# Отчет по тестовому заданию

**ТЕСТОВОЕ ЗАДАНИЕ**

Этап 1.

1. Изучить паспорт торговой модели a1, предоставленный Нанимателем.
2. Внести предложения о возможных дальнейших направлениях для улучшения результата бизнес метрик.
3. Указать на слабые места.
4. Оценить срок для воспроизведения работающего кода торговой модели (без самой реализации).

Этап 2.

1. Рассказать какой новый подход, кроме обычного one-shot learning, предлагается в статье https://arxiv.org/pdf/1503.03832.pdf, код на гите https://github.com/ikonushok/siamese-triplet.
2. Нарисовать блок схемы формирования триплетов для вариантов 3,4,5 вариантов кода.
3. Необходимо запустить код датасете, предоставленном Нанимателем. В датасете дано 4 класса, то есть, 2 класса buy и 2 класса sell (их нужно объединить в один датасет с 4мя классами [0, 1, 2, 3]).
4. Оформить код в гите, передать ссылку. Критерий: код запускается Нанимателем без ошибок и понятен.

# Этап 1. Анализ торговой модели

# 1. Изучение паспорта торговой модели a1

- Изучен файл Паспорт a1.pdf
- Изучена статья https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

Описание паспорта, саммари:

В качестве входных данных модели используются данные OHCLV с временным интервалом 20 минут для валютной пары eurusd. Для формирования анкеров используется кластеризация (k-means). Для предсказаний - сиамская нейронная сеть, обучение на малом количестве данных. Модель выбрана из-за ограничения небольшого количества данных паттернов. PyTorch используется для обучения сети (LazySNN.py - кастомная модель). Торговая система выделяет паттерны buy/sell с помощью Optuna по заранее заданным параметрам, проводит оптимизацию гиперпараметров, валидацию и оценку метрик (доходность,

просадка, торговые коэффициенты). Учитывается ограничение на торговлю в определенные дни (пятница), также Average True Range для определения уровней stop-loss (статический) и запрет на торговлю против тренда. Эффективность определяется по метрикам: доходность, просадка, Sharpe Ratio.

## 2. Предложения по улучшению бизнес-метрик

Марковские цепи и Байесовские сети доверия могут быть интегрированы в текущую систему в качестве дополнительных методов анализа. Эти модели могут генерировать сигналы buy/sell на основе вероятностных зависимостей цены, объемы торгов и др, затем сравниваться с предсказаниями сиамской нейросети, после чего приниматься торговое решение.

Марковские цепи позволяют оценивать вероятность перехода из одного состояния рынка в другое, байесовские сети позволяют оценить вероятность событий (рост, падение), и обновляются по мере поступления новых данных.

Оба подхода менее ресурсоемки по сравнению с нейросетями, не имеют проблемы переобучения, лучше интерпретируемы чем нейросети.

Источники:

A Bayesian Network Approach to Stock Market Trading

Bayesian Networks for Financial Market Prediction

Markov Chains in Financial Modeling

Application of Hidden Markov Models in Stock Market Prediction

## 3. Указание на слабые места

Слабые места торговой системы:

1. Слабая интерпретируемость результатов (проблема всех нейросетей), сложно понять почему нейросеть приняла то или иное решение. Как следствие - сложность принятия обоснованных торговых решений. Использование Байесовской сети и Марковских цепей как дополнительный подход, позволит иметь более понятный и что удобно - доступный для сравнения результат.

2. Необходимость часто повторяющегося регулярного обучения модели, для актуализации паттернов и предсказаний. Может быть затратно с точки зрения ресурсов. Решением может быть использование fine-tuning, для адаптации модели только к новым данным. Усложнит задачу, в случае изменения паттернов.Но можно сократить количество обучений, например делать базовое обучение чуть реже, и fine-tune в перерывах.

3. Проблема переобучения, недостаток данных. Необходимость использования сиамской нейросети из-за недостаточного набора данных. Обучение на малом объеме данных не позволяет использовать методы регуляризации (L1, L2, Dropout), это приводит к переобучению и низкой эффективности предсказаний на новых данных без дообучения.

   Возможное решение: Увеличение объема данных, увеличение кол-ва паттернов, полученных, например, из других валютных пар с высокой корелляицей к eurusd или создание искуственных данных на основе имеющихся. Появится возможность тестирования других видов нейронных сетей, возможность использования методов регуляризации - решение проблемы переобучения.

4. Модель очень кастомизирована и показывает положительный результат на маловолатильной паре eurusd. Это может быть плюсом, так как задача решается эффективно, но при торговле на других парах потребуется серьезная адаптация. Возможно пригодится разработка модульной архитектуры модели, для удобства работы с разными валютными парами.

# 4. Оценка срока для воспроизведения кода

*Оценка срока на воспроизведение кода: ~ 2.5 месяца*

1. Анализ и подготовка данных: 1-2 недели
   - Анализ данных, нормализация и стандартизация.
   - Формирование триплетов, подготовка датасетов.
2. Разработка архитектуры модели: 1-2 недели
   - Разработка и настройка архитектуры нейросети
   - Тестирование архитектуры, тестирование модели на небольших наборах данных.
3. Обучение и тестирование нейронной сети: 3-4 недели
   - Обучение модели, получение и оценка результатов
   - Подбор параметров, оптимизация (Optuna)
   - Тестирование оптимизированных моделей.
4. Риск-менеджмент и мета-алгоритмы: 1-2 недели

- Разработка и тестирование мета-алгоритмов для управления рисками.
- Поиск и внедрение “календарных фильтров”

5. Тестирование на исторических данных: 2-3 недели
    - Проверка модели на исторических данных.
    - Валидация модели на различных периодах времени.

---

# Этап 2. Реализация модели на основе статьи

## 1. Рассказать какой новый подход, кроме обычного one-shot learning, предлагается в статье

- Изучена статья: https://arxiv.org/pdf/1503.03832.pdf
- изучен код на гите: https://github.com/ikonushok/siamese-triplet

В статье описывается модель FaceNet - определят схожесть лиц основываясь на обучении представления лиц в виде точек в евклидовом пространстве (близкие точки - похожие лица, удаленные точки - разные лица).

Новый подход - negative mining: В процессе обучения используется метод постоянного усложнения триплетов по мере обучения. Также, положительный пример должен быть почти отрицательным, а отрицательный - почти положительным, то есть выбираются такие отрицательные примеры, которые максимально затрудняют обучение модели.

Обучение и оптимизация происходит с использованием триплетов (триплетной функции потерь triplet loss) - три изображения, исходное (анкор), позитивный пример - изображение того же лица, негативный пример - изображение другого лица.

Модель обновляет веса минимизируя расстояние между анкором (начальным) и положительным примером (схожим с анкором) и максимизирует расстояние между анкором и отрицательным примером (k-nn).

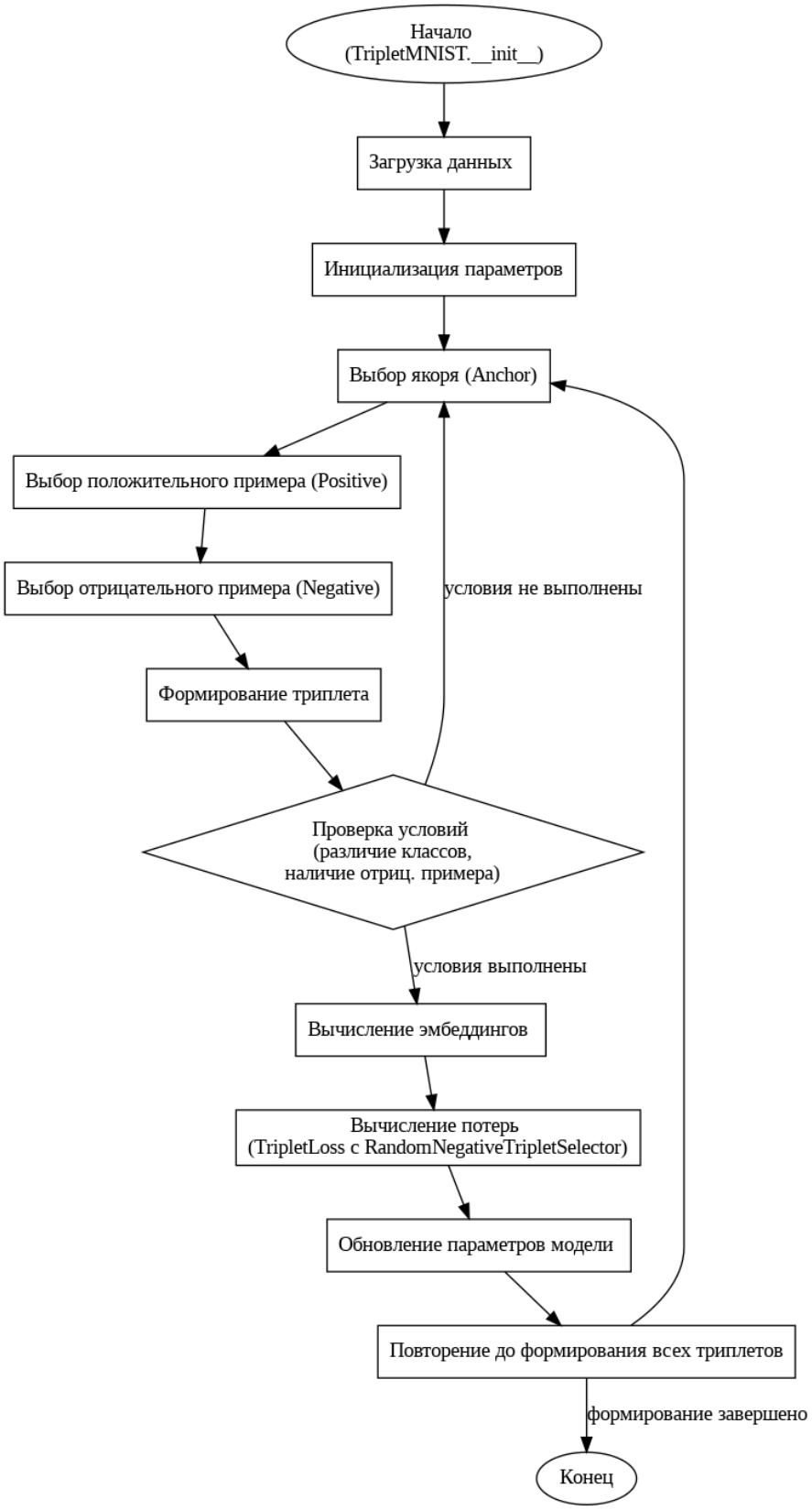
Также, отличие от предыдущих подходов - модель обучается выдавать сразу компактное embedding представление на начальных этапах модели и

кластеризовать. В отличие от других моделей, которые создают промежуточное представление большой размерности, что усложняет модель, так как нужно уменьшать размерность перед классификацией.

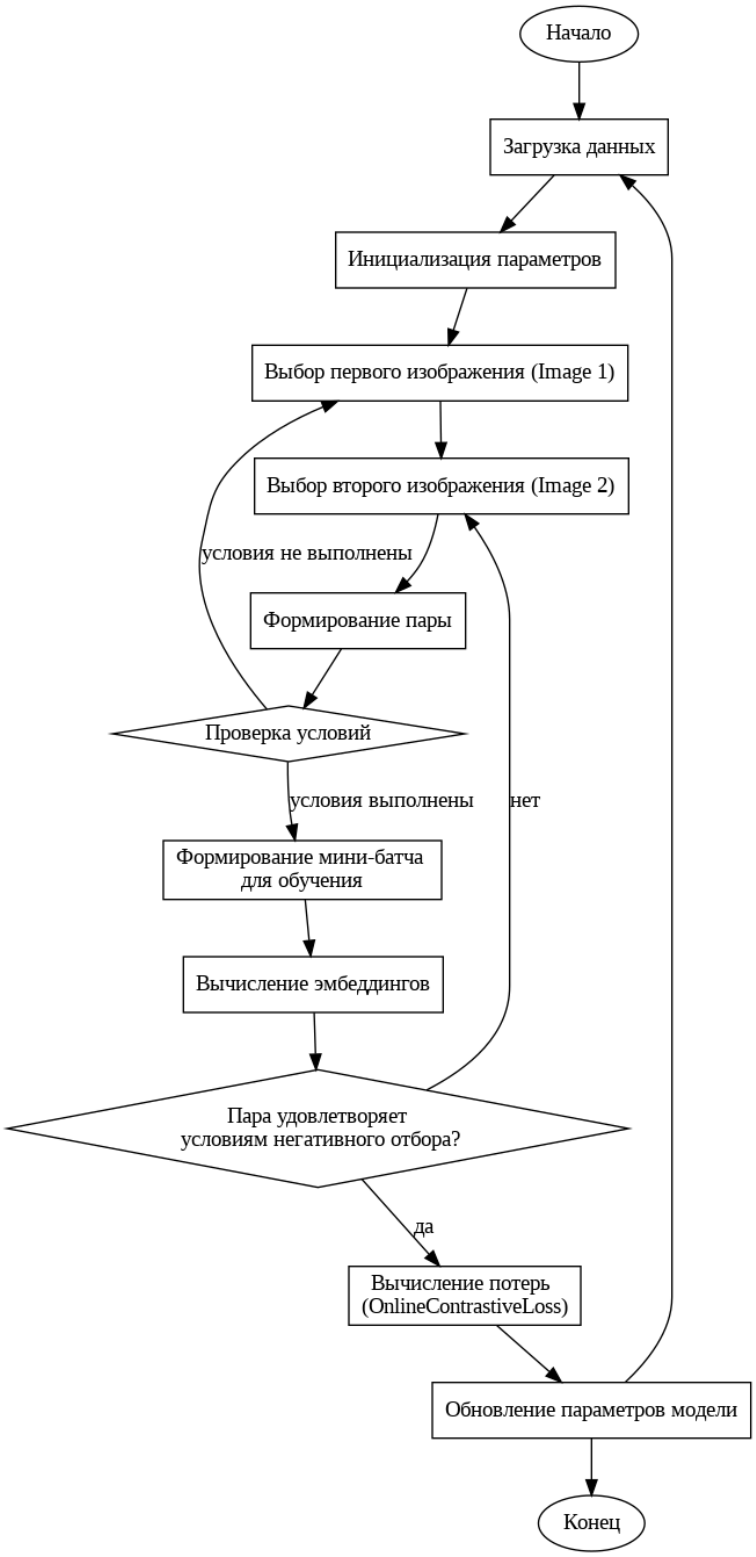
## 2. Блок-схемы формирования триплетов для вариантов кода 3, 4, 5

(сл. страница)

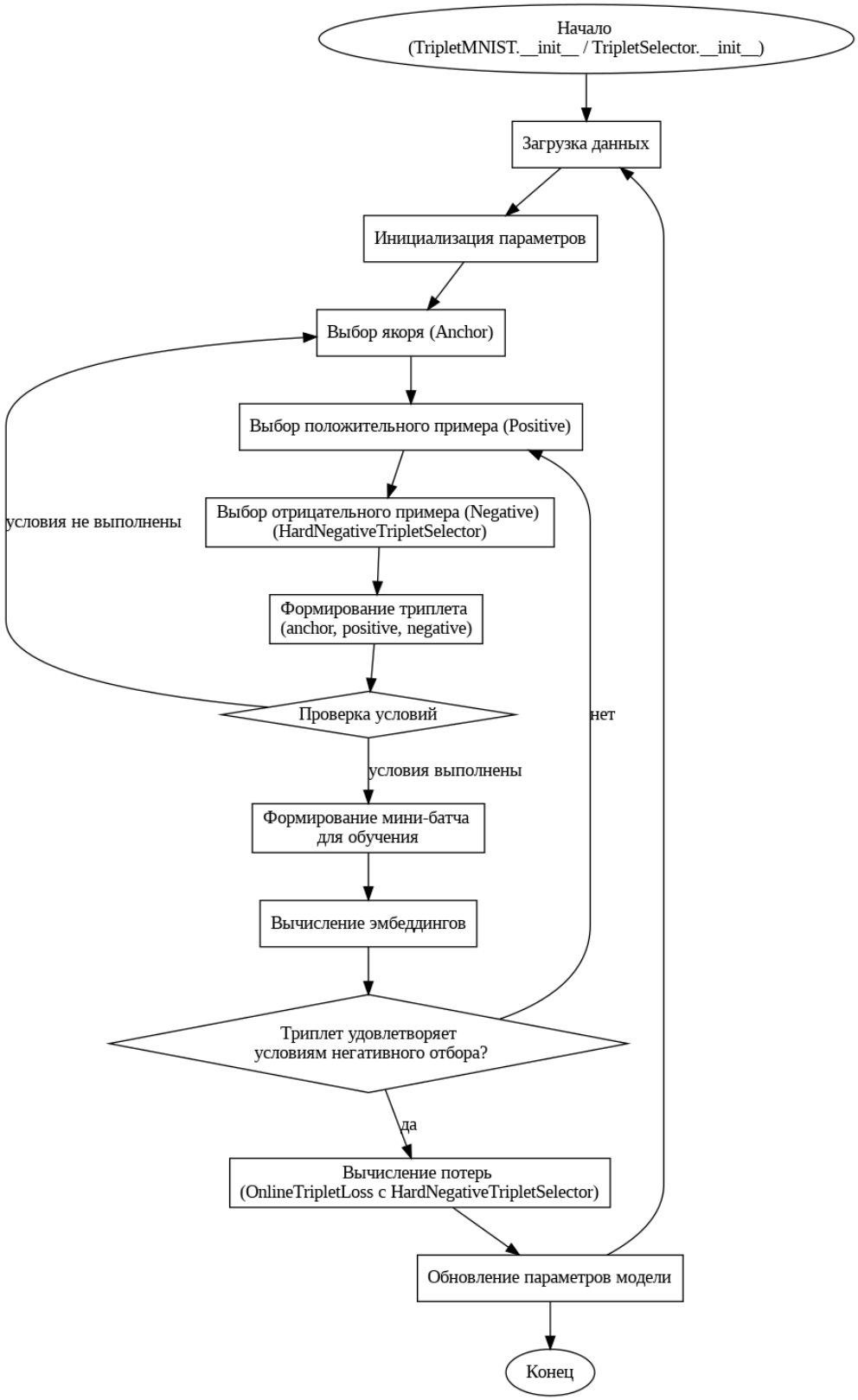
## 3. Сиамская сеть с триплетной потерей



## 4. Контрастивная потеря с негативным отбором



## 5. Триплетная потеря с негативным отбором



# 3. Запуск кода на предоставленном датасете

Описание процесса подготовки данных и запуска модели.

В файле main.py полученные данные из файлов объединяются в один датасет с 4-мя классами [0, 1, 2, 3], нормализуются по среднему значению и стандартному отклонению.

Для обработки кастомного датасета подготовлен класс CustomDataset в файле datasets.py. В этом классе данные и метки преобразуются в тензоры, разделяются на обучающую и тестовую выборки, трансформируются в массив.

**Подход 1: Использование исходной размерности данных**

Изначально данные передавались в оригинальной размерности [1, 60, 5], один канал с размером 60x5. Из-за того, что “изображение” получалось слишком “узким”, а размер сверточного ядра слишком большой, сверточные слои приводили к уменьшению размера тензора до нуля после операции max_pool2d в EmbeddingNet и ошибкам в обработке тензоров.

Изменение параметров сверточного ядра, добавление паддинга в слоях свертки для сохранения размерности тензора не устранили проблему. Отладка и изменение параметров слоя max_pool2d в EmbeddingNet, приводили к новым ошибкам. Было принято решение отказаться от этого подхода и найти более простой и быстрый способ подружить данные с исходным кодом.

**Подход 2. Преобразование данных в формат 28x28**

Для исключения ошибок, и для тестирования было принято решение, изменить формат данных, транформировать матрицу под размеры MNIST (28,28), с которым код умеет работать.

Входные данные для каждого паттерна имеют размер (60, 5), эти данные переформатируем в размер (20, 15) и вставляем в нулевой массив размером 28x28, заполняя верхнюю левую часть “изображения”. Данный подход использовался для предварительного теста.

Формат данных подошел, но результаты разбиения на кластеры ожидаемо были неудачными, предположительно из-за того, что меняя размер под 28 на 28 мы теряем зависимости заложенные природой данных.

**Подход 3. Преобразование данных в формат 60x60**

Для сохранения зависимостей в данных, формы и размерности временных шагов, характеристик, учитывая природу данных и чтобы при этом сохранить квадратную форму матрицы для адаптации к коду, было принято решение переформатировать входные данные для каждого паттерна (60,5) в размер (60,60) и вставить данные в центральную часть нового массива. Тогда временные шаги сохраняются в строках, характеристики сохраняются в столбцах, как в исходных данных.

Реализовано в файле datasets.py, функция def reshape(self, data) в классе class CustomDataset(Dataset).

Далее, данные передаются через светочную нейросеть  - класс class EmbeddingNet(nn.Module), файл networks.py. Размерность данных после сверток стала 64x12x12. Для адаптации нового формата данных, необходимо изменить входной размер полносвязной части сети self.fc, чтобы она принимала данные формата 64x12x12 после прохождения через сверточные и подвыборочные слои [*self.fc = nn.Sequential(nn.Linear(64 * 12 * 12, 256)]*.

# 4. Оформление кода на GitHub

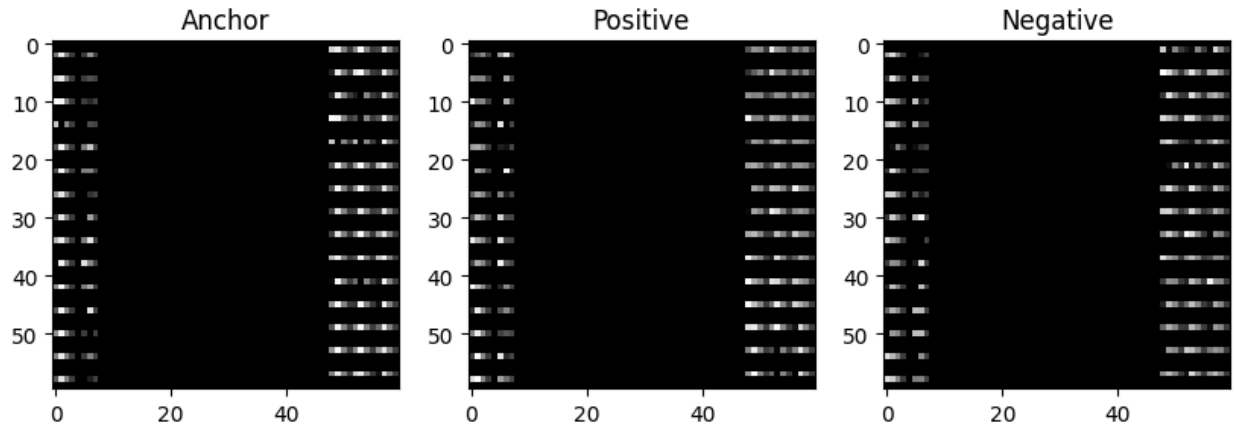
*Ссылка на репозиторий с кодом:*

https://github.com/timurkupaev/siamese-triplet-custom-dataset

# 5. Обсуждение результатов

С учетом изменения формата данных под матрицу 60x60 не удалось добиться разделения на кластеры. В случае с моделью Классификацией с softmax это можно объяснить малым количеством данных для данного подхода. В вариантах кода с триплетами, триплеты (anchor positive negative) создаются и формируются, но неверно передаются в модель, из-за чего модель не обучается. Требуется более детальное погружение в структуру данных, дальнейшая настройки модели, дополнительная проверка передаваемых данных, изменение процесса передачи триплета в модель для обучения.

*Пример сформированного триплета (формируется, но не передается в модель)*




Ожидаемые результаты: Разбиение на два кластера данных, для моделей сиамской сети, контрастивной и триплетной потерь с негативным отбором. Отсутствие видимого разделения для классификации с softmax.

С учетом ограничений по времени, были выбраны “быстрые” решения с надеждой на быструю победу, которые не привели к ожидаемому результату. В частности - переход на использование квадратной матрицы 60x60, вместо более детального и глубокого погружения в причину проблем и ошибок возникающих со слоем max_pool2d при использовании оригинального размера данных.

## 6. Процесс выполнения задач

**27 мая**

- изучение паспорта
- изучение статьи: https://arxiv.org/pdf/1503.03832.pdf (FaceNet)
- изучение статьи: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
- анализ слабых мест модели, поиск альтернатив, обзор статей по марковским цепям, байесовским сетям доверия
- составление документа с рекомендациями, описанием паспорта, завершение этапа 1

**28 мая**

- детальное изучение статьи по FaceNet
- изучение кода https://github.com/ikonushok/siamese-triplet
- замешательство в условиях тестового задания, уточнение информации

- составление блок схем вариантов формирования триплетов для вариантов кода 3,4,5

**29 мая**

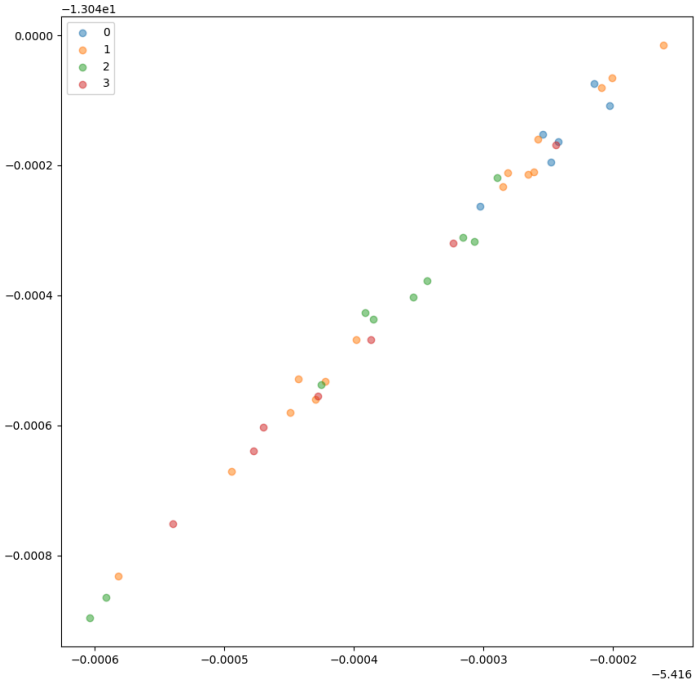
- запуск, изучение кода на гите
- формирование и интеграция класса для обработки кастомного датасета
- изменение параметров слоев в EmbeddingNet
- отладка и работа с ошибками, возникающими при использовании max_pool2d в EmbeddingNet

**30 мая**

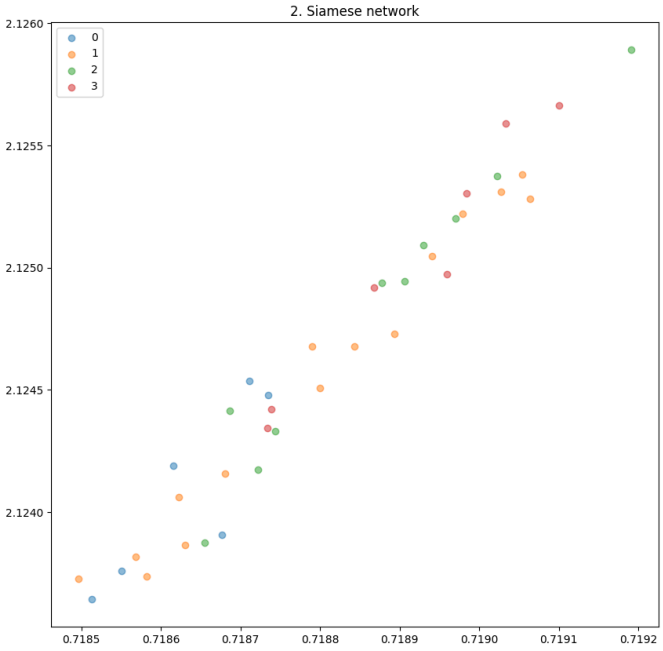
- преобразование данных в формат 28x28 для тестирования
- преобразование данных в формат 60x60 и адаптация данных к исходному коду
- запуск обучения, настройка гиперпараметров
- проверка формирования данных, поиск ошибок, отладка
- оформление файла описания, загрузка кода на гитхаб, завершение этапа 2

# Appendix:

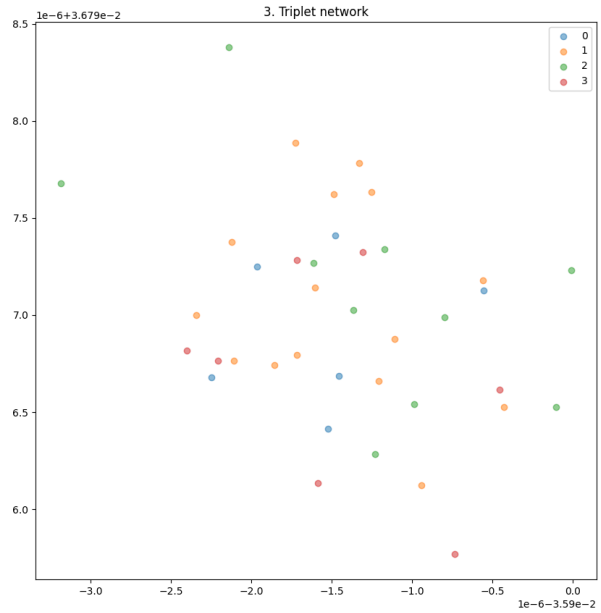
1. Classification with softmax



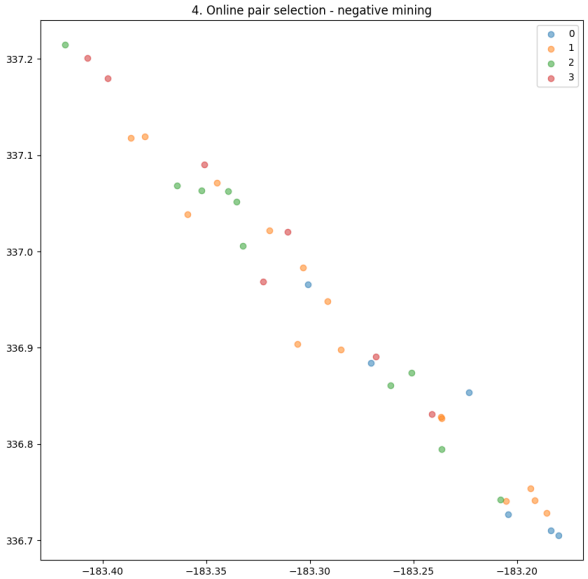
2. Siamese network



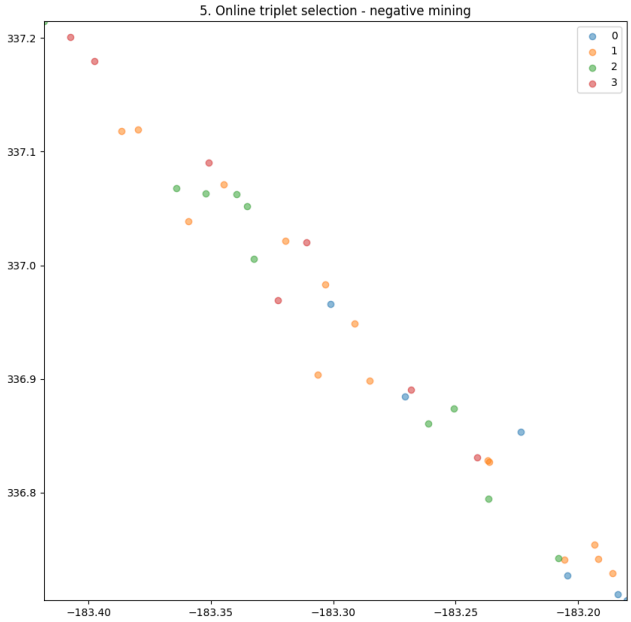

3. Triplet network




4. Online pair selection - negative mining




5. Online triplet selection negative mining


5. Online triplet selection - negative mining
0
1
2
3
337.2
337.1
337.0
336.9
336.8
−183.40
−183.35
−183.30
−183.25
−183.20