матриц, причем эти три матрицы будут иметь определенные ограничения, связанные с заданной матрицей.

Преимущества:

- Эффективный метод (с помощью алгоритма Ланцоша или аналогичного ему можно применять к очень большим матрицам).
- Имеет тенденцию работать хорошо для большинства наборов данных

Недостатки:

- Если данные являются сильно нелинейными, он может работать не так хорошо.
- Результаты не всегда подходят для визуализации
- Может отбрасывать полезную информацию

1.3.4 Линейный дискриминантный анализ

Линейный дискриминантный анализ, или сокращенно LDA, – это алгоритм прогностического моделирования для многоклассовой классификации. Линейный дискриминантный анализ также работает как алгоритм снижения размерности, это означает, что он уменьшает число измерений от исходного до C - 1 числа признаков, где C – число классов.

Он обеспечивает проекцию обучающего набора данных, которая наилучшим образом разделяет примеры по их назначенному классу. В частности, модель стремится найти линейную комбинацию входных переменных, которая обеспечивает максимальное разделение образцов между классами (центроиды или средние значения классов) и минимальное разделение образцов внутри каждого класса.

Идея, лежащая в основе LDA, проста. Необходимо найти новое пространство признаков для проецирования данных, чтобы максимизировать разделимость классов. Первым шагом является поиск способа измерения

способности к разделению каждого кандидата на новое пространство признаков. Расстояние между проецируемыми средними каждого класса может быть одной из мер, однако только это расстояние будет не очень хорошей метрикой, потому что оно не учитывает разброс данных.

В 1988 году статистик Рональд Фишер предложил следующее решение: максимизировать функцию, которая представляет собой разницу между средними, нормированную на меру внутриклассовой изменчивости. Предложение Фишера заключается в том, чтобы максимизировать расстояние между средними каждого класса и минимизировать разброс внутри самого класса. Таким образом, мы получаем две меры: внутриклассовую и межклассовую.

Однако такая формулировка возможна, только с предположением, что набор данных имеет нормальное распределение. Это предположение может привести к недостаткам, так как если распределение данных значительно отличается от гауссовского, LDA может работать не очень хорошо.

1.4 Автокодировщики

Если PCA и LDA – это методы, то автокодировщики – это семейство методов.

Автокодировщик можно определить как нейронную сеть, основной целью которой является изучение пространства признаков в наборе данных. Автокодировщик пытается восстановить входные данные по выходным. Автокодировщик обычно состоит из двух частей: кодировщика, который преобразует вход в скрытый код, и декодировщика, который восстанавливает вход из скрытого кода.

Автокодировщики являются семейством, потому что единственное имеющееся ограничение – это то, что входной и выходной слой будут одинаковой размерности, внутри можно создать любую архитектуру, чтобы иметь возможность кодировать данные высокой размерности.

Автокодировщики начинают с некоторого случайного низкоразмерного представления и выполняют градиентный спуск к своему решению путем изменения весов, соединяющих входной слой со скрытым слоем, а скрытый слой - с выходным.

Поскольку есть полный контроль над внутренней частью сети, имеется возможность разработать кодировщики, которые будут способны выбирать очень сложные взаимосвязи между признаками. Это дает большую гибкость в том, как именно будут сжиматься данные.

Автокодировщики являются мощным инструментом снижения размерности данных и в некоторых случаях показали отличные результаты по сравнению с другими методами.

1.5 Преимущества автокодировщиков

Автокодировщик – более сложная и комплексная техника, чем, например, PCA, которая может моделировать относительно сложные взаимосвязи и нелинейности в данных.

Автокодировщик с одним слоем и линейной активацией имеет такую же производительность, как и PCA. Автокодировщик с несколькими слоями и нелинейной функцией активации, называемый глубоким автокодировщиком, склонен к переобучению, но это может контролироваться регуляризацией и тщательным проектированием. При этом, такая модель способна лучше остальных алгоритмов работать на наборе однотипных данных.

Еще один большой плюс автокодировщиков в том, что поскольку к концу обучения есть веса, ведущие к скрытому слою, можно обучать их на определенных входных данных, и, если позже появятся новые данные, можно будет уменьшить их размерность с помощью этих весов без повторного обучения. Это будет работать только в том случае, если точка данных в некоторой степени похожа на данные, на которых производилось обучение.

2. МАТЕРИАЛЫ И МЕТОДЫ

2.1 Описание набора данных

Спектрографические методы анализа, основанные на анализе спектров собственной флуоресценции, получили большое распространение в различных экологических задачах.

В данной работе для выделения признаков используется набор данных, представляющий собой серии спектров собственной флуоресценции цианобактерий. Каждая серия состоит из 7 спектров, полученных использованием облучающих лазеров с различной длиной волны, а именно: 405, 458, 476, 488, 496, 514 и 543 нм. Каждый образец относится к одному из 24 классов: 398, 457, 535, 550, 601, 624, 666, 756, 824, 1072, 1126, 1315, 1336, 1409, 1415, 1416, 1712, 1713, 1715, 1718, 1750, 1763, 1804, 1817.

К имеющимся данным уже была применена предварительная обработка, описанная в работе [3, p. 47], а именно:

- Интерполяция спектров с шагом 1 нм
- Нормировка на максимальную интенсивность
- Отсечение интенсивностей не больше 5% от максимальной
- Центрирование спектров
- Приведение спектров к единой длине
- Двухэтапное сглаживание спектров оконными функциями

Таким образов были получены 547 серий из 7 спектров с областью определения от -200 до +200.

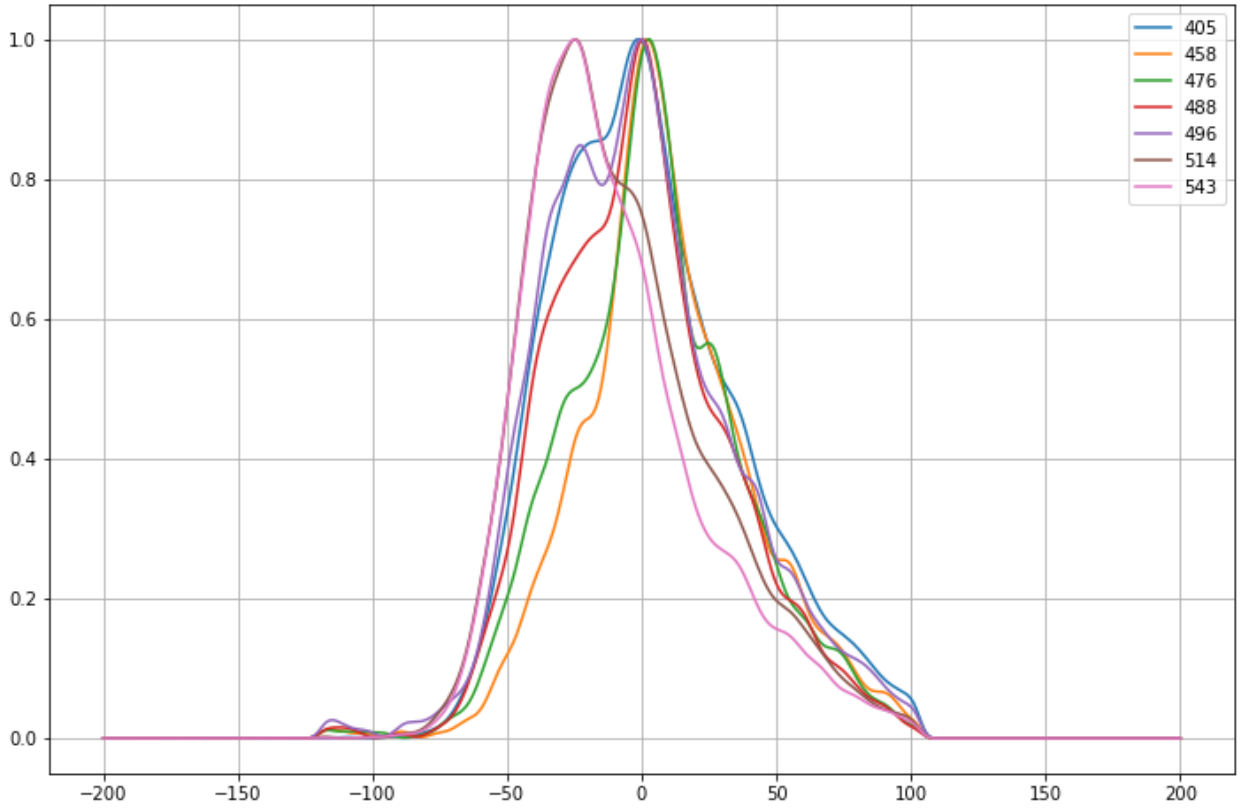

Рисунок 3 - Пример серии спектров из исходного набора данных

Пример входных данных, используемых в данной работе, представлен на рис. 3.

2.2 Модели автокодировщиков

В предыдущей главе были упомянуты автокодировщики – нейронные сети, которые обучены пытаться копировать свой вход в свой выход. Внутри нейронной сети имеется скрытый слой h, который описывает код, используемый для представления входных данных.

Все автокодировщики обладают следующими особенностями:

- Зависимость от данных: Автокодировщики способны осмысленно сжимать только те данные, на которых они были обучены. Поскольку они изучают особенности, специфичные для данных обучения, они отличаются от стандартных алгоритмов сжатия данных, таких как gzip. Поэтому мы не можем ожидать, что автокодировщик, обученный на рукописных цифрах, сможет сжимать фотографии пейзажей.

- Наличие потерь: Выходной сигнал автокодировщика не будет точно таким же, как входной, это будет близкое, но ухудшенное представление.
- Автокодировщики считаются методом обучения без учителя, поскольку им не нужны явные метки для обучения. Но если быть точнее, они являются самоконтролируемыми (self-supervised), поскольку генерируют свои собственные метки из обучающих данных.

Сеть можно рассматривать как состоящую из двух частей: кодирующей функции $h = f(x)$ и декодера, который производит реконструкцию $r = g(h)$. Эта архитектура представлена на рисунке 4.

Если автокодировщику удается просто научиться устанавливать $g(f(x)) = x$ везде, то он не особенно полезен. Вместо этого автокодировщики разрабатываются таким образом, чтобы они не могли научиться идеальному копированию. Обычно их ограничивают таким образом, чтобы они могли копировать только приблизительно, и копировать только те входные данные, которые похожи на обучающие. Поскольку модель вынуждена определять приоритеты того, какие аспекты входных данных должны быть скопированы, она часто узнает полезные свойства данных.

Идея автокодировщиков была частью исторического ландшафта нейронных сетей в течение десятилетий. Традиционно автокодировщики использовались для уменьшения размерности или обучения признаков. Недавно теоретические связи между автокодировщиками и моделями латентных переменных вывели автокодировщики на передний план генеративного моделирования. Автокодировщики можно рассматривать как особый случай сетей прямого распространения и обучать их с помощью всех тех же методов, как правило, мини-пакетного градиентного спуска по градиентам, вычисленным методом обратного распространения.

Рассмотрим подробнее различные модели автокодировщиков.

2.2.1 Стандартный автокодировщик

Рассмотрим детально автокодировщик, кодировщик (encoder) и декодировщик (decoder). И кодировщик, и декодировщик представляют собой полносвязные нейронные сети с прямым распространением, по сути, ИНС, которые были рассмотрены в первой главе. Код (code) – это один слой ИНС с размерностью по нашему выбору. Количество узлов в слое кода (размер кода) – это гиперпараметр, который задается перед обучением автокодировщика.

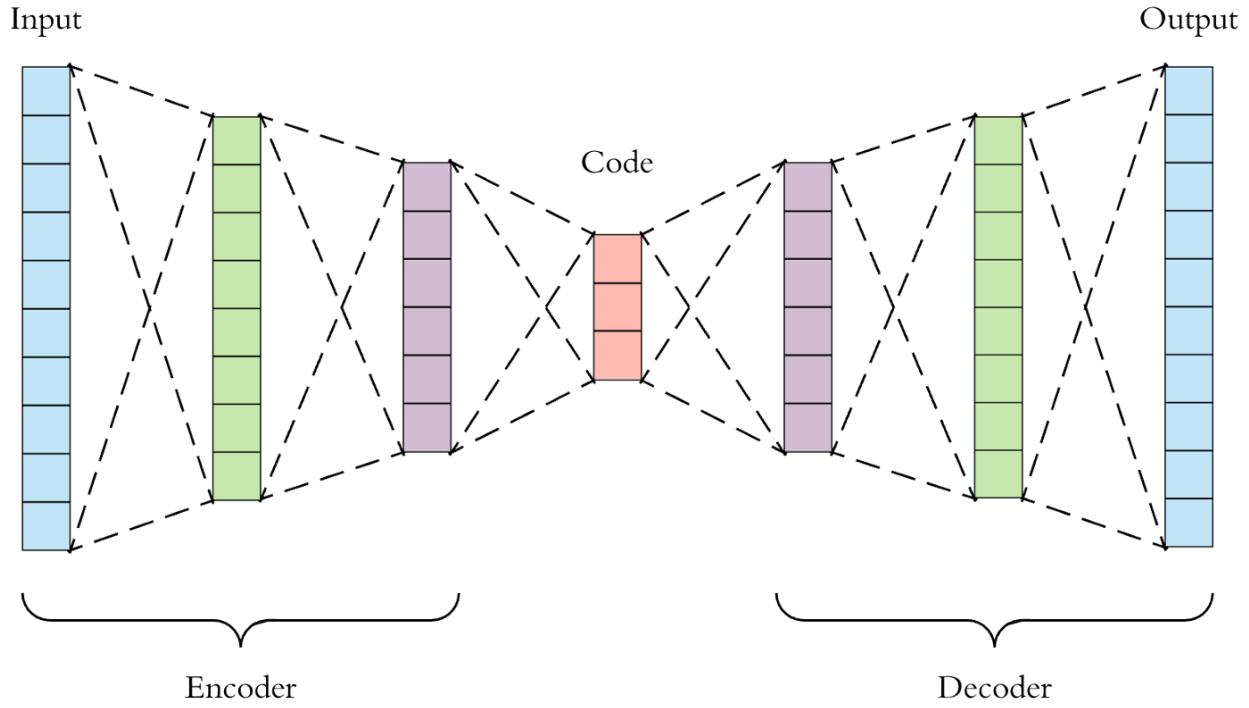

Рисунок 4 - Модель автокодировщика

Сначала входной сигнал проходит через кодировщик, который представляет собой полностью связанную ИНС, для получения кода. Затем декодировщик, который имеет аналогичную структуру ИНС, производит выходной сигнал, используя только код. Цель состоит в том, чтобы получить выходной сигнал, идентичный входному.

Примечательно, что архитектура декодера является зеркальным отражением архитектуры кодера. Это не является обязательным условием, но обычно так и происходит. Единственное требование - размерность входа и выхода должна быть одинаковой. Остальная часть модели может быть устроена многими различными способами.

Существует 4 гиперпараметра, которые необходимо задать перед обучением автокодировщика:

- Размер кода: количество узлов в среднем слое. Меньший размер приводит к большему сжатию.
- Количество слоев: автокодировщик может быть настолько глубоким, насколько это необходимо. На рисунке 4 изображены 2 слоя в кодировщике и декодировщике, без учета входа и выхода.
- Количество узлов на слой: количество узлов на слой уменьшается с каждым последующим слоем кодировщика и увеличивается обратно в декодировщике. Также декодировщик симметричен кодировщику с точки зрения структуры слоев.
- Функция потерь: чаще всего в автокодировщиках используют либо среднюю квадратичную ошибку (mse), либо бинарную кроссэнтропию. Если входные значения находятся в диапазоне [0, 1], то мы обычно используем кроссэнтропию, в противном случае мы используем среднюю квадратичную ошибку.

Среднеквадратичная ошибка (MSE), если вектор из n прогнозов генерируется на основе выборки из n точек данных по всем переменным, а Y – вектор наблюдаемых значений прогнозируемой переменной, при этом $\hat{Y}$ – это прогнозируемые значения (например, по методу наименьших квадратов), то MSE предиктора в пределах выборки вычисляется как:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Бинарная кроссэнтропия, или BCE, измеряет производительность модели, выход которой представляет собой значение между 0 и 1. Потеря кроссэнтропии увеличивается по мере того, как предсказанное значение расходится с фактической меткой. Так, предсказание вероятности .012 при фактической метке наблюдения 1 будет плохим и приведет к высоким потерям. Идеальная модель имела бы логарифмическую потерю 0. Если вектор

из n прогнозов генерируется на основе выборки из n точек данных по всем переменным, а Y – вектор наблюдаемых значений прогнозируемой переменной, при этом $\hat{Y}$ – это прогнозируемые, то L_{BCE} (значение функции потерь) предиктора в пределах выборки вычисляется как

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Автокодировщики обучаются так же, как и ИНС, методом обратного распространения.

2.2.2 Разреженный автокодировщик

Разреженный автокодировщик [4] (Sparse Autoencoder) применяет "разреженное" ограничение на активацию скрытых нейронов, чтобы избежать переобучения и повысить устойчивость. Это заставляет модель иметь только небольшое количество скрытых нейронов, активированных одновременно, или, другими словами, один скрытый нейрон должен быть инактивирован большую часть времени.

Допустим, в l-ом скрытом слое есть s_l нейронов, и функция активации для j-го нейрона в этом слое обозначена как $a_j^l(.), j = 1, \dots, s_l$. Доля активации этого нейрона $\hat{p}_j$ должна быть небольшим числом p, известным как параметр разреженности; обычно $p = 0.05$.

$$\hat{p}_j{}^l = \frac{1}{n}\sum_{i=1}^{n} a_j^l(x^i) \approx p$$

Это ограничение достигается путем добавления штрафного члена в функцию потерь L: $L\left(x, f\left(g(x)\right)\right) + \Omega(h)$, где h – размер кода. Ω – обычный регуляризатор, например L_1, g и f – кодировщик и декодировщик соответственно.

Разреженный автокодировщик не обязательно сужается к центру. Его код может иметь и большую размерность, чем входной сигнал. Обучаясь приближать тождественную функцию x = f(g(x)), он учится в коде выделять

полезные свойства сигнала. Из-за регуляризатора даже расширяющийся к центру разреженный автокодировщик не может выучить тождественную функцию напрямую, т.к. регуляризация вынуждает автокодировщик находить значимые признаки.

2.2.3 Денуасифицирующий автокодировщик

Поскольку автокодировщик обучается функции тождества, возникает риск переобучения, когда параметров сети больше, чем количество точек данных.

Для того чтобы избежать переобучения и повысить устойчивость, в работе [5] предложена модификация базового автокодировщика. Входные данные частично искажаются путем добавления шумов или маскировки некоторых значений входного вектора стохастическим образом и не является специфическим для определенного типа процесса искажения (например, маскирующий шум, гауссовский шум и т.д.). Затем модель обучается для восстановления исходного входного сигнала. Пример подобной архитектуры представлен на рис. 5

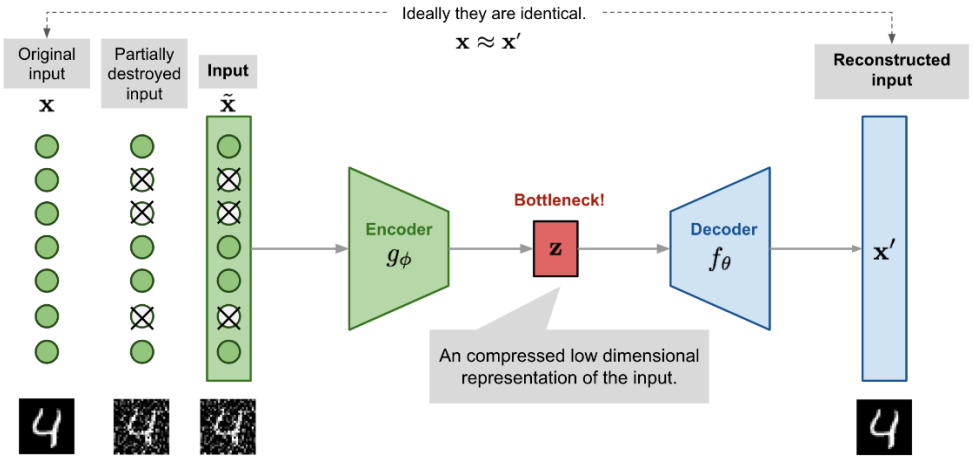

Рисунок 5 - Архитектура денуасифицирующего автокодировщика

Такая конструкция мотивирована тем, что человек может легко распознать объект или сцену даже при частичной окклюзии или повреждении изображения. Чтобы восстановить частично разрушенный входной сигнал, денуасифицирующий автокодировщик (DAE) должен обнаружить и

зафиксировать взаимосвязь между измерениями входного сигнала, чтобы вывести недостающие фрагменты.

Для высокоразмерных входных данных с высокой избыточностью, таких как изображения, модель, скорее всего, будет полагаться на данные, собранные из комбинации многих измерений входных данных, для восстановления денуазированной версии, а не на избыточную подгонку одного измерения. Это закладывает основу для обучения хорошей репрезентации.

В эксперименте, проведенном в оригинальной статье DAE, шум применяется таким образом: фиксированная доля входных измерений выбирается случайным образом, и их значения принудительно устанавливаются равными 0.

2.3 Сверточные нейронные сети

Сверточная нейронная сеть [6] (ConvNet/CNN) – это алгоритм глубокого обучения, который может принимать входное изображение, присваивать значение (обучаемые веса и смещения) различным аспектам/объектам на изображении и отличать один от другого.

Архитектура ConvNet аналогична структуре связи нейронов в человеческом мозге и была вдохновлена организацией зрительной коры. Отдельные нейроны реагируют на стимулы только в ограниченной области зрительного поля, известной как рецептивное поле. Набор таких полей перекрывается и охватывает всю зрительную область.

Сеть ConvNet способна успешно улавливать пространственные и временные зависимости в изображении путем применения соответствующих фильтров. Архитектура лучше подходит к набору данных изображений благодаря уменьшению количества задействованных параметров и возможности повторного использования весов. Другими словами, сеть может быть обучена для лучшего понимания сложности изображения.

В данной архитектуре используются слои свертки и пуллинга.

Свёрточный слой является основным структурным элементом CNN, и именно в нём происходит большая часть вычислений. Для его работы требуется несколько компонентов: входные данные, фильтр (детектор признаков) и карта признаков. Детектор признаков, также известный как ядро или фильтр, который перемещается по рецептивным полям изображения, проверяя, присутствует ли признак. Этот процесс известен как свертка. Пример свертки на рис. 6.

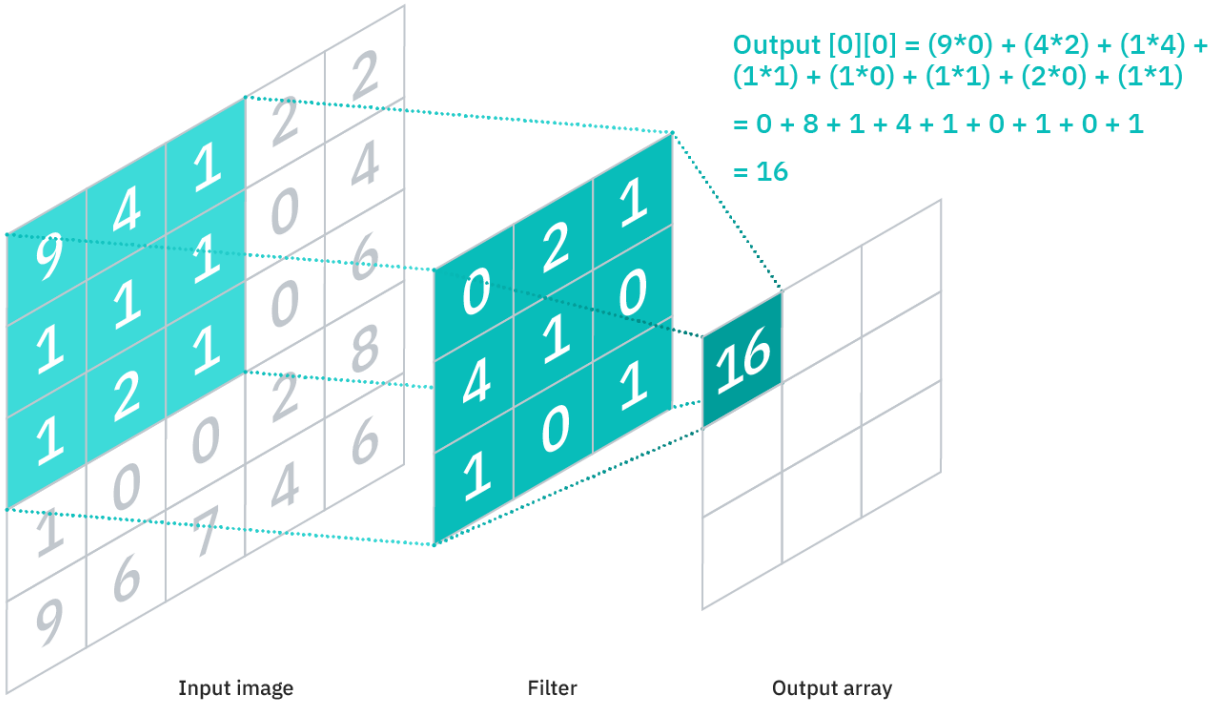

Рисунок 6 - Работа детектора признаков

Детектор признаков это собой двумерный (2-D) массив весов, который представляет собой часть изображения. Фильтр применяется к области изображения, и вычисляется точечное произведение между входными пикселями и фильтром. Затем это точечное произведение поступает в выходной массив. После этого фильтр сдвигается на строку, повторяя процесс до тех пор, пока ядро не охватит все изображение. Конечный результат серии точечных произведений входного сигнала и фильтра известен как карта признаков.

После каждой операции свертки CNN применяет преобразование Rectified Linear Unit (ReLU) к карте признаков, внося нелинейность в модель.

Слой пуллинга, также известный как слой субдискретизации, осуществляет снижение размерности, уменьшая количество параметров на входе. Как и в случае со сверточным слоем, операция объединения проводит фильтр по всем входным данным, но разница в том, что этот фильтр не имеет весов. Вместо этого ядро применяет функцию агрегирования к значениям в пределах рецептивного поля, заполняя выходной массив. Пример работы слоя субдискретизации на рис. 7

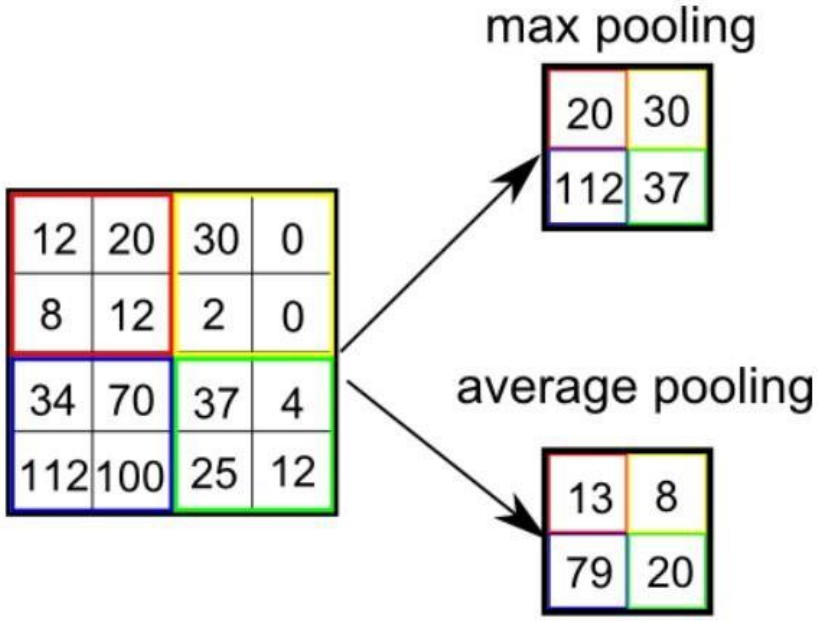

Рисунок 7 - Пример субдискретизации

Существует два основных типа пуллинга:

- Максимальный (max pooling): По мере продвижения фильтра по входу он выбирает пиксель с максимальным значением для отправки в выходной массив. Этот подход, как правило, используется чаще по сравнению со средним пуллингом.
- Среднее значение (average pooling): По мере продвижения фильтра по входу он вычисляет среднее значение в пределах рецептивного поля для отправки на выходной массив.

Хотя на слое субдискретизации теряется много информации, он также имеет ряд преимуществ для CNN. Он помогает снизить сложность, повысить эффективность и ограничить риск переобучения.

2.4 Сверточный автокодировщик

Сверточный автокодировщик [7] (CAE) расширяет базовую структуру обыкновенного автокодировщика путем замены полностью связанных слоев на слои свертки. Как и в простом автокодировщике, размер входного слоя такой же, как и выходных слоев, но сеть кодировщика меняется на слои свертки.

В целом, CAE в основном используются для уменьшения и сжатия размера входных данных, удаления шумов с одновременным сохранением полезной информации и извлечения достоверных признаков.

Более конкретно, CAE состоят из двух моделей CNN – кодировщика и декодировщика, как показано на рисунке 8.

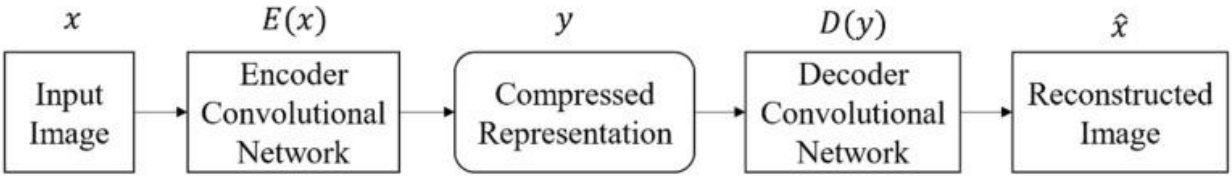

Рисунок 8 - Модель сверточного автокодировщика

Кодировщик в основном используется для кодирования исходного входного изображения в скрытое представление, которое имеет меньшую размерность. С другой стороны, декодировщик отвечает за восстановление сжатого латентного представления, создавая выходное изображение, максимально похожее на исходное.

3. ОПИСАНИЕ РЕШЕНИЯ

3.1 Построение моделей

Для обучения используется алгоритм Adam со стандартными параметрами из библиотеки Keras, т.к. в большинстве случаев он дает хорошие результаты, и все модели должны быть обучены одинаковым образом, чтобы была возможность сравнивать их производительность.

В качестве числа эпох выбрано значение 90, т.к. после нескольких тестов было установлено, что этого как раз достаточно для того, чтобы представленные модели достигали минимума ошибки.

Размер батча при использовании градиентного спуска был установлен в 16, что связано с ограничениями на GPU.

Перед обучением исходный набор данных был разбит на две части – для обучения и оценки качества модели, в пропорции 75/25.

Как уже было описано в пункте 2.2.1, перед обучением автокодировщика необходимо задать 4 гиперпараметра, а именно:

- Размер кода
- Число слоев
- Число нейронов на слой
- Функцию ошибки

Так как входные данные нормированы от 0 до 1, в данной работе в качестве функции ошибки будет использоваться бинарная кроссэнтропия.

Влияние числа нейронов на слой на работу модели будет рассмотрено в п. 3.1.3. Рассмотрим различные сочетания остальных параметров.

3.1.1 Стандартная модель

Вход и выход модели имеют размер 2807 по размеру входных данных (7 спектров по 401 значению каждый, представленные в виде одного вектора). Скрытые слои модели устроены следующим образом:

- Последний слой сети кодировщика, представляющий выделенные признаки, его размер будет изменяться

- До двух скрытых слоев, размер которых увеличивается на 100% по мере удаления от слоя с выделенными признаками. Число таких слоев также будет изменяться

Для каждого сочетания параметров модель была обучена 3 раза, после чего значение функции потерь было усреднено для повышения устойчивости результата. Результат обучения на рис. 9.

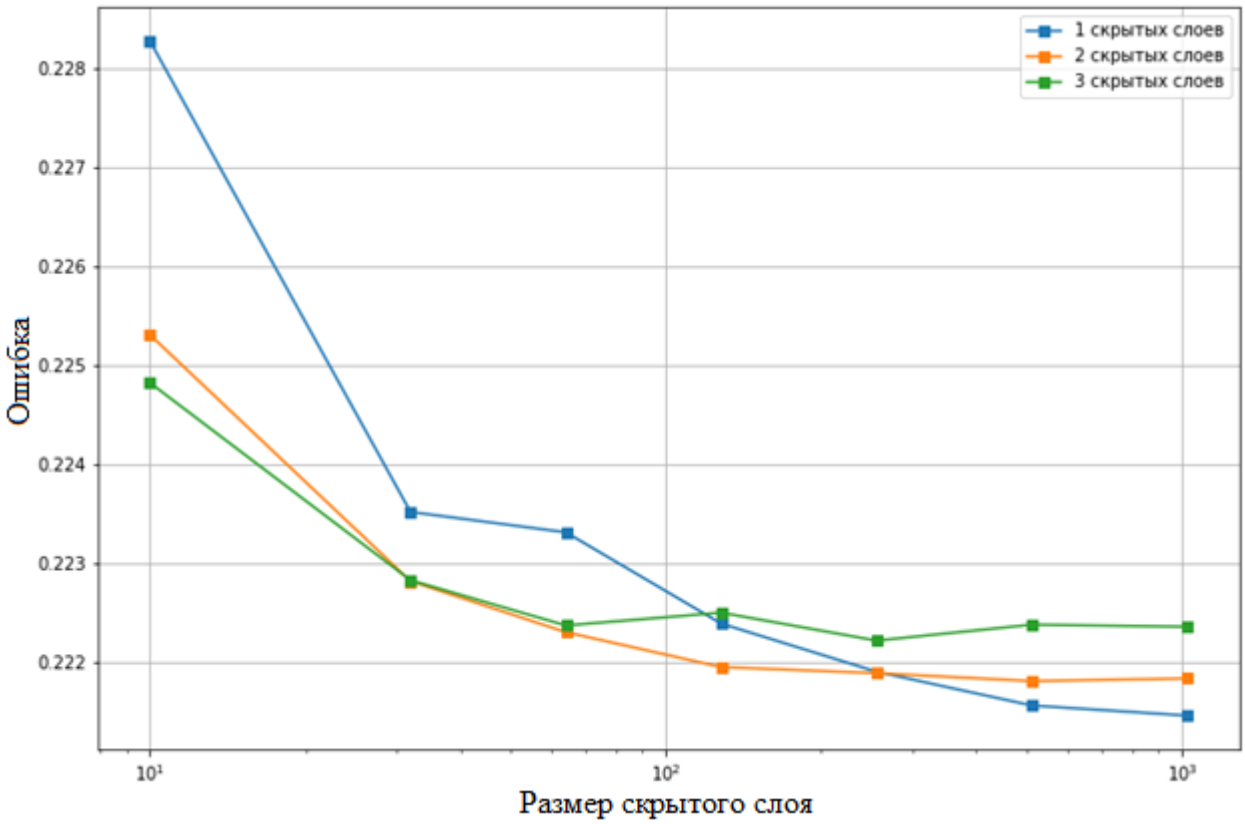

Рисунок 9 - Результат обучения

В качестве размера последнего слоя кодировщика были выбраны следующие значения: 10, 32, 64, 128, 256, 512, 1024. Это позволяет удобно отобразить данные на логарифмической шкале и оценить зависимость ошибки от числа выделяемых признаков.

Из результатов обучения можно сделать следующие выводы:

- Наименьшей ошибки достигает сеть с одним скрытым слоем, но только при увеличении числа выделяемых признаков до больших значений, не имеющих практического применения

- Каждая модель при достижении определенного оптимального числа признаков почти перестает изменять значение функции ошибки. При дальнейшем увеличении числа признаков ошибка будет расти из-за переобучения.
- Модель с двумя слоями имеет минимальную ошибку при адекватном количестве выделяемых признаков (64, 128)

На данный момент оптимальной кажется модель с двумя слоями и числом признаков около ста. Применим регуляризацию и изучим изменения в работе моделей.

3.1.2 Применение регуляризации

Применяться будет L_1 регуляризация [8] со стандартным параметром регуляризации 10^{-5}, на каждый скрытый слой сети. Результаты на рис. 10

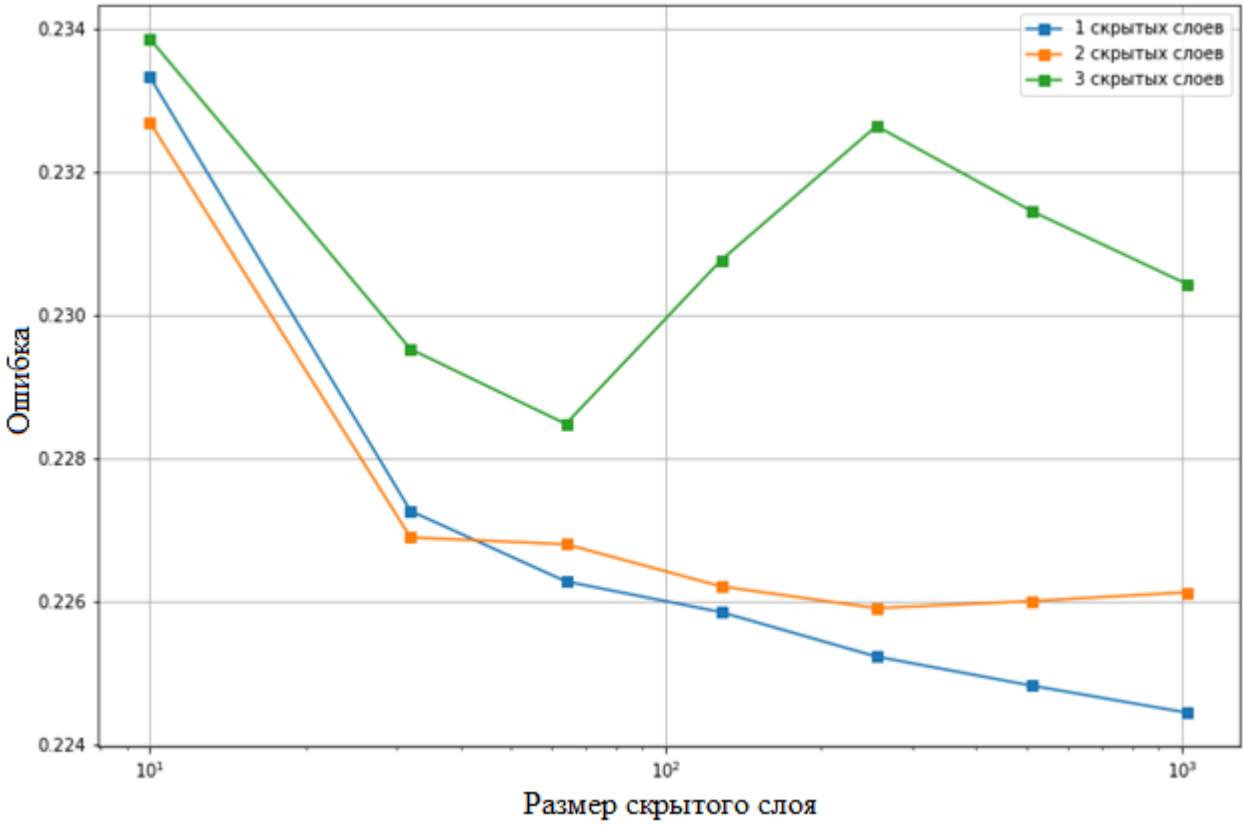

Рисунок 10 - Применение регуляризации

Из рисунка видно, что применение регуляризации ухудшило результат работы сети, причем для сети с тремя скрытыми слоями кодировщика ошибка

растет при увеличении размера последнего скрытого слоя кодировщика. Это может происходить по нескольким причинам:

- Стандартным параметр регуляризации оказался слишком велик
- Сам факт применения регуляризации каким-то образом ухудшает работу сети

Проверим первую гипотезу и уменьшим коэффициент регуляризации. Результат на рис. 11

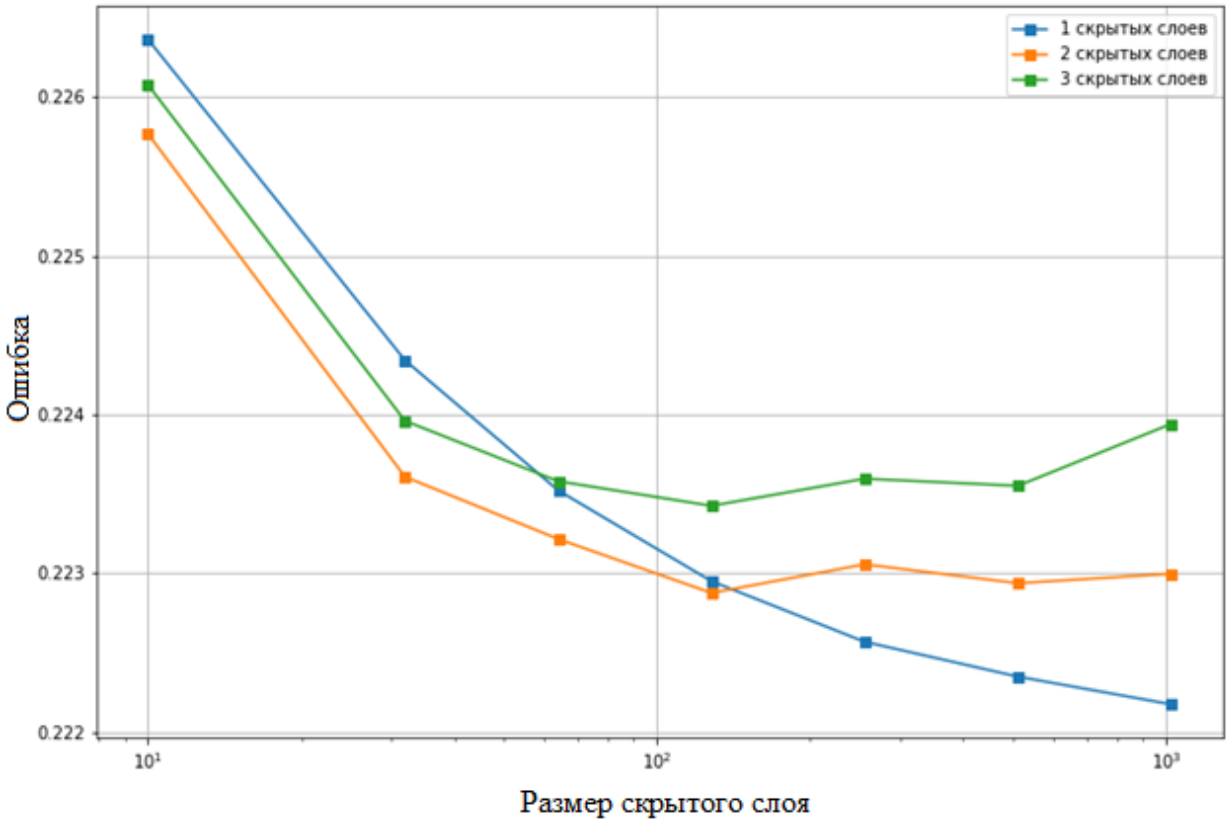

Рисунок 11 - Результаты после изменения коэффициента регуляризации

Ошибка сети уменьшилась по сравнению с предыдущим случаем, но она все еще выше, чем без регуляризации. Попробуем применить регуляризацию только к последнему слою кодировщика, а не ко всем скрытым слоям сети. Результат на рис. 12.

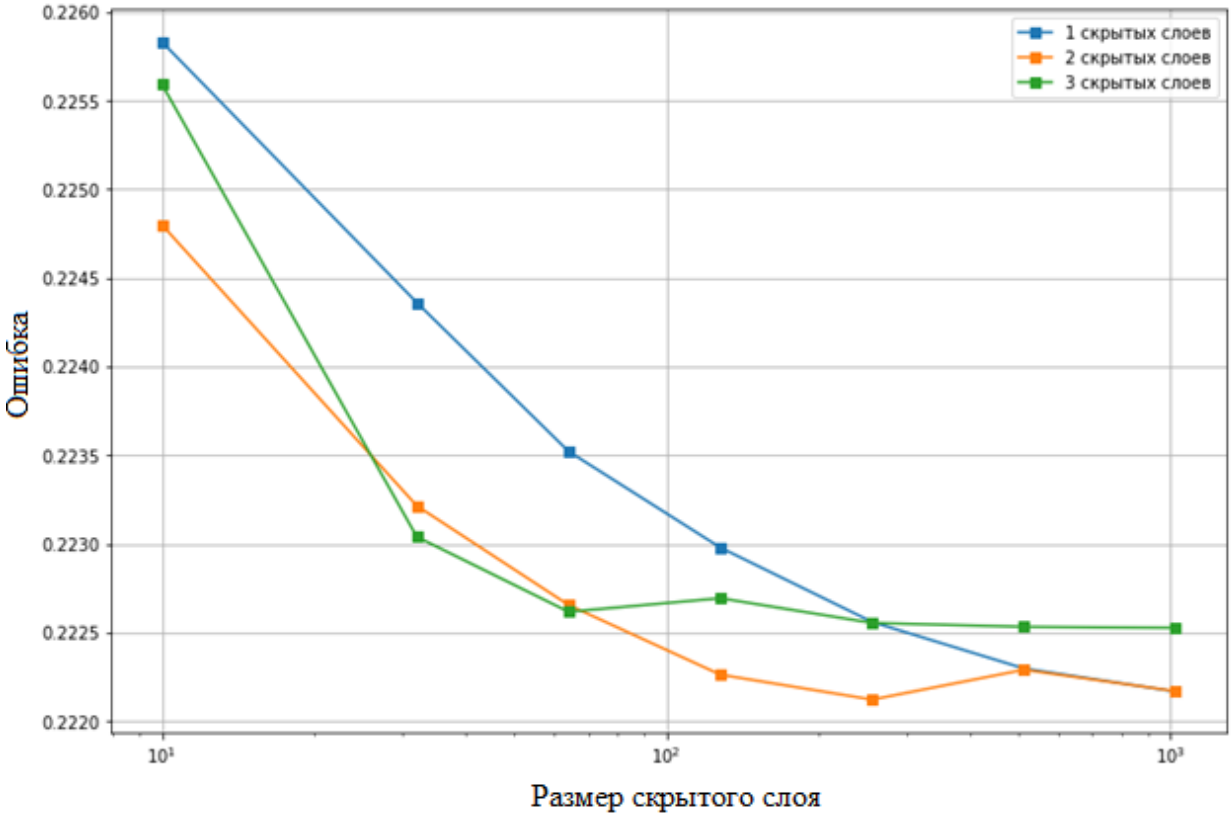

Рисунок 12 - Ошибки моделей при применении регуляризации только к одному слою

Из рисунка 12 видно, что модель с одним слоем значительно ухудшила результаты относительно остальных моделей, т.к. в ней регуляризация применяется к единственному скрытому слою. Остальные модели также работают хуже, чем без введения регуляризации.

Таким образом, сам факт введения L_1 регуляризации ухудшает работу модели. Связано это может быть с тем, что данная регуляризация побуждает выделяемые признаки, близкие к нулю, принимать нулевое значение. Если модель обучается выделять важные признаки, многие из которых близки к нулю, это может увеличить ошибку. Данная гипотеза будет проверена в 4 разделе данной главы.

3.1.3 Изменение размеров слоев

В предыдущих случаях были рассмотрены модели, в которых размеры скрытых слоев были фиксированы. Возьмем лучшую модель из исследованных (2 слоя в кодировщике, 64 признака) и исследуем изменение

функции ошибки при изменении размеров скрытого слоея от 0% до 100% от последнего слоя кодировщика с шагом в 10% путем умножения размера последнего слоя на коэффициент k, изменяющийся соответственно от 1 до 2 с шагом 0.1. Результат на рис. 13.

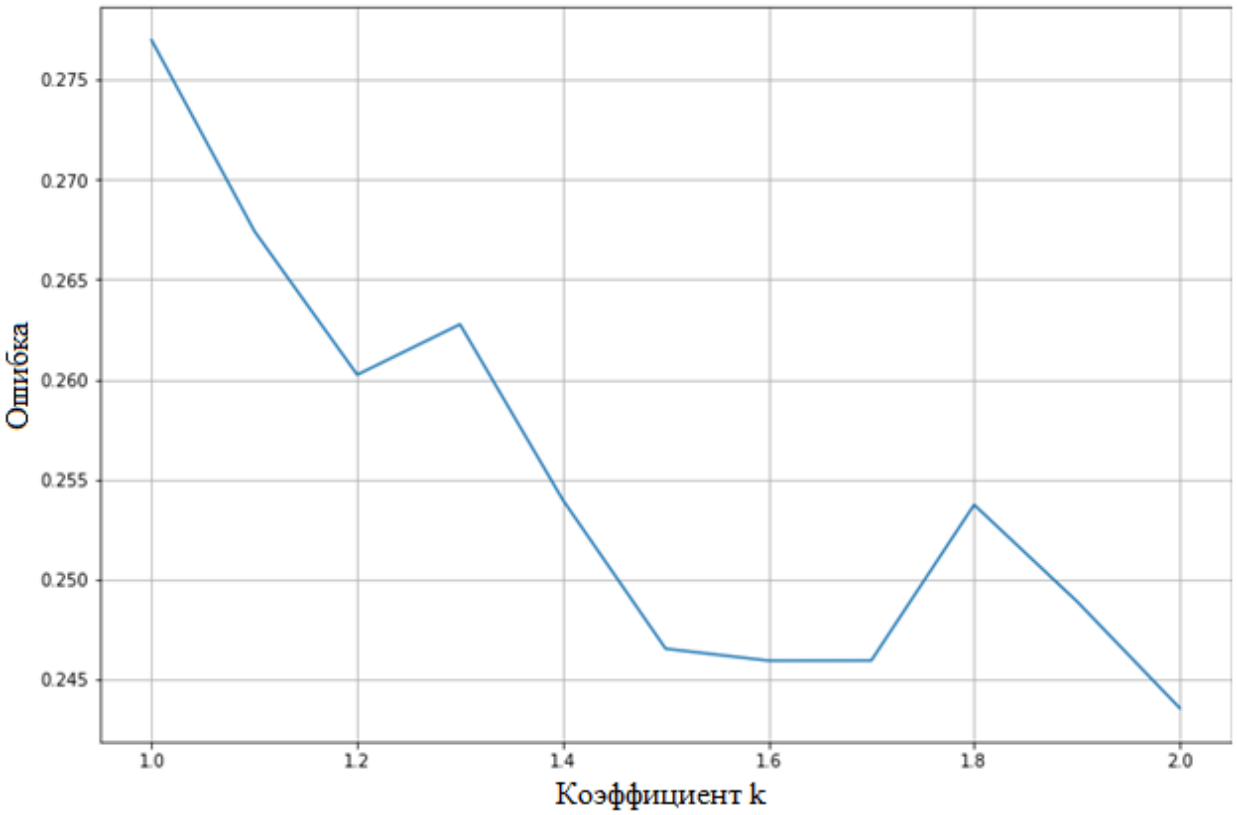

Рисунок 13 - Изменение ошибки при изменении размера скрытого слоя

Можно сделать следующий вывод: уменьшение размера слоя в целом увеличивает ошибку. При этом, увеличение слоя на 60% периодически дает лучший результат. Это может быть связано со множеством различных факторов, например разбиения датасета, и в среднем выгоднее не изменять предыдущую модель.

3.1.4 Изменение функции активации

Во всех предыдущих моделях использовалась стандартная функция активации ReLU [9]. Для моделей с большим числом скрытых слоев это может приводить к проблеме затухающего градиента. В этом случае ReLU всегда выдает одно и то же значение для любого входа. Это достигается путем обучения большого отрицательного порогового значения для весов сети.

Если ReLU окажется в таком состоянии, он вряд ли сможет восстановиться, поскольку градиент функции в точке 0 также равен 0, поэтому обучение по алгоритму градиентного спуска не изменит весовые коэффициенты.

Попробуем улучшить работу модели с тремя скрытыми слоями кодировщика путем изменения функции активации на ELU. Результат на рис. 14.

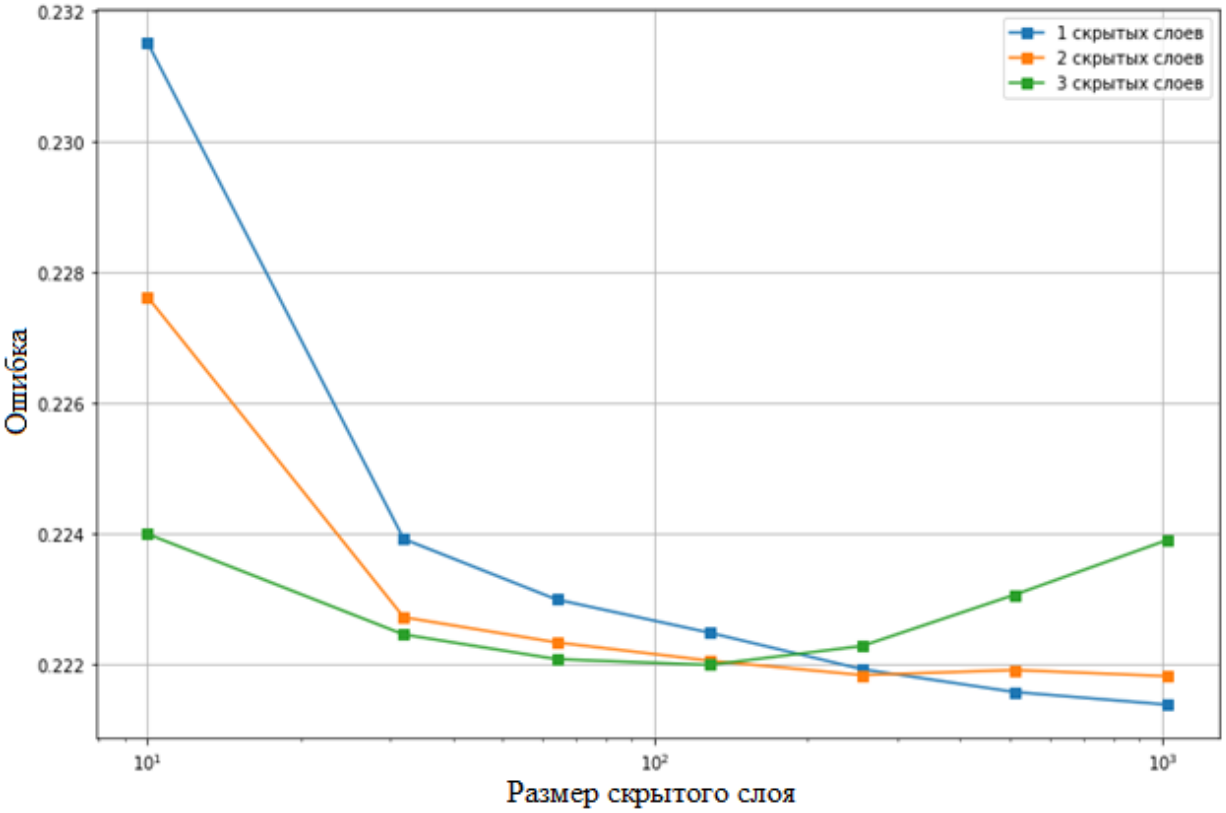

Рисунок 14 - Результаты изменения функции активации

Изменение функции активации позволило значительно улучшить результат модели с тремя слоями кодировщика относительно модели с двумя слоями. Для 64 выделяемых признаков ошибка данной сети насыщается на меньшем значении, чем раньше.

3.1.5 Искажение входных данных

Применим архитектуру денуасифицирующего автокодировщика для решения задачи выделения признаков. Искажать данные будем следующим образом: на входе модели с двумя слоями будем выставлять в нуль

определенное количество входных значений (от 0% до 95% с шагом в 5%). Для этого применим слой Dropout. Результат на рис. 15.

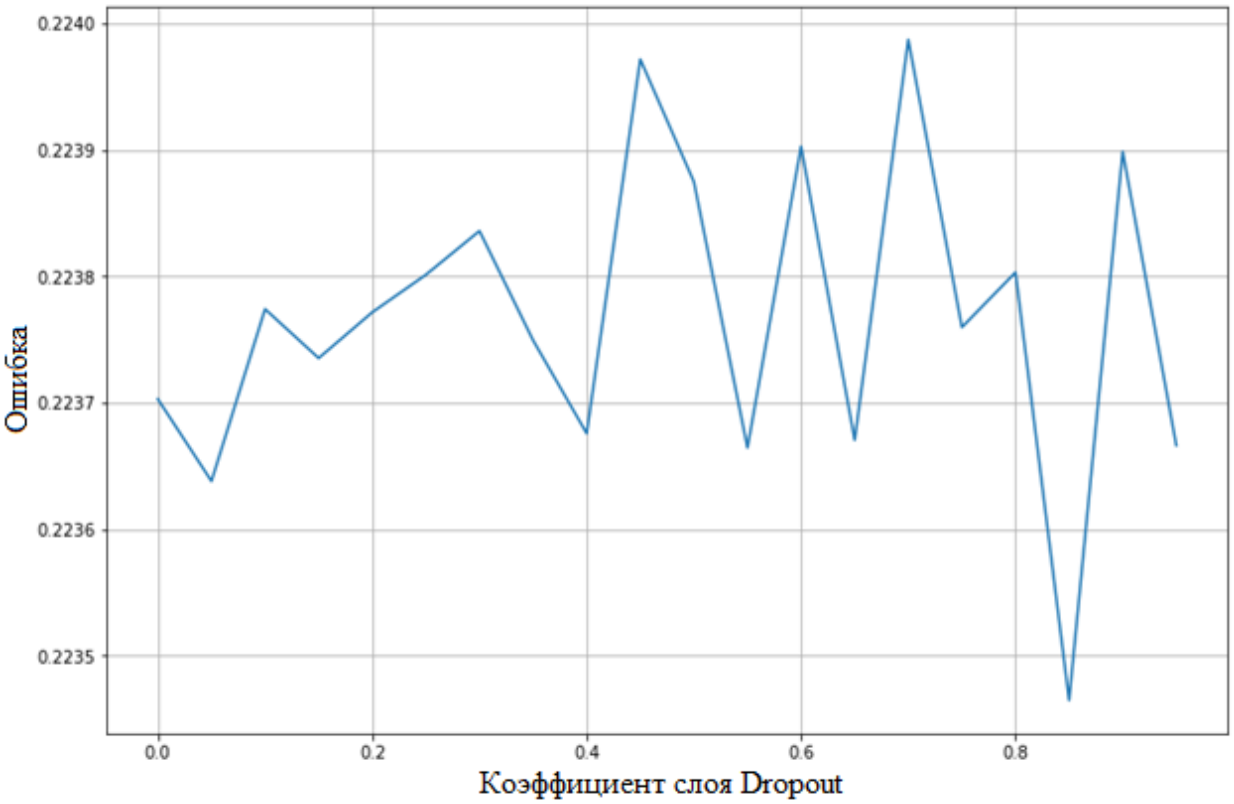

Рисунок 15 - Денуасификация (результаты)

Из рисунка видно, что данная архитектура дает очень разные результаты. Не прослеживается зависимость коэффициента слоя Dropout и получаемой ошибки. Это происходит из-за особенности входных данных. Серии спектров обработаны так, что у большинства входных данных много нулевых значений (до 25%), а устанавливаются в 0 случайные значения.

Таким образом, применение архитектуры денуасифицирующего автокодировщика в данной задаче не целесообразно.

3.1.6 Сверточная модель

Попробуем применить модель сверточного автокодировщика. В модели будут использоваться одномерные слои свертки и субдискретизации. Входные данные представим в виде 7 отдельных векторов по 400 значений; каждый вектор, соответствующий лазеру определенной длины волны, будет интерпретироваться как отдельный канал свертки.

Число фильтров чаще всего выбирается вдвое большим числа каналов. Исследуем модели с различным количеством фильтров от 6 до 26. Размер ядра свертки будет изменяться от 3 до 11 с шагом 2.

Первая модель будет иметь две пары слоев свертки и субдискретизации с параметрами, описанными выше. После обучения модели, усредним ошибку по размеру фильтра и рассмотрим зависимость ошибки модели от числа фильтров (рис. 16).

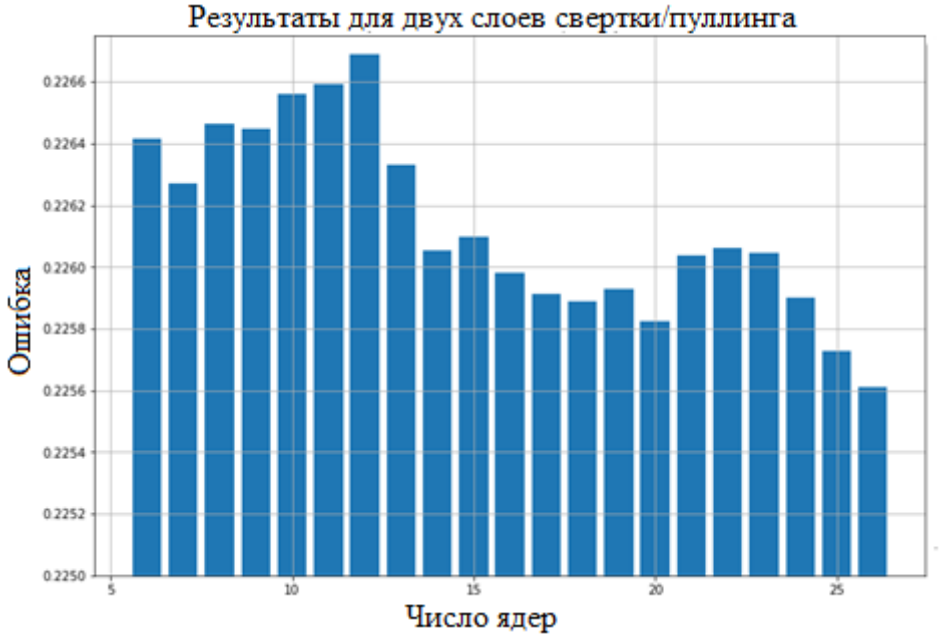

Рисунок 16 - Ошибка для каждого числа фильтров

Из рисунка можно сделать вывод о том, что с увеличением числа фильтров ошибка модели в среднем снижается. Можно также выделить три группы моделей, в которых ошибка в среднем ниже предыдущей группы. Примечательно, что размер этих групп близок к числу каналов входных данных. Отобразим эти группы разными цветами и без усреднения по размеру фильтра (рис. 17).

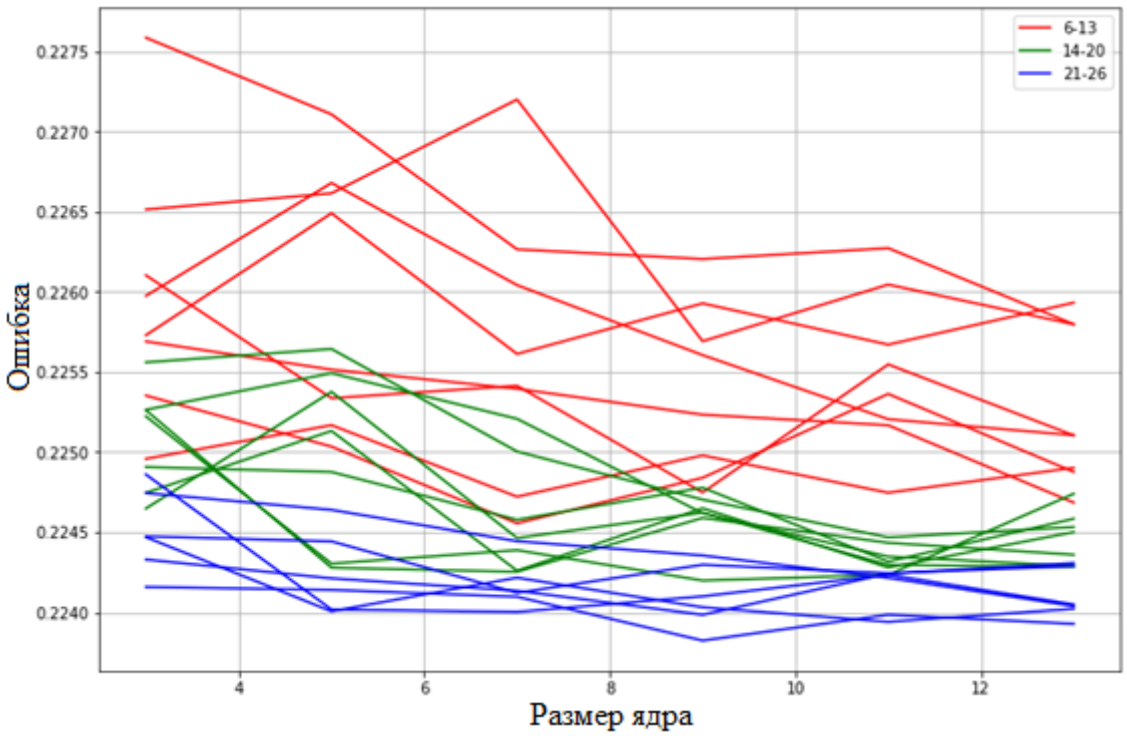

Рисунок 17 - Зависимость ошибки от размера ядра по числу фильтров

Ошибка резко увеличивается или уменьшается при определенных сочетаниях размера фильтра и их количества. С повышением числа фильтров линии на графике располагаются ближе друг к другу, т.к. ошибка уменьшается все медленнее.

Для трех слоев свертки/субдискретизации получены аналогичные результаты (рис. 18), при этом ошибка выше, чем у модели с двумя парами слоев. Это может происходить из-за потери большого количества признаков по мере увеличения количества слоев субдискретизации.

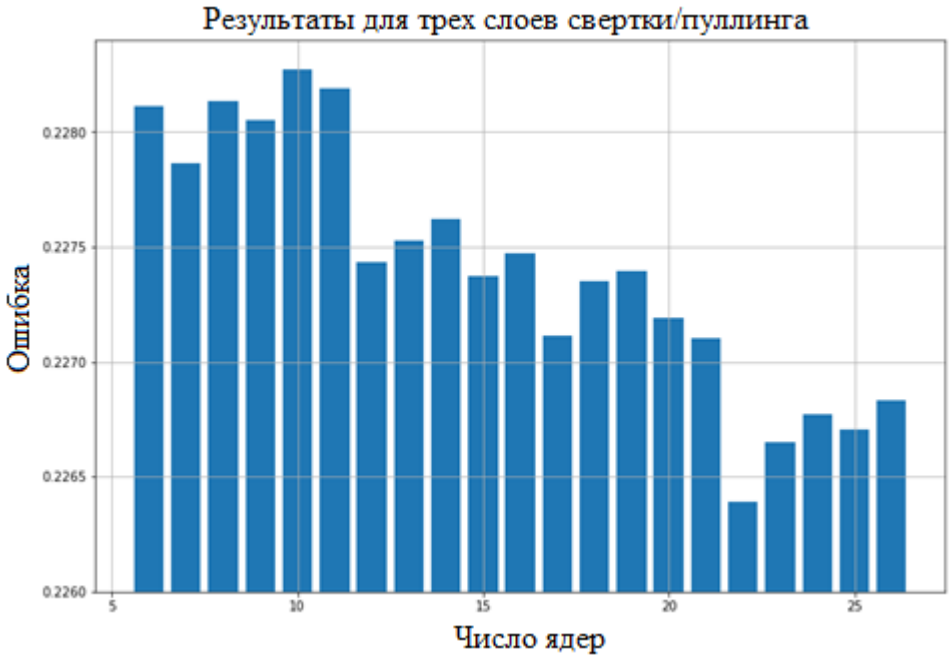

Рисунок 18 - Ошибка для второй сверточной модели

Далее для двух рассмотренных моделей был изменен слой субдискретизации. В предыдущих моделях использовался max pooling, эти слои были заменены на average pooling. Данный способ субдискретизации также используется на практике [10], но в данной задаче он также не подошел – во всех случаях ошибка была выше. Результат для модели с двумя парами слоев в кодировщике представлен на рис. 19.

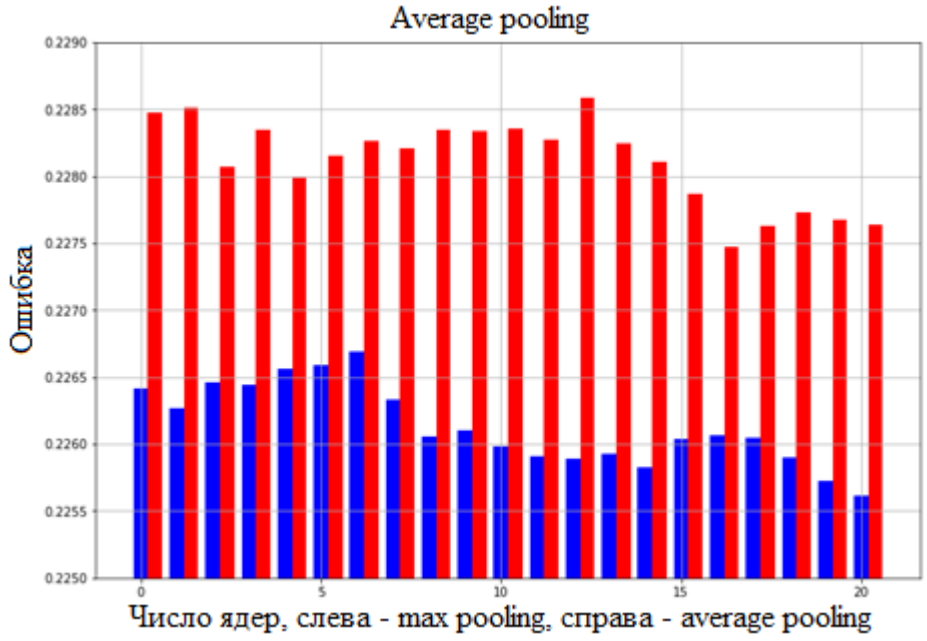

Рисунок 19 - Сравнение типов субдискретизации

Таким образом, из рассмотренных сверточных моделей оптимальной оказалась модель с двумя слоями свертки, после каждого из которых идет слой субдискретизации. Параметрами, при которых ошибка модели минимальна, оказались:

- Число фильров – 17
- Размер фильтра – 7
- Параметр субдискретизации – 2

3.2 Выбор модели

Все рассмотренные модели сравнивались по двум важным для задачи выделения признаков параметрам – число выделяемых признаков и значение функции ошибки в сравнении с аналогичными моделями.

Наилучшими оказались модели:

- Сверточной сети, описанной в конце раздела 3.1
- Полносвязной сети с двумя скрытыми слоями кодировщика, у которой первый слой имеет размер 256, а второй – 128
- Полносвязной сети с тремя скрытыми слоями кодировщика, с размерами слоев 192, 128 и 64 соответственно. Функция активации заменена на ELU.

Сравним модели между собой (см. табл. 1)

Таблица 1 – сравнение моделей

Модель	Число обучаемых параметров	Число выделяемых признаков	Среднее значение функции потерь
сверточная	7810	1700	0.22180
двухслойная	1506167	128	0.22224
трехслойная	1146935	64	0.22268

Модель со слоями свертки и субдискретизации имеет значительно меньшее количество параметров, так как обрабатывает каждый спектр из

серии фильтрами параллельно. Также данная модель показала наименьшее значение функции потерь. При этом, уменьшение ошибки на 0.00044 достигнуто за счет большого увеличения числа выделяемых признаков. При увеличении коэффициента субдискретизации или числа пар слоев ошибка сильно увеличивается. Уменьшение такого большого пространства признаков всего на 40% не найдет применения на практике.

Полносвязная модель с двумя скрытыми слоями в кодировщике показывает меньшее среднее значение ошибки, но выделяет признаков в два раза больше, чем трехслойная модель. Небольшое уменьшение ошибки за счет роста числа выделяемых признаков на 100% говорит о том, что данные признаки намного менее полезны.

Таким образом, была выбрана модель с тремя скрытыми слоями в кодировщике размерами 192, 128 и 64 и функцией активации ELU. Модель была обучена на большом числе эпох (160), была достигнута ошибка 0.22249. Затем модель была сохранена для дальнейшего использования.

3.3 Применимость выделенных признаков

Полученная модель была использована для выделения признаков из набора данных. Выделенные признаки были использованы при обучении пяти наиболее популярных алгоритмов классификации, широко используемые при решении прикладных задач [11], а именно:

- Метод k-ближайших соседей (K-Nearest Neighbors);
- Метод опорных векторов (Support Vector Machines);
- Случайный лес (Random Forests);
- Наивный байесовский метод (Naive Bayes);
- Линейный дискриминантный анализ (Linear Discriminant Analysis);

3.3.1 Проведение классификации

Для обучения моделей классификации и оценки метрик их качества набор данных с выделенными признаками был разделен в пропорции 75/25.

Так как классов объектов достаточно много относительно общего количества объектов, при разбиении набора данных была использована стратификация. Для повышения достоверности результатов разбиение данных и обучение каждой модели были произведены 10 раз, и полученные результаты были усреднены.

Для каждого метода были посчитаны следующие метрики:

- Accuracy – доля правильно классифицированных объектов
- Precision – доля релевантных экземпляров среди найденных экземпляров
- Recall – доля релевантных экземпляров, которые были найдены
- F_1 - это среднее гармоническое значение показателей precision и recall. Максимально возможное значение F-score равно 1, что означает идеальную точность и отзыв, а минимально возможное значение равно 0.

3.3.2 Анализ результатов

Результаты обучения классификаторов представлены в табл. 2

Таблица 2 – результаты классификации

Метод	Accuracy	Precision	Recall	F_1
Метод k-ближайших соседей	0.8328	0.7874	0.7879	0.7720
Метод опорных векторов	0.8883	0.8551	0.8642	0.8524
Случайный лес	0.8854	0.8454	0.8378	0.8330
Наивный байесовский метод	0.8445	0.8097	0.7877	0.7882
Линейный дискриминантный анализ	0.8737	0.8667	0.8754	0.8543

Из табл. 2 можно сделать следующие выводы:

- Точность (accuracy) во всех случаях выше остальных метрик, т.к. в наборе данных много классов, представленных в разном

количестве, а метрика точность не учитывает ложноположительных и ложноотрицательных результатов

- Худший результат показал метод k-ближайших соседей, т.к. он самый примитивный. Схожий результат у наивного байесовского метода, его особенностью является быстрая работа с меньшей точностью результата
- Лучший результат показал метод опорных векторов, т.к. он хорошо работает в пространствах большой размерности. Максимальная достигнутая точность данного метода при обучении составила 91.97%, что является прекрасным результатом

Таким образом, даже при применении достаточно простых алгоритмов классификации со стандартными параметрами из библиотеки scikit-learn были достигнуты приемлемые точности классификации, что свидетельствует о том, что выделенные признаки действительно значимы и могут использоваться для решения практических задач, таких как классификация.

3.4 Исследование выделенных признаков

Для проведения классификации в предыдущем разделе все серии спектров были преобразованы в вектора признаков размером 64. Возьмем первую серию спектров из набора данных (рис. 20)

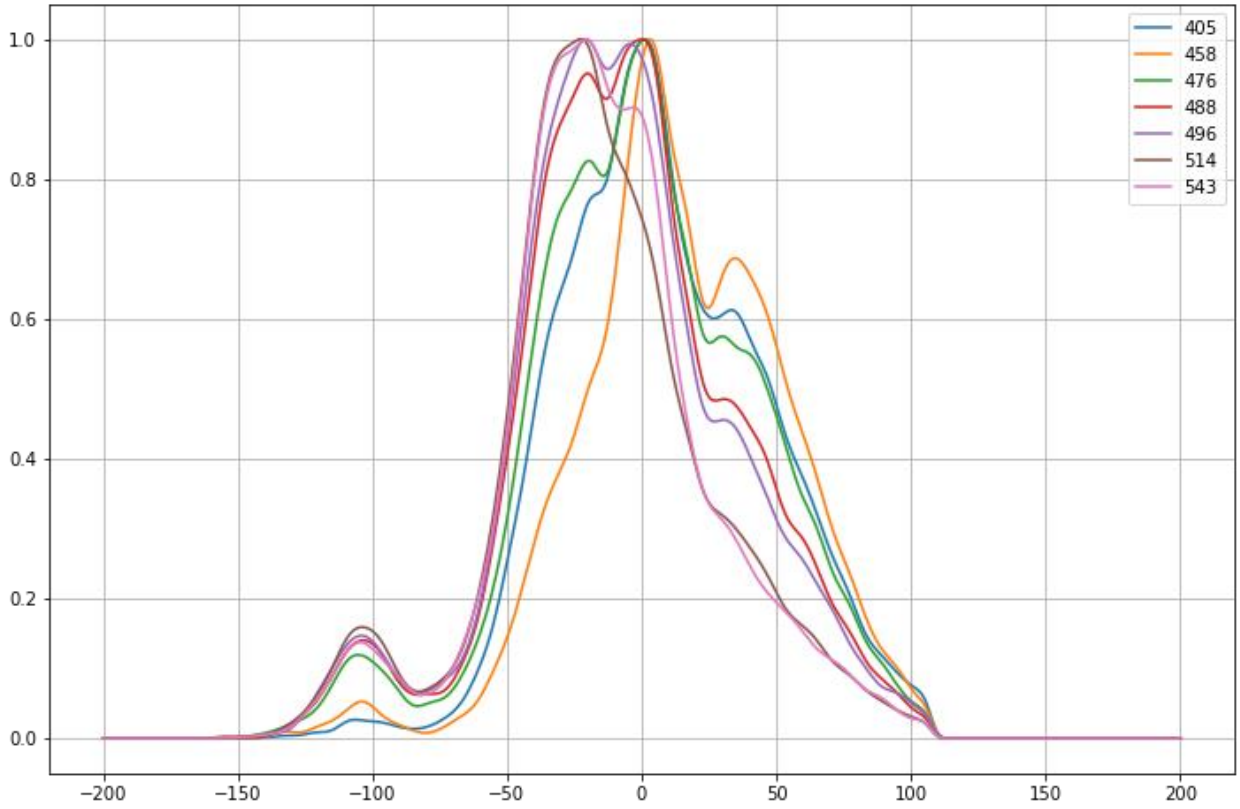

Рисунок 20 - Первая серия спектров из набора данных

Из данной серии спектров были выделен следующий вектор признаков (рис. 21).

```
array([-0.99266183, -0.99729055,  3.075067  , -0.9794937 , -0.605777  ,
       -0.9938787 ,  3.8848064 ,  1.2573565 , -0.8114589 ,  7.2926555 ,
       -0.99889183,  5.5185986 , -0.87234104,  4.1647778 , -0.99870414,
        2.7585442 , -0.99723065,  5.9743266 , -0.9997629 , -0.7418927 ,
        1.358774  ,  6.064234  , -0.9993083 , -0.04230851, -0.9706859 ,
       -0.073421  ,  1.3022487 ,  1.5708218 , -0.99760276, -0.9993394 ,
        3.66295   , -0.96893966,  5.375111  , -0.9978033 ,  4.923403  ,
        3.936654  , -0.9998771 , -0.9984808 ,  5.5795774 , -0.9999721 ,
       -0.99987954, -0.999131  ,  6.2617345 ,  1.9881719 , -0.86038584,
        4.32771   , -0.9869506 ,  1.0124575 , -0.7454656 ,  7.174304  ,
       -0.9430481 , -0.405627  ,  2.9615297 , -0.99927145,  7.974781  ,
       -0.76618207,  3.1820464 , -0.9979615 ,  3.1741073 ,  7.663921  ,
        6.1466684 , -0.9977458 , -0.99809605,  0.6131362 ], dtype=float32)
```

Рисунок 21 - Выделенные признаки

Сеть-декодировщик из данных признаков восстанавливает следующую серию спектров (рис. 22).

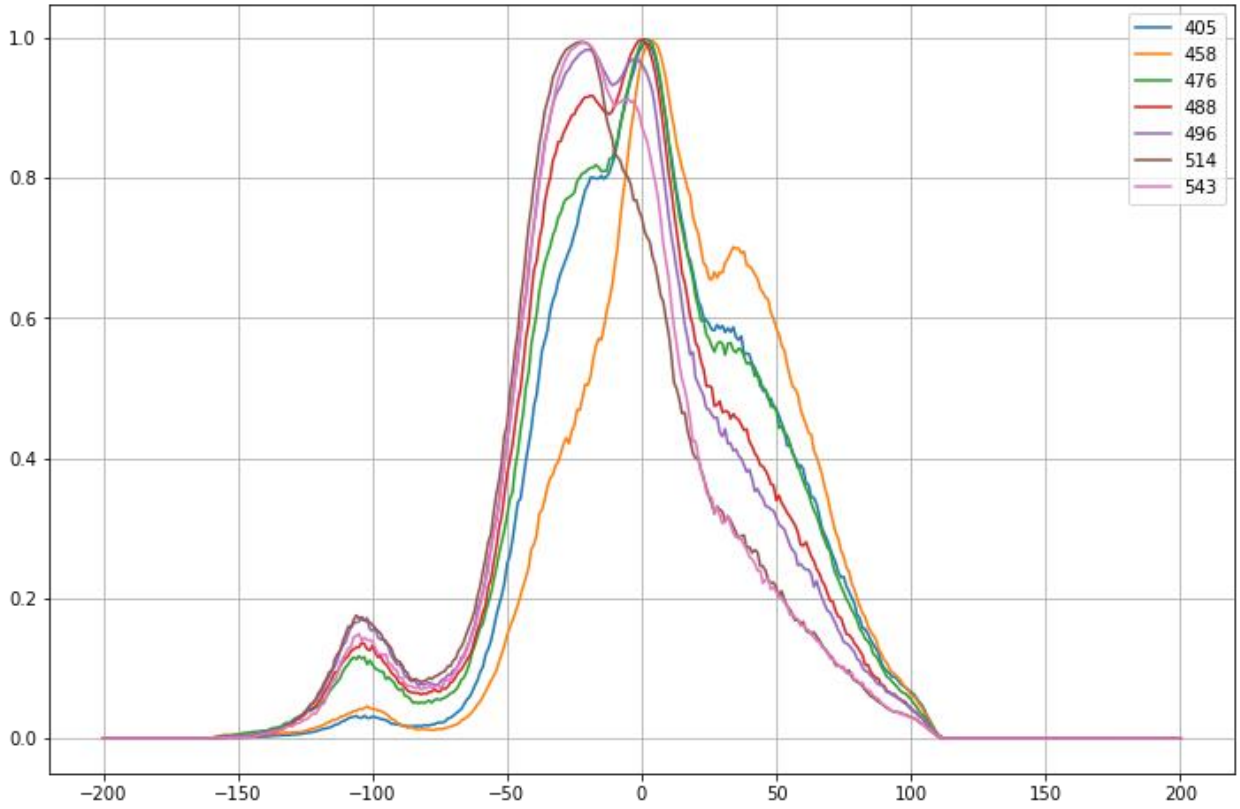

Рисунок 22 - Восстановленная серия спектров

Видно, что восстановленная серия спектров соответствует исходной, при этом присутствуют небольшие погрешности в виде шума.

Таким образом, выделенные признаки не только можно использовать для решения практических задач, но и восстанавливать из них исходную серию спектров. Рассмотрим подробнее выделенные признаки.

3.4.1 Характеристики признаков

Выделенные признаки не подчиняются нормальному распределению (p-value << 1). Это неудивительно, так как выбранная модель автокодировщика не способна контролировать распределение выделяемых признаков. Для этого используются вариационные автокодировщики, не подходящие для задачи выделения признаков и не рассматриваемые в данной работе.

Рассчитаем для каждого из 64 признаков среднее значение, медиану, диапазон и среднеквадратичного отклонение. После этого, отсортируем признаки по убыванию отклонения. Предполагается, что признаки с наибольшим отклонением кодируют наиболее значимые данные, по которым

серии спектров сильнее всего различаются. 15 признаков с наибольшим отклонением представлены на рис. 23.

	mean	median	min	max	range	std_dev
59	5.186203	4.286103	-0.749348	13.634843	14.384192	3.734059
38	5.065187	4.675764	-0.852354	15.828714	16.681068	3.460859
9	5.314063	5.012269	-0.744578	15.720603	16.465181	3.329548
60	6.388295	7.490508	-0.630768	13.558391	14.189158	3.271975
6	3.519131	3.566488	-0.997535	14.352055	15.349590	3.168456
58	2.306315	1.722854	-0.979052	14.648397	15.627449	3.061054
32	5.725231	6.316377	-0.967861	10.495151	11.463012	2.555536
8	0.304233	-0.853090	-0.994210	12.245433	13.239642	2.524314
42	5.356998	5.110942	-0.785899	13.364835	14.150734	2.462658
25	2.489779	3.021027	-0.998511	8.675647	9.674158	2.452663
34	5.031476	4.858825	-0.999238	11.675266	12.674503	2.423459
13	2.960796	2.750903	-0.947535	9.917129	10.864663	2.391432
21	7.012331	7.514291	-0.954434	11.104239	12.058674	2.363825
11	5.797557	5.954314	-0.983175	11.022033	12.005207	2.356703
56	3.814904	3.618156	-0.864706	9.738698	10.603404	2.319594

Рисунок 23 - Наиболее значимые признаки

Построим гистограмму отклонений всех признаков (рис. 24).

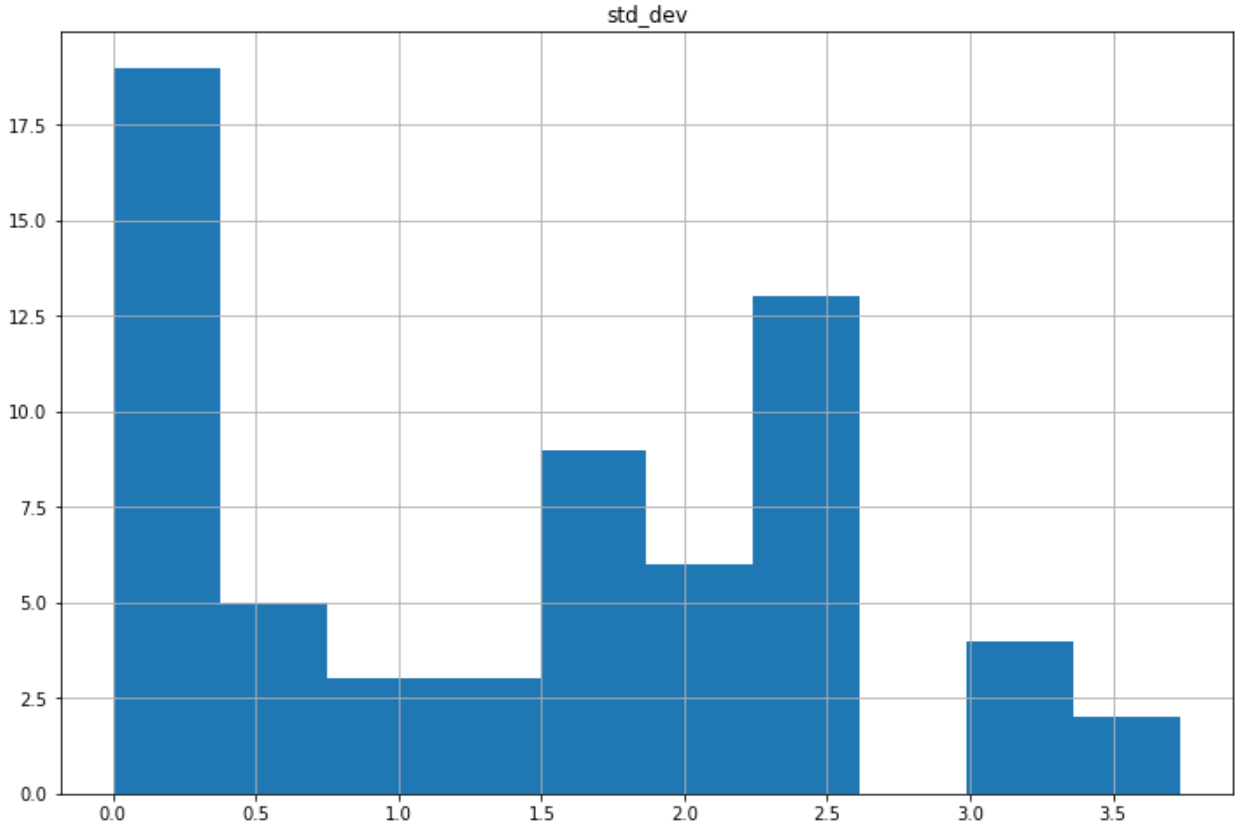

Рисунок 24 - Гистограмма среднеквадратичных отклонений

На этих рисунках можно выделить две интересные группы признаков:

- 6 признаков с самым большим отклонением (более 3.0): 59, 38, 9, 60, 6, 58
- 19 признаков, наименее отличающихся между входными данными. 17 из них имеют отклонение меньше 0.06

3.4.2 Анализ признаков

Возьмем случайную серию спектров и установим 59 признак, имеющий самое большое отклонение, в минимальное, среднее и максимальное значения. После каждого изменения подадим вектор признаков на вход декодировщика. Результаты на рис. 25 – рис. 27.

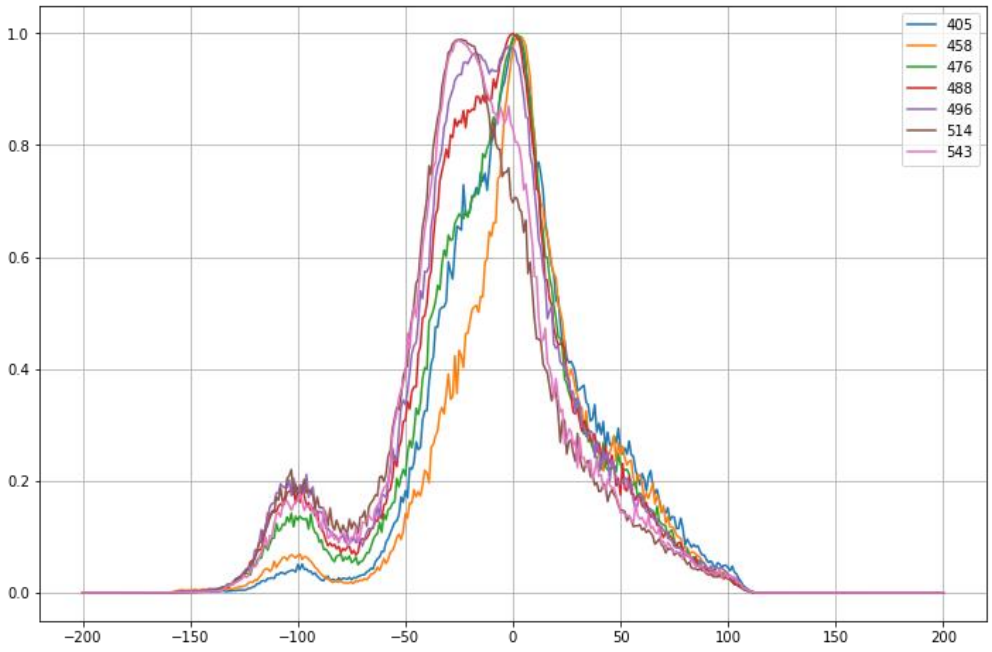

Рисунок 25 - Минимальное значение признака 59

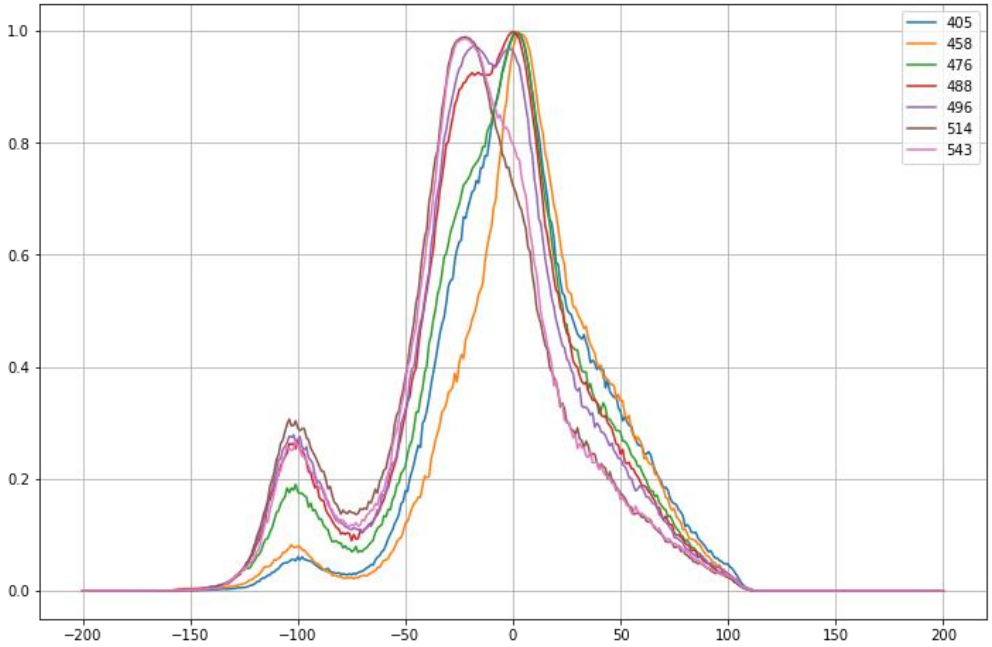

Рисунок 26 - Среднее значение признака 59

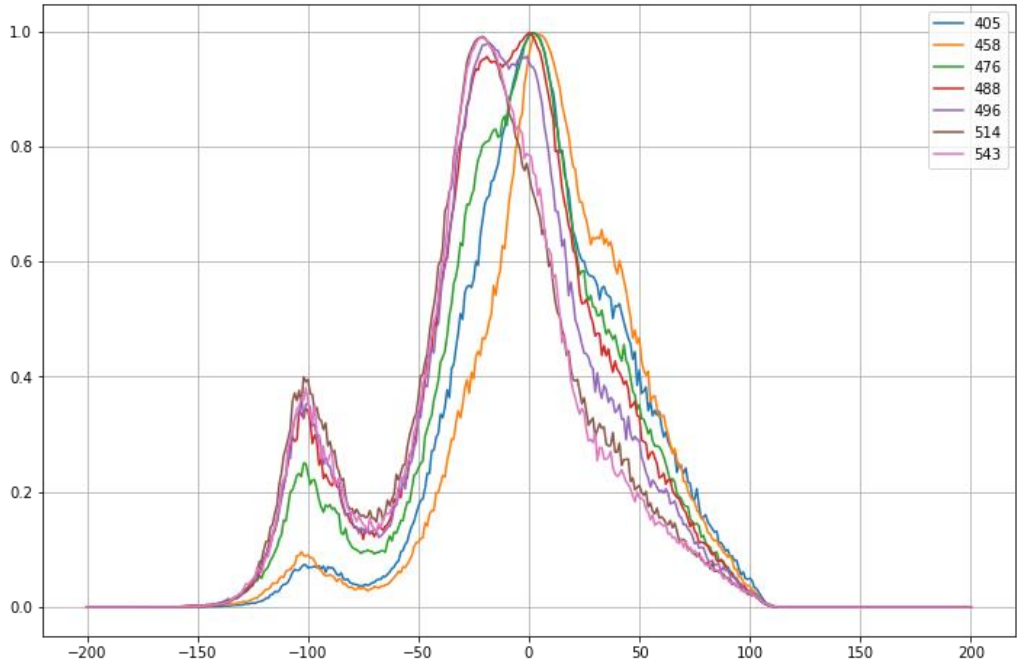

Рисунок 27 - Максимальное значение признака 59

Из данных рисунков можно сделать следующие выводы:

- По мере приближения значения признака к максимальному или минимальному значению, шум в выходных данных декодировщика усиливается, что свидетельствует о наличии взаимной зависимости признаков

- Признак 59 сильнее всего влияет на возвышение линий спектров с большой длиной волны в районе точки -100

Применим аналогичные операции к признаку с номером 38. Результаты на рис. 28 – рис. 30.

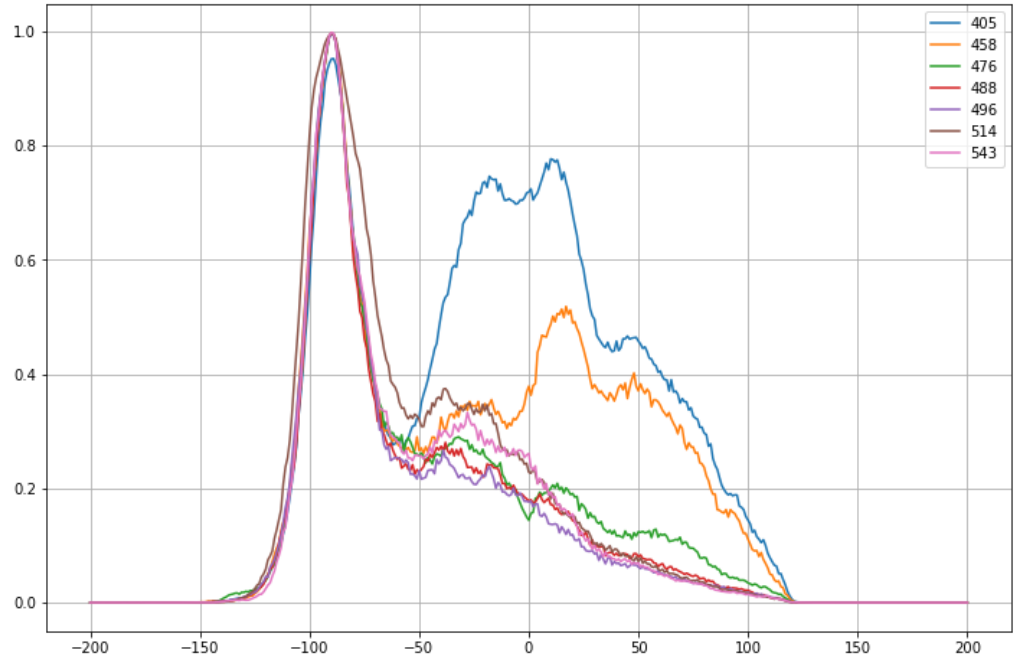

Рисунок 28 - Минимальное значение признака 38

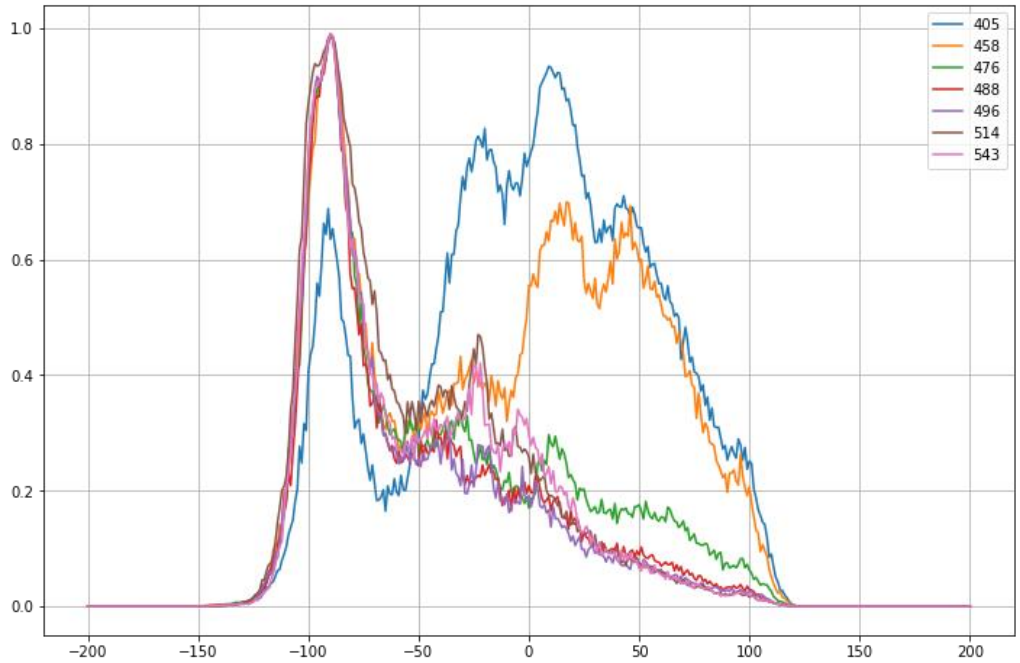

Рисунок 29 - Среднее значение признака 38

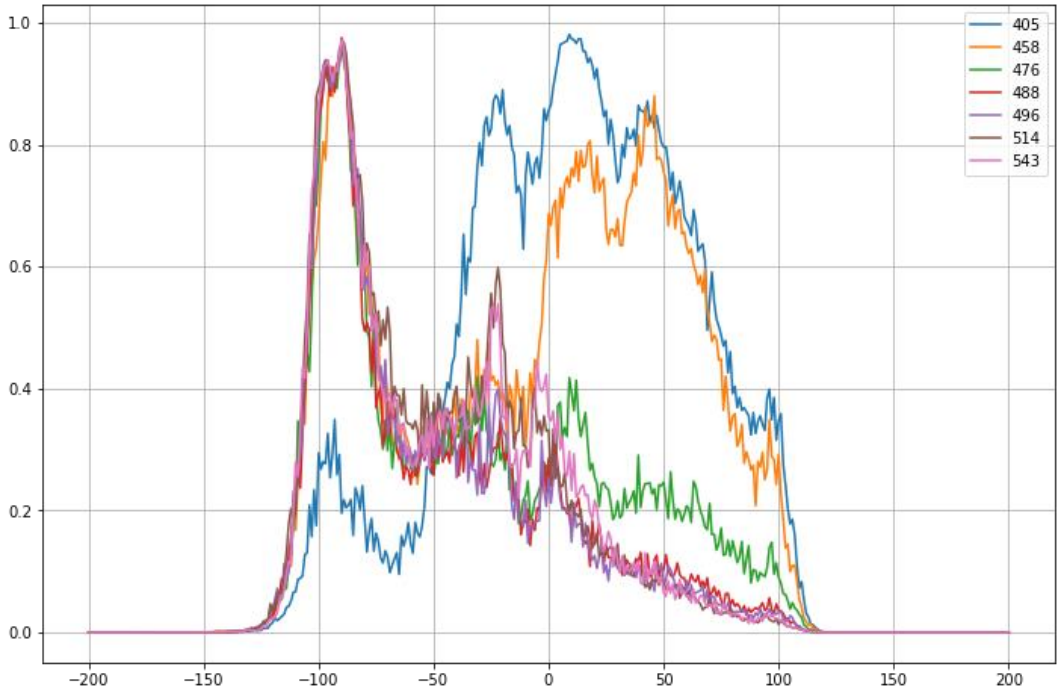

Рисунок 30 - Максимальное значение признака 38

Из рисунков видно, что увеличение данного признака влияет на спектры полученные с помощью лазеров с меньшей длиной (особенно 405 нм). Пик спектра, выделенного синим, в окрестности точки -100 уменьшается. В отличие от предыдущего случая, шум при декодировании сильнее всего увеличивается по мере удаления значения признака от изначального значения, что также свидетельствует о взаимной зависимости признаков.

При аналогичном рассмотрении признака с номером 6, кажется, что он отвечает за плотность линий спектра, особенно на интервале от 25 до 100, а изменения признака 9 не дали понять, за что конкретно он отвечает.

Очевидно, что признаки зависимы друг от друга. Построим совместные распределения шести признаков с самым большим отклонением, а также гистограммы их распределений. Результат на рис. 31.

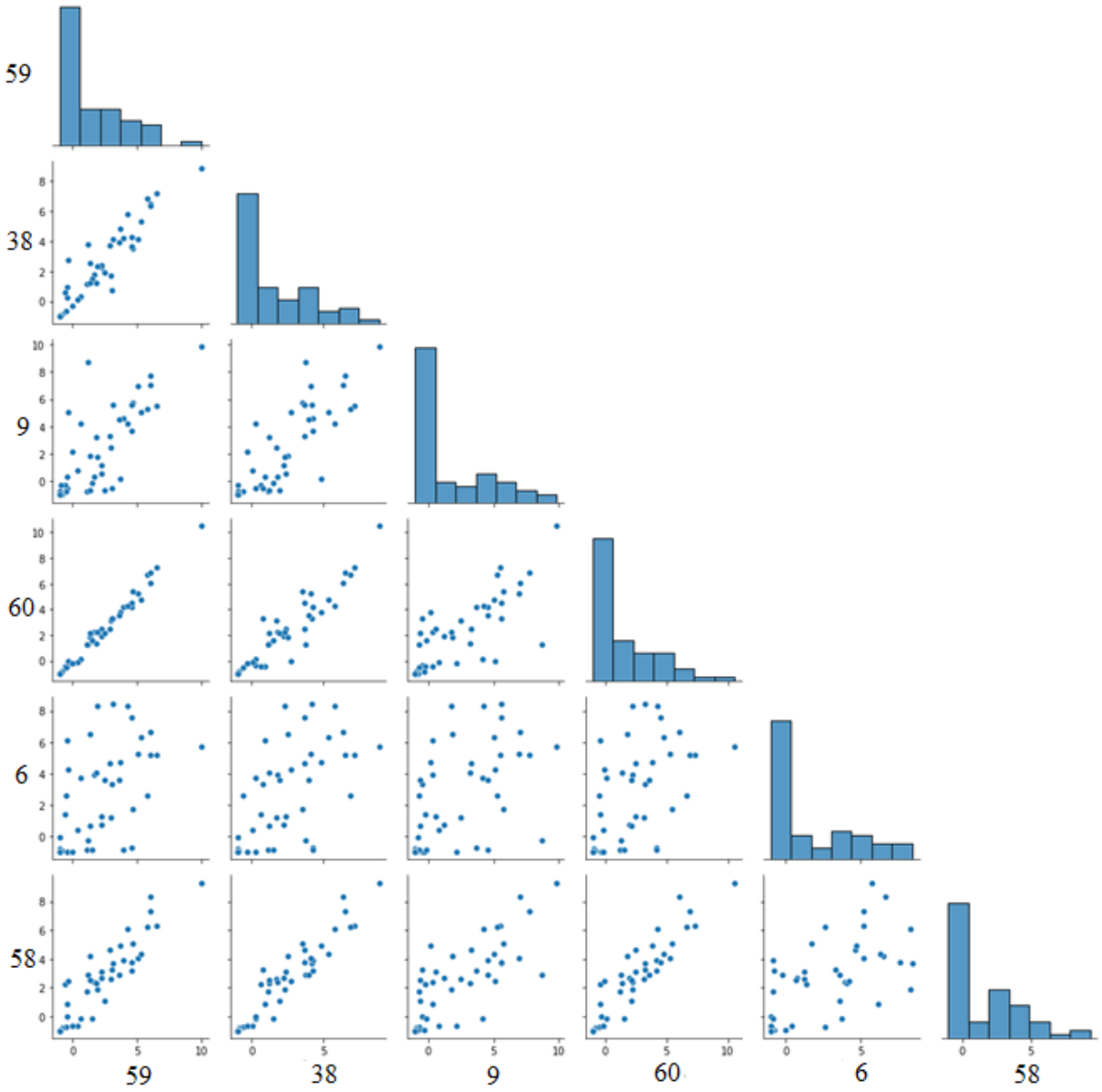

Рисунок 31 - Совместные распределения признаков по номерам

Из данного рисунка можно сделать следующие выводы:

- Самые важные из выделенных признаков в большинстве случаев имеют значения, близкие к нулю, что подтверждает гипотезу из первой части данной главы о том, почему применение регуляризации L_1 ухудшает работу модели
- Между большинством признаков присутствует зависимость. Это связано с использованием автокодировщика. Как уже упоминалось в первой главе, одна из особенностей

автокодировщиков – зависимость между выделяемыми признаками, в отличие от, например, PCA.

3.4.3 Анализ декодировщика

Подадим на вход декодировщика вектор из 64 нулей. Результат на рис. 32.

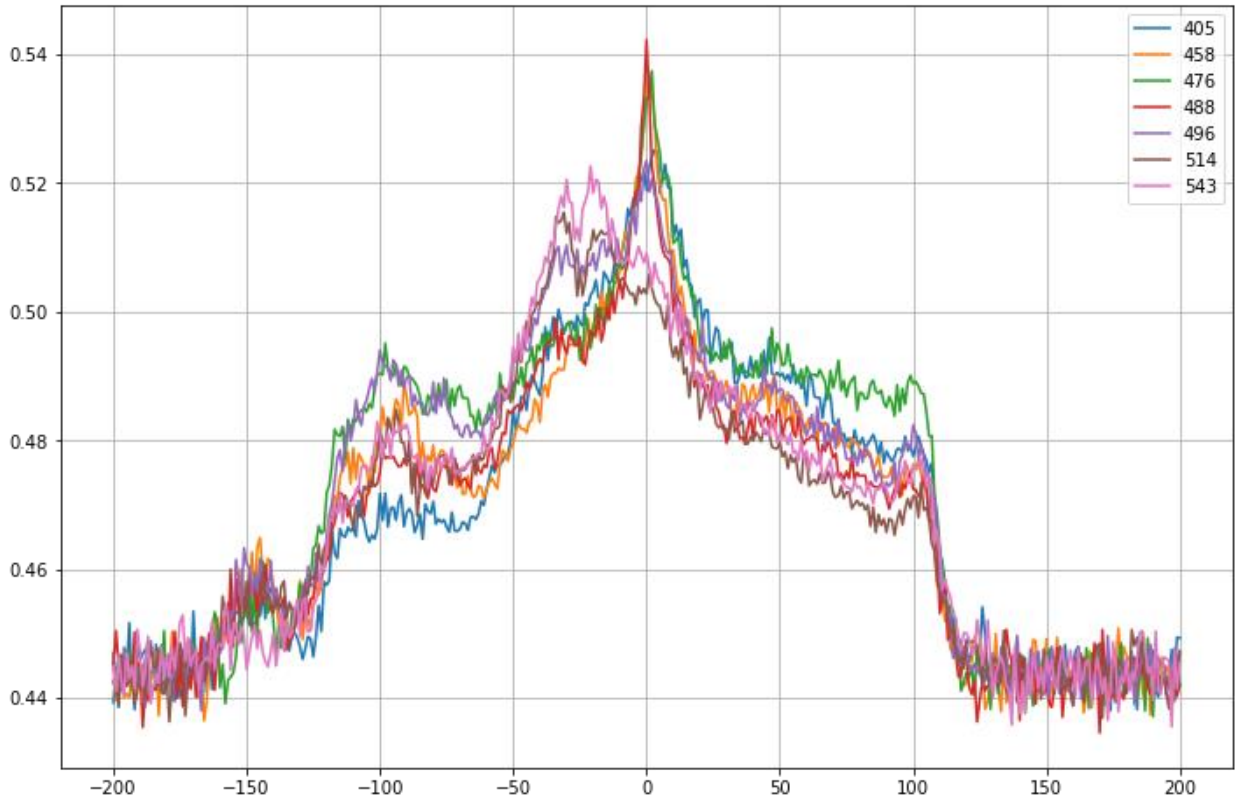

Рисунок 32 - Вывод декодировщика при нулевом входе

На этом рисунке заметны некоторые особенности серий спектров из набора данных – пик в точках -100, 0, -150, общая форма графика. При этом, края линий спектров не равны нулю. Это говорит о том, что информация о данных, на которых обучается автокодировщик, содержится как в выделенных признаках, так и в весах декодировщика.

Теперь подадим на вход декодировщика средние значения признаков. Результат на рис. 33.

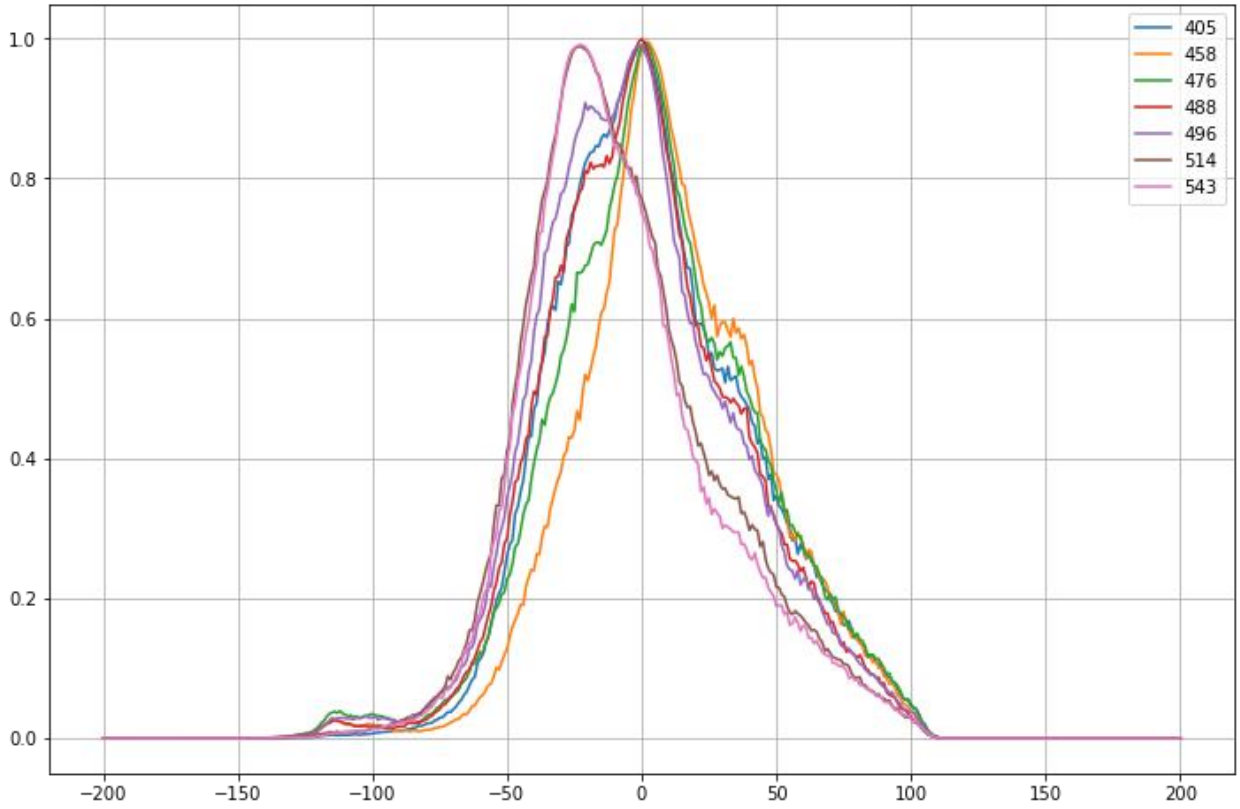

Рисунок 33 - Вывод декодировщика при подаче средних значений признаков

Вывод выглядит как стандартная серия спектров из набора данных.

Подадим на вход декодировщика минимальные и максимальные значения признаков (рис. 34 и 35).

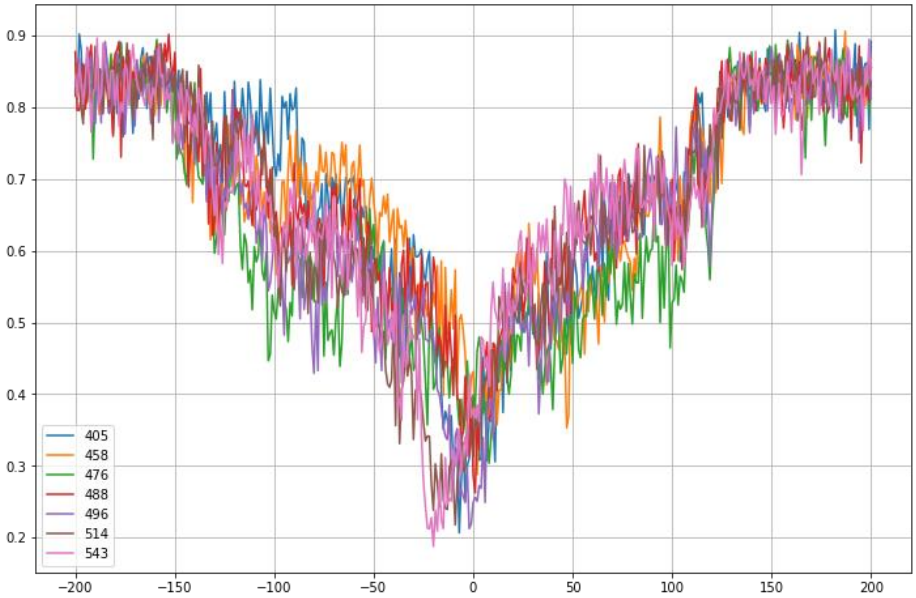

Рисунок 34 - Вывод декодировщика при подаче минимальных значений признаков

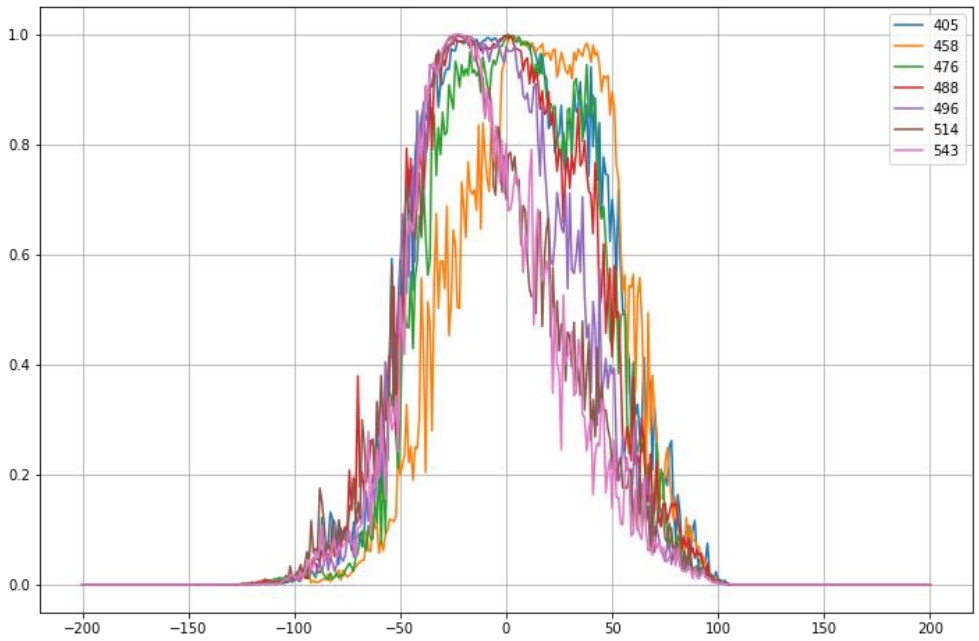

Рисунок 35 - Вывод декодировщика при подаче максимальных значений признаков

Рисунок 34 совершенно не похож на серию спектров из набора данных. Связано это с тем, что минимальные значения признаков, близкие к -1, обусловлены выбором функции активации.

При установке значений признаков на максимальные, все пики спектров сливаются. Как было показано в прошлом разделе, важной информацией в признаках часто является наличие пиков графиков в определенных точках. За эти пики отвечает определенное соотношение признаков. При этом, максимальные значения признаков наблюдаются достаточно редко и отвечают за «нестандартные» пики. Поэтому установка всех признаков к максимальному значению сливает все возможные пики в один большой.

3.4.4 Вывод

Таким образом, из анализа выделенных признаков и сети-декодировщика, можно сделать следующие выводы:

- Информация об особенностях входных данных, полученная нейросетевым автокодировщиком в процессе обучения, содержится как в выделенных признаках, так и в весах сети-декодировщика

- Хотя выделенные признаки хорошо описывают входные данные и могут использоваться в практических задачах, не все они являются содержательными. Это особенность работы автокодировщиков
- Изменение наиболее значимых признаков при их декодировании влияет в разной степени на группы спектров разной длины, т.е. важные признаки во входных данных исходят из сочетаний значений различных спектров
- Выделенные признаки кодируют сложные зависимости, которые не удалось бы выделить при помощи стандартных алгоритмов, таких как PCA. При этом, выделенные признаки зависят друг от друга, и в большинстве случаев нельзя однозначно утверждать, за что конкретно отвечает отдельный признак. Это еще одна особенность нейросетевых автокодировщиков.

4. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР

4.1 Концепция экономического обоснования

Данная выпускная квалификационная работа посвящена разработке модели нейросетевого автокодировщика, способного выделять важные признаки из серии спектров собственной флуоресценции цианобактерий. Эти признаки в дальнейшем могут быть использованы для решения прикладных задач в области экологии.

Актуальность разработки обосновывается большой размерностью результатов работы спектрофотометрических методов, что затрудняет обработку данных.

В разделе экономического обоснования рассчитываются:

1. Трудоемкость выполнения ВКР.
2. Заработная плата разработчика и руководителя;
3. Отчисления на социальные нужды;
4. Затраты на сырье и приобретение расходных материалов;
5. Затраты на эксплуатацию и содержания оборудования;
6. Затраты на услуги сторонних организаций.
7. Амортизационные отчисления;
8. Накладные расходы

4.2 Трудоемкость выполнения работ

Определим перечень выполненных работ и их продолжительность.

В таблице 3 отражена трудоемкость работ по созданию ВКР.

Таблица 3 – Трудоемкость работ

№	Наименование работы	Длительность работы, чел.дн.	
		Руководитель	Исполнитель
1	Обзор литературы по теме работы	4	6
2	Построение моделей автокодировщиков	–	18
3	Выбор модели	–	3

4	Изучение применимости выделенных признаков	—	2
5	Изучение характеристик выделенных признаков	–	5
6	Оформление пояснительной записки	2	8
7	Оформление иллюстрационного материала	–	1
8	Защита проекта	3	3
ИТОГО		9	44

4.3 Заработная плата разработчика и руководителя

На статью «Расходы на оплату труда» относят заработную плату научных сотрудников, инженеров и прочего инженерно-технического персонала, непосредственно занятых выполнением работы.

В Приказе ректора № ОД/0539 от 27.09.2019 «Об увеличении уровня оплаты труда работникам университета и об изменении размеров минимальных должностных окладов и должностных окладов по профессионально-квалификационным группам», должностной оклад руководителя равен 39400,00 руб., исполнителя – 14000,00 руб.

Ставка заработной платы за единицу времени (день) на основе оклада за месяц и при количестве рабочих дней (21 рабочий день):

- для руководителя –1876,19 руб.;
- для исполнителя – 666,67 руб.

Расходы на основную заработную плату исполнителей определяются по формуле (1)

$$З_{\text{осн.з/пл}} = \sum_{i=1}^{k} T_i * C_i \qquad (1)$$

где $З_{зосн/.пл}$ – расходы на основную заработную плату исполнителей, руб.; k – количество исполнителей; T_i – время, затраченное i-м исполнителем на проведение исследования, дни или часы; C_i – ставка i-го исполнителя, руб./день или руб./час.

$$З_{\text{осн.з/пл}} = 9 * 1876{,}19 \ + 44 * \ 666{,}67 = 46219{,}19 \text{ руб.}$$

Согласно «Положению об оплате труда работников университета» от 15.10.2020, норматив дополнительной заработной платы составляет 8,3%.

Расходы на дополнительную заработную плату исполнителей определяются по формуле (2)

$$З_{\text{доп.з/пл}} = З_{\text{осн.з/пл}} * \frac{Н_{\text{доп}}}{100} \quad (2)$$

где $З_{доп.з/пл}$ – расходы на дополнительную заработную плату исполнителей, руб.; $З_{осн.з/пл}$ – расходы на основную заработную плату исполнителей, руб.; $Н_{доп}$ – норматив дополнительной заработной платы, %.

$$З_{\text{доп.з/пл}} = 46219{,}19 * 0{,}083 = \ 3836{,}19 \text{ руб.}$$

Итого:

$$З_{\text{з/пл}} = З_{\text{осн.з/пл}} + З_{\text{доп.з/пл}}$$

$$З_{\text{з/пл}} = 46219{,}19 + \ 3836{,}19 = 50055{,}38 \text{ руб.}$$

4.4 Отчисления на социальные нужды

На статью «Отчисления на социальные нужды» относят затраты, связанные с выплатой социальных отчислений с заработной платы.

Страховые взносы отчисляются в следующие государственные внебюджетные фонды социального назначения:

- Пенсионный фонд России (ПФР);
- Федеральный фонд обязательного медицинского страхования (ФФОМС);
- Фонд социального страхования (ФСС).

Отчисления на страховые взносы составляют 30,2% от основной и дополнительной заработной платы. Данные отчисления на страховые взносы рассчитываются по формуле (3)

$$З_{\text{соц}} = (З_{\text{доп.з/пл}} + З_{\text{осн.з/пл}}) * \frac{Н_{\text{соц}}}{100} \quad (3)$$

где $З_{соц}$ – отчисления на социальные нужды с заработной платы, руб.; $З_{осн.з/пл}$ – расходы на основную заработную плату исполнителей, руб.; $З_{доп.з/пл}$ – расходы на дополнительную заработную плату исполнителей, руб.; $Н_{соц}$ –

норматив отчислений страховых взносов на обязательное социальное, пенсионное и медицинское страхование, %.

$$З_{соц} = (46219{,}19 + 3836{,}19) * 0{,}302 = 15116{,}72 \text{ руб.}$$

4.5 Материалы

На статью «Материалы» относят затраты на сырье, основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, необходимые для выполнения работы с учетом транспортно-заготовительных расходов.

Произведем расчёт количества и стоимости материалов с учетом транспортно-заготовительных расходов по формуле (4), результаты представлены в табл. 4.

$$З_м = \sum_{l=1}^{L} G_l * Ц_l * (1 + \frac{Н_{т.з.}}{100}) \quad (4)$$

где $З_м$ – затраты на сырье и материалы, руб.; l – индекс вида сырья или материала; G_l – норма расхода l-го материала на единицу продукции, ед.; $Ц_l$ – цена приобретения единицы l-го материала, руб./ед.; $Н_{т.з.}$ – норма транспортно-заготовительных расходов, %.

$$З_м = 1790{,}00 * (1 + 0{,}1) = 1969{,}00 \text{ руб.}$$

Таблица 4 – Расходные материалы

Материалы	**Количество, ед.**	**Цена, руб.**	**Сумма, руб.**
Бумага офисная	1	439,00	439,00
Комплект картриджей для принтера	1	932,00	932,00
USB Флеш-накопитель	1	419,00	419,00
ИТОГО:			**1790,00**
Транспортные расходы (10%)			**179,00**
ВСЕГО:			**1969,00**

4.6 Спецоборудование

На статью «Спецоборудование» относятся затраты на приобретение (или изготовление) специальных приборов, стендов, другого специального оборудования, необходимого для выполнения работы.

В данной работе расходы на спецоборудование не предусмотрены.

4.7 Расходы на содержание и эксплуатацию оборудования

На статью «Расходы на содержание и эксплуатацию оборудования» относят затраты на содержание и эксплуатацию всех видов оборудования, используемого в работе.

Рассчитывается по формуле (5):

$$З_{эо} = \sum_{i=1}^{m} C_i^{мч} * t_i^{м} \quad (5)$$

где $З_{эо}$ – затраты на содержание и эксплуатацию оборудования, руб.; $C_i^{мч}$ – расчетная себестоимость одного машино-часа работы оборудования на i-й технологической операции, руб./м-ч; $t_i^{м}$ – количество машино-часов, затрачиваемых на выполнение i-й технологической операции, м-ч.

Длительность работ с компьютером составила 445 ч, с принтером – 1 ч.

Стоимость 1 кВт/ч по тарифу – 5,42 руб.

Компьютер потребляет 0.65 кВт/ч, принтер – 0.02 кВт/ч.

Таким образом стоимость одного машино-часа компьютера:

$$С_к = 0{,}65 * 5{,}42 = 3{,}52 \frac{руб}{м.ч}$$

Стоимость одного машино-часа принтера:

$$С_п = 0{,}02 * 5{,}42 = 0{,}11 \frac{руб}{м.ч}$$

Затраты на содержание и эксплуатацию оборудования составили:

$$З_{эо} = 3{,}52 * 445 + 0{,}11 * 1 = 1566{,}51 \text{ руб.}$$

4.8 Затраты по работам, выполняемым сторонними организациями

На статью «Затраты по работам, выполняемым сторонними организациями», относят затраты по оплате всех видов работ, выполняемых сторонними организациями.

Для выполнения работы использовался доступ в сеть Интернет. Стоимость услуги составила 550,00 руб. в месяц. Количество целых месяцев составляет 2. Стоимость без НДС – 458,00 руб. Таким образом, затраты на оказание услуги составили

$$З_{\text{без НДС}} = З_{\text{с НДС}} * (1 - \frac{20\%}{120\%})$$

$$З_{\text{и}} = 2 * 458{,}00 = 916{,}00 \text{ руб.}$$

4.9 Амортизационные отчисления

В статье «Амортизационные отчисления» учитываются амортизационные отчисления по всем видам основных средств, используемым при выполнении ВКР.

Амортизационные отчисления по основному средству i за год определяются как:

$$А_i = Ц_{\text{п.н.}i} * \frac{H_{ai}}{100}$$

Стоимость компьютера составляет 57000,00 руб., принтера – 4000,00 руб.

Согласно Постановлению Правительства РФ от 01.01.2002 №1 (ред. от 07.07.2016) «О классификации основных средств, включаемых в амортизационные группы», нормативный срок полезного использования используемого оборудования 2 – 3 года.

Определим годовую норму амортизации как обратную величину от срока полезного использования, умноженную на 100%:

$$Н_{ai} = 100\% * \frac{1}{2} = 50\%$$

Определим амортизационные отчисления за год по основным средствам:

$$А_{\text{к}} = 57000{,}00 * \frac{50}{100} = 28500{,}00 \text{ руб.}$$

$$А_{\text{п}} = 4000{,}00 * \frac{50}{100} = 2000{,}00 \text{ руб.}$$

Для определения величины амортизационных отчислений по основным средствам, используемым в процессе выполнения ВКР, необходимо определить время, в течение которого использует основное средство по формуле

$$А_{i\text{ВКР}} = А_i * \frac{T_{i\text{ВКР}}}{12}$$

где $А_{i\text{ВКР}}$ – амортизационные отчисления по i-му основному средству, используемому в работе над ВКР в руб.; $А_i$ – амортизационные отчисления за

год по i-му основному средству в руб.; $T_{i\text{ВКР}}$ – время, в течение которого использовалось i-ое основное средство, мес.

Основные средства использовались в процессе выполнения ВКР в течение 2-х месяцев.

$$\text{А}_{\text{кВКР}} = 28500{,}00 * \frac{2}{12} = 4750{,}00 \text{ руб.}$$

$$\text{А}_{\text{пВКР}} = 2000{,}00 * \frac{2}{12} = 333{,}33 \text{ руб.}$$

Итого:

$$\text{А}_{\text{ВКР}} = 4750{,}00 + 333{,}33 = 5083{,}33 \text{ руб.}$$

4.10 Накладные расходы

В статью «Накладные расходы» включаются расходы на управление и хозяйственное обслуживание (величина накладных расходов определяется на основе норматива, установленного в конкретной организации, где производится выполнение работ).

Норматив накладных расходов примем равным 20 %, высчитывается от расходов на оплату труда

$$\text{З}_{\text{н}} = 20\% * (\text{З}_{\text{осн.з/пл}} + \text{З}_{\text{доп.з/пл}})$$

$$\text{З}_{\text{н}} = 20\% * 50055{,}38 = 10011{,}08 \text{ руб.}$$

4.11 Полная себестоимость

Расчет себестоимости приведен в табл. 5.

Таблица 5 – Калькуляция затрат на ВКР

№ п/п	Наименование статьи	Сумма, руб.
1.	Расходы на оплату труда	50055,38
2.	Отчисления на социальные нужды	15116,72
3.	Материалы	1969,00
4.	Затраты по работам, выполняемым сторонними организациями	916,00
5.	Расходы на содержание и эксплуатацию оборудования	1566,51
6.	Амортизационные отчисления	5083,33

7.	Накладные расходы	10011,08
8.	Спецоборудование	–
ИТОГО		84718,02

4.12 Вывод

В разделе «Экономическое обоснование ВКР» рассчитана себестоимость разработки модели нейросетевого автокодировщика, способного выделять важные признаки из серии спектров собственной флуоресценции цианобактерий.

Были рассчитаны следующие статьи калькуляции:

- Расходы на оплату труда;
- Отчисления на социальные нужды;
- Материалы;
- Затраты по работам, выполняемым сторонними организациями;
- Расходы на содержание и эксплуатацию оборудования;
- Амортизационные отчисления;
- Накладные расходы.

Определена себестоимость разработки проекта 84718,02 руб.

В последние годы существует глобальный тренд на заботу об экологии, экологические проекты, из чего можно сделать вывод о целесообразности данной работы.

ВЫВОДЫ

В ходе выполнения данной работы были выделены признаки из спектров флуоресценции, чтобы их можно было в дальнейшем использовать в различных прикладных задачах. Данная цель была решена при помощи особой архитектуры искусственных нейронных сетей – автокодировщика.

Для выполнения данной цели были построены различные модели автокодировщика, изучено влияние параметров моделей на качество их работы, выявлена оптимальная модель, изучены выделенные признаки и возможность их применения в прикладных задачах.

В результате была получена модель автокодировщика, способная автоматически выделять полезные для решения экологических задач признаки, что облегчит работу экологов.

СПИСОК ЛИТЕРАТУРЫ

1. Discover Feature Engineering, How to Engineer Features and How to Get Good at It [Electronic resource]. URL: https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/ (accessed: 26.05.2022).
2. Springer I.T.J. Principal Component Analysis, Second Edition.
3. Т. Р. Жангиров, А. С. Перков, А. А. Лисс. Применение линейного дискриминантного анализа для классификации цианобактерий по спектрам собственной флуоресценции.
4. Zhang C. et al. Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status // Journal of Control Science and Engineering. Hindawi Limited, 2018. Vol. 2018.
5. Vincent P. et al. Extracting and Composing Robust Features with Denoising Autoencoders.
6. Lecun Y. et al. Gradient-Based Learning Applied to Document Recognition.
7. Masci J. et al. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction.
8. Ng A.Y. Feature selection, L 1 vs. L 2 regularization, and rotational invariance.
9. Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. 2018.
10. Maggipinto M. et al. A Convolutional Autoencoder Approach for Feature Extraction in Virtual Metrology // Procedia Manufacturing. Elsevier B.V., 2018. Vol. 17. P. 126–133.
11. Yuvalı M., Yaman B., Tosun Ö. Classification Comparison of Machine Learning Algorithms Using Two Independent CAD Datasets // Mathematics. MDPI, 2022. Vol. 10, № 3.